Untargeted Attack against Federated Recommendation Systems via Poisonous Item Embeddings and the Defense

Yang Yu[1,2], **Qi Liu**[1,2*], **Likang Wu**[1,2], **Runlong Yu**[1,2], **Sanshi Lei Yu**[1,2], **Zaixi Zhang**[1,2]

[1]Anhui Province Key Laboratory of Big Data Analysis and Application,
School of Computer Science and Technology, University of Science and Technology of China
[2]State Key Laboratory of Cognitive Intelligence
{yflyl613, wulk, yrunl, zaixi}@mail.ustc.edu.cn, qiliuql@ustc.edu.cn, meet.leiyu@gmail.com

Abstract

Federated recommendation (FedRec) can train personalized recommenders without collecting user data, but the decentralized nature makes it susceptible to poisoning attacks. Most previous studies focus on the targeted attack to promote certain items, while the untargeted attack that aims to degrade the overall performance of the FedRec system remains less explored. In fact, untargeted attacks can disrupt the user experience and bring severe financial loss to the service provider. However, existing untargeted attack methods are either inapplicable or ineffective against FedRec systems. In this paper, we delve into the untargeted attack and its defense for FedRec systems. (i) We propose *ClusterAttack*, a novel untargeted attack method. It uploads poisonous gradients that converge the item embeddings into several dense clusters, which make the recommender generate similar scores for these items in the same cluster and perturb the ranking order. (ii) We propose a **unifo**rmity-based defe**n**se mechanism (*UNION*) to protect FedRec systems from such attacks. We design a contrastive learning task that regularizes the item embeddings toward a uniform distribution. Then the server filters out these malicious gradients by estimating the uniformity of updated item embeddings. Experiments on two public datasets show that *ClusterAttack* can effectively degrade the performance of FedRec systems while circumventing many defense methods, and *UNION* can improve the resistance of the system against various untargeted attacks, including our *ClusterAttack*.

1 Introduction

Over recent years, personalized recommender systems have been widely used to alleviate the information overload problem for users (Wu et al. 2019; Li et al. 2020; Jin et al. 2020). Most conventional recommenders are trained on centralized user data, which has the risk of data leakage and raises privacy concerns (Wu et al. 2018; Yang et al. 2020). Moreover, some privacy regulations, such as GDPR[1] and CCPA[2], make it more difficult to collect user data for centralized model training. Federated learning (FL) is a decentralized training paradigm that enables multiple clients to collaboratively learn a global model without sharing their local data (McMahan et al. 2017). Several studies have applied FL to train privacy-preserving federated recommendation (FedRec) systems (Lin et al. 2021; Liang, Pan, and Ming 2021).

Unfortunately, FL is known to be vulnerable to poisoning attacks (Bhagoji et al. 2019; Bagdasaryan et al. 2020; Cao, Jia, and Gong 2021). Its decentralized training procedure allows the attacker to arbitrarily modify the local training data or the uploaded gradient to achieve certain malicious goals. Based on the goal of the attacker, poisoning attacks can be classified into targeted and untargeted attacks. In the FedRec scenario, previous studies mainly focus on targeted attacks that try to promote certain target items (Zhang et al. 2022a; Rong et al. 2022). The untargeted attack that aims to degrade the overall performance of the FedRec system is still less explored (Wu et al. 2022). In fact, without an effective defense mechanism, the untargeted attack can continuously disrupt the user experience of the system, which will lead to severe losses of customers and revenue for the service provider (Li et al. 2016). Thus, it is necessary to explore the untargeted attack and its defense for FedRec systems.

The untargeted poisoning attack against FedRec systems faces several critical challenges. First, the attack method must be effective even with a small fraction of malicious clients. Considering that a recommender system usually has millions of users, it is impractical for the attacker to control a large number of clients. Second, the attacker can only access a small set of data stored on the malicious clients as the clients never share their local training data in the FL framework. Since existing poisoning attack methods against centralized recommendation model learning usually require a strong knowledge of the full training data (Fang et al. 2018; Wu et al. 2021a), they are infeasible in the FedRec scenario. Third, the untargeted attack aims to degrade the overall performance of FedRec systems on arbitrary inputs. It is more challenging than the targeted attack that only manipulates the model output on certain target items. Fourth, many recommenders are trained on implicit user feedback with heavy noise, which makes them robust to malicious perturbation to a certain degree (Yu and Qin 2020; Wang et al. 2021).

To address these challenges, in this paper, we first propose a novel untargeted model poisoning attack method named *ClusterAttack*, which can effectively degrade the overall performance of FedRec systems with a small fraction of malicious clients. Its main idea is to upload poisonous gradients

*Corresponding Author.

[1]https://gdpr-info.eu
[2]https://oag.ca.gov/privacy/ccpa

that converge the item embeddings of the recommendation model into several dense clusters, which can let the recommender generate similar scores for these close items in the same cluster and perturb the ranking order. Specifically, we split the item embeddings into several clusters with an adaptive clustering mechanism and compute the malicious gradient that reduces the within-cluster variance. To make our attack more difficult to be detected, we clip the malicious gradient with a norm bound estimated from the normal gradients before uploading it to the server. As most existing defense methods cannot effectively defend *ClusterAttack*, we further propose a uniformity-based defense mechanism (*UNION*) to protect FedRec systems from such attacks. We require all benign clients to train the local recommendation model with an additional contrastive learning task that regularizes the item embeddings toward a uniform distribution in the space. Then the server identifies these malicious gradients by estimating the uniformity of updated item embeddings. In addition, our *UNION* mechanism can be combined with many existing Byzantine-robust FL methods to provide more comprehensive protection for FedRec systems. Extensive experiments on two public datasets show that our *ClusterAttack* can effectively degrade the performance of FedRec systems without being detected, and our *UNION* mechanism can improve the resistance of the system against many untargeted attacks, including our *ClusterAttack*[3].

The main contributions of our work are listed as follows:

- We propose *ClusterAttack*, a novel untargeted model poisoning attack method, which reveals the security risk of FedRec systems even with existing defense methods.
- We propose *UNION*, a defense mechanism that improves the resistance of FedRec systems against various untargeted poisoning attacks. To our best knowledge, it is the first defense mechanism specialized for FedRec systems.
- Extensive experiments on two public datasets validate the effectiveness of our *ClusterAttack* and *UNION* methods.

2 Related Work

2.1 Attack & Defense for Recommender Systems

Poisoning attacks against recommender systems and their defense have been widely studied in the past decades (Lam and Riedl 2004; Zhou et al. 2016; Aktukmak, Yilmaz, and Uysal 2019). However, these researches mainly focus on the centralized training of recommendation models. They require the attacker or the server to have strong knowledge of the full training data to perform effective attacks or defenses, such as all user profiles (Burke et al. 2006) or the entire rating matrix (Li et al. 2016). These methods are infeasible under the FL setting since the server cannot access the data of the clients. Recently, some targeted poisoning attack methods have been proposed to boost certain target items in the FedRec scenario (Zhang et al. 2022a; Rong et al. 2022), while the untargeted attack and its defense are still less explored. Wu et al. (2022) is a recent untargeted data poisoning attack against FedRec systems. It subverts the local model training by choosing items closest to the user embedding as negative samples and the farthest ones as positive samples. However, it only manipulates the input training data while keeping the local training algorithm unmodified, which limits its attack effect. Our *ClusterAttack* utilizes the vulnerability of FL and performs a more powerful model poisoning attack, which can effectively degrade the performance of the FedRec system with a small fraction of malicious clients.

[3]Our code is available at https://github.com/yflyl613/FedRec.

数据投毒攻击在联邦推荐系统中2022年提出的

2.2 Attack & Defense for Federated Learning

In the general FL domain, several untargeted poisoning attack methods have been proposed and can be directly applied to degrade the performance of FedRec systems (Fang et al. 2020; Tolpegin et al. 2020). For example, Baruch, Baruch, and Goldberg (2019) propose to add a small amount of noise to each dimension of the average normal gradient, where the intensity of noise is estimated by the ratio of malicious clients. Fang et al. (2020) propose to perturb the uploaded gradient by adding noise in opposite directions inferred from normal updates. However, these methods usually require a large fraction of malicious clients (e.g., 20%) to achieve a significant performance degradation, which is unrealistic for a FedRec system with millions of users. To protect the FL system from potential poisoning attacks, researchers have also proposed several Byzantine-robust FL methods in the past few years (Yin et al. 2018; Zhang et al. 2022b). Although these defense methods can guarantee the convergence of the global model, we found that most of them perform poorly against carefully-designed poisoning attacks in the FedRec scenario. Due to the diversity of user interests, the training data on each client is highly non-IID. Some gradients uploaded by benign clients may also deviate from others, which makes it more difficult for the server to distinguish these malicious gradients. Our *UNION* mechanism can be combined with existing Byzantine-robust FL methods and improves their performance against many untargeted poisoning attacks, especially our *ClusterAttack*.

3 Preliminaries

In this section, we introduce the settings of federated recommendation systems and the threat model used in this work.

3.1 Federated Recommendation Systems

Let $\mathcal{I}$ and $\mathcal{U}$ denote the sets of M items and N users/clients in a recommender system, respectively. These clients try to train a global model collaboratively without sharing their private data. We assume that the parameters of the recommendation model Θ consist of three parts: an item model Θ_{item} that converts the item ID into the item embedding, a user model Θ_{user} that infers the user interest embedding from the user profile (e.g., the user ID or historical interacted items), and a predictor model Θ_{pred} that predicts a ranking score given an item embedding and a user embedding. In each training round, the server first distributes the current global model parameters $[\Theta_{\text{item}}; \Theta_{\text{pred}}]$ to n randomly selected clients. Then each selected client computes the update gradient $\mathbf{g} = [\mathbf{g}_{\text{item}}; \mathbf{g}_{\text{user}}; \mathbf{g}_{\text{pred}}]$ with their local data. Following previous works (Rong, He, and Chen 2022; Yu et al. 2022), we assume they use BPR (Rendle et al. 2009) with

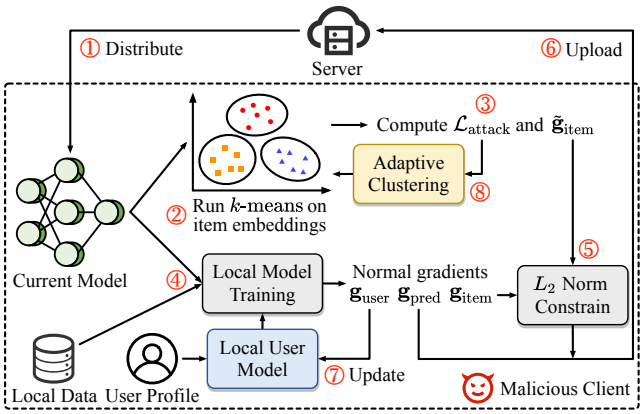

Figure 1: The procedure of our *ClusterAttack*.

L_2 regularization to train the local model, i.e., the gradient $\mathbf{g}$ is generated by optimizing the following loss function:

$$\mathcal{L}_{\text{rec}} = -\log\left(\sigma\left(\hat{y}_{\text{p}} - \hat{y}_{\text{n}}\right)\right) + \lambda \left\|\Theta\right\|_2^2, \tag{1}$$

where σ is the sigmoid function. $\hat{y}_{\text{p}}$ and $\hat{y}_{\text{n}}$ are the predicted ranking scores of the positive and negative items. Next, the client uploads $[\mathbf{g}_{\text{item}}; \mathbf{g}_{\text{pred}}]$ to the server and updates the local user model with $\mathbf{g}_{\text{user}}$, which is not uploaded due to its privacy sensitivity (Wu et al. 2021b,c). Finally, the server aggregates all the received gradients with certain aggregation rules and updates the global model. Such training round proceeds iteratively until convergence.

3.2 Threat Model

Attack Goal. The attacker aims to degrade the overall performance of the FedRec system on arbitrary inputs.

Attack Capability and Knowledge. The attacker controls a set of malicious clients $\mathcal{U}_{\text{mal}}$ which accounts for $m\%$ of $\mathcal{U}$. As there are usually millions of users in a recommender system, we assume that m should be small (e.g., $m = 1$). Following previous works (Wu et al. 2022; Zhang et al. 2022a), we assume that the attacker has access to the training code, local model, and user data on the devices of malicious clients while cannot access the data or gradients of other benign clients. The attacker can arbitrarily modify the gradients uploaded by the malicious clients. We also assume the attacker does not know the aggregation rule used by the server.

4 Methodology

In this section, we introduce the details of our untargeted model poisoning attack method *ClusterAttack* and our defense mechanism *UNION* for FedRec systems.

4.1 *ClusterAttack*

To degrade the overall performance of FedRec systems, our *ClusterAttack* aims to poison the item embeddings, which are widely used in most recommendation models (Kang and McAuley 2018; Wang et al. 2019). Since a recommendation model generally predicts the ranking score based on a user embedding and an item embedding, our main idea is to converge these item embeddings into several dense clusters. Thus, the recommender tends to generate similar scores for these close items in the same cluster and mess up the ranking order. Figure 1 illustrates the procedure of *ClusterAttack*.

Algorithm 1: Adaptive Clustering

Input: Number of clusters K, range of number of clusters $[K_{\min}, K_{\max}]$, threshold R, and decay rate β.
Init: Set $\tilde{\mathcal{L}}_{\text{attack}}^{(0)}$, n_{inc}, n_{dec} and t as 0.
// Repeat after each round of attack
$t \leftarrow t + 1$;
Calculate $\mathcal{L}_{\text{attack}}^{(t)}$ with Equation (2);
$\tilde{\mathcal{L}}_{\text{attack}}^{(t)} \leftarrow \beta \cdot \tilde{\mathcal{L}}_{\text{attack}}^{(t-1)} + (1-\beta) \cdot \mathcal{L}_{\text{attack}}^{(t)}$;
$\hat{\mathcal{L}}_{\text{attack}}^{(t)} \leftarrow \tilde{\mathcal{L}}_{\text{attack}}^{(t)} / \left(1 - \beta^t\right)$;
if $\hat{\mathcal{L}}_{\text{attack}}^{(t)} > \hat{\mathcal{L}}_{\text{attack}}^{(t-1)}$ **then** $n_{\text{inc}} \leftarrow n_{\text{inc}} + 1$;
else $n_{dec} \leftarrow n_{\text{dec}} + 1$;
if $n_{\text{inc}} - n_{\text{dec}} \geqslant R$ **then**
    $K \leftarrow \min\left(\left\lfloor K + \sqrt{K_{\max} - K} \right\rfloor, K_{\max}\right)$;
    Reset $n_{\text{inc}}, n_{\text{dec}}$ and t as 0;
end if
if $n_{\text{dec}} - n_{\text{inc}} \geqslant R$ **then**
    $K \leftarrow \max\left(\left\lfloor K - \sqrt{K - K_{\min}} \right\rfloor, K_{\min}\right)$;
    Reset $n_{\text{inc}}, n_{\text{dec}}$ and t as 0;
end if

When selected for model training, the malicious client receives the latest global model from the server, which contains the item embeddings $\{\boldsymbol{v}_i\}_{i=1}^{M}$ (Step 1). We first apply k-means (Lloyd 1982) to split them into K clusters $\{C_i\}_{i=1}^{K}$ with centroids $\{\boldsymbol{c}_i\}_{i=1}^{K}$ (Step 2). Then we compute the following loss function to measure the within-cluster variance:

$$\mathcal{L}_{\text{attack}} = \sum_{i=1}^{K} \sum_{\boldsymbol{v}_j \in C_i} \left\|\boldsymbol{v}_j - \boldsymbol{c}_i\right\|_2^2. \tag{2}$$

The malicious gradient of each item embedding is computed to minimize the above attack loss, i.e., $\tilde{\boldsymbol{g}}_{\boldsymbol{v}_i} = \partial \mathcal{L}_{\text{attack}} / \partial \boldsymbol{v}_i$ (Step 3). To make our attack stealthier, we further clip $\tilde{\boldsymbol{g}}_{\boldsymbol{v}_i}$ with an estimated norm of normal item embedding gradients. Specifically, for each malicious client $u^{(j)} \in \mathcal{U}_{\text{mal}}$, we compute the normal gradient with the original loss function $\mathcal{L}_{\text{rec}}$ and his local training data (Step 4). Then we calculate the mean μ and standard deviation σ of the L_2 norms of all normal item embedding gradients. Assuming these norms follow a Gaussian distribution, we generate a reasonable norm bound $b_i^{(j)} = \mu + \lambda_i^{(j)} \sigma$ for each item embedding $\boldsymbol{v}_i$ on the malicious client $u^{(j)}$, where $\lambda_i^{(j)}$ is a number randomly sampled from $[0, 3]$. Therefore, the clipped malicious item embedding gradients are formulated as follows:

$$\hat{\mathbf{g}}_{\boldsymbol{v}_i}^{(j)} = \frac{\tilde{\mathbf{g}}_{\boldsymbol{v}_i}}{\max\left(1, \left\|\tilde{\mathbf{g}}_{\boldsymbol{v}_i}\right\|_2 / b_i^{(j)}\right)}. \tag{3}$$

The malicious gradient of the item model is set as $\hat{\mathbf{g}}_{\text{item}}^{(j)} = \left[\hat{\mathbf{g}}_{\boldsymbol{v}_1}^{(j)}; \hat{\mathbf{g}}_{\boldsymbol{v}_2}^{(j)}; \cdots; \hat{\mathbf{g}}_{\boldsymbol{v}_M}^{(j)}\right]$. Finally, the malicious client uploads $\hat{\mathbf{g}}^{(j)} = \left[\hat{\mathbf{g}}_{\text{item}}^{(j)}; \mathbf{g}_{\text{pred}}^{(j)}\right]$ to the server and updates its local user model with the normal gradient $\mathbf{g}_{\text{user}}^{(j)}$ (Step 6 & 7).

Algorithm 2: Defense Procedure on the Server Side

Input: A set of received model gradients $\{\mathbf{g}^{(i)}\}_{i=1}^{n} = \left\{\left[\mathbf{g}_{\text{item}}^{(i)}; \mathbf{g}_{\text{pred}}^{(i)}\right]\right\}_{i=1}^{n}$, current global model $\mathbf{\Theta} = [\mathbf{\Theta}_{\text{item}}; \mathbf{\Theta}_{\text{pred}}]$, learning rate η, the item set $\mathcal{I}$, and number of random sampling T.

Output: A set of filtered model gradients $\mathcal{G}_{\text{filter}}$.

for $i \leftarrow 1$ **to** n **do**
  Update the item model $\mathbf{\Theta}_{\text{item}}^{(i)\prime} \leftarrow \mathbf{\Theta}_{\text{item}} - \eta \cdot \mathbf{g}_{\text{item}}^{(i)}$;
  Randomly select T items $\{v_i\}_{i=1}^{T}$ from $\mathcal{I}$ to estimate the uniformity of updated item embeddings $d_i \leftarrow \frac{2}{T(T-1)} \sum_{j=1}^{T} \sum_{k=j+1}^{T} \|f(v_j) - f(v_k)\|_2^2$;
end for
if $\text{GapStatistics}(\{d_i\}_{i=1}^{n})$ **then**
  Running k-means to split $\{d_i\}_{i=1}^{n}$ into two clusters;
  $\mathcal{G}_{\text{filter}} \leftarrow$ all the gradients in the larger cluster;
else
  $\mathcal{G}_{\text{filter}} \leftarrow \{\mathbf{g}^{(i)}\}_{i=1}^{n}$;
end if
return $\mathcal{G}_{\text{filter}}$

Considering that the number of clusters K will greatly impact the attack effect, we further design an adaptive clustering mechanism to automatically adjust the value of K after each round of attack (Step 8), which is shown in **Algorithm 1**. Since the only feedback of the attack effect for the attacker is $\mathcal{L}_{\text{attack}}$ in Equation (2), we track it during the attack process and compute its bias-corrected exponential moving average $\hat{\mathcal{L}}_{attack}$. We use two counters n_{inc} and n_{dec} to record the number of rounds in which the smoothed attack loss increases and decreases, respectively. If $\hat{\mathcal{L}}_{attack}$ increased in most of the past few rounds, we assume that the current value of K is too small, and the attack loss cannot converge well. Thus, we increase the value of K to make the attack easier. Contrarily, if $\hat{\mathcal{L}}_{attack}$ keeps descending, we further decrease the value of K to perform a stronger attack.

4.2 *UNION* Mechanism

Our *ClusterAttack* reveals that maintaining the distribution of item embeddings is essential for protecting FedRec systems. Thus, we further propose a uniformity-based defense mechanism (*UNION*) that regularizes the item embeddings toward a uniform distribution in the space with a contrastive learning task. Then the server filters out these malicious gradients that lead to abnormally distributed item embeddings.

Client Side. We require all benign clients to train the local recommendation model with an additional contrastive learning (CL) task. Specifically, denote the item set interacted by a client as $\mathcal{I}_u = \{v_i\}_{i=1}^{L}$. For each $v_i \in \mathcal{I}_u$, we randomly select another item v_i^+ from $\mathcal{I}_u$ as the positive sample, and P items $\{v_i^-\}_{i=1}^{P}$ from $\mathcal{I} \backslash \mathcal{I}_u$ as the negative samples. We adopt InfoNCE (Oord, Li, and Vinyals 2018) as the contrastive loss function. It is formulated as follows:

$$\mathcal{L}_{\text{cl}} = -\sum_{i=1}^{L} \log \frac{e^{f(v_i)^\top f(v_i^+)}}{e^{f(v_i)^\top f(v_i^+)} + \sum_{j=1}^{P} e^{f(v_i)^\top f(v_j^-)}}, \quad (4)$$

Algorithm 3: Gap Statistics

Input: A set of estimated uniformity $\{d_i\}_{i=1}^{n}$, and number of sampling B.

Output: Whether there is more than one cluster.

$\left\{\tilde{d}_i\right\}_{i=1}^{n} \leftarrow$ apply min-max normalization to $\{d_i\}_{i=1}^{n}$;
for $k \in \{1, 2\}$ *in parallel* **do**
  Apply k-means on $\left\{\tilde{d}_i\right\}_{i=1}^{n}$ to get clusters $\{D_i\}_{i=1}^{k}$ with centroids $\{\mu_i\}_{i=1}^{k}$;
  $w_k \leftarrow \sum_{i=1}^{k} \sum_{\tilde{d}_j \in D_i} \left\|\tilde{d}_j - \mu_i\right\|_2^2$;
  for $b \leftarrow 1$ **to** B **do**
    Uniformly sample n points $\{t_i\}_{i=1}^{n}$ from $[0, 1]$;
    Apply k-means on $\{t_i\}_{i=1}^{n}$ to get clusters $\{D_i'\}_{i=1}^{k}$ with centoids $\{\mu_i'\}_{i=1}^{k}$;
    $w_b^* \leftarrow \sum_{i=1}^{k} \sum_{t_j \in D_i'} \|t_j - \mu_i'\|_2^2$;
  end for
  $\overline{w} = \frac{1}{B} \sum_{b=1}^{B} \log(w_b^*)$;
  $gap_k \leftarrow \overline{w} - \log(w_k)$;
  $s_k \leftarrow \sqrt{\frac{1+B}{B^2} \sum_{b=1}^{B} [\log(w_b^*) - \overline{w}]^2}$;
end for
return $gap_1 < gap_2 - s_2$

where f denotes the item model. The overall loss function on the client side is $\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha \cdot \mathcal{L}_{\text{cl}}$. As proved by Wang and Isola (2020), the contrastive loss $\mathcal{L}_{\text{cl}}$ asymptotically optimizes the uniformity of the distribution induced from the learned embeddings, which is measured as follows:

$$\mathcal{L}_{\text{uniform}}(f; t) = \mathbb{E}_{x,y \overset{\text{i.i.d}}{\sim} p_{\text{data}}} e^{-t\|f(x)-f(y)\|_2^2}, \ t > 0. \quad (5)$$

Therefore, the additional CL task can regularize the item embeddings toward a uniform distribution in the space while training with the recommendation task. Since such an optimization objective is opposite to the goal of *ClusterAttack*, the CL task can mitigate its attack effect and also makes it easier for the server to distinguish these malicious gradients.

Server Side. Since now all benign clients train the model with the CL task that optimizes the item embeddings toward a uniform distribution, we let the server estimate the uniformity of updated item embeddings for each received gradient. Here we measure the uniformity in terms of the average L_2 distance between any two item embeddings, i.e.,

$$\mathcal{L}'_{\text{uniform}}(f) = \mathbb{E}_{x,y \overset{\text{i.i.d}}{\sim} p_{\text{data}}} \|f(x) - f(y)\|_2^2, \quad (6)$$

which is a simplified version of Equation (5). The defense procedure on the server side is shown in **Algorithm 2**. We further adopt the Gap Statistics algorithm (Tibshirani, Walther, and Hastie 2001) to estimate the number of clusters in the set of estimated uniformity $\{d_i\}_{i=1}^{n}$, which is shown in **Algorithm 3**. Generally, it compares the change of within-cluster dispersion with that expected under a uniform distribution to determine the number of clusters in a set of data. If the algorithm estimates that there is more than one cluster, we believe there are some malicious gradients that lead to abnormally distributed item embeddings. Hence, we further

Model	Attack Method	ML-1M		Gowalla	
		HR@5	NDCG@5	HR@5	NDCG@5
MF	No Attack	0.03549 (-)	0.02226(-)	0.02523 (-)	0.01697 (-)
	LabelFlip	0.03561 (-0.34%)	0.02238 (-0.54%)	0.02541 (-0.71%)	0.01711 (-0.82%)
	FedAttack	0.03358 (5.38%)	0.02118 (4.85%)	0.02371 (6.02%)	0.01585 (6.60%)
	Gaussian	0.03555 (-0.17%)	0.02224 (0.09%)	0.02528 (-0.20%)	0.01701 (-0.24%)
	LIE	0.03259 (8.17%)	0.02062 (7.37%)	0.02316 (8.20%)	0.01571 (7.42%)
	Fang	0.03038 (14.40%)	0.01897 (14.78%)	0.02131 (15.54%)	0.01448 (14.67%)
	ClusterAttack	**0.02451 (30.94%)**	**0.01545 (30.59%)**	**0.01664 (34.05%)**	**0.01117 (34.18%)**
SASRec	No Attack	0.10810 (-)	0.07053 (-)	0.03251 (-)	0.02217 (-)
	LabelFlip	0.10857 (-0.43%)	0.07071 (-0.26%)	0.03270 (-0.58%)	0.02222 (-0.23%)
	FedAttack	0.10013 (7.37%)	0.06572 (6.82%)	0.03054 (6.06%)	0.02087 (5.86%)
	Gaussian	0.10769 (0.38%)	0.07055 (-0.03%)	0.03226 (0.77%)	0.02222 (-0.23%)
	LIE	0.09677 (10.48%)	0.06281 (10.95%)	0.03008 (7.47%)	0.02021 (8.84%)
	Fang	0.08964 (17.08%)	0.05909 (16.22%)	0.02797 (13.96%)	0.01883 (15.07%)
	ClusterAttack	**0.06547 (39.44%)**	**0.04130 (41.44%)**	**0.02223 (31.62%)**	**0.01544 (30.36%)**

Table 1: Model performance under different untargeted attack methods with no defense. The percentages in parentheses indicate the relative performance degradation compared with the no-attack scenario.

apply k-means to split $\{d_i\}_{i=1}^{n}$ into two clusters and filter out all the gradients belonging to the minor one.

It is noted that *UNION* is a general mechanism that aims to preserve the distribution of item embeddings. It can be easily combined with many existing Byzantine-robust FL methods (Blanchard et al. 2017; Wang et al. 2020) to provide more comprehensive protection for FedRec systems. These methods can learn a more accurate model on the set of filtered model gradients returned by our *UNION* mechanism while maintaining their original convergence guarantee.

5 Experiments

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

- **RQ1**: How does our *ClusterAttack* perform compared with existing untargeted attack methods?
- **RQ2**: Can our *ClusterAttack* circumvent existing defense methods while preserving its attack performance?
- **RQ3**: How does our *UNION* mechanism perform against existing untargeted attacks and our *ClusterAttack*?
- **RQ4**: How does the ratio of malicious clients affect the performance of our methods?
- **RQ5**: Is the proposed adaptive clustering mechanism in our *ClusterAttack* effective?
- **RQ6**: How difficult it is to defend our *ClusterAttack*, and why does our *UNION* mechanism work?

5.1 Datasets and Experimental Settings

We conduct experiments with two public datasets. The first is ML-1M (Harper and Konstan 2016), a movie recommendation dataset. The second is Gowalla (Liang et al. 2016), a check-in dataset obtained from the Gowalla website. We adopt the 10-core version used in (Wang et al. 2019), i.e., retaining users and items with at least ten interactions. The statistics of the two datasets are shown in Table 2. Following previous works (He et al. 2017; Sun et al. 2019), we adopt the leave-one-out approach and hold out the latest interacted item of each user as the test data. We use the item before the last one for validation and the rest for training.

Dataset	#Users	#Items	#Actions	Avg. length	Density
ML-1M	6,040	3,706	1,000,209	165.6	4.47%
Gowalla	29,858	40,981	1,585,043	53.1	0.13%

Table 2: Detailed statistics of the two datasets.

In our experiments, we choose the widely used MF (Rendle et al. 2009) and SASRec (Kang and McAuley 2018) as the recommendation model. The hidden dimension of models is 64. We use FedAvg (McMahan et al. 2017) with Adam optimizer (Kingma and Ba 2015) as the FL framework. Each user is treated as a client in the FedRec system. 50 clients are randomly selected in each round for model training. We randomly select 1% of users from the entire user set $\mathcal{U}$ and take them as malicious clients. The detailed experimental settings are listed in the Appendix. Following (Zhang et al. 2022a; Wu et al. 2022), we use the Hit Ratio (HR) and the Normalized Discounted Cumulative Gain (NDCG) over the top 5 ranked items to measure the performance of the recommendation model. Note that the metrics are only calculated on benign clients using the all-ranking protocol, i.e., all items not interacted with by the user are used as candidates. All the hyper-parameters are tuned on the validation set. We repeat each experiment 5 times and report the average results.

5.2 Attack Performance Evaluation (RQ1)

We compare the attack performance of *ClusterAttack* with the following *data poisoning attack* methods:

- *LabelFlip* (Tolpegin et al. 2020), which flips the label of the training sample on malicious clients;
- *FedAttack* (Wu et al. 2022), which chooses items that are most similar to the user embedding as negative samples and the farthest ones as positive samples;

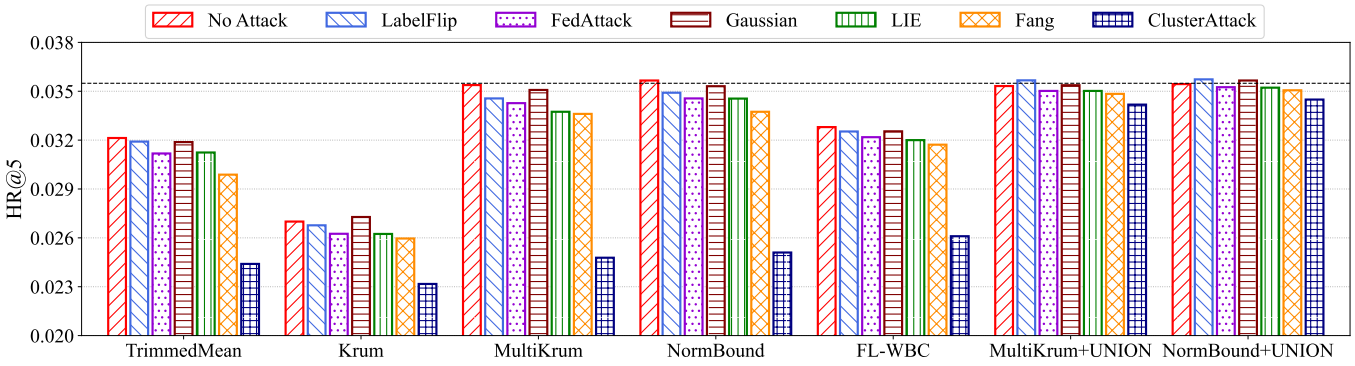

Figure 2: Model performance under different untargeted attack methods with different defense mechanisms. The black dashed line represents the model performance without any attack or defense.

and the following *model poisoning attack* methods:

- *Gaussian* (Fang et al. 2020), which computes the mean and standard deviation of normal gradients and then uploads samples from the estimated Gaussian distribution;
- *LIE* (Baruch, Baruch, and Goldberg 2019), which adds small amounts of noise to the average normal gradient;
- *Fang* (Fang et al. 2020), which adds noise in opposite directions to the average normal gradient.

The experimental results are shown in Table 1. From the results, we have several findings. First, not all untargeted poisoning attacks are effective against FedRec systems when the number of malicious clients is limited. Especially, *LabelFlip* and *Gaussian* even slightly raise the performance of the model. This may be because such limited perturbations make the recommendation model more robust to the noise in user behaviors. Second, well-designed model poisoning attacks (*LIE*, *Fang*, and *ClusterAttack*) usually perform better than data poisoning attacks (*LabelFlip* and *FedAttack*). This is because these model poisoning attacks directly modify the uploaded gradient, which is more flexible and effective than manipulating the training data. Third, our *ClusterAttack* consistently outperforms other baselines by a large margin. The reason is that our method uploads the poisonous gradient that converges the item embeddings into several dense clusters. It is more effective than these baseline model poisoning attacks that only add certain perturbation noise to the normal gradients. Besides, our adaptive clustering mechanism can automatically adjust the number of clusters after each round of attack, which also leads to better attack performance.

5.3 Attack Effectiveness under Defense (RQ2)

To verify the effectiveness of *ClusterAttack* under defense, we compare the attack performance of different attack methods against the following Byzantine-robust FL methods:

- *TrimmedMean* (Yin et al. 2018), which computes the coordinate-wise trimmed mean of the received gradients.
- *Krum* (Blanchard et al. 2017), which selects the gradient closest to its neighboring gradients for the model update.
- *MultiKrum* (Blanchard et al. 2017), which selects multiple gradients via *Krum* and calculates their average.

Defense Method	Attack Method	HR@5
MultiKrum+UNION	ClusterAttack	0.03378 (4.82%)
	ClusterAttack+CL	0.03525 (0.68%)
NormBound+UNION	ClusterAttack	0.03449 (2.82%)
	ClusterAttack+CL	0.03566 (-0.48%)

Table 3: Attack performance of *ClusterAttack+CL*.

- *NormBound* (Wang et al. 2020), which clips the L_2 norm of received gradients with a threshold before aggregation.
- *FL-WBC* (Sun et al. 2021), in which all benign clients upload partially masked gradients with Laplace noise to mitigate the attack effect on the global model.

We use MF and ML-1M in the following experiments (the results with SASRec and Gowalla show similar trends and are omitted due to the space limit). The results are shown as the *left five* groups in Figure 2. We find that some Byzantine-robust FL methods such as *TrimmedMean* and *Krum* severely degrade the performance of the recommendation model even without any attack. We attribute it to the highly non-IID training data on each client. Some benign gradients may also deviate from others. Therefore, they may be incorrectly filtered out by these defense methods, which will impair the performance of the global model. The partially masked gradients used in *FL-WBC* will cause certain information loss which also leads to performance degradation. In contrast, *MultiKrum* and *NormBound* do not substantially hurt the model performance and can mitigate the impact of most existing attacks. However, none of these existing defense methods can effectively defend *ClusterAttack*. The reason is that we only manipulate the item embedding gradients and further bound their L_2 norms by the one estimated from normal gradients, which makes it less likely to be detected as an outlier by these defense methods. Besides, our further analysis finds that the non-IID training data on each client also covers our attack (See Section 5.7).

5.4 Defense Performance Evaluation (RQ3)

In this subsection, we evaluate the effectiveness of our *UNION* mechanism. Since *MultiKrum* and *NormBound* will

$m\%$	No Attack	FedAttack	LIE	Fang	ClusterAttack
0.5%	0.03549	0.03491	0.03465	0.03426	0.03001
1%	0.03549	0.03358	0.03259	0.03038	0.02451

Table 4: Attack performance of different untargeted attacks with different ratios of malicious clients.

Defense Method	FedAttack	LIE	Fang	ClusterAttack
No Defense	0.03195	0.03147	0.02793	0.01950
MultiKrum+UNION	0.03438	0.03447	0.03454	0.03291
MormBound+UNION	0.03490	0.03464	0.03464	0.03351

Table 5: Defense performance of *UNION* against different untargeted attacks with 5% malicious clients.

not greatly hurt the model performance, we combine them with our *UNION* mechanism respectively and test their defense performance against different attacks. The results are shown as the *right two* groups in Figure 2. We first find that our *UNION* mechanism can significantly improve the resistance of these defense methods against *ClusterAttack*. The reason is that the additional CL task optimizes the item embeddings toward a uniform distribution, which is opposite to the goal of *ClusterAttack*. Besides, the server also keeps filtering out the gradients leading to abnormally distributed item embeddings. Thus, the goal of *ClusterAttack* cannot be effectively achieved. We also find that our *UNION* mechanism enhances the performance of these defense methods against other baseline attacks. This verifies that regularizing the distribution of item embeddings is beneficial to protecting FedRec systems from various untargeted attacks.

Since our *UNION* mechanism modifies the training algorithm on the client side, which can be known by the attacker via malicious clients, we wonder whether the attacker can avoid detection by generating malicious gradients along with the CL task. Thus, we further evaluate the defense performance of *UNION* against a new attack method *ClusterAttack+CL*, which generates the malicious gradient by optimizing $\mathcal{L}'_{\text{attack}} = \mathcal{L}_{\text{attack}} + \alpha \cdot \mathcal{L}_{\text{cl}}$. As shown in Table 3, the extra CL task weakens the attack effect of *ClusterAttack*. This is because $\mathcal{L}_{\text{attack}}$ tries to converge the item embeddings into clusters, while $\mathcal{L}_{\text{cl}}$ regularizes these embeddings to be uniformly distributed. Such opposite goals cannot be jointly optimized well and lead to poor attack effects.

5.5 Influence of the Ratio of Malicious Clients (RQ4)

In this subsection, we further conduct several experiments to explore how the ratio of malicious clients affects the performance of our methods. First, we set the ratio of malicious clients $m\%$ as 0.5% and 1% respectively and compare the performance of our *ClusterAttack* with various baseline untargeted attacks. The HR@5 of the model are shown in Table 4. It shows that most existing attack methods are ineffective with very few malicious clients, while our *ClusterAttack* can still degrade the model performance by 15.49% even with 0.5% malicious clients. This verifies that our malicious gradients which aim to converge the item embeddings into dense clusters are highly effective in perturbing the ranking order of the FedRec system. Next, to validate the robustness of our *UNION* mechanism, we increase the ratio of malicious clients to 5% and evaluate its performance against various attacks. The HR@5 of the model shown in Table 5 demonstrate that after combining with our *UNION* mechanism, *MultiKrum+UNION* and *NormBound+UNION* can well protect the FedRec system against various untargeted attacks even with a large number of malicious clients.

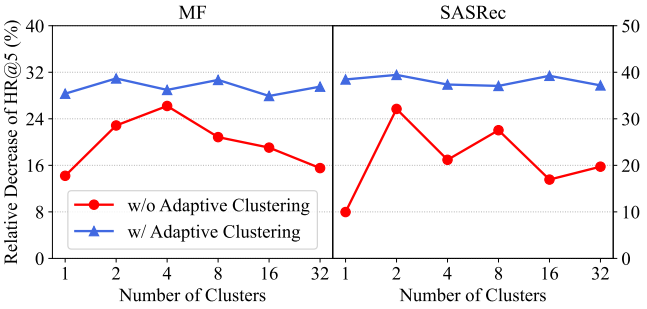

Figure 3: Impact of adaptive clustering.

5.6 Impact of Adaptive Clustering (RQ5)

In this subsection, we conduct experiments to verify the impact of the adaptive clustering mechanism in *ClusterAttack*. We set the initial number of clusters K as $\{1, 2, 4, 8, 16\}$ respectively and compare the attack performance of *ClusterAttack* and its variant with the adaptive clustering mechanism removed. The results on the ML-1M dataset are shown in Figure 3. We find that without the adaptive clustering mechanism, the attack effect does not always improve with the decreasing number of clusters as intuitively thought. This is because when the value of K is too small, the attack loss cannot effectively converge with limited malicious clients. Results also show that the attack effect varies significantly with different numbers of clusters. The attack performance is consistently better when the adaptive clustering mechanism is adopted to adjust the value of K based on the attack effect after each round of attack, and it is less influenced by the selection of the initial number of clusters.

5.7 Gradients and Uniformity Analysis (RQ6)

To better understand how difficult it is to defend *ClusterAttack*, we save all the model gradients received by the server during the training procedure. Every 10 rounds, we use PCA to reduce the dimension of these saved gradients and visualize them in the first row of Figure 4. We find that these malicious gradients have similar PCA components to most normal gradients, while some gradients uploaded by benign clients seem to be outliers. This verifies that it is quite difficult to distinguish these malicious gradients due to the highly non-IID training data on each client. We further save all the item embedding uniformities estimated by the server with our *UNION* mechanism and its variant when the CL task on the client side is removed. We use kernel density estimation (KDE) (Parzen 1962) to estimate their probability density distributions, which are visualized in the last two

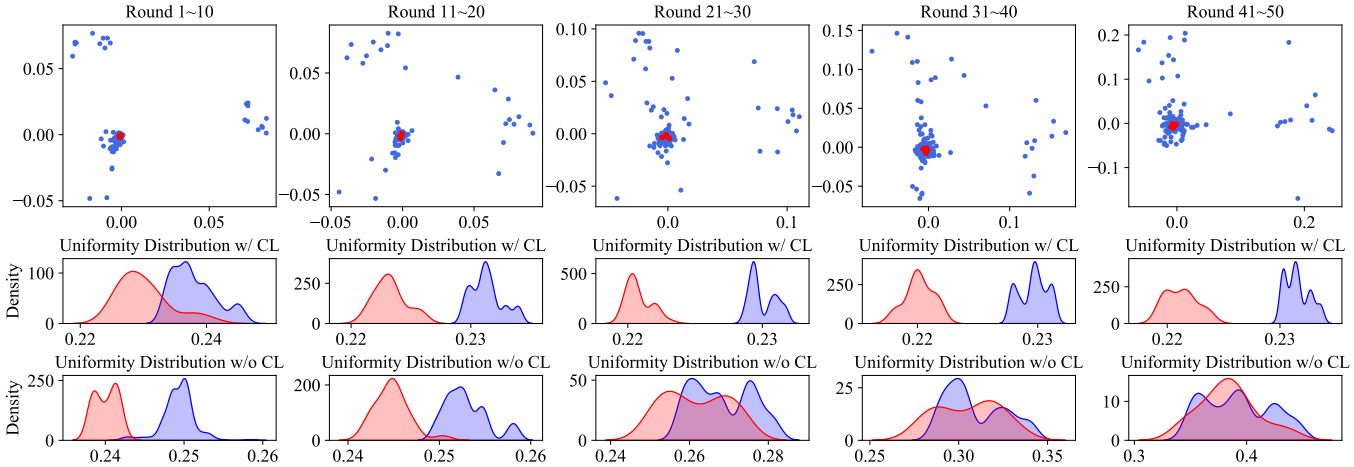

Figure 4: Visualization of the uploaded gradients and the uniformity distribution in different rounds of model training. The blue color and red color denote benign clients and malicious clients, respectively.

rows of Figure 4. It shows that without the CL task, the item embeddings updated with normal gradients and malicious gradients tend to have similar uniformity. In contrast, the estimated uniformities fall into two well-separated parts with the CL task. Therefore, the server can better filter out these malicious gradients and train a more accurate model.

6 Conclusion

In this paper, we studied the untargeted poisoning attack and its defense for FedRec systems. We first presented a novel untargeted attack named *ClusterAttack*. By uploading poisonous gradients that converge the item embeddings into several dense clusters, it can effectively degrade the overall performance of the FedRec system. Meanwhile, we proposed a uniformity-based defense mechanism to protect the system from such attacks. It requires all clients to train the recommendation model with an additional contrastive learning task, which enables the server to distinguish these malicious gradients based on the estimated uniformity of updated item embeddings. Extensive experiments validated the effectiveness of our attack and defense methods. Our work reveals the security risk of FedRec systems and provides a general defense mechanism that can be combined with existing Byzantine-robust FL methods to better protect the system from potential untargeted attacks in the real world.

Appendix A Experimental Settings

In our experiments, the hidden dimensions of both MF and SASRec models are set to 64. We set the dropout ratio as 0.2 to mitigate overfitting. The maximum sequence length of SASRec is set to 200 and 100 for ML-1M and Gowalla, respectively. We use FedAvg (McMahan et al. 2017) with Adam optimizer (Kingma and Ba 2015) as the FL framework. The FL training procedure runs for at most 6,000 rounds to ensure the convergence of the global recommendation model. The number of clients randomly selected per round n is 50. In our *ClusterAttack*, we set the initial number of clusters as 2 and 8 for ML-1M and Gowalla respectively considering the size of their item sets. The range of the number of clusters and the threshold R is set to $[1, 50]$ and 100 for both datasets. In our *UNION* mechanism, we set the number of negative samples P in the contrastive learning task as 15. The coefficient α is set to 1. The number of random sampling for uniformity estimation T is 500. In the Gap Statistics algorithm, the number of random sampling B is 50. The hyper-parameter settings specific to each model and dataset are listed in Table 6.

Hyper-parameters	ML-1M		Gowalla	
	MF	SASRec	MF	SASRec
Learning rate η	2e-3	1e-3	2e-3	1e-3
L_2 regularization coefficient λ	1e-5	1e-5	1e-6	1e-5
L_2 norm bound for *NormBound*	0.1	1.5	0.1	2.5

Table 6: Hyper-parameter settings.

Appendix B Experimental Environment

We conduct experiments on a Linux server with CentOS 7.9.2009. All experiments were run on an NVIDIA GeForce RTX 3090 GPU with CUDA 11.0. The CPU is Intel(R) Xeon(R) Gold 6226R CPU @2.90GHz and the total memory is 376GB. We use Python 3.9.12 and PyTorch 1.7.1.

Acknowledgements

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2021YFF0901003), and the National Natural Science Foundation of China (No. 61922073 and U20A20229).

References

Aktukmak, M.; Yilmaz, Y.; and Uysal, I. 2019. Quick and accurate attack detection in recommender systems through user attributes. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 348–352.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How To Backdoor Federated Learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, 2938–2948.

Baruch, G.; Baruch, M.; and Goldberg, Y. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *Advances in Neural Information Processing Systems*, volume 32, 8632–8642.

Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing Federated Learning through an Adversarial Lens. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 634–643.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, volume 30, 119–129.

Burke, R. D.; Mobasher, B.; Williams, C.; and Bhaumik, R. 2006. Classification Features for Attack Detection in Collaborative Recommender systems. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 542–547.

Cao, X.; Jia, J.; and Gong, N. Z. 2021. Provably Secure Federated Learning against Malicious Clients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8): 6885–6893.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. Z. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, 1605–1622.

Fang, M.; Yang, G.; Gong, N. Z.; and Liu, J. 2018. Poisoning Attacks to Graph-Based Recommender Systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*, 381–392.

Harper, F. M.; and Konstan, J. A. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4): 1–19.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International World Wide Web Conference*, 173–182.

Jin, B.; Lian, D.; Liu, Z.; Liu, Q.; Ma, J.; Xie, X.; and Chen, E. 2020. Sampling-Decomposable Generative Adversarial Recommender. In *Advances in Neural Information Processing Systems*, volume 33, 22629–22639.

Kang, W.; and McAuley, J. J. 2018. Self-Attentive Sequential Recommendation. In *IEEE International Conference on Data Mining*, 197–206.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Lam, S. K.; and Riedl, J. 2004. Shilling Recommender Systems for Fun and Profit. In *Proceedings of the 13th International World Wide Web Conference*, 393–402.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. In *Advances in Neural Information Processing Systems*, volume 29, 1885–1893.

Li, Z.; Wu, B.; Liu, Q.; Wu, L.; Zhao, H.; and Mei, T. 2020. Learning the Compositional Visual Coherence for Complementary Recommendations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 3536–3543.

Liang, D.; Charlin, L.; McInerney, J.; and Blei, D. M. 2016. Modeling User Exposure in Recommendation. In *Proceedings of the 25th International World Wide Web Conference*, 951–961.

Liang, F.; Pan, W.; and Ming, Z. 2021. FedRec++: Lossless Federated Recommendation with Explicit Feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 4224–4231.

Lin, G.; Liang, F.; Pan, W.; and Ming, Z. 2021. FedRec: Federated Recommendation With Explicit Feedback. *IEEE Intelligent Systems*, 36(5): 21–30.

Lloyd, S. P. 1982. Least Squares Quantization in PCM. *IEEE Transations on Information Theory*, 28(2): 129–136.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, 1273–1282.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748.

Parzen, E. 1962. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3): 1065–1076.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 452–461.

Rong, D.; He, Q.; and Chen, J. 2022. Poisoning Deep Learning Based Recommender Model in Federated Learning Scenarios. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2204–2210.

Rong, D.; Ye, S.; Zhao, R.; Yuen, H. N.; Chen, J.; and He, Q. 2022. FedRecAttack: Model Poisoning Attack to Federated Recommendation. In *IEEE 38th International Conference on Data Engineering*, 2643–2655.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1441–1450.

Sun, J.; Li, A.; DiValentin, L.; Hassanzadeh, A.; Chen, Y.; and Li, H. 2021. FL-WBC: Enhancing Robustness against Model Poisoning Attacks in Federated Learning from a Client Perspective. In *Advances in Neural Information Processing Systems*, volume 34, 12613–12624.

Tibshirani, R.; Walther, G.; and Hastie, T. 2001. Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2): 411–423.

Tolpegin, V.; Truex, S.; Gursoy, M. E.; and Liu, L. 2020. Data Poisoning Attacks Against Federated Learning Systems. In *European Symposium on Research in Computer Security*, 480–501.

Wang, H.; Sreenivasan, K.; Rajput, S.; Vishwakarma, H.; Agarwal, S.; Sohn, J.-y.; Lee, K.; and Papailiopoulos, D. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *Advances in Neural Information Processing Systems*, volume 33, 16070–16084.

Wang, Q.; Yao, J.; Gong, C.; Liu, T.; Gong, M.; Yang, H.; and Han, B. 2021. Learning with Group Noise. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11): 10192–10200.

Wang, T.; and Isola, P. 2020. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, 9929–9939.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 165–174.

Wu, C.; Lian, D.; Ge, Y.; Zhu, Z.; and Chen, E. 2021a. Triple Adversarial Learning for Influence based Poisoning Attack in Recommender Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1830–1840.

Wu, C.; Wu, F.; Qi, T.; Huang, Y.; and Xie, X. 2022. FedAttack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4164–4172.

Wu, C.; Wu, F.; Wang, X.; Huang, Y.; and Xie, X. 2021b. Fairness-aware News Recommendation with Decomposed Adversarial Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 4462–4469.

Wu, J.; Liu, Q.; Huang, Z.; Ning, Y.; Wang, H.; Chen, E.; Yi, J.; and Zhou, B. 2021c. Hierarchical Personalized Federated Learning for User Modeling. In *Proceedings of the 30th International World Wide Web Conference*, 957–968.

Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019. Session-Based Recommendation with Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1): 346–353.

Wu, Z.; Li, G.; Liu, Q.; Xu, G.; and Chen, E. 2018. Covering the Sensitive Subjects to Protect Personal Privacy in Personalized Recommendation. *IEEE Transactions on Services Computing*, 11(3): 493–506.

Yang, L.; Tan, B.; Zheng, V. W.; Chen, K.; and Yang, Q. 2020. Federated Recommendation Systems. In *Federated Learning*, 225–239. Springer.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 5650–5659.

Yu, R.; Liu, Q.; Ye, Y.; Cheng, M.; Chen, E.; and Ma, J. 2022. Collaborative List-and-Pairwise Filtering From Implicit Feedback. *IEEE Transactions on Knowledge and Data Engineering*, 34(6): 2667–2680.

Yu, W.; and Qin, Z. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 861–870.

Zhang, S.; Yin, H.; Chen, T.; Huang, Z.; Nguyen, Q. V. H.; and Cui, L. 2022a. PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1415–1423.

Zhang, Z.; Cao, X.; Jia, J.; and Gong, N. Z. 2022b. FLDetector: Defending Federated Learning Against Model Poisoning Attacks via Detecting Malicious Clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2545–2555.

Zhou, W.; Wen, J.; Xiong, Q.; Gao, M.; and Zeng, J. 2016. SVM-TIA a Shilling Attack Detection Method Based on SVM and Target Item Analysis in Recommender Systems. *Neurocomputing*, 210(C): 197–205.