PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion

Shijie Zhang
The University of Queensland
shijie.zhang@uq.edu.au

Hongzhi Yin*
The University of Queensland
h.yin1@uq.edu.au

Tong Chen
The University of Queensland
tong.chen@uq.edu.au

Zi Huang
The University of Queensland
huang@itee@uq.edu.au

Quoc Viet Hung Nguyen
Griffith University
quocviethung.nguyen@griffith.edu.au

Lizhen Cui
Shandong University
clz@sdu.edu.cn

ABSTRACT

Due to the growing privacy concerns, decentralization emerges rapidly in personalized services, especially recommendation. Also, recent studies have shown that centralized models are vulnerable to poisoning attacks, compromising their integrity. In the context of recommender systems, a typical goal of such poisoning attacks is to promote the adversary's target items by interfering with the training dataset and/or process. Hence, a common practice is to subsume recommender systems under the decentralized federated learning paradigm, which enables all user devices to collaboratively learn a global recommender while retaining all the sensitive data locally. Without exposing the full knowledge of the recommender and entire dataset to end-users, such federated recommendation is widely regarded 'safe' towards poisoning attacks. In this paper, we present a systematic approach to backdooring federated recommender systems for targeted item promotion. The core tactic is to take advantage of the inherent popularity bias that commonly exists in data-driven recommenders. As popular items are more likely to appear in the recommendation list, our innovatively designed attack model enables the target item to have the characteristics of popular items in the embedding space. Then, by uploading carefully crafted gradients via a small number of malicious users during the model update, we can effectively increase the exposure rate of a target (unpopular) item in the resulted federated recommender. Evaluations on two real-world datasets show that 1) our attack model significantly boosts the exposure rate of the target item in a stealthy way, without harming the accuracy of the poisoned recommender; and 2) existing defenses are not effective enough, highlighting the need for new defenses against our local model poisoning attacks to federated recommender systems.

CCS CONCEPTS

• **Information systems → Collaborative filtering**.

*Corresponding author.

KEYWORDS

Federated Recommender System; Poisoning Attacks; Deep Learning

ACM Reference Format:
Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. 2021. PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

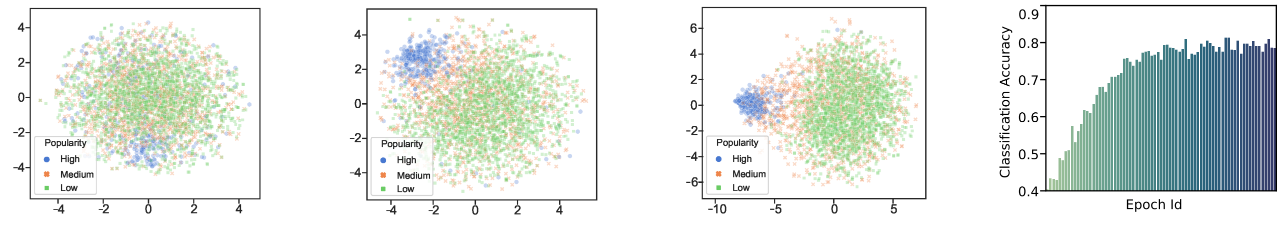
1 INTRODUCTION

The demand for recommender systems has increased more than ever before. Over the years, various recommendation algorithms have been proposed and proven successful in various applications. Collaborative filtering (CF), which infers users' potential interests from users' historical behavior data, lies at the core of modern recommender systems. Recently, enhancing CF with deep neural networks has been a widely adopted means for modelling the complex user-item interactions [20] and demonstrated state-of-the-art performance in a wide range of recommendation tasks.

The conventional recommender systems centrally store users' personal data to facilitate the centralized model training, which are, however, increasing the privacy risks [40]. Apart from privacy issues, another major risk emerges w.r.t. the correctness of the learned recommender in the presence of malicious users, where the trained model can be induced to deliver altered results as the adversary desires, e.g., promoting a target product[1] such that it gets more exposures in the item lists recommended to users. This is termed ***poisoning attack*** [12, 25], which is usually driven by financial incentives but incurs strong unfairness in a trained recommender. In light of both privacy and security issues, there has been a recent surge in decentralizing recommender systems, where federated learning [28, 33, 34] appears to be one of the most representative solutions. Specifically, a federated recommender allows users' personal devices to locally host their data for training, and the global recommendation model is trained in a multi-round fashion by submitting a subset of locally updated on-device models to the central server and performing aggregation (e.g., model averaging).

Subsuming a recommender under the federated learning paradigm naturally protects user privacy as the data is no longer uploaded to a central server or cloud. Moreover, it also prevents a recommender from being poisoned. The rationale is, in recommendation, the predominant poisoning method for manipulating item

[1]Product demotion works analogously, hence we mainly focus on promoting an unpopular product in this paper.



(a) 5th Epoch (b) 20th Epoch (c) 40th Epoch (d) Popularity Classification Accuracy (F1)

Figure 1: Visualization on the popularity bias of a CF-based federated recommender. (a), (b) and (c) are the t-SNE projection of item embeddings at different training stages, where more details on popularity labels can be found in Section 4.1. (d) shows the convergence curve of a popularity classifier using $F1$ score.

promotion is data poisoning, where the adversary manipulates malicious users to generate fake yet plausible user-item interactions (e.g., ratings), making the trained recommender biased towards the target item [12, 13, 25, 36]. However, these aforementioned data poisoning attacks operate under the assumption that the adversary has prior knowledge about the entire training dataset. Apparently, such assumption is voided in federated recommendation as the user data is distributively held, thus restricting the adversary's access to the entire dataset. Though the adversary can still conduct data poisoning by increasing the amount of fake interactions and/or malicious user accounts, it inevitably makes the attack more identifiable from either the abnormal user behavioral footprints [23, 38] or heavily impaired recommendation accuracy [25].

Thus, we aim to answer this challenging question: *can we backdoor federated recommender systems via poisoning attacks?* In this paper, we provide a positive answer to this question with a novel attack model that manipulates the target item's exposure rate in this non-trivial decentralized setting. Meanwhile, our study may also shed some light on the security risks of federated recommenders. Unlike data poisoning, our attack approach is built upon the model poisoning scheme, which compromises the integrity of the model learning process by manipulating the gradients [4, 5] of local models submitted by several malicious users. Given a federated recommender, we assume an adversary controls several malicious users/devices, each of which has the authority to alter the local model gradients used for updating the global model. Since the uploaded parameters directly affect the global model, it is more cost-effective for an adversary to poison the federated recommender from the model level, where a small group of malicious users will suffice.

Despite the efficacy of model poisoning in many federated learning applications, the majority of these attacks are only focused on perturbing the results in classification tasks (e.g., image classification and word prediction) [3, 11]. However, designed for personalized ranking tasks, federated recommenders are optimized towards completely different learning objectives, leaving poisoning attacks for item promotion largely unexplored. In the meantime, the federated environment significantly limits our adversary's prior knowledge to only partial local resources (i.e., malicious users' data and models) and the global model. Moreover, the poisoned global model should maintain high recommendation accuracy, so as to promote the target item stealthily while providing high-quality recommendations to benign users.

To address these challenges, we take advantage of a common characteristics of recommendation models, namely the popularity bias. As pointed out by previous studies [1, 39, 41, 42], data-driven recommenders, especially CF-based methods are prone to amplify the popularity by over-recommending popular items that are frequently visited. Intuitively, because such inherent bias still exists in federated recommenders, if we can disguise our target item as the popular items by uploading carefully crafted local gradients via malicious users, we can effectively trick the federated recommender to become biased towards our target item, thus boosting its exposure. Specifically, we leverage the learnable item embeddings as an interface to facilitate model poisoning. To provide a proof-of-concept, in Figure 1, we use t-SNE to visualize the item embeddings learned by a generic federated recommender (see Section 3.1 for model configuration). As the model progresses from its initial state towards final convergence (Figure 1(a)-(c)), items from three different popularity groups gradually forms distinct clusters. To further reflect the strong popularity bias, we train a simplistic popularity classifier (see Section 3.4) that predicts an item's popularity group given its learned embeddings, and show its testing performance in Figure 1(d). In short, the high prediction accuracy ($F1 > 0.8$) again verifies that the item embeddings in a well-trained federated recommender are highly discriminative w.r.t. their popularity.

To this end, we propose **p**oisoning **attack** for **i**tem **p**romotion (PipAttack), a novel poisoning attack model targeted on recommender systems in the decentralized, federated setting. Apart from optimizing PipAttack under the *explicit promotion constraint* that straightforwardly pushes the recommender to raise the ranking score of the target item, we innovatively design two learning objectives, namely the *popularity obfuscation constraint* and *distance constraint* so as to achieve our attack goal via fewer model updates without imposing dramatic accuracy drops. On one hand, popularity obfuscation confuses the federated recommender by aligning the target item with popular ones in the embedding space, thus greatly benefiting the target item's exposure rate via the popularity bias. On the other hand, to avoid harming the usability of the federated recommender and being detected, our attack model bears a distance constraint to prevent the manipulated gradients uploaded by malicious users from deviating too far from the original ones. We summarize our main contributions as follows:

- To the best of our knowledge, we present the first systematic approach to poisoning federated recommender systems, where an adversary only has limited prior knowledge compared with existing centralized scenarios. Our study reveals the existence of recommenders' security backdoors even in a decentralized environment.

- We propose PipAttack, a novel attack model that induces the federated recommender to promote the target item by uploading carefully crafted local gradients through several malicious users. PipAttack assumes no access to the full training dataset and other benign users' local information, and innovatively takes advantage of the inherent popularity bias to effectively boost the exposure rate of the target item.
- Experiments on two real-world datasets demonstrate the advantageous performance of PipAttack even when defensive strategies are in place. Furthermore, compared with all baselines, PipAttack is more cost-effective as it can reach the attack goal with fewer model updates and malicious users.

2 PRELIMINARIES

In this section, we first revisit the fundamental settings of federated recommendation and then formally define our research problem.

2.1 Federated Recommender Systems

Let $\mathcal{V}$ and $\mathcal{U}$ denote the sets of N items and M users/devices, respectively. Each user $u_i \in \mathcal{U}$ owns a local training dataset $\mathcal{D}_i$ consisting of implicit feedback tuples (u_i, v_j, r_{ij}), where $r_{ij} = 1$ if u_i has visited item $v_j \in \mathcal{V}$ (i.e., a positive instance), and $r_{ij} = 0$ if there is no interaction between them (i.e., a negative instance). Note that the negative instances are downsampled using a positive-to-negative ratio of $1 : q$ for each u_i due to the large amount of unobserved user-item interactions. Then, for every user, a federated recommender (FedRec) is trained to estimate the feedback $\hat{r}_{ij} \in [0, 1]$ between u_i and all items, where $\hat{r}_{ij}$ is also interpreted as the ranking/similarity score for a user-item pair. With the ranking score computed, FedRec then recommends an item list for each user u_i by selecting K top-ranked items w.r.t. $\hat{r}_{ij}$. It can be represented as:

$$FedRec(u_i|\Theta) = \{v_j | \hat{r}_{ij} \text{ is top-}K \text{ in } \{\hat{r}_{ij'}\}_{j' \in \bar{\mathcal{I}}(i)}\}, \quad (1)$$

where $\bar{\mathcal{I}}(i)$ denotes unrated items of u_i and Θ denotes all the trainable parameters in FedRec. For notation simplicity, we directly use Θ to represent the recommendation model.

Federated Learning Protocol. In FedRec, a central server coordinates individual user devices, of which each keeps $\mathcal{D}_i$ and a copy of the recommendation model locally. To train FedRec, the local model on the i-th user device is optimized locally by minimizing the following loss function:

$$\mathcal{L}_i^{rec} = -\sum_{(u_i,v_j,r_{ij}) \in \mathcal{D}_i} r_{ij} \log \hat{r}_{ij} + (1 - r_{ij}) \log(1 - \hat{r}_{ij}), \quad (2)$$

where we treat the estimated $\hat{r}_{ij} \in [0, 1]$ as the probability of observing the interaction between u_i and v_j. Then, the above cross-entropy loss quantifies the difference between $\hat{r}_{ij}$ and the binary ground truth r_{ij}. It is worth mentioning that, other popular distance-based loss functions (e.g., hinge loss and Bayesian personalized ranking loss [31]) are also applicable and behave similarly in FedRec. With the user-specific loss $\mathcal{L}_i^{rec}$ computed, we can derive the gradients of u_i's local model Θ_i, denoted by $\nabla\Theta_i$. At iteration t, a subset of users $\mathcal{U}_t$ are randomly drawn. Each $u_i \in \mathcal{U}_t$ then downloads the latest global model Θ_t and updates its local gradients w.r.t. $\mathcal{D}_i$, denoted by $\nabla\Theta_i^t$. After the central server receives all local gradients submitted by $|\mathcal{U}_t|$ users, it aggregates the collected gradients to facilitate global model update. Specifically, FedRec follows the commonly used gradient averaging [27] to obtain the updated model

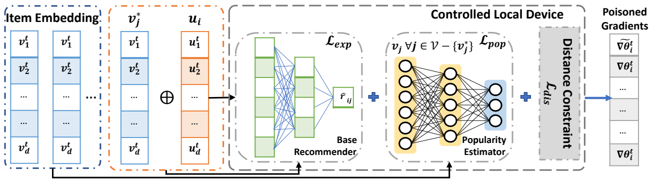


Figure 2: Overview of PipAttack

Θ_{t+1} with learning rate η:

$$\Theta_{t+1} = \Theta_t - \eta \frac{1}{|\mathcal{U}^t|} \sum_{u_i \in \mathcal{U}_t} \nabla\Theta_i^t. \quad (3)$$

The training proceeds iteratively until convergence. Unlike centralized recommenders, FedRec collects only each user's local model gradients instead of her/his own data $\mathcal{D}_i$, making existing poisoning attacks [9, 25, 36] fail due to their ill-posed assumptions on the access to the entire dataset. Furthermore, different local gradients are not shared across users, which further reduces the amount of prior knowledge that a malicious party can acquire.

2.2 Poisoning Attacks

Following [11], we assume an adversary can compromise a small proportion of users in FedRec, which we term malicious users. The attack is scheduled to start at epoch t, and all malicious users participate in the training of FedRec normally before that. From epoch t, the malicious users start poisoning the global model by replacing the real local gradients $\nabla\Theta_i^t$ with purposefully crafted gradients $\widetilde{\nabla\Theta_i^t}$, so as to gradually guide the global model to recommend the target item more frequently. The task is formally defined as follows:

Problem 1. **Poisoning FedRec for Item Promotion.** Given a federated recommender parameterized by Θ, our adversary aims to promote a target item $v_j^* \in \mathcal{V}$ by altering the local gradients submitted by every compromised malicious user device $u_i^* \in \mathcal{U}^*$ at each update iteration t, i.e., $f : \nabla\Theta_i^t \mapsto \widetilde{\nabla\Theta_i^t}$. Note that each malicious user will produce its unique deceptive gradients $\widetilde{\nabla\Theta_i^t}$. Hence, the deceptive gradients $\{\widetilde{\nabla\Theta_i^t}\}_{\forall u_i \in \mathcal{U}^*}$ for all malicious users are parameters to be learned during the attack process, such that for each benign user $u_i \in \mathcal{U} \backslash \mathcal{U}^*$, the probability that v_j^* appears in $FedRec(u_i|\Theta)$ is maximized.

3 POISONING FEDREC FOR ITEM PROMOTION

In this section, we introduce our proposed PipAttack in details.

3.1 Base Recommender

Federated learning is compatible with the majority of latent factor models. Without loss of generality, we adopt neural collaborative filtering (NCF) [20], a performant and widely adopted latent factor model, as the base recommender of FedRec. In short, NCF extends CF by leveraging an L-layer feedforward network $FFN(\cdot)$ to model the complex user-item interactions and estimate $\hat{r}_{ij}$:

$$\hat{r}_{ij} = \sigma(\mathbf{h}^\top FFN(\mathbf{u}_i \oplus \mathbf{v}_j)), \quad (4)$$

where $\mathbf{u}_i, \mathbf{v}_j \in \mathbb{R}^d$ are respectively user and item embeddings, $\oplus$ is the vector concatenation, $\mathbf{h} \in \mathbb{R}^{d_L}$ denotes the projection weights that corresponds to the d_L-dimensional output of the last network layer, and σ is the sigmoid function that rectifies the output to fall between 0 and 1. Meanwhile, it is worth noting that our attack method is generic and applicable to many other recommenders like feature-based [32] and graph-based [19] ones.

3.2 Prior Knowledge of PipAttack

In summary, the parameters Θ to be learned in FedRec are: the projection vector $\mathbf{h}$, all weights and biases in the L-layer FFN, as well as embeddings $\mathbf{u}_i$ and $\mathbf{v}_j$ for all users and items. Meanwhile, an important note is that, as user embeddings are regarded highly sensitive as they directly reflect users' personal interests, they are commonly disallowed to be shared across users in federated recommenders [8, 30] for privacy reasons. In FedRec, this is achieved by withholding $\mathbf{u}_i$ and its gradients on every user device, and the user embeddings are updated locally. Hence, the gradients of $\mathbf{u}_i$ are excluded from both $\nabla\Theta$ and $\widetilde{\nabla\Theta}$ that will be respectively submitted by benign and malicious users during the update of FedRec.

To perform poisoning attacks on FedRec, traditional attack approaches designed for centralized recommenders are inapplicable. This is due mainly to the prior knowledge that needs to be accessible to the adversary, e.g., all user-item interactions [13, 36] and even other benign users' embeddings [12, 25], which becomes an ill-posed assumption for federated settings as most of such information is retained at the user side. In short, FedRec substantially narrows down the prior knowledge and capability of an attack model, which is restricted to the following:

I. The adversary can access the global model Θ_t at any iteration t, excluding all benign users' embeddings, i.e., $\{\mathbf{u}_i\}_{\forall u_i \in \mathcal{U}\backslash\mathcal{U}^*}$.
II. The adversary can access and alter all malicious users' local models and their gradients.
III. The adversary knows the whole item set (not interactions) which is commonly available on any e-commerce platform, as well as side information that reflects each item's popularity. We will expand on this in Section 3.4.

In what follows, we present the key components of PipAttack as shown in Fig 2 for learning all deceptive gradients $\{\widetilde{\nabla\Theta}_i^t\}_{\forall u_i \in \mathcal{U}^*}$, which induces FedRec to promote the target item v_j^*.

3.3 Explicit Promotion

Like many studies on poisoning attacks on recommender systems, our adversary's goal is to manipulate the learned global model such that the generated recommendation results meet the adversary's demand. Essentially, to give the target item v_j^* more exposures (i.e., to be recommended to more users), we need to raise the ranking score $\hat{r}_{ij}$ whenever a user u_i is paired with v_j^*. As a minimum requirement, we need to ensure all malicious users can receive v_j^* in their top-K recommendations. With the adversary's control over a group of malicious users $\mathcal{U}^*$, we can explicitly boost the ranking score of v_j^* for every $u_i^* \in \mathcal{U}^*$ via the following objective function:

$$\mathcal{L}_{exp} = -\sum_{u_i^* \in \mathcal{U}^*, v_j^*} \log \hat{r}_{ij}, \tag{5}$$

which encourages a large similarity score between every malicious user and the target item. Theoretically, this mimics the effect of inserting fake interaction records (i.e., data poisoning) with the target item via malicious users, which can gradually drive the CF-based global recommender to predict positive feedback on v_j^* for other benign users who are similar to $u_i^* \in \mathcal{U}^*$.

3.4 Popularity Obfuscation

Unfortunately, Eq.(5) requires the adversary to manipulate a relatively large number of malicious users in order to successfully and efficiently poison FedRec. Otherwise, after aggregating the local gradients from sampled users, the large user base of a recommender system can easily dilute the impact of deceptive gradients uploaded by a small group of $\mathcal{U}^*$. In this regard, on top of explicit promotion, we propose a more cost-effective tactic in PipAttack from the popularity perspective. As discussed in Section 1, it is well acknowledged that CF-based recommenders are intrinsically biased towards popular items that are frequently visited [2], and FedRec is no exception. Compared with long-tail/unpopular items, popular items are more widely and frequently trained in a recommender, thus amplifying their likelihood of receiving a larger ranking score and gaining advantages in exposure rates.

Hence, we make full use of the inherent popularity bias rooted in recommender systems, where we aim to learn deceptive gradients that can trick FedRec to 'mistake' v_j^* as a popular item. In a latent factor model like FedRec, this can be achieved by poisoning the item embeddings in the global recommender, so that embedding $\mathbf{v}_j^*$ is semantically similar to popular items in $\mathcal{V}$. Intuitively, if a popular item and v_j^* are close to each other in the latent space, so will their ranking scores for the same user. Given the existence of popularity bias, $\mathbf{v}_j^*$ will be promoted to more users' recommendation lists.

Availability of Popularity Information. Our popularity obfuscation strategy needs prior knowledge about items' popularity. However, directly obtaining such information via user-item interaction frequencies is infeasible as it requires access to the entire dataset. Fortunately, though the user behavioral data is protected, the prosperity of online service platforms has brought a wide range of visible clues on the popularity of items. For example, hot tracks are always displayed on music applications (e.g., Spotify) without revealing identities of their listeners, and e-commerce sites (e.g., eBay) records the sales volume of each product while keeping the purchasers anonymized. Furthermore, as PipAttack is aware of the item set, item popularity can be easily looked up on a fully public platform (e.g., Yelp). Thus, popularity information can be fetched from various public sources to assist our poisoning attack, and the prerequisites of FedRec remain intact.

Popularity Estimator. The most straightforward way to facilitate such popularity obfuscation is to incorporate a distance metric to penalizes any $\widetilde{\nabla\Theta}_i$ that enlarges the distance between v_j^* and a randomly sampled popular item $v_{j'}$. However, being a personalized model, whether an item can be recommended to u_i is not only determined by its popularity, but also the user-item similarity. Therefore, such constraints will work the best only if the selected $v_{j'}$ accounts for each user's personal preference, which is infeasible due to the fact that all benign users within $\mathcal{U}\backslash\mathcal{U}^*$ are intransparent to the adversary. In PipAttack, we propose a novel popularity estimator-based method that collectively engages all the popular items in the item set. Specifically, we can assign each item a discrete label w.r.t. its popularity level obtained via public information. Then, suppose there are C popularity classes, the popularity estimator $f_{est}(\cdot)$ is a deep neural network (DNN) that inputs a learned item embedding $\mathbf{v}$ at iteration step t, and computes a C-dimensional probability distribution vector $\hat{\mathbf{y}}$ via the final softmax layer. Each element $\hat{\mathbf{y}}[c] \in \hat{\mathbf{y}}$ ($c = 1, 2, ..., C$) represents the probability that v_i belongs to class c. We train $f_{est}(\cdot)$ with cross-entropy loss on all $(\mathbf{v}, \mathbf{y})$ pairs for all $v \neq v_j^*$, where $\mathbf{y} = \{0, 1\}^C$ is the one-hot label:

$$\mathcal{L}_{est} = -\sum_{\forall v \neq v_j^*} \sum_{c=1}^{C} \mathbf{y}[c] \log \hat{\mathbf{y}}[c]. \quad (6)$$

Boosting Item Popularity. Once we obtain a well-trained $f_{est}(\cdot)$, it is highly reflective of the popularity characteristics encoded in each item embedding, where items at the same popularity level will have high semantic similarity in the embedding space. So, at the t-th training iteration of FedRec, we aim to boost the predicted popularity of v_j^* by making targeted updates on embedding $\mathbf{v}_j^* \in \Theta^{t+1}$ with crafted deceptive gradients $\widetilde{\nabla\Theta}_i^t$. This is achieved by minimizing the negative log-likelihood of class c_{top} (i.e., the highest popularity level) in the output of $f_{est}(\mathbf{v}_j^*)$:

$$\mathcal{L}_{pop} = -\log f_{est}(\mathbf{v}_j^*)[c_{top}], \ \mathbf{v}_j^* \in \Theta_{t+1}. \quad (7)$$

Note that $f_{est}(\cdot)$ stays fixed and will no longer be updated after the popularity obfuscation starts. Essentially, by optimizing $\mathcal{L}_{pop}$, PipAttack generates gradients $\widetilde{\nabla\Theta}_i^t$ that enforces v_j^* to approximate the characteristics of all items from the top popularity group in the embedding space, resulting in a boosted exposure rate.

3.5 Distance Constraint

Generally, aggressively fabricated deceptive gradients $\widetilde{\nabla\Theta}_i^t$ may help the adversary achieve the attack goal with fewer training iterations, but will also incur strong discrepancies with the real gradients $\nabla\Theta_i^t$, leading to a higher chance to be detected by the central server [5] and harm the performance of the global model. Hence, the crafted gradients should maintain a certain level of similarity with the genuine ones. As such, we place a constraint on the distance between each $\widetilde{\nabla\Theta}_i^t$ for $u_j^* \in \mathcal{U}^*$ and all malicious users' original local gradient $\nabla\Theta_i^t$:

$$\mathcal{L}_{dis} = \sum_{u_i \in \mathcal{U}^*} ||\widetilde{\nabla\Theta}_i^t - \frac{1}{|\mathcal{U}^*|} \sum_{u_i \in \mathcal{U}^*} \nabla\Theta_i^t||_p, \quad (8)$$

where $||\cdot||_p$ is p-norm distance and we adopt $||\cdot||_2$ in PipAttack.

3.6 Optimizing PipAttack

In this subsection, we define the loss function of PipAttack for model training. Instead of training each component separately, we combine their losses and use joint learning to optimize the following objective function:

$$\mathcal{L} = \mathcal{L}_{exp} + \alpha\mathcal{L}_{pop} + \gamma\mathcal{L}_{dis}, \quad (9)$$

where α and γ are non-negative coefficients to scale and balance the effect of each part.

4 EXPERIMENTS

In this section, we first outline the evaluation protocols for our PipAttacks and then conduct experiments on two real-world datasets to evaluate the performance of PipAttack. Particularly, we aim to answer the following research questions (RQs) via experiments:

RQ1: Can PipAttack perform poisoning attacks effectively on federated recommender systems?

RQ2: Does PipAttack harm the recommendation performance of the federated recommender significantly?

RQ3: How does PipAttack benefit from each key component?

RQ4: What is the impact of hyperparameters to PipAttack?

RQ5: Can PipAttack bypass defensive strategies deployed at the server side?

4.1 Experimental Datasets

We adopt two popular public datasets for evaluation, namely MovieLens-1M (ML) [16] and Amazon (AZ) [17]. ML contains 1 million ratings involving 6,039 users and 3,705 movies, while AZ contains 103,593 ratings involving 13,174 users and 5,970 cellphone-related products. Following [18, 20, 21], we have binarized the user feedback, where all ratings are converted to $r_{ij} = 1$, and negative instances are sampled $q = 4$ times the amount of positive ones. PipAttack needs to obtain the popularity labels of items. As described in Section 3.4, such information can be easily obtained in real-life scenarios (e.g., page views) without sacrificing user identity in FedRec. However, as our datasets are mainly collected for pure recommendation research, there is no such side information available. Hence, we segment items' popularity into three levels by sorting all items by the amount of interactions they receive, with intervals of the top 10% (high popularity), top 10% to 45% (medium popularity), and the last 55% (low popularity). Note that we only use the full dataset once to compensate for the unavailable side information about item popularity, and the interaction records of each user are locally stored throughout the training of FedRec and poisoning attack.

4.2 Evaluation Protocols

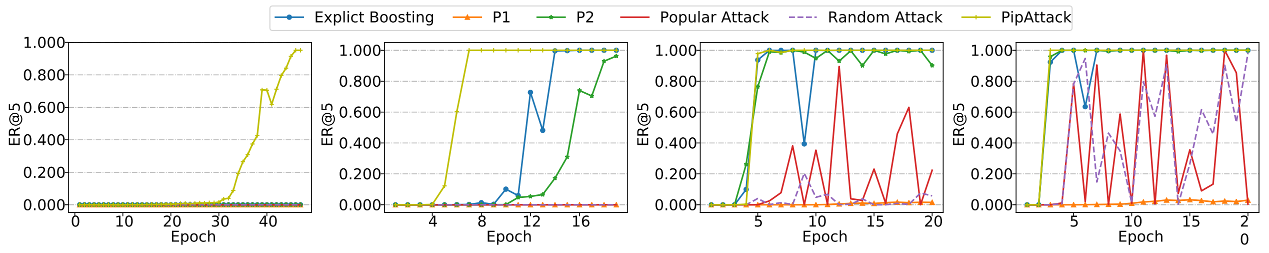
Following the common setting for attacking federated models [5], we first train FedRec without attacks (malicious users behave normally in that period) for several epochs, then PipAttack starts poisoning FedRec by activating all malicious users. To ensure fairness in evaluation, all tested methods are asked to attack the same pre-trained FedRec model. We introduce our evaluation metrics below.

Poisoning Attack Effectiveness. We use exposure rate at rank K (*ER@K*) as our evaluation metric. Suppose each user receives K recommended items, then for the target item to be promoted, *ER@K* is the fraction of users whose K recommended items include the target item. Correspondingly, larger *ER@K* represents stronger poisoning effectiveness. Notably, from both datasets, we select the least popular item as the target item in our experiments as this is the hardest possible item to promote. For each user, all items excluding positive training examples are used for ranking.

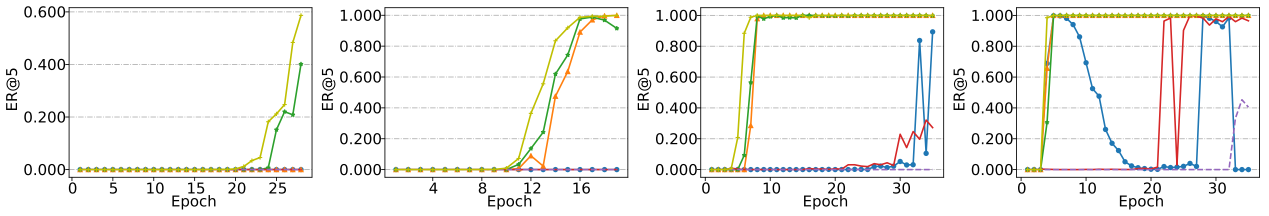
Recommendation Effectiveness. It is important that the poisoning attack does not harm the accuracy of FedRec. To evaluate the recommendation accuracy, we adopt the leave-one-out approach [21] to hold out ground truth items for evaluation. An item is held out for each user to build a validation set. Following [20], we employ hit ratio at rank K (*HR@K*) to quantify the fraction of observing the ground truth item in the top-K recommendation lists.

4.3 Baselines

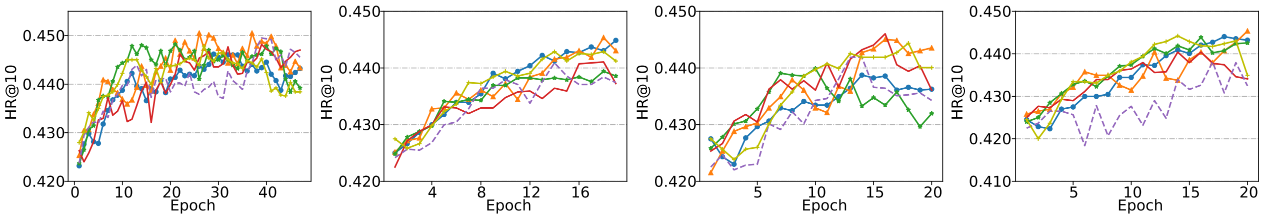
We compare PipAttack with five poisoning attack methods, where the fist three are model poisoning and the last two are data poisoning methods. Notably, many recent models are only designed to attack centralized recommenders, thus requiring prior knowledge



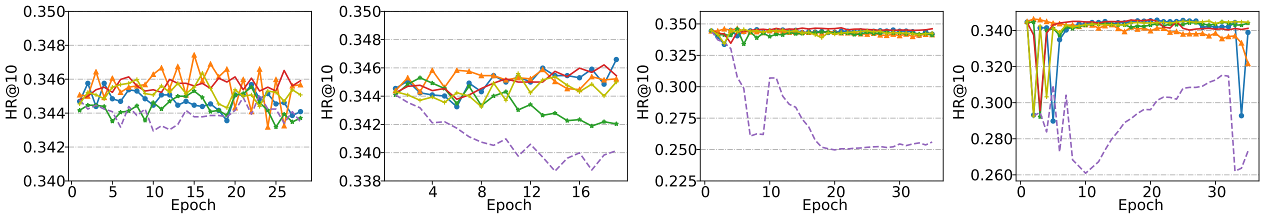
(a) ER@5 with ζ = 10% on ML (b) ER@5 with ζ = 20% on ML (c) ER@5 with ζ = 30% on ML (d) ER@5 with ζ = 40% on ML



(e) ER@5 with ζ = 5% on AZ (f) ER@5 with ζ = 10% on AZ (g) ER@5 with ζ = 20% on AZ (h) ER@5 with ζ = 30% on AZ



(i) HR@10 with ζ = 10% on ML (j) HR@10 with ζ = 20% on ML (k) HR@10 with ζ = 30% on ML (l) HR@10 with ζ = 40% on ML



(m) HR@10 with ζ = 5% on AZ (n) HR@10 with ζ = 10% on AZ (o) HR@10 with ζ = 20% on AZ (p) HR@10 with ζ = 30% on AZ

Figure 3: Attack ((a)-(h)) and recommendation ((i)-(p)) results on ML and AZ. Note that the curves start from the first epoch after the attack starts.

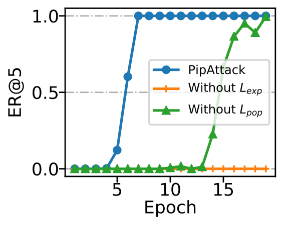
that cannot be obtained in the federated setting. Hence, we choose the following baselines that do not hold assumptions on those inaccessible knowledge: **P1** [5]: This work aims to poison federated learning models by directing the model to misclassify the target input. **P2** [4]: This is a general approach for attacking distributed models and evading defense mechanisms. **Explicit Boosting (EB)**: The adversary directly optimizes the explicit item promotion objective $\mathcal{L}_{exp}$. **Popular Attacks (PA)** [15]: It injects fake interactions with both target and popular items via manipulated malicious users to promote the target item. **Random Attacks (RA)** [15]. It poisons a recommender in a similar way to PA, but uses fake interactions with the target item and randomly selected items.

4.4 Parameters Settings

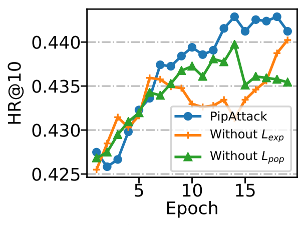
In FedRec, we set the latent dimension d, learning rate, local batch size to 64, 0.01 and 64, respectively. Each user is considered as an individual device, where 10% and 5% of the users (including both benign and malicious users) are randomly selected on ML and AZ at every iteration. Model parameters in FedRec are randomly initialized using Gaussian distribution ($\mu = 0, \sigma = 1$). The popularity estimator $f_{est}(\cdot)$ is formulated as a 4-layer deep neural network with 32, 16, 8, and 3 hidden dimensions from bottom to top. In Pipattack, the local epoch and learning rate are respectively 30 and 0.01 on both datasets. We also examine the effect of different fractions ζ of malicious users with $\zeta \in \{10\%, 20\%, 30\%, 40\%\}$ on ML and $\zeta \in \{5\%, 10\%, 20\%, 30\%\}$ on AZ. For the coefficients in loss function $\mathcal{L}$, we set $\alpha = 60$ when $\zeta = 10\%$ and $\alpha = 20$ when $\zeta \in \{20\%, 30\%, 40\%\}$ on ML, and $\alpha = 10$ on AZ across all malicious user fractions. We apply $\gamma = 0.0005$ on both datasets.

4.5 Attack Effectiveness (RQ1)

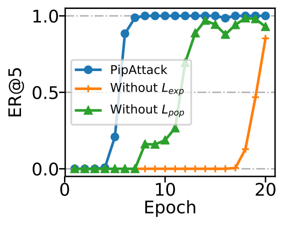
Figure 3 shows the performance of $ER@5$ w.r.t. to different malicious user fractions. It is clear that PipAttack is successful at inducing the recommender to recommend the target item. In fact, the recommender becomes highly confident in its recommendation results and recommend the target item to all the users after the 40th epoch, with just a small number of malicious users (i.e., 10% on ML and 5% on AZ). By increasing the fraction of fraudsters, the attack performance improves significantly, as it is easier for the adversary to manipulate the parameters of the global model



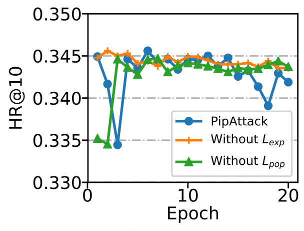
(a) ER@5 on ML



(b) HR@10 on ML

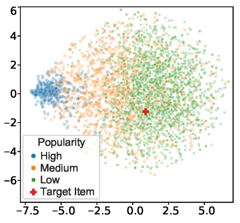


(c) ER@5 on AZ

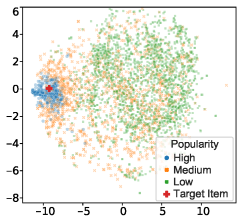


(d) HR@10 on AZ

Figure 4: Ablation test with different model architectures.

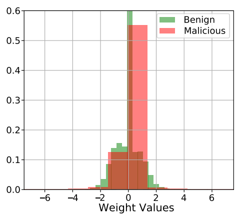


(a) Before Poisoning Attack

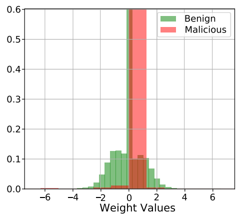


(b) After Poisoning Attack ($ER@5 = 1$)

Figure 5: Visualization of item embeddings before and after being attacked by PipAttack.



(a) With Distance Constraint



(b) Without Distance Constraint.

Figure 6: Comparison of gradient distributions between benign and malicious users.

via more malicious users. Additionally, PipAttack outperforms all baselines consistently in terms of $ER@5$, showing strong capability to promote the target items to all users. Furthermore, as the attack proceeds, our model can consistently reach 100% exposure rate in most scenarios, while there are obvious fluctuations in the performance of other baseline methods. Finally, we observe that model poisoning methods (i.e., P1, P2 and Explicit Boosting) generally perform better than data poisoning methods (i.e., Random and Popular attack), indicating the superiority of model poisoning paradigm since the global recommender can be manipulated arbitrarily for mean even with limited number of compromised users.

4.6 Recommendation Accuracy (RQ2)

Recommendation accuracy plays a significant role in the success of adversarial attacks. On one hand, it is an important property that the server can use to detect anomalous updates. On the other hand, only a recommender that is highly accurate will have a large user base for promoting the target item. Figure 3 shows the overall performance curve of FedRec after the attack starts. The first observation is that, PipAttack, which is the most powerful for item promotion, still achieves competitive recommendation results on both datasets. It proves that PipAttack can avoid harming the usefulness of FedRec. Another observation is that there is an overall upward trend for all baseline models except random attack. Finally, compared with model poisoning attacks, data poisoning attacks, especially random attack, cause a larger decrease on recommendation accuracy, which further confirms the effectiveness of model poisoning methods.

4.7 Effectiveness of Model Components (RQ3)

We conduct ablation analysis to better understand the performance gain from the major components proposed in PipAttack. We set $\zeta = 20\%$ throughout this set of experiments, and we discuss the key components below.

4.7.1 *Explicit Promotion Constraint.*

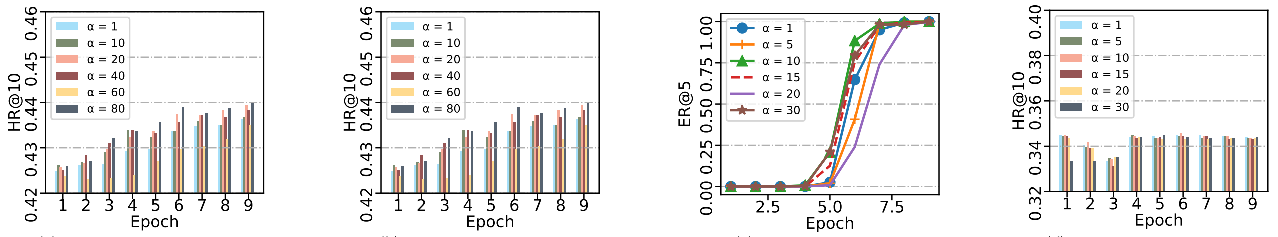
We first remove explicit promotion constraint $\mathcal{L}_{exp}$ and plot the new results in Figure 4. Apparently, it leads to severe performance drop on two datasets. As the main adversarial objective is to increase the ranking score for the target item, removing $\mathcal{L}_{exp}$ directly limits PipAttack's capability. In addition, the degraded PipAttack is more effective on the AZ than that on ML. A possible reason is that AZ is sparser and thus is easier to attack, despite the removal of explicit promotion constraint.

4.7.2 *Popularity Obfuscation Constraint.*

We validate our hypothesis of leveraging the popularity bias in FedRec for poisoning attack. In Figure 4, without the popularity obfuscation constraint $\mathcal{L}_{pop}$, PipAttack suffers from the obviously inferior performance on both two datasets. For instance, $ER@5 = 100\%$ is respectively achieved at 17th and 14th epochs on ML and AZ, which are far behind the full version of PipAttack. It confirms that by taking advantage of the popularity bias, PipAttack can effectively accomplish the attack goal with less attack attempts. To further verify the efficacy of popularity obfuscation in PipAttack, we visualize the item embeddings via t-SNE in Figure 5. Obviously, after the poisoning attack, the target item embedding moves from the cluster of unpopular items to the cluster of the most popular items. Hence, the use of popularity bias is proved beneficial to promoting the target item.

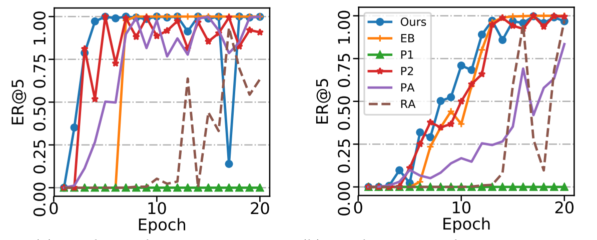
4.7.3 *Distance Constraint.*

As introduced in [5], the server can use weight update statistics to detect anomalous updates and even refuse the corresponding updates. To verify the contribution of the distance constraint in Eq.(9), we derive one variant of PipAttack by removing the distance constraint $\mathcal{L}_{dis}$. In Figure 6, on the ML dataset, the averaged gradients' distributions for all benign updates and malicious updates are plotted. It is clear that the deceptive gradients' distribution under the distance constraint is similar to that of benign users. In contrast, the gradients are much sparser and diverges far from benign users' gradients. Furthermore, we calculate the KL-divergence to measure the difference between two distributions in each figure. Specially, the result achieved with distance constraint (0.005) is much lower than the result achieved without it (0.177). Hence, PipAttack is stealthier, and is harder for the server to flag abnormal gradients.



(a) ER@5 on MovieLen (b) HR@10 on MovieLen (c) ER@5 on Amazon (d) HR@10 on Amazon

Figure 7: Parameters sensitivity.



(a) With Buylan strategy. (b) With Trimmed mean strategy.

Figure 8: Defense results.

4.8 Hyperparameter Analysis (RQ4)

We answer RQ4 by investigating the performance fluctuations of PipAttack with varied hyperparameters on both dataset. Specifically, we examine the values of α in $\{1, 10, 20, 40, 60, 80\}$ on ML and $\{1, 5, 10, 15, 20, 30\}$ on AZ, respectively ($\zeta = 20\%$). Note that we omit the effect of γ as it brings much less variation to the model performance. Figure 7 presents the results with varied α. As can be inferred from the figure, all the best attack results are achieved with $\alpha = 20$ on ML and $\alpha = 10$ on AZ. Meanwhile, altering this coefficient on the attack objective has less impact on the recommendation accuracy. Overall, setting $\alpha = 20$ on ML and $\alpha = 10$ on AZ is sufficient for promoting target item, while ensuring the high-quality recommendation of FedRec.

4.9 Handling Defenses (RQ5)

In general, the mean aggregation rule used in FedRec assumes that the central server works in a secure environment. To answer RQ5, we investigate how our different attack models perform in the presence of attack-resistant aggregation rules, namely Bulyan [14] and Trimmed Mean [35]. Note that though Krum [6] is also a widely used defensive method, it is not considered in our work due to the severe accuracy loss it causes. We benchmark all methods' performance on the ML dataset, and Figure 8 shows the *ER*@5 results with $\zeta = 20\%$. The first observation we can draw from the results is that, PipAttack consistently outperforms all baselines and maintains $ER@5 = 1$, which further confirms the superiority of PipAttack. The main goal of those resilient aggregation methods is to ensure convergence of the global model under attacks, while our model's distance constraint effectively ensures this. Our experiments reveal the limited robustness of existing defenses in federated recommendation, thus emphasising the need for more advanced defensive strategies on such poisoning attacks.

5 RELATED WORK

A growing number of online platforms are deploying centralized recommender systems to increase user interaction and enrich shopping potential [29, 37, 40]. However, many studies show that attacking recommender systems can affect users' decisions in order to have target products recommended more often than before [9, 22, 24]. In [10], a reinforcement learning (RL)-based model is designed to generate fake profiles by copying the benign users' profiles in the source domain, while [36] firstly constructed a simulator recommender based on the full user-item interaction network, and then the results of the simulator system is used to design rewards function for policy training. [9, 25] generate fake users through Generative Adversarial Networks (GAN) to achieve adversarial intend. [13] and [12] formulate the attack as an optimization problem by maximizing the hit ratio of the target item. However, many of these attack approaches fundamentally rely on the white-box model, in which the attacker requires the adversary to have full knowledge of the target model and dataset.

Another variant of federated recommenders are devised to inhibit the availability of dataset [26, 28, 30, 34]. For federated recommender systems, expecting complete access to the dataset and model is not realistic. [7, 15] study the influence of low-knowledge attack approaches to promote an item (e.g., random, popular attack), but the performance is unsatisfactory. Despite previous attack methods success with various adversarial objectives under federated setting such as misclassification [4, 5] and increased error rate [11], the study of promoting an item for federated recommender system still remains largely unexplored. Therefore, we propose a novel framework that does not have full knowledge of the target model to attack under the federated setting to fill this gap.

6 CONCLUSION

In this paper, we present a novel model poisoning attack framework for manipulating item promotion named PipAttack. We also prove that our attack model can work on the presence of defensive protocols. With three innovatively designed attack objectives, the attack model is able to perform efficient attacks against federated recommender systems and maintain high-quality recommendation results generated by the poisoned recommender. Furthermore, the distance constraint plays an essential role in disguising malicious users as benign users to avoid detection. The experimental results based on two real-world datasets demonstrate the superiority and practicality of PipAttack over peer methods even in the presence of defensive strategies deployed at the server side.

7 ACKNOWLEDGMENTS

This work is supported by Australian Research Council Future Fellowship (Grant No. FT210100624), Discovery Project (Grant No. DP190101985) and Discovery Early Career Research Award (Grant No. DE200101465).

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *RecSys*. 42–46.

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. In *RMSE Workshop*.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *AISTATS*. 2938–2948.

[4] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *NeurIPS*. 8632–8642.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *ICML*. 634–643.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*. 118–128.

[7] Robin Burke, Bamshad Mobasher, and Runa Bhaumik. 2005. Limited knowledge shilling attacks in collaborative filtering systems. In *ITWP 2005*. 17–24.

[8] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Secure federated matrix factorization. *IEEE Intelligent Systems* (2020).

[9] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *RecSys*. 322–330.

[10] Wenqi Fan, Tyler Derr, Xiangyu Zhao, Yao Ma, Hui Liu, Jianping Wang, Jiliang Tang, and Qing Li. 2021. Attacking Black-box Recommendations via Copying Cross-domain User Profiles. In *ICDE*. 1583–1594.

[11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX*. 1605–1622.

[12] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *WWW*. 3019–3025.

[13] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *ACSAC*. 381–392.

[14] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *ICML*. 3521–3530.

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

[16] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems* (2015), 1–19.

[17] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.

[18] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[21] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*. 549–558.

[22] Nguyen Quoc Viet Hung, Huynh Huu Viet, Nguyen Thanh Tam, Matthias Weidlich, Hongzhi Yin, and Xiaofang Zhou. 2018. Computing Crowd Consensus with Partial Agreement. *IEEE Transactions on Knowledge and Data Engineering* (2018), 1–14.

[23] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *WSDM*. 333–341.

[24] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *WWW*. 393–402.

[25] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking recommender systems with augmented user profiles. In *CIKM*. 855–864.

[26] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, and Xiuzhen Cheng. 2020. Meta Matrix Factorization for Federated Rating Predictions. In *SIGIR*. 981–990.

[27] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[28] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. Fedfast: Going beyond average for faster training of federated recommender systems. In *SIGKDD*. 1234–1242.

[29] Quoc Viet Nguyen, Chi Thang Duong, Thanh Tam Nguyen, Matthias Weidlich, Karl Aberer, Hongzhi Yin, and Xiaofang Zhou. 2017. Argument Discovery via Crowdsourcing. *VLDBJ* (2017), 511–535.

[30] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-Preserving News Recommendation Model Learning. In *EMNLP*. 1423–1432.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[32] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.

[33] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2021. Fast-adapting and Privacy-preserving Federated Recommender System. *VLDBJ* (2021).

[34] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).

[35] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*. 5650–5659.

[36] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical data poisoning attack against next-item recommendation. In *WWW*. 2458–2464.

[37] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. 2021. Graph Embedding for Recommendation against Attribute Inference Attacks. In *WWW*. 3002–3014.

[38] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *SIGIR*. 689–698.

[39] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR*. 11–20.

[40] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *WSDM*. 717–726.

[41] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-Opportunity Bias in Collaborative Filtering. In *WSDM*. 85–93.

[42] Ziwei Zhu, Jianling Wang, and James Caverlee. 2020. Measuring and mitigating item under-recommendation bias in personalized ranking systems. In *SIGIR*. 449–458.