Rethinking Byzantine Robustness in Federated Recommendation from Sparse Aggregation Perspective

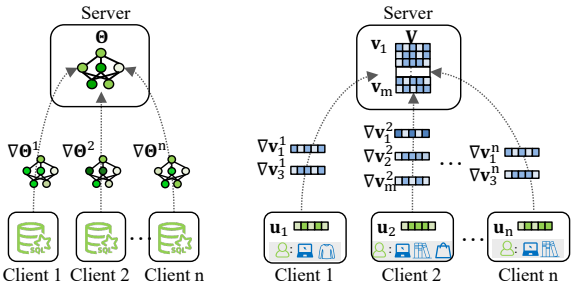
Zhongjian Zhang[1]*, Mengmei Zhang[2]*, Xiao Wang[3], Lingjuan Lyu[4]
Bo Yan[1], Junping Du[1], Chuan Shi[1]†
[1]Beijing University of Posts and Telecommunications
[2]China Telecom Bestpay
[3]Beihang University
[4]Sony AI
{zhangzj, boyan, junpingdu, shichuan}@bupt.edu.cn, zhangmengmei@bestpay.com.cn, xiao_wang@buaa.edu.cn, lingjuan.lv@sony.com

Abstract

To preserve user privacy in recommender systems, federated recommendation (FR) based on federated learning (FL) emerges, keeping the personal data on the local client and updating a model collaboratively. Unlike FL, FR has a unique sparse aggregation mechanism, where the embedding of each item is updated by only partial clients, instead of full clients in a dense aggregation of general FL. Recently, as an essential principle of FL, model security has received increasing attention, especially for Byzantine attacks, where malicious clients can send arbitrary updates. The problem of exploring the Byzantine robustness of FR is particularly critical since in the domains applying FR, e.g., e-commerce, malicious clients can be injected easily by registering new accounts. However, existing Byzantine works neglect the unique sparse aggregation of FR, making them unsuitable for our problem. Thus, we make the first effort to investigate Byzantine attacks on FR from the perspective of sparse aggregation, which is non-trivial: it is not clear how to define Byzantine robustness under sparse aggregations and design Byzantine attacks under limited knowledge/capability. In this paper, we reformulate the Byzantine robustness under sparse aggregation by defining the aggregation for a single item as the smallest execution unit. Then we propose a family of effective attack strategies, named **Spattack**, which exploit the vulnerability in sparse aggregation and are categorized along the adversary's knowledge and capability. Extensive experimental results demonstrate that Spattack can effectively prevent convergence and even break down defenses under a few malicious clients, raising alarms for securing FR systems.

Introduction

As an essential way to alleviate information overload, recommender systems are widely used in e-commerce (Ying et al. 2018), media (Wang et al. 2018; Wu et al. 2019), and social network (Fan et al. 2019), recommending items that users may be interested in. Despite the remarkable success, conventional recommender systems require centrally storing users' personal data for training, increasing privacy risks.



(a) Dense aggregation in general FL. **(b) Sparse aggregation in FR.**

Figure 1: Comparisons between dense aggregation of general FL and the unique sparse aggregation of FR.

Recently, federated learning (FL) (McMahan et al. 2016) has emerged as a privacy-preserving paradigm and successfully applied to the recommendation area. In federated recommendation (FR) (Sun et al. 2024; Luo, Xiao, and Song 2022), the global item embeddings are uploaded to a central server for aggregation. Meanwhile, each user's interaction data and privacy features are kept on the local client. In this way, the privacy of local data is well protected.

Unlike general FL systems, FR has a unique sparse aggregation mechanism. As shown in Fig. 1, for general FL, each element (circle) of model parameters can be updated by all n clients, named dense aggregation. While for FR, the interactions of users and items are usually sparse (Ma et al. 2008), resulting in each item's embedding can only be updated by partial clients. For example, client n can only produce and send substantive gradients $\{\nabla \boldsymbol{v}_1^n, \nabla \boldsymbol{v}_3^n\}$ for its interacted items $\{v_1, v_3\}$. For the remaining items, the updates are zero vectors or empty, named sparse aggregation.

By far, FR has provided satisfactory performance without collecting users' private data, extending recommendation applications to privacy-sensitive scenarios. Despite success, the model security, as an essential principle, has received increasing attention. Here we consider the worst-case attack, i.e., Byzantine attack (Fang et al. 2024, 2019; Blanchard et al. 2017a), where attackers are omniscient and collusive, and can control several clients to upload arbitrary

*These authors contributed equally.
†Corresponding author.

malicious gradients. Note that Byzantine robustness is especially critical for FR, since in the domains applying FR, e.g., e-commerce, malicious clients can be injected easily by registering new accounts. This raises one question naturally: *With the unique sparse aggregations, how robust the federated recommendation model is against Byzantine attacks?*

For this question, existing Byzantine works cannot be directly employed, since they mainly focus on dense aggregation in general FL (Rodr'iguez-Barroso et al. 2022; Xu et al. 2021; Blanchard et al. 2017b). Despite a few prior attacks against FR emerges (Yuan et al. 2023; Yu et al. 2023; Rong et al. 2022; Wu et al. 2022), they also neglect to analyze how the sparse aggregation affects the robustness of FR. We answer this problem by solving two challenges: (1) How to define Byzantine robustness under sparse aggregations? Existing Byzantine attacks and defenses are mainly defined based on the dense aggregation mechanism in the general FL. In FR, due to sparse user-item interaction, for an item, its embedding is updated only by its interacted users, and the remaining users upload zero-valued or empty updates. So the aggregated item embedding may be skewed towards zero-valued when directly applying existing dense aggregators, since the zero-value update is majority. Hence, it is vital to transfer them into FR and re-examine their theoretical guarantee and effectiveness. (2) How to design general Byzantine attacks against FR for attackers with different levels of knowledge and capability in reality. Specifically, it is hard to have full knowledge of all users, due to the large number of participating users in FR. Besides, since user-item interactions are usually sparse, Byzantine clients should not update too many items. Otherwise, a monitor based on the number of user interactions can be triggered easily under such aggressive modifications (Wu et al. 2022).

In this paper, we make the first effort to investigate the Byzantine robustness of federated recommendation from the perspective of sparse aggregations. For the first challenge, we transfer the existing aggregators to FR by treating the aggregation for a single item as the smallest execution unit. Namely, for each item embedding, the gradients are collected and aggregated separately and concurrently. Based on this, we further point out that such a sparse aggregation mechanism of FR will lead to a unique Byzantine vulnerability: items with different degrees receive different amounts of updates, leading to individual robustness. The degrees of all items usually meet long tail distribution in reality (Abdollahpouri, Burke, and Mobasher 2019), where most items (named tailed items) are only interacted with by a few users, making them extremely fragile. For the second challenge, we design a series of attack strategies, named **Spattack**, based on the vulnerability from sparse aggregation in FR. Then we categorize them along the attacker's knowledge and capability into four classes. To be specific, following (Xie, Koyejo, and Gupta 2020; Fang et al. 2019; Baruch, Baruch, and Goldberg 2019), we consider both omniscient attacker (Spattack-O) and limited non-omniscient attacker (Spattack-L). Then we further divide them depending on whether limiting the maximum number of each client's poisoned items or not. In summary, our contributions are three folds:
(1) We first systematically study the Byzantine robustness of FR from the perspective of unique sparse aggregation, by treating the aggregation for a single item as the smallest execution unit. We theoretically analyze its convergence guarantee and point out a special vulnerability of FR.
(2) We propose a family of effective attack strategies, named Spattack, utilizing the vulnerability from sparse aggregation. Then Spattack can be categorized into four different types along attacker's knowledge and capability.
(3) We perform experiments on multiple benchmark datasets for different FR systems. The results show that our Spattack can prevent the convergence of vanilla even defense FR models by only controlling a few malicious clients.

Background and Preliminary

Centralized Recommendation

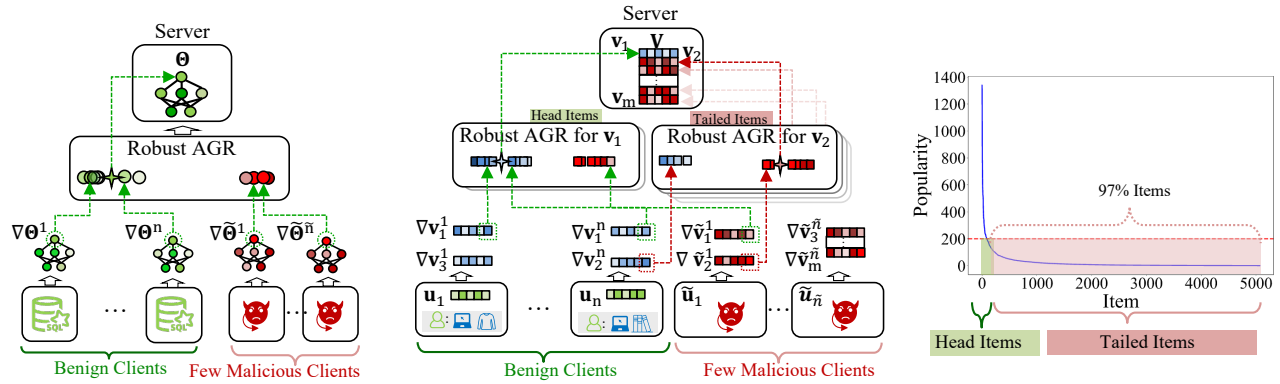
Here, a recommender system contains a set of users $\mathcal{U} = \{u_1, \cdots, u_n\}$ and a set of items $\mathcal{V} = \{v_1, \cdots, v_m\}$, where n and m are the numbers of users and items, respectively. Each user $u_i \in \mathcal{U}$ has a local training dataset $\mathcal{D}_i$, consisting of implicit feedback tuples (u_i, v_j, r_{ij}). These tuples represent user-item interactions (e.g., purchased, clicked), where $r_{ij} = 1$ and $r_{ij} = 0$ indicate positive and negative instances, respectively, i.e., whether u_i interacted with v_j. For each user u_i, we define $\mathcal{V}_{u_i} = \{v_j \in \mathcal{V} | (u_i, v_j, r_{ij}) \in \mathcal{D}_i\}$ as the set of the items that interact with u_i. Let $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m]$ denote the embeddings of users and items, respectively. The recommender system is trained to predict the rating score $\hat{r}_{ij} = f_{\boldsymbol{\Theta}}(\boldsymbol{u}_i, \boldsymbol{v}_j)$ between u_i and v_j, where $\hat{r}_{ij}$ represents how much u_i likes v_j, $f_{\boldsymbol{\Theta}}$ is the score function, $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\Theta}\}$ are learnable parameters. Then, the system recommends an item list for each user that they might be interested in by sorting the rating scores. In traditional centralized training, the personal dataset $\mathcal{D}_i$ of each user u_i is stored on a central server, yielding a total dataset $\mathcal{D}$ for model training, which will increase the privacy risks.

Federated Recommendation

Considering privacy issues, in FR, the privacy data $\mathcal{D}_i$ of user u_i is kept on the local device. The shared model parameters $\boldsymbol{V}$ and $\boldsymbol{\Theta}$ are aggregated over clients by sending the local gradients to a central server. According to the base recommender, the parameters $\boldsymbol{\Theta}$ are different: in Matrix Factorization (MF) recommender models, the interaction function is fixed and $\boldsymbol{\Theta}$ is an empty set. In deep learning-based recommender models, $\boldsymbol{\Theta}$ is the set of weights of neural networks. Following (Rong et al. 2022), we adopt the classic and widely used MF as the base recommender for simplicity, where f is fixed to be dot product, i.e., $\hat{r}_{ij} = \boldsymbol{u}_i \odot \boldsymbol{v}_j$. Following (Rong et al. 2022), we take Bayesian Personalized Ranking (BPR) (Rendle et al. 2009), a pairwise personalized ranking loss, as the local loss of each client:

$$\mathcal{L}_i(\boldsymbol{u}_i, \boldsymbol{V}) = - \sum_{\substack{v_j,\, v_k \in \mathcal{V}_{u_i} \\ r_{ij}=1 \,\wedge\, r_{ik}=0}} \ln \sigma\left(\hat{r}_{ij} - \hat{r}_{ik}\right), \qquad (1)$$

where σ is the logistic sigmoid function. It assumes that the user prefers the positive items over all negative items.



(a) Existing Byzantine attack against general FL. **(b) Our Byzantine attack against FR.** **(c) Item popularity of Steam dataset.**

Figure 2: Analysis of Byzantine robustness in general FL and FR. Under Byzantine attacks, a robust aggregator can filter outliers in general FL, but fails to defend for tailed items' embedding.

In each training iteration, the central server sends the current item embeddings $\boldsymbol{V}^t$ to all clients. For each user u_i, the client computes loss $\mathcal{L}_i(\boldsymbol{u}_i^t, \boldsymbol{V}^t)$ then locally updates its private user embedding at epoch t as follows:

$$\boldsymbol{u}_i^{t+1} \leftarrow \boldsymbol{u}_i^t - \eta \cdot \nabla \boldsymbol{u}_i^t, \tag{2}$$

where η is the learning rate. Then u_i uploads its local item embedding gradients $\nabla \boldsymbol{V}^{i,t}$ to a central server. After collecting gradients from all clients, the server updates $\boldsymbol{V}^t$ by:

$$\boldsymbol{V}^{t+1} \leftarrow \boldsymbol{V}^t - \eta \cdot \sum_{i \in [n]} \nabla \boldsymbol{V}^{i,t}. \tag{3}$$

As shown in Fig. 1(b), for each user u_i, the private interaction history (item list) and user embedding (green $\boldsymbol{u}_i$ vector) is preserved on the local client device, and only the gradients of item embeddings $\boldsymbol{V}$ are sent. Throughout the training stage, all users' privacy is well protected.

Byzantine Attack and Defense

Byzantine Attack. In Byzantine attacks, the attacker aims to degrade model performance and even prevent convergence by controlling a few malicious clients. As shown in Fig. 2(a), malicious client $\tilde{u}_i$ is allowed to send arbitrary (red) gradient $\nabla \tilde{\boldsymbol{\Theta}}^i$. Following existing Byzantine attack studies (Xu et al. 2021; Fang et al. 2019; Baruch, Baruch, and Goldberg 2019), considering the worst case, we assume attackers have full knowledge of all benign gradients $\{\nabla \boldsymbol{\Theta}^1, \cdots, \nabla \boldsymbol{\Theta}^n\}$ and all the malicious clients are collusive by default, which help to understand the severity of model poisoning threats.

Byzantine Defense. Since servers have no access to the raw training data of clients, the defense is generally implemented on the server side as a robust aggregator, which can filter Byzantine updates and guarantee model convergence. As shown in Fig. 2(a), let $\{\nabla \boldsymbol{\Theta}^1, \cdots, \nabla \boldsymbol{\Theta}^n\}$ be the gradient vectors of n benign clients in FL. The server collects and aggregates the training gradient of each client model using a federated aggregator. In non-robust FL settings, coordinate-wise Mean in form of $\text{MEAN}(\nabla \boldsymbol{\Theta}^1, \cdots, \nabla \boldsymbol{\Theta}^n) = \frac{1}{n} \sum_{i=1}^n \nabla \boldsymbol{\Theta}^i$ is an effective aggregation rule. However, MEAN can be manipulated by several malicious clients (Blanchard et al. 2017b). Therefore, multiple robust aggregators (Yin et al. 2018; Blanchard et al. 2017b; Xu et al. 2021) are proposed to filter the Byzantine updates. For example, coordinate-wise Median aggregator computes the median for each element Θ_i in parameter $\boldsymbol{\Theta}$ across all clients, yielding 0.5 breakdown point (Yin et al. 2018). Namely, when the fraction of malicious clients is less than 0.5, the Median aggregator can guarantee the model convergence under Byzantine attacks, yielding the correct gradient (green star) in Fig. 2(a).

Methodology

Problem Definition

For Byzantine attacks, attackers can inject some Byzantine users $\tilde{\mathcal{U}} = \{\tilde{u}_1, \cdots, \tilde{u}_{\tilde{n}}\}$, limiting the proportion of malicious ones less than ρ, i.e., $\tilde{n}/(n+\tilde{n}) < \rho$. A malicious client $\tilde{u}_i$ can upload arbitrary gradient values $\nabla \tilde{\boldsymbol{V}}^{i,t}$ at any epoch t, to directly perturb the item embedding. The server will collect and aggregate all gradients including benign $\{\nabla \boldsymbol{V}^{1,t}, \cdots, \nabla \boldsymbol{V}^{n,t}\}$ and malicious $\{\nabla \tilde{\boldsymbol{V}}^{1,t}, \cdots, \nabla \tilde{\boldsymbol{V}}^{\tilde{n},t}\}$. Let $\text{AGR}(\cdot)$ be the aggregation operator of federated learning, which can be the most common $\text{MEAN}(\cdot)$ or statistically robust $\text{MEDIAN}(\cdot)$. Our Byzantine attacker aims to prevent model convergence, namely, keeping the recommendation loss $\mathcal{L}_i$ from decreasing. Formally, in FR, the objective of the Byzantine attack is defined as the following optimization problem:

$$\begin{aligned}
&\max_{\{\nabla \tilde{\boldsymbol{V}}_i^t : i \in \tilde{n}\}} \sum_{i=1}^{n} \left(\mathcal{L}_i(\boldsymbol{u}_i^{t+1}, \boldsymbol{V}^{t+1}) - \mathcal{L}_i(\boldsymbol{u}_i^t, \boldsymbol{V}^t) \right), \\
&\text{s.t. } \boldsymbol{V}^{t+1} = \boldsymbol{V}^t - \eta \cdot \text{AGR}(\{\nabla \boldsymbol{V}^{i,t} : i \in [n]\} \\
&\qquad \cup \{\nabla \tilde{\boldsymbol{V}}^{i,t} : i \in [\tilde{n}]\}), \\
&\boldsymbol{u}_i^{t+1} = \boldsymbol{u}_i^t - \eta \nabla \boldsymbol{u}_i^t, \quad \text{for } i \in [n], \\
&\frac{\tilde{n}}{n+\tilde{n}} \leq \rho,
\end{aligned} \tag{4}$$

where attackers aim to find the optimal set of malicious gradients $\{\nabla\tilde{\boldsymbol{V}}_i^t : i \in [\tilde{n}]\}$, to raise the loss after updating. Before solving this optimization problem, we find that FR has a unique sparse aggregation mechanism defined as follows:

Definition 1. ***(Dense/Sparse Aggregation).*** *Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be the shared model parameter vector. If there exists an element θ_i ($i \in [d]$) for which only a subset of clients can produce valuable updates, the parameter $\boldsymbol{\theta}$ is sparsely aggregated. If all clients can support it, it is a dense aggregation.*

As shown in Fig. 2(a), in general FL, each element (green circle) of model parameters $\boldsymbol{\Theta}$ is assumed to be involved in all clients' loss functions, i.e., dense aggregation. Different from FL, for a client u_i in FR, not all item embeddings $\boldsymbol{V} = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\}$ are employed in local loss $\mathcal{L}_i$ in Eq. 1, i.e., sparse aggregation. For example, client u_1 in Fig. 2(b) only computes valuable gradients $\{\nabla\boldsymbol{v}_1^1, \nabla\boldsymbol{v}_3^1\}$ for items v_1 and v_3, while the gradients for the remaining items are either zero or an empty set. When directly applying the aggregators on all $\boldsymbol{V}_i$, the embedding will be skewed towards zero value. Therefore, we need to adapt them to FR.

Adapting Dense Aggregator to Sparse. We adapt existing aggregators from dense to sparse aggregation by treating the aggregation for a single item as the smallest execution unit. As shown in Fig. 2(b), the aggregator is conducted separately for each item. Taking the embedding of j-th item as an example, the embedding is updated by:

$$\boldsymbol{v}_j^{t+1} = \boldsymbol{v}_j^t - \eta \cdot \text{AGR}(\{\nabla\boldsymbol{v}_j^{i,t} | \text{ user } i \in \mathcal{U}_{v_j}\}), \quad (5)$$

where $\mathcal{U}_{v_j}$ is the set of users that the item v_j interacts with, $\nabla\boldsymbol{v}_j^{i,t}$ is the gradient of item v_j sent from client u_i at epoch t. Only if the user u_i has interaction with item v_j, the gradients $\nabla\boldsymbol{v}_j^{i,t}$ can be aggregated to $\boldsymbol{v}_j^{t+1}$ separately and concurrently. Intuitively, the numbers of received gradients are varied for different items, leading to each item having personal robustness. Therefore, we need to theoretically re-examine the convergence guarantee of existing aggregators against Byzantine attacks under the sparse aggregation.

Byzantine Robustness Analysis

Robustness of FR without Defense. Like general FL, FR without defense often uses the Mean aggregator to compute the average of input gradients, which is highly susceptible to Byzantine attacks. Even one malicious client can also destroy the Mean aggregator as stated in Proposition 1.

Proposition 1. *For each item v_j, let $\{\nabla\boldsymbol{v}_j^{i,t} | \text{ user } i \in \mathcal{U}_{v_j}\}$ be the set of benign gradient vectors at epoch t. Consider a Mean aggregator averaging updates for each element. Let $\nabla\tilde{\boldsymbol{v}}_j$ be a malicious update with arbitrary values in $\mathbb{R}^d$. The output of MEAN$(\{\nabla\boldsymbol{v}_j^{i,t} | \text{ user } i \in \mathcal{U}_{v_j}\} \cup \nabla\tilde{\boldsymbol{v}}_j) = \frac{1}{|\mathcal{U}_{v_j}|+1}(\sum_{i\in\mathcal{U}_{v_j}} \nabla\boldsymbol{v}_j^{i,t} + \nabla\tilde{\boldsymbol{v}}_j)$ can be controlled as zero vector by only single malicious $\nabla\tilde{\boldsymbol{v}}_j$. When all the items are attacked, one malicious client can prevent convergence.*

Proof. If the attacker registers one malicious client, where the embedding gradient of each item v_j is $\nabla\tilde{\boldsymbol{v}}_j = -\sum_{i\in\mathcal{U}_j} \nabla\boldsymbol{v}_j^{i,t}$, the output of aggregator is zero vector, which can prevent convergence. □

Robustness of FR with Defense. The most common defense method is to use aggregators that are statistically more robust against outliers than Mean. In these defenses, FL models have a consistently high breakdown point, e.g., when $\rho < 50\%$, Median can theoretically guarantee the convergence of FL as proved in (Yin et al. 2018). However, we find that FR models have varied breakdown points for different items, which depends on the item's degree. Specifically, each item embedding can only be updated by specific clients with whom the item interacts. Obviously, the popular item with massive updates is more robust. Unfortunately, in FR, only a few items interact frequently (head items), while the remaining items interact less frequently (tailed items). We plot the popularity (item degree) of the Steam recommendation dataset (Cheuque, Guzmán, and Parra 2019) in Fig. 2(c). We find that 97% tailed items (red long tail area) have interactions less than 200 times, and only 3% head items (green area) interact frequently over 200 times. Therefore, existing statistically-based FL defenses will fail to guarantee the convergence of most items. Formally, let x be the degree of an item and $p(x)$ be its probability. We assume that the probability distribution can be defined as a typical power-law distribution $p(x) = Cx^{-\beta}$, where C is a normalization constant and β is the scaling parameter. The failure of defenses can be formally characterized as follows:

Proposition 2. *Let α be the breakdown point of robust federated aggregator, the amount of benign and malicious clients are n and $\tilde{n}$ respectively, and β is the scaling parameter of the power-law distribution of items' degree with constant C. Then at least $1 - \frac{C}{\beta-1}(\frac{1-\alpha}{\alpha}\tilde{n})^{(1-\beta)}$ percent of items' embeddings can be broken down.*

The proof of Proposition 2 refers to the Appendix. Taking the Steam dataset as an example, the degree distribution of items can be modeled as a typical form of power-law distribution as shown in Fig. 2(c). For example, if an attacker can control $\rho = 5\%$ clients, each item can receive 197 malicious gradients at most. Clearly, for 97% tailed items that interact less than 200 times, few malicious (red) updates can become the majority and dominate the aggregation. In this case, the statistically robust Median aggregator will pick the majority (red circles), yielding the malicious output (red star). In conclusion, due to the sparse aggregation vulnerability of FR, statistically robust aggregators in FL can also be easily broken down by Byzantine attacker.

Spattack: Byzantine Attack Strategies

Intuition. In Eq. 4, the attacker aims to keep the recommendation loss from decreasing to prevent recommender convergence. Considering the unique vulnerability from sparse aggregation, i.e., the majority of tailed items have a lower breakdown point, we can conclude that: (1) The gradients are farther away from true gradients, the more considerable corruption is. (2) More items are disrupted in the training process, leading to more powerful attacks. Therefore, the attack objective of the proposed Spattack can be simplified to maximally uploading gradients farther away from true gradients and greedily disrupting the embeddings of items.

Attack Taxonomy. In real scenarios, depending on the

Spattack	O-D	O-S	L-D	L-S
Knowledge	✓	✓	✗	✗
Capability	✓	✗	✓	✗

Table 1: Attack Taxonomy. For each malicious client in Spattack, knowledge means knowing benign gradients, and capability refers to poisoning all items.

attacker's knowledge about benign gradients and the maximum number of poisoned items in each malicious client, we outline different scenarios of Spattack that can be launched. As shown in Tab. 1, we have four possible scenarios:

Spattack-O-D is considered a worst case, where the attacker is both omniscient and omnipotent, i.e., attackers can obtain benign gradients at each epoch and the maximum number of poisoned items is not limited. Following the first intuition that the malicious gradients farther away from true gradients can cause larger corruption, attackers upload the gradients in the opposite direction of the benign ones. Formally, for a item v_j, we collect benign gradient $\nabla \boldsymbol{v}_j^{i,t}$ from u_i, where u_i interacts with v_j, i.e., $u_i \in \mathcal{U}_{v_j}$. Then we compute the sum of the collected benign gradients to obtain the expected gradient $\nabla \bar{\boldsymbol{v}}_j^t = \sum_{u_i \in \mathcal{U}_{v_j}} \nabla \boldsymbol{v}_j^{i,t}$. Lastly, each malicious client $\tilde{u}_i \in \tilde{\mathcal{U}}$ will upload malicious gradients $\nabla \tilde{\boldsymbol{v}}_j^{i,t} = -\frac{1}{|\tilde{\mathcal{U}}|} \nabla \bar{\boldsymbol{v}}_j^t$. Following the second intuition that greedily disrupts items, the attack effectiveness will be maximized by uploading poisoning gradients for all items. In this attack, the non-robust Mean aggregator will output zero gradients, while statistically robust aggregators will select malicious gradients for the majority of tailed items, preventing the convergence of item embeddings. According to Proposition 1 and Proposition 2, even only having a small portion of malicious clients, Spattack-O-D can still guarantee to disrupt the majority of item embeddings.

Spattack-L-D uploads random noise as malicious gradients for all items, where attackers are non-omniscient but omnipotent, i.e., attackers do not have any knowledge about the benign gradients but can attack all items. Specifically, attackers construct the malicious gradient by randomly sampling from the Gaussian noise and keeping the same noise in all malicious clients. Under the Mean aggregator, the aggregated gradients can be skewed by such noise. Even worse, the statistically robust aggregators, e.g., Median, can pick the uploaded random noise as output for tailed items. So this attack can still prevent model convergence.

Spattack-O-S and Spattack-L-S only upload malicious gradients for partial items, where attackers are non-omnipotent. Let $\tilde{m}_{max}$ be the maximum number of poisoned items in each malicious client. The larger $\tilde{m}_{max}$, the stronger the attack, but the excessive $\tilde{m}_{max}$ may lead to the attack being detected. To limit malicious users to behaving like benign users, we restrict $\tilde{m}_{max}$ as the maximum number of interactions in benign clients. Specifically, to make the injections of malicious clients as imperceptible and effective as possible, based on the distribution of item popularity, we use a sampling operation to determine the poisoned items for each malicious client. Therefore, the attacker can automatically assign more malicious gradients to the items having more interactions. Then we generate malicious gradients based on the opposite benign gradients (Spattack-O-S) or random noise (Spattack-L-S), respectively.

Dataset	#Users	#Items	#Edges	Sparsity
ML100K	943	1,682	100,000	93.70 %
ML1M	6,040	3,706	1,000,209	95.53 %
Steam	3,753	5,134	114,713	99.40 %

Table 2: Statistics of datasets.

Experiment

We conduct extensive experiments to answer the following research questions. **RQ1**: How does Spattack perform compared with existing Byzantine attacks? **RQ2**: Can Spattack break the defenses deployed on FR? **RQ3**: Can Spattack transfer to different FR systems? **RQ4**: How do hyperparameters impact on Spattack? Given the limited space, please refer to the Appendix for more detailed experiments.

Experimental Setup

Datasets and Federated Recommender Systems. Following (Rong et al. 2022), Spattack is evaluated on three widely used datasets, including movie recommendation datasets ML1M and ML100K (Harper and Konstan 2016), and game recommendation dataset Steam (Cheuque, Guzmán, and Parra 2019). The dataset statistics refer to Tab. 2. The test set is divided with the leave-one-out method, where the latest interaction of a user is left as the test set and the remaining interactions as the training set. FedMF (Rong et al. 2022) and the SOTA FedGNN (Wu et al. 2021) are selected as evaluation models. More dataset and reproducibility details are in the Appendix.

Evaluation Protocols. We utilize two common evaluation protocols, including hit ratio (HR) and normalized discounted cumulative gain (nDCG) at ranks 5 and 10. For each user, since ranking the test item among all items is time-consuming, following the widely-used strategy (He et al. 2017), we randomly sample 100 items that do not interact with the user, then rank the test item among the 100 items. Notably, all metrics are only calculated on benign clients.

Baselines. We compare Spattack with two categories of methods. First is the data poisoning attack, where attackers generate malicious gradients by modifying training data. LabelFlip (Tolpegin et al. 2020) flips training labels for poisoning, while FedAttack (Wu et al. 2022) uses misaligned samples. Second is model poisoning attacks, where attackers directly modify the uploaded gradients. Gaussian (Fang et al. 2019) estimates the Gaussian distribution of benign gradients and then samples from it. LIE (Baruch, Baruch, and Goldberg 2019) adds small amounts of noise towards the average of benign gradients. Cluster (Yu et al. 2023) uploads malicious gradients that aim to make item embeddings collapse into several dense clusters. Fang (Fang et al. 2019) adds noise to opposite directions of the average normal gradient. More details can be found in the Appendix.

Byzantine Defense Strategies. We evaluate Spattack performance under the following defense strategies: Mean is the vanilla non-robust aggregator that computes the mean value

Dataset	Metric	Clean	LabelFlip	FedAttack	Gaussian	LIE	Cluster	Fang	Type L-S	Type L-D	Type O-S	Type O-D
ML100K	HR@5	0.2513	0.2517 (-3%)	0.2550 (-2%)	0.2550 (-2%)	0.2539 (-2%)	0.2461 (-5%)	0.1957 (-25%)	0.1018 (-59%)	0.0721 (-71%)	0.0594 (-76%)	**0.0530 (-79%)**
	nDCG@5	0.1643	0.1706 (-3%)	0.1721 (-2%)	0.1729 (-1%)	0.1724 (-2%)	0.1678 (-4%)	0.1229 (-30%)	0.0620 (-62%)	0.0380 (-77%)	0.0362 (-78%)	**0.0339 (-79%)**
	HR@10	0.4051	0.4083 (-2%)	0.4094 (-2%)	0.4116 (-2%)	0.4116 (-2%)	0.3982 (-5%)	0.2919 (-30%)	0.2163 (-47%)	0.1601 (-60%)	0.0997 (-75%)	**0.0944 (-77%)**
	nDCG@10	0.2131	0.2206 (-2%)	0.2213 (-2%)	0.2230 (-1%)	0.2229 (-1%)	0.2166 (-4%)	0.1541 (-32%)	0.0980 (-54%)	0.0658 (-69%)	0.0492 (-77%)	**0.0470 (-78%)**
ML1M	HR@5	0.3121	0.3051 (-1%)	0.3056 (-1%)	0.3053 (-1%)	0.3054 (-1%)	0.3033 (-2%)	0.2827 (-9%)	0.1007 (-68%)	0.0921 (-71%)	0.0925 (-70%)	**0.0907 (-71%)**
	nDCG@5	0.2054	0.2013 (-2%)	0.2021 (-1%)	0.2017 (-1%)	0.2018 (-1%)	0.2004 (-2%)	0.1858 (-9%)	0.0581 (-72%)	**0.0521 (-75%)**	0.0553 (-73%)	0.0549 (-73%)
	HR@10	0.4626	0.4632 (-1%)	0.4634 (-1%)	0.4634 (-1%)	0.4634 (-1%)	0.4592 (-2%)	0.3977 (-15%)	0.2141 (-54%)	0.1935 (-58%)	0.1753 (-62%)	**0.1679 (-64%)**
	nDCG@10	0.2539	0.2522 (-1%)	0.2528 (-1%)	0.2526 (-1%)	0.2526 (-1%)	0.2506 (-2%)	0.2231 (-12%)	0.0939 (-63%)	0.0846 (-67%)	0.0817 (-68%)	**0.0793 (-69%)**
Steam	HR@5	0.5729	0.4792 (-15%)	0.4798 (-15%)	0.4879 (-14%)	0.4862 (-14%)	0.4263 (-25%)	0.0278 (-95%)	0.0426 (-93%)	**0.0139 (-98%)**	0.0671 (-88%)	0.0685 (-88%)
	nDCG@5	0.3815	0.3157 (-17%)	0.3172 (-16%)	0.3216 (-15%)	0.3209 (-15%)	0.2750 (-27%)	0.0160 (-96%)	0.0261 (-93%)	**0.0080 (-98%)**	0.0390 (-90%)	0.0408 (-89%)
	HR@10	0.6933	0.6429 (-7%)	0.6431 (-7%)	0.6474 (-6%)	0.6471 (-6%)	0.6220 (-10%)	0.0619 (-91%)	0.0834 (-88%)	**0.0322 (-95%)**	0.1308 (-81%)	0.1287 (-81%)
	nDCG@10	0.4207	0.3685 (-12%)	0.3700 (-12%)	0.3732 (-11%)	0.3730 (-11%)	0.3386 (-19%)	0.0269 (-94%)	0.0391 (-91%)	**0.0138 (-97%)**	0.0593 (-86%)	0.0601 (-86%)

Table 3: Comparison of Spattack with baselines under a 3% malicious rate. Lower scores represent better attack effectiveness. We additionally report the performance drop (%) compared with the performance on the clean model.

of gradients for each dimension. Median (Yin et al. 2018) is a statistically robust aggregator with a 0.5 breakdown point, computing the element-wise median value. Trimmed-mean (Yin et al. 2018) trims several extreme values for each dimension and then averages the rest. Krum (Blanchard et al. 2017b) picks the gradient that is the most similar to other uploaded gradients. Norm (Suresh et al. 2019) clips the norm of gradients with a given threshold.

Attack Performance Evaluation (RQ1)

We compare the proposed Spattack against existing SOTA attack baselines under 3% malicious ratio. The experimental results are reported in Tab. 3, we find:

• Spattack can prevent FR convergence by controlling a few malicious clients. For example, Spattack can achieve a 47%-98% performance drop under 3% malicious clients, demonstrating that FR is extremely vulnerable to Spattack.

• Spattack significantly outperforms other baselines. The explanation is that Spattack fully utilizes the sparse aggregation vulnerability by greedily breaking more items. Specifically, LabelFlip and FedAttack only indirectly manipulate the gradient by modifying data, while LIE, Cluster and Fang directly manipulate the gradients and thus can achieve higher attack impacts. Although Fang also perturbs in opposite directions of benign gradients, the malicious gradients are skewed to zero vector without considering the sparse aggregation of FR, leading to less effective attacks.

• The results on Steam overall drop more than ML100K and ML1M. A possible reason is that Steam involves fewer interactions on average (referring to the sparsity in Tab. 2), meaning there are more tailed items, which makes the model more susceptible to attacks.

Attack Effectiveness under Defense (RQ2)

We evaluate Spattack under different defenses, where the malicious ratios ρ are set to 1%, 3% and 5% for omniscient Spattack-O, and 5%, 10% and 15% for the harder non-omniscient Spattack-L. Since TrimM and Krum assume the number of malicious updates is fixed for each item, but Spattack-O/L-S uploads varying numbers of updates, making them cannot be applied. As shown in Fig. 3, we find:

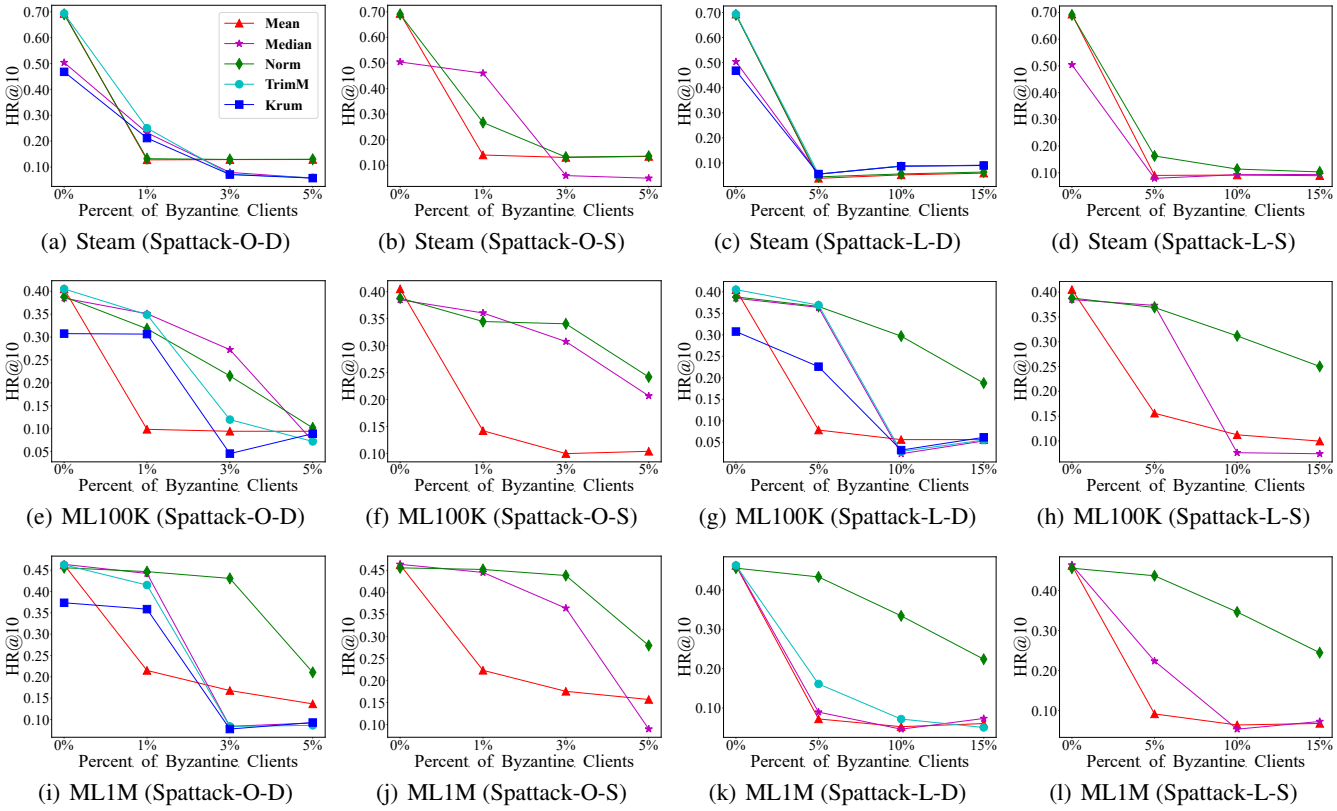
• With only 5% malicious clients, Spattack can dramatically degrade recommendation performance and even prevent convergence. The explanation is that different items have varied amounts of updates, and the defense of tailed items can be more easily broken than head items.

• When the attacker's knowledge and capability are limited, with the increasing malicious ratio ρ, the performance of defense FR consistently decreases even reaching an untrained model. The results also demonstrate that hiding the gradients of benign clients cannot protect FR, because the attacker can break the defense using only random noise.

Transferability of Attack (RQ3)

To assess the the Spattack's generalization, we perform attack on the SOTA FedGNN (Wu et al. 2021) by extra uploading the malicious gradients of the GNN model. No Defense and Defense correspond to mean and median aggregators, respectively. The malicious ratio ρ is 10%. More results and analysis are in Appendix. As shown in Fig. 4, we find:

• The performance of FedGNN dramatically drops under Spattack, demonstrating the common vulnerability of FedMF and FedGNN. Even though the parameters of GNN are densely aggregated, the attacker can still prevent convergence of model training by poisoning item embeddings.

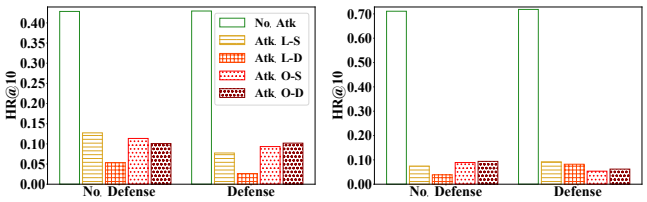


(a) Steam (Spattack-O-D) (b) Steam (Spattack-O-S) (c) Steam (Spattack-L-D) (d) Steam (Spattack-L-S)

(e) ML100K (Spattack-O-D) (f) ML100K (Spattack-O-S) (g) ML100K (Spattack-L-D) (h) ML100K (Spattack-L-S)

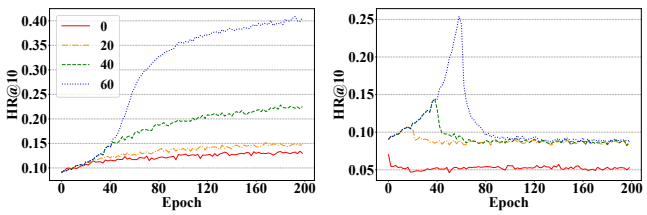
(i) ML1M (Spattack-O-D) (j) ML1M (Spattack-O-S) (k) ML1M (Spattack-L-D) (l) ML1M (Spattack-L-S)

Figure 3: Performance of Spattack against multiple defense strategies under different ratios of Byzantine clients.



(a) ML100K (b) Steam

Figure 4: Attack performance on FedGNN model.



(a) Spattack-O-D (c) Spattack-L-D

Figure 5: Attack performance on different start epochs.

• Spattack can achieve more effective attacks under defense. A possible reason is that for most items, i.e., tailed item, the malicious gradients can easily be the majority in its aggregation, so the Median AGR tends to pick the malicious gradient as output, while the poisoning in Mean AGR will be in remission by averaging malicious and benign gradients.

Hyperparameter Analysis (RQ4)

In Fig. 5, we show the convergence of FR under mean aggregators on Steam, where the malicious ratio is 10%, and the results correspond to starting attacks at 0, 20, 40 and 60. More results refer to the Appendix. We find Spattack with a small starting epoch tends to have better attack performance because the model has converged under a large starting epoch. Moreover, when Spattack is launched, Spattack-O prevents the model from continuing to converge, while Spattack-L causes the performance dramatically drops. The reason is that Spattack-O uploads malicious gradients with an average equal to the negative of the benign gradients' average, resulting in zero gradients after aggregation.

Conclusion

In this paper, we first systematically study the Byzantine robustness of FR from the perspective of sparse aggregation, where the item embedding can only be updated by partial clients, instead of full clients (dense aggregation in general FL). Then we design a series of attack strategies, called Spattack, based on the vulnerability from sparse aggregation in FR. Spattack can be employed by attackers with different levels of knowledge and capability. Extensive experimental results demonstrated that FR is extremely fragile to Spattack. In the future, we aim to design a more robust aggregator in FR from the perspective of sparse aggregation, which focuses on the robust aggregation for tailed items.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2023YFF0725103), the National Natural Science Foundation of China (U22B2038, 62322203, 62172052, 62192784), and the Young Elite Scientists Sponsorship Program (No.2023QNRC001) by CAST.

References

Abdollahpouri, H.; Burke, R.; and Mobasher, B. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. In *The Florida AI Research Society*.

Baruch, G.; Baruch, M.; and Goldberg, Y. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *Neural Information Processing Systems*.

Blanchard, P.; Mhamdi, E. M. E.; Guerraoui, R.; et al. 2017a. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*.

Blanchard, P.; Mhamdi, E. M. E.; Guerraoui, R.; et al. 2017b. Machine learning with adversaries: byzantine tolerant gradient descent. *Neural Information Processing Systems*.

Cheuque, G.; Guzmán, J.; and Parra, D. 2019. Recommender Systems for Online Video Game Platforms: the Case of STEAM. *Companion Proceedings of The 2019 World Wide Web Conference*.

Fan, W.; Ma, Y.; Yin, D.; et al. 2019. Deep social collaborative filtering. *Proceedings of the 13th ACM Conference on Recommender Systems*.

Fang, M.; Cao, X.; Jia, J.; et al. 2019. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX Security Symposium*.

Fang, M.; Zhang, Z.; Hairi; Khanduri, P.; Liu, J.; Lu, S.; Liu, Y.; and Gong, N. 2024. Byzantine-robust decentralized federated learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2874–2888.

Harper, F. M.; and Konstan, J. A. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*

He, X.; Liao, L.; Zhang, H.; et al. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*.

Luo, S.; Xiao, Y.; and Song, L. 2022. Personalized Federated Recommendation via Joint Representation Learning, User Clustering, and Model Adaptation. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*.

Ma, H.; Yang, H.; Lyu, M. R.; et al. 2008. SoRec: social recommendation using probabilistic matrix factorization. In *International Conference on Information and Knowledge Management*.

McMahan, H. B.; Moore, E.; Ramage, D.; et al. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; et al. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *ArXiv*.

Rodr'iguez-Barroso, N.; L'opez, D. J.; Luz'on, M. V.; et al. 2022. Survey on Federated Learning Threats: concepts, taxonomy on attacks and defences, experimental study and challenges. *ArXiv*.

Rong, D.; Ye, S.; Zhao, R.; et al. 2022. FedRecAttack: Model Poisoning Attack to Federated Recommendation. *2022 IEEE 38th International Conference on Data Engineering (ICDE)*.

Sun, Z.; Xu, Y.; Liu, Y.; He, W.; Kong, L.; Wu, F.; Jiang, Y.; and Cui, L. 2024. A survey on federated recommendation systems. *IEEE Transactions on Neural Networks and Learning Systems*.

Suresh, A. T.; McMahan, B.; Kairouz, P.; et al. 2019. Can You Really Backdoor Federated Learning. *arXiv: Learning*.

Tolpegin, V.; Truex, S.; Gursoy, M. E.; et al. 2020. Data Poisoning Attacks Against Federated Learning Systems. *Cornell University - arXiv*.

Wang, H.; Zhang, F.; Xie, X.; et al. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. *Proceedings of the 2018 World Wide Web Conference*.

Wu, C.; Wu, F.; An, M.; et al. 2019. NPA: Neural News Recommendation with Personalized Attention. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Wu, C.; Wu, F.; Lyu, L.; et al. 2021. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*.

Wu, C.; Wu, F.; Qi, T.; et al. 2022. FedAttack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Xie, C.; Koyejo, O.; and Gupta, I. 2020. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, 261–270. PMLR.

Xu, J.; Huang, S.-L.; Song, L.; et al. 2021. Byzantine-robust Federated Learning through Collaborative Malicious Gradient Filtering. *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*.

Yin, D.; Chen, Y.; Ramchandran, K.; et al. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *arXiv: Learning*.

Ying, R.; He, R.; Chen, K.; et al. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Yu, Y.; Liu, Q.; Wu, L.; et al. 2023. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yuan, W.; Nguyen, Q. V. H.; He, T.; et al. 2023. Manipulating Federated Recommender Systems: Poisoning with Synthetic Users and Its Countermeasures. In *SIGIR*.