Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training

Chenwang Wu
University of Science and Technology of China
wcw1996@mail.ustc.edu.cn

Defu Lian*
University of Science and Technology of China
liandefu@ustc.edu.cn

Yong Ge
University of Arizona
yongge@arizona.edu

Zhihao Zhu
University of Science and Technology of China
zzh98@mail.ustc.edu.cn

Enhong Chen
University of Science and Technology of China
cheneh@ustc.edu.cn

Senchao Yuan
University of Science and Technology of China
yuansc@mail.ustc.edu.cn

ABSTRACT

Recent studies have shown that recommender systems are vulnerable, and it is easy for attackers to inject well-designed malicious profiles into the system, leading to biased recommendations. We cannot deny these data's rationality, making it imperative to establish a robust recommender system. Adversarial training has been extensively studied for robust recommendations. However, traditional adversarial training adds small perturbations to the parameters (inputs), which do not comply with the poisoning mechanism in the recommender system. Thus for the practical models that are very good at learning existing data, it does not perform well. To address the above limitations, we propose adversarial poisoning training (APT). It simulates the poisoning process by injecting fake users (ERM users) who are dedicated to minimizing empirical risk to build a robust system. Besides, to generate ERM users, we explore an approximation approach to estimate each fake user's influence on the empirical risk. Although the strategy of "fighting fire with fire" seems counterintuitive, we theoretically prove that the proposed APT can boost the upper bound of poisoning robustness. Also, we deliver the first theoretical proof that adversarial training holds a positive effect on enhancing recommendation robustness. Through extensive experiments with five poisoning attacks on four real-world datasets, the results show that the robustness improvement of APT significantly outperforms baselines. It is worth mentioning that APT also improves model generalization in most cases.

CCS CONCEPTS

• **Security and privacy → Social network security and privacy**.

*Corresponding author.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462914

KEYWORDS

Robust Recommender Systems, Adversarial Training, Poisoning Attacks.

ACM Reference Format:
Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, Enhong Chen, and Senchao Yuan. 2021. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462914

1 INTRODUCTION

With the advent of the era of big data, the information we can obtain has exploded and far beyond our processability. As an important means to solve information overload, recommender systems have been widely used in many applications such as e-commerce [36], music service [40], and mobility prediction [25].

Unfortunately, due to the recommendation model's openness and collaboration, recent studies [22, 44] have shown that poisoning a few malicious profiles into the system is enough to make biased recommendations. Specifically, unscrupulous producers can easily poison fake profiles to promote their products or demote the competitors' ones (e.g., Amazon's online retailer attached a link to a sex manual next to the spiritual guide by constructing some fake ratings[1]). Thus the vulnerability alarms that the recommender system does not seem to be as robust as we expect.

The defense against poisoning attacks is mainly launched from two aspects [9]: (1) detecting and removing fraudsters in the data processing stage, and (2) building a robust recommendation model. Extant fraudster detections will inevitably misclassify some normal users and eliminate them in pursuit of a high detection rate, which is irrational for these excluded normal users. In this work, we are concerned about boosting the recommendation robustness even if malicious data are invariably available. Moreover, a few works [3, 34] have shown that adversarial robustness can provide critical insights and finds for the model interpretability. Therefore, from both the model's security or interpretability, the study of robust recommender systems is indispensable indeed.

Adversarial training [29] is widely recognized as an effective method to enhance recommendation robustness [6, 15, 38, 43]. Its

[1] https://www.cnet.com/news/amazon-blushes-over-sex-link-gaffe/

core idea is to add adversarial perturbations into model parameters (or inputs) to maximize the recommendation error, while the model performs empirical risk minimization for self-improvement and finally establishes a robust model in the adversarial game. However, it is not impeccable. Models in practical applications are highly irregular, resulting in learning algorithms that are very good at studying existing data. This strong coupling is directly manifested in the model's excellent defense against the specific attack used in adversarial training [28]. However, it is arduous for a poisoning attacker to directly destroy the parameters (or existing rating), significantly reducing practical performance.

Given the deficiency of existing adversarial training, we explore an adversarial poisoning training (APT) to improve the recommendation robustness by injecting fake data into the system, which we call "fight fire with fire". The motivation is that malicious attackers can devise bogus users to destroy recommendations, thus in theory, defenders can also design positive users to improve model robustness. To be specific, APT turns training into a min-min problem. The inner objective is minimized towards finding poisoning users (ERM users) who are committed to minimizing empirical risk, while the outer minimization aims to correct the model's parameters by standard supervised training on the mixed dataset. Clearly, unlike the zero-sum game of traditional adversarial training, APT performs a positive-sum game strategy in which the two players alternately improve model robustness in the way of alliance cooperation. The major challenge of APT is the inner ERM user generation. Here, we propose using the influence function [19] to estimate each poisoning user's effect on the empirical risk to find ERM users. In a theoretical Gaussian latent factor model, we prove that APT improves the upper bound of adversarial robustness against poisoning attacks. Besides, we provide the first theoretical proof of adversarial training's positive effect on building a robust recommender system, lacking in previous works [15, 38, 43].

Our contributions are outlined as follows.

- Given the Gaussian latent factor model, we give the upper bound of model robustness against poisoning attacks and derive the first proof that adversarial training can improve the poisoning robustness of recommender systems.
- We propose a novel robust training strategy, adversarial poisoning training, by poisoning fake ERM users into the recommender system for robust adversarial training. Besides, to generate ERM users, we explore an influence-based method to estimate fake users' impact on empirical risk.
- Through theoretical analysis, we prove the effectiveness of APT in improving model robustness, which provides a new avenue for adversarial training.
- We evaluate the robustness against five poisoning attacks on four real-world datasets. The results verify that APT markedly enhances the recommendation robustness while ensuring generalization.

2 PRELIMINARIES

2.1 Matrix Factorization for Recommender Systems

In this paper, we mainly focus on the matrix-factorization-based (MF-based) recommender system. Matrix factorization is widely known in recommender systems due to its simplicity and effectiveness[35]. The typical paradigm of MF is to decompose the user-item interaction matrix into the product of two low-dimensional latent matrices. Formally, suppose $r_i \in \mathbb{R}^m$ is the rating vector of user i, and $r_{i,j}$ denotes the rating given by user i to item j. There are n users and m items in total. Let $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{m\times d}$ are the latent factor matrix of users and items respectively, and d is the latent feature dimension, then the unseen rating $\hat{r}_{i,j}$ is predicted by $\hat{r}_{i,j} = U_i V_j^T$. Solving U and V can be transformed into minimizing the following problem:

$$\arg\min_{U,V} \sum_{(i,j)\in\Omega} (r_{i,j} - U_i V_j^T)^2 + \lambda(\|U\|_F^2 + \|V\|_F^2), \tag{1}$$

where Ω represents the set of all rated user-item pairs, $\|\cdot\|_F$ denotes the Frobenius norm, and λ is the regularization parameter.

2.2 Adversarial Training for Recommender

One explanation [17] of machine learning's vulnerability is that the model overfits normal data's robust features but ignores the incomprehensible non-robust features. However, these two features are equally important concerning supervised learning. At this time, adversarial training plays a crucial role in learning non-robust features. In recommendation tasks[15, 32, 38], adversarial training adds perturbations to the model parameters (or inputs) to force the recommender system to adapt to such noise and learn the perturbation's non-robust features, thereby enhancing the robustness facing adversarial attacks. The specific framework [9] is as follows:

$$\min_{\theta_R} \max_{\Delta, \|\Delta\|\le\epsilon} (\mathcal{L}(\mathcal{D}, \theta_R) + \lambda_{adv}\mathcal{L}(\mathcal{D}, \theta_R + \Delta)). \tag{2}$$

Δ is the added perturbation dedicated to destroying the recommendation, $\epsilon > 0$ limits the magnitude of perturbation, and λ_{adv} controls the impact of perturbations. $\mathcal{D}$, θ_R, and $\mathcal{L}$ are the dataset, parameters, and training loss of the recommender system, respectively. Briefly speaking, adversarial training performs a zero-sum game between the recommender and attack adversary and seeks a robust model in the game.

2.3 Threat Model

Attack goal. According to attackers' intents, they can design various attacks, including promotion attacks, demotion attacks, and availability attacks [22]. The promotion attack (demotion attack) typically aims to increase (decrease) the popularity of the target item (e.g., improve (decline) the recommendation ranking of an item on the ranking-based model [24]). For the availability attack, the attacker desires to maximize the recommendation error to render the model useless ultimately. We cannot delete the existing rating, so demotion attacks can be achieved by increasing the popularity of non-target items until the target item is not in the user's recommendation list [42], which in a sense is equivalent to promotion attacks. Besides, availability attacks are a hybrid of promotion attacks and demotion attacks in essence. For simplicity, we primarily concentrate on the defense against promotion attacks.

Attack capability. Theoretically, as long as the injected fake profile is large enough, any model is fragile. Apparently, the model robustness and attack capability cannot be decoupled. Also, injecting excessive fake users is arduous to operate and will inevitably

Algorithm 1: Adversarial Poisoning Training

Input: The epochs of training T, pre-training T_{pre}, and poisoning interval T_{inter}.

Randomly initialize the user set $\mathcal{D}^*$ defined in Definition 3.1.
for T_{pre} *epochs* **do**
  Do standard training on the dataset $\mathcal{D}$;
end
$\mathcal{D}' = \mathcal{D}$;
for $T - T_{pre}$ *epochs* **do**
  for *per* T_{inter} *epochs* **do**
    Calculate the influence vector $\mathcal{I}$ according to Eq. 5;
    for *each ERM user in* $\mathcal{D}^*$ **do**
      Select m^* items in Φ with probability $\frac{exp(-tI_i)}{\sum_{j\in\Phi} exp(-tI_j)}$ and rate the selected items with normal distribution $(\mu_i + r^+, \sigma_i)$ at random;
    end
    $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}^*$;
  end
  Do standard training on the dataset $\mathcal{D}'$;
end

produce different signatures from normal users, leading to being detected. Given the above considerations, we limit the attacker to register up to n' malicious users, and the upper limit of the number of ratings for each user is m'.

Attack knowledge. We consider the research under the white-box attack; that is, the attacker can master everything about the recommender system, including the algorithm used, specific parameter settings, and even the users' historical behavior. Although the challenge of obtaining knowledge makes this ideal attack less practical than black-box attacks, the focus of our work is the defense. The white-box attack provides the worst hypothesis for defense and the most rigorous test for evaluating recommendation robustness, which is beyond the black-box attack.

3 ADVERSARIAL POISONING TRAINING

Although adversarial training has been empirically proven to enhance adversarial robustness in recent works[15, 32, 38, 51], it still has deficiencies against poisoning attacks. Adversarial training usually uses extant attack methods to approximate perturbations, and it intuitively forces practical models that are very good at learning existing data to adapt to these attacks [28]. However, unlike evasion attacks, the poisoner cannot modify the existing ratings in recommender systems, let alone the model parameters. Obviously, adding perturbations to the parameters (inputs) in adversarial training cannot truly reflect poisoning attacks, making the defense performance often deficient in the face of practical attacks.

Now we consider an interesting question: an attacker can construct malicious users to maximize the recommendation error; can the defender also carefully design "favorable" fake data to improve the system's robustness? Given this motivation and the deficiency of existing adversarial training, we propose adversarial poisoning training (APT) by poisoning well-designed fake users to achieve the effect of fighting poison with poison. Next, we give a formal definition of the robust training method proposed in the paper.

Definition 3.1 (Adversarial poisoning training). Assume that $\mathcal{D} = \{r_1, \ldots, r_n\}$ is the dataset that may be contaminated. Besides, $\mathcal{D}^* = \{r_1^*, \ldots, r_{n^*}^*\}$ is a set of n^* fake users dedicated to minimizing the empirical risk, then the framework of adversarial poisoning training is defined as follows:

$$\min_{\theta_R} \min_{\mathcal{D}^*, |\mathcal{D}^*|=n^*} \mathcal{L}(\mathcal{D} \cup \mathcal{D}^*, \theta_R). \tag{3}$$

In contrast to traditional adversarial training described in Formula 2, the inner objective of APT is the minimization operation, and the perturbations used in APT are the poisoning users, which we call ERM users or defense users. Essentially, APT simulates the adversary's poisoning process, which can solve the pseudo-poisoning issue of adversarial training in the recommendation. The current major challenge is how to optimize the inner objective of Formula 3 to generate ERM users. Thanks to the influence function's application in decision-making prediction [19], we derive an approximate solver for generating ERM users by estimating each user's influence on empirical risk.

The influence function assesses the sensitivity of data changes on decision-making. To be specific, for a point z in the training set, we add a small perturbation δ to it to become $z' = z + \delta$, then the impact of this modification on the test point z_{test} is defined as

$$\mathcal{I}_{pert,loss}(z, \delta) := -\frac{1}{n} \nabla_\theta \mathcal{L}(z_{test}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_z \nabla_\theta \mathcal{L}(z, \hat{\theta}) \delta,$$

where $\mathcal{L}$ is the training loss of Formula 1, and H_θ is the Hessian matrix of training loss. The full proof can be found in [19].

If we use empirical risk loss $\mathcal{L}_{ER}$ instead of the loss of z_{test}, we can measure the influence of data modifications on empirical risk. Besides, considering poisoning a user z' to the system, it is equivalent to copying an existing user z in the dataset first, and then adding a perturbation $\delta = z' - z$ to the newly copied z, so the influence of poisoning z' can be considered as $\mathcal{I}_{poison}(z') = \mathcal{I}_{copy}(z) + \mathcal{I}_{pert}(z, \delta)$. When z is fixed, the influence $\mathcal{I}_{copy}(z)$ is the same for any fake user and can be ignored. Furthermore, owing to data sparsity in the recommender system, suppose the minimum rate is 0 and $z = \arg\min_{r, r\in\mathcal{D}} \|r\|$, then z will be close to zero vector. Based on the above observations, we can approximate the influence of each poisoning user z' on empirical risk as follows:

$$I_{poison}(z') \approx \mathcal{I}^T z', \tag{4}$$

where the influence vector $\mathcal{I}$ is

$$\mathcal{I} = -\frac{1}{n} \nabla_{\theta_R} \mathcal{L}_{ER}(\hat{\theta}_R)^T H_{\hat{\theta}_R}^{-1} \nabla_z \nabla_{\theta_R} \mathcal{L}(z, \hat{\theta}_R). \tag{5}$$

Here we define $\mathcal{L}_{ER}(\hat{\theta}_R) = \sum_{r_{i,j}} \frac{1}{1+exp(-(r_{i,j}-\hat{r}_{i,j})^2)}$ instead of training loss, because it is more sensitive to minor changes in the rating and can better perceive the abnormal impact of the attack.

The i-th element in the influence vector $I \in \mathbb{R}^m$ reflects item i of poisoning user z' on the empirical risk. The smaller the influence, the more beneficial selecting this item to minimize the empirical risk. Based on this property, we contribute an influence-based approach to generate ERM users, which can be decomposed into two steps: (1) selecting m^* items to be rated. Let the item set $\Phi = \{i | \mathcal{I}_i < 0\}$. It

is easy to know from Eq. 4 that rating the item in the complement $\overline{\Phi}$ is bound to increase influence. Consequently, we only select items in Φ, and the probability that we choose item $i \in \Phi$ is $\frac{exp(-tI_i)}{\sum_{j\in\Phi} exp(-tI_j)}$, where t is used to scale the influence. (2) Rating each item selected in step 1. Intuitively, poisoning robustness requires more real users (we will prove it in Theorem 4.3), so we are inclined to rate with the normal distribution of existing ratings. However, Eq. 4 inspires us that the injected user z' has a small influence when the selected item's rate reaches the maximum (the selected item $i \in \Phi$), exposing its fake features. To tackle the contradiction, we make a compromise to randomly rate each selected item from the distribution $\mathcal{N}(\mu_i + r^+, \sigma_i)$. $\mathcal{N}(\mu_i, \sigma_i)$ is the rating distribution of item i, Adding $r^+ > 0$ indicates that minimizing the influence prefer to increase the rates.

The algorithm of APT is shown in Alg. 1. We first do T_{pre} rounds of standard pre-training. Then we generate new ERM users using the approach proposed in the previous paragraph and combine them with the original data for standard training. In addition, considering the sufficiency and efficiency of model learning, we generate new ERM users every T_{inter} rounds and use implicit Hessian-vector products [19] to approximate $\nabla_{\theta_R}\mathcal{L}_{ER}(\hat{\theta}_R)^T H^{-1}_{\hat{\theta}_R}$ in Eq. 5 efficiently.

4 THEORETICAL RESULTS

In this section, we theoretically analyze the poisoning robustness of the recommender system. Precisely, we follow the latent factor model, which using latent features to associate users with products for recommendations. For simplicity, we only focus on a single item's recommendation (see appendix for all proofs). Note that these analysis results can be easily expanded to multiple items.

Definition 4.1 (Gaussian model). Let $\overline{u} \in \mathbb{R}^d$ be the mean vector of user embedding and let $\sigma > 0$ be the variance parameter. Then the $(\overline{u}, \sigma)$-Gaussian model is defined by the following distribution over $(u, r) \in \mathbb{R}^d \times \{\pm 1\}$: First, draw a rating $r \in \{\pm 1\}$ uniformly at random, then sample the user embedding $u \in \mathbb{R}^d$ from $\mathcal{N}(r \cdot \overline{u}, \sigma^2 I)$.

Here we limit the rating r to $\{-1, 1\}$, which denotes whether the model recommends the item (r =1) or not ($r = -1$) to the user.

Definition 4.2 (Gaussian recommender). Let $(u_1, r_1), \ldots, (u_n, r_n) \in \mathbb{R}^d \times \{\pm 1\}$ be drawn i.i.d. from a $(\overline{u}, \sigma)$-Gaussian model. Let $\hat{v} \in \mathbb{R}^d$ be the item embedding estimator: $\hat{v} = \sum_{i=1}^n r_i u_i$. Then the recommendation model $f_{\hat{v}} : \mathbb{R}^d \to \{\pm 1\}$ is defined as $f_{\hat{v}} = sign(\langle \hat{v}, u \rangle)$.

Based on the above two definitions, we can define the recommendation error $\beta = \mathbb{P}_{(u,r)}[f_{\hat{v}}(u) \neq r]$. Next, we give the upper bounds of robustness against poisoning attacks under no defense, adversarial training, and adversarial poisoning training.

Theorem 4.3. *Let $(u_1, r_1), \ldots, (u_n, r_n) \in \mathbb{R}^d \times \{\pm 1\}$ be drawn i.i.d. from $(\overline{u}, \sigma)$ -Gaussian model, and $(u'_1, r'_1), \ldots, (u'_{n'}, r'_{n'}) \in \mathbb{R}^d \times \{\pm 1\}$ are poisoning data. Suppose $\hat{v} = \frac{1}{n}\sum_{i=1}^n r_i u_i$, $\|\hat{v}\|_2 = \alpha$, and limit $\max \|u\|_\infty = \gamma$. If $\langle \hat{v}, \overline{u} \rangle - n' d\gamma^2/n \geq 0$, then the recommendation model $f_{\hat{v}}$ has robust recommendation error at most β with a probability of at least $1 - 2\exp(-\frac{d}{8(\sigma^2+1)})$ if*

$$n' \leq \frac{n}{d\gamma^2} \cdot \left(\frac{\alpha\sqrt{d}(2\sqrt{n}-1)}{2\sqrt{n}+4\sigma} - \alpha\sigma\sqrt{2\log 1/\beta} \right).$$

Table 1: Statistics of datasets

Dataset	users	items	ratings	sparsity
FilmTrust	796	2011	30880	98.07%
ML-100K	943	1682	100000	93.70%
ML-1M	6040	3706	1000209	95.53%
Yelp	14575	25602	569949	99.85%

Given the recommendation error bound β, Theorem 4.3 gives the upper bound of the poisoning users tolerated by the standard training model. If the upper bound exists (the right side of the inequality is greater than 0), it is easy to find that the larger the number of normal users n, the more the number of acceptable poisoning users n'. This means that recommendation robustness needs more real users. Note that similar findings [33] were found in evasion attacks.

Theorem 4.4. *Assume a $(\bar{u}, \sigma)$ -Gaussian model uses adversarial training under the assumptions of Theorem 4.3, and ϵ is adversarial perturbation level. If $\min_{i=1,\ldots,n} \|u_i\| = \tau$, then the recommendation model $f_{\hat{v}_{AT}}$ has robust recommendation error at most β with probability at least $1 - 2\exp\left(-\frac{d}{8(\sigma^2+1)}\right)$ if*

$$n' \leq \frac{n}{d\gamma^2 - \epsilon\tau} \cdot \left(\frac{\alpha\sqrt{d}(2\sqrt{n}-1)}{2\sqrt{n}+4\sigma} - \alpha\sigma\sqrt{2\log 1/\beta} + \epsilon\tau \right).$$

For the recommendation model that uses adversarial training, Theorem 4.4 gives the upper bound of poisoning robustness. Compared with Theorem 4.3, it is easy to find that adversarial training improves model robustness. In particular, to our best knowledge, this is the first theoretical proof that adversarial training can improve the poisoning robustness in recommender systems.

Theorem 4.5. *Assume a $(\bar{u}, \sigma)$ -Gaussian model under the assumptions of Theorem 4.3. If $(u_1^*, r_1^*), \ldots, (u_{n^*}^*, r_{n^*}^*) \in \mathbb{R}^d \times \{\pm 1\}$ are ERM users dedicated to minimizing empirical risk. Suppose $\min_{i=1,\ldots,n} \|u_i\| = \tau$, then the recommendation model $f_{\hat{v}}$ has robust recommendation error at most β with probability at least $1 - 2\exp\left(-\frac{d}{8(\sigma^2+1)}\right)$ if*

$$n' \leq \frac{1}{d\gamma^2} \cdot \left(\frac{n\alpha\sqrt{d}(2\sqrt{n}-1)}{2\sqrt{n}+4\sigma} - n\alpha\sigma\sqrt{2\log 1/\beta} + n^*\gamma\tau \right).$$

Similarly, comparing Theorem 4.5 with Theorem 4.3, APT can also provably boost the upper bound of tolerable poisoning data, which provides a new idea for adversarial training.

Moreover, comparing the upper bound of Theorem 4.4 with Theorem 4.5, we can conclude that when $n^* > (n + n')\epsilon/\gamma$ (the proportion of injected ERM users is larger than the one of parameter perturbation), APT is more robust than adversarial training.

5 EXPERIMENTS

5.1 Experiments Settings

5.1.1 Datasets. We use four real-world datasets commonly used in the security studies [8, 13, 16, 26, 37, 43] of the recommender

system, including FilmTrust[2], ML-100K[3] (MovieLens-100K), ML-1M[4] (MovieLens-1M), and Yelp[4]. ML-100K includes 943 users who have rated 1,682 movies for 100,000 ratings. ML-1M comprises 6,040 users who have rated 3,706 movies about one million times. For FilmTrust, the same pretreatment as [26] is used to filter cold-start users who seriously affect the recommender system (the rating number is less than 15), leaving 796 users with trust ratings for 2011 movies. Yelp is a common dataset for business, and we use the subset create by [16] which also filters users with ratings less than 15. Table 1 lists the detailed statistics of these datasets. All ratings are from 1 to 5, and we normalized them to $[0, 1]$ in the experiments. For each dataset, we randomly select a positive sample from each user for testing, and the rest are used as the training set and verification set in a 9:1 ratio.

5.1.2 Attack Approaches. We use the following poisoning attack for robustness validation:

Random Attack [21]: This attack assigns the maximum rating to the target item and rates selected items according to the normal distribution of all user ratings at random.

Average Attack [21]: The only difference from Random Attack is that the non-target selected item i is randomly rated with the normal rating distribution of item i.

AUSH Attack [26]: This attack uses GAN to generate each fake user to carry out attacks imperceptibly and assigns the highest rating to the target item.

PGA Attack [22]: This attack builds an attack objective and uses SGD to update the poisoned user's ratings to optimize the objective. Finally, the first m' items with the largest ratings are selected as the fake user's filler items.

TNA Attack [12]: This attack selects a subset of the most influential users in the dataset and optimizes the rating gap between the target item and top-K items in the user subset. Here we use $\mathcal{S}$-TNA.

5.1.3 Baselines. We compare the proposed APT with the following robust algorithms:

Adversarial Training(AT) [16]: In each training step, it first uses SGD to optimize the inner objective to generate small perturbations and adds them to the parameters, and then performs training.

PCMF [2]: Unlike traditional matrix factorization to learn two matrices U and V, this method only optimizes one user-item interaction matrix $U \in \mathcal{R}^{n \times m}$, and then estimates the complete matrix by $UU^T Y$, where $Y \in \mathcal{R}^{n \times m}$ is the training matrix.

APT-rand: The non-influential version of APT. It generates "ERM" users at random rather than based on influence. The primary purpose of introducing this method is to verify the effectiveness of the influence function.

Remark: The experiments are concerned with the MF-based model and evaluate the defense method's effectiveness in improving model robustness. Robust algorithms [7, 49] strongly coupled with non-MF models cannot be compared fairly, so they are omitted here.

5.1.4 Evaluation Protocol. We first use HR@K (Hit Ratio), which calculates the proportion of test items that appear in the user's top-K recommendation list. Besides, we introduce Rank shift, which is defined as the difference between the specific item's rank before and after the attack. The closer the value is to 0, the smaller the impact of the attack. Lastly, we define robustness improvement $RI = 1 - (HR_{defense} - HR_{origin})/(HR_{attack} - HR_{origin})$. The closer the value is to 1, the better the robustness. We report the average results of 30 independent repeated experiments and perform paired t-test to judge the statistical significance when necessary.

5.1.5 Parameters Setting. We concern the MF-based recommendation model described in Section 2.1, and we set the latent factor dimension d to 64. The model is trained for 25 epochs, and the Adagrad optimizer is used for training. In APT, we inject 5% of ERM users without special mention, and set ϵ in AT to 0.03 for a fair comparison, which is about 5% of the parameter weight. In FilmTrust, ML-100K, ML-1M, and Yelp, t is set to 40, 20, 20, 10, T_{pre} is set to 5, 15, 15, 10, r^+ is set to 2, 1, 1, 1, the number of ERM user's rated items m^* is set to 100, 400, 700, 2500, and ERM users are generated every 3, 2, 2, and 2 epochs, respectively.

In FilmTrust, ML-100K, and ML-1M, we use HR@50[5], while HR@1000 is used in Yelp. This is because Yelp is difficult to attack, and setting a large K helps make apparent comparisons between defense methods. A similar treatment is reflected in the attack size, set to 5% in Yelp and 3% in other datasets. For the target items of attacks, we learn two types of items: (1) random items randomly selected from all items, and (2) unpopular items randomly selected from items with the number of rates less than 5. In each attack, we set the number of target items to 5 and set the number of filler items m' to the average number of ratings per user. The source code of APT is available at https://github.com/Daftstone/APT.

5.2 Performance Comparison

In this section, we compare the robustness and generalization of the model configured with APT and other defense methods.

5.2.1 Robustness. We evaluate the hit ratio of target items in attack and defense, as shown in Table 2. The Origin denotes the unperturbed model, and the Attack represents the perturbed model with no defense. First, the target item's HR is significantly promoted after the attack, which means that the recommender system is vulnerable. Second, these defense methods are positive in weakening the attack's damage with respect to HR in most cases. Third, the proposed APT achieves remarkable defense results, almost close to the unperturbed model performance. On average, we reduce the impact of attacks on random items by over 89% and unpopular items by over 87%, which effortlessly outperforms baselines. Finally, we notice that the performance of APT against AUSH is slightly inferior when compared with the defense against other attacks, and a similar phenomenon also appears in adversarial training. We suspect that AUSH's poisoning data based on GAN are closer to the real data, making it more formidable for adversarial training to discover and learn adversary data's non-robust features.

Moreover, Fig. 1 shows the Rank shift distribution of target items (unpopular items) under the TNA attack. Consistent with the findings in Table 2, the attack significantly promotes the target item's rank among all users. After using adversarial training, the rank

[2]https://www.librec.net/datasets/flmtrust.zip
[3]https://grouplens.org/datasets/movielens/
[4]https://www.yelp.com/dataset/download

[5]Collaborative filtering is often used for candidate selection in practical recommendations, so it is more instructive to select a larger K to ensure a high recall [15].

Table 2: The performance in target items (robustness). *, **, and * indicate that the improvements over the best results of baselines are statistically significant for $p < 0.05$, $p < 0.01$, and $p < 0.001$, respectively.**

Dataset	Attack	Random items						Unpopular items					
		origin	Attack	PCMF	AT	APT-rand	APT	origin	Attack	PCMF	AT	APT-rand	APT
FilmTrust (HR@50)	Average	0.1290	0.3126	0.3653	**0.1308**	0.3348	0.1119	0.0000	0.0320	0.3386	0.0054	0.0298	**0.0041**
	Random	0.1290	0.3735	0.3650	0.1695	0.4303	**0.1233****	0.0000	0.0424	0.3373	0.0063	0.0360	**0.0044****
	AUSH	0.1290	0.6235	0.2657	0.3092	0.6567	**0.1564*****	0.0000	0.3721	0.2433	0.1015	0.2799	**0.0208*****
	PGA	0.1290	0.4814	0.3495	0.2728	0.5257	**0.1307*****	0.0000	0.1635	0.3340	0.0627	0.1201	**0.0052*****
	TNA	0.1290	0.8855	0.3306	0.8000	0.8315	**0.2129*****	0.0000	0.6689	0.3198	0.5751	0.4559	**0.0152*****
ML-100K (HR@50)	Average	0.0460	0.4787	0.0725	0.3348	0.1981	**0.0554***	0.0000	0.9292	0.0470	0.8572	0.0773	**0.0009*****
	Random	0.0460	0.2366	0.0746	0.1641	0.1272	**0.0394*****	0.0000	0.5199	0.0500	0.4265	0.0165	**0.0007*****
	AUSH	0.0460	0.5567	**0.0645**	0.4160	0.2367	0.0761	0.0000	0.9773	0.0363	0.9112	0.1195	**0.0015*****
	PGA	0.0460	0.3131	0.0564	0.2410	0.1274	**0.0394****	0.0000	0.5576	0.0425	0.5402	0.0107	**0.0002*****
	TNA	0.0460	0.5774	**0.0620**	0.4012	0.2356	0.0871	0.0000	0.9897	0.0382	0.9336	0.1399	**0.0019*****
ML-1M (HR@50)	Average	0.0001	0.4241	0.0232	0.1791	0.1330	**0.0224**	0.0000	0.9682	0.0403	0.9297	0.2102	**0.0242*****
	Random	0.0001	0.1028	**0.0226**	0.0472	0.0316	0.0302	0.0000	0.8350	0.0390	0.5625	0.0219	**0.0064*****
	AUSH	0.0001	0.4600	**0.0340**	0.1163	0.1285	0.0537	0.0000	0.9909	0.0453	0.9581	0.2747	**0.0980*****
	PGA	0.0001	0.8050	0.0156	0.0381	0.0161	**0.0010*****	0.0000	0.4837	0.0373	0.3383	0.0042	**0.0012*****
	TNA	0.0001	0.4572	**0.0185**	0.2172	0.1508	0.0323	0.0000	0.9703	**0.0188**	0.9274	0.2281	0.0192
Yelp (HR@1000)	Average	0.0143	0.0496	0.7913	0.0325	0.0175	**0.0155***	0.0000	0.0427	0.7971	0.0247	0.0126	**0.0076***
	Random	0.0143	0.0402	0.7890	0.0347	0.0182	**0.0160***	0.0000	0.0407	0.8018	0.0245	0.0182	**0.0098****
	AUSH	0.0143	0.1127	0.5536	0.0650	**0.0616**	0.1106	0.0000	0.1207	0.7921	0.0759	**0.0724**	0.1023
	PGA	0.0143	0.0257	0.7437	0.0167	0.0189	**0.0142**	0.0000	0.0251	0.7310	0.0109	0.0137	**0.0058**
	TNA	0.0143	0.0876	0.8717	0.0603	**0.0381**	0.0443	0.0000	0.0663	0.8612	0.0592	0.0386	**0.0355***

change caused by the attack can be eased, but it is only slight. On the contrary, APT impels the distribution of rank shift obviously tends to 0, which means that applying APT can produce more stable recommendations in a disturbed environment.

In conclusion, these results confirm the positive effect of APT in boosting recommendation robustness against poisoning attacks.

5.2.2 Generalization. It is meaningless to improve the robustness at the cost of apparently sacrificing the generalization of standard recommendations. Table 3 records the HR of various defense methods in the holdout test set. We can find that the baseline PCMF is only effective on Movielens datasets. Faced with more sparse FilmTrust and Yelp, the performance is significantly decreased, which indicates that the expression ability of a single user-item interaction matrix is still insufficient. Besides, AT's performance is dramatically reduced, even by 0.1 to HR, whereas the proposed APT's generalization accuracy only fluctuates slightly. More encouragingly, APT surprisingly improves the generalization on ML-100K, ML-1M, and Yelp, and the improvement is above 0.01 in terms of HR. These results confirm that APT effectively guarantees the model's generalization while performing high-quality defense. Although AT and APT are both based on adversarial training, the generalization is quite different. We reasonably suspect that performing the zero-sum game in AT improves robustness at the massive price of sacrificing generalization, in line with the finding in [47].

5.3 Performance under Different Attack Sizes

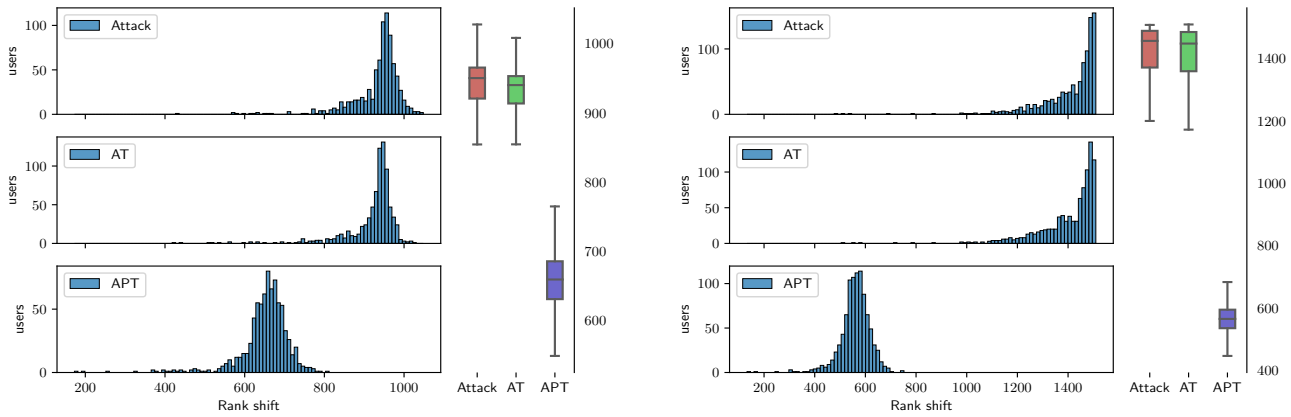
We conduct the robustness improvement test of APT under different attack sizes, as illustrated in Fig. 2. On the one hand, the overall defense performance of APT remains at a high level, although there will be individual cases where it performs poorly (e.g., injecting 2% of fake users on Yelp by AUSH). On the other hand, as the attack intensity increases, the robustness against attacks will also be attenuated. Especially in Yelp, RI is reduced by up to 20%. This signifies that high-intensity attacks bring severe defense challenges.

5.4 Effectiveness of the Influence

In this paper, we generate ERM users based on evaluating the empirical risk by influence function. To study its effectiveness, we define APT-rand as a non-influential version of APT. The specific comparison is shown in the last two columns of Table 2 and Table 3. From Table 2, we observe that although APT-rand can mitigate the harm of attacks, its performance improvement is tiny compared to APT. On average, the robustness of APT is increased by 22 times that of APT-rand concerning HR. Besides, Table 3 shows that APT-rand reduces model generalization, even is the worst among all the compared approaches in ML-1M. We reasonably believe that the injected users derived by APT-rand are irregular, and these random data destroy the original distribution and increases the difficulty of model training. These comparison results highlight that the use of the influence function in APT apparently enhances the robustness and generalization of the recommender system.

5.5 Sensitivity w.r.t the Number of ERM Users

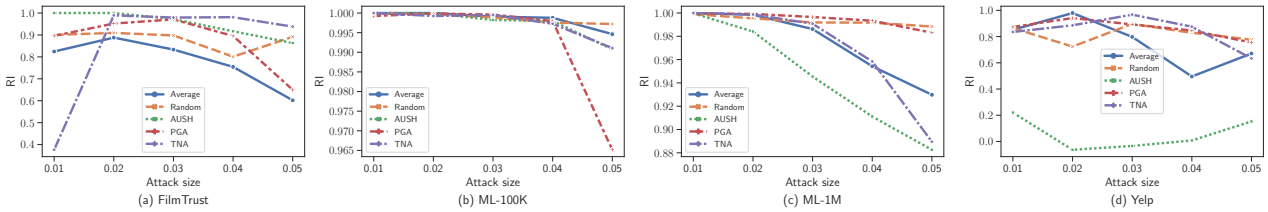
We test the impact of injected ERM users from 1% to 20% on the unpopular item's defense performance, and the results are illustrated in Fig. 3. 0.00 on the abscissa denotes that no defense method has been employed. The Origin is used as the benchmark, which represents a clean model without any attacks. First, we are delighted to find that using only 1% of ERM users can drastically weaken the attacks' impact, particularly in the ML-100K and ML-1M. Next, as the number of ERM users injected increases, the defense performance is gradually improving. Most of them reach saturation at 5% injection, and the HR of target items is close to the benchmark, which means that the attack is invalid. Finally, there are the same findings as in Section 5.2.1 that the defensive performance against the more real AUSH fake users on Yelp is not good. It inspires subsequent



(a) FilmTrust (b) ML-100K

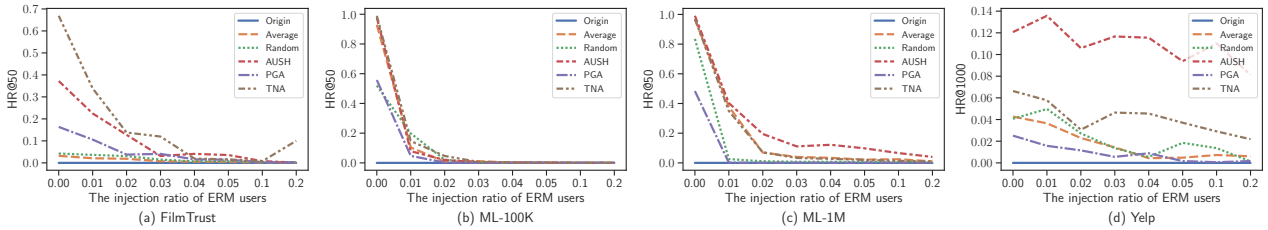
(c) ML-1M (d) Yelp

Figure 1: The distribution of rank shift. In each dataset, top: TNA attack; middle: adversarial training on TNA attack; bottom: adversarial poisoning training on TNA attack; boxplot: statistical distributions of rank shift. The closer the rank shift is to 0, the smaller the damage of the attack.



(a) FilmTrust (b) ML-100K (c) ML-1M (d) Yelp

Figure 2: Robustness improvement (RI) under different attack sizes when attacking unpopular items.



(a) FilmTrust (b) ML-100K (c) ML-1M (d) Yelp

Figure 3: The impact of the number of ERM users on defense performance.

Table 3: The performance in test set (generalization). *, **, and * indicate that the improvements over the unperturbed model are statistically significant for $p < 0.05$, $p < 0.01$, and $p < 0.001$, respectively.**

Dataset	Attack	random items						unpopular items					
		origin	Attack	PCMF	AT	APT-rand	APT	origin	Attack	PCMF	AT	APT-rand	APT
FilmTrust (HR@50)	Average	0.8539	0.8550	0.8421	0.7631	**0.8518**	0.8377	0.8539	0.8554	0.8417	0.7485	**0.8504**	0.8363
	Random	0.8539	0.8557	0.8413	0.7687	**0.8515**	0.8394	0.8539	0.8565	0.8425	0.7528	**0.8520**	0.8383
	AUSH	0.8539	0.8556	0.8443	0.7476	**0.8512**	0.8334	0.8539	0.8543	0.8451	0.7507	**0.8497**	0.8352
	PGA	0.8539	0.8518	0.8431	0.7340	**0.8508**	0.8338	0.8539	0.8537	0.8436	0.7480	**0.8498**	0.8343
	TNA	0.8539	0.8441	**0.8467**	0.7460	0.8464	0.8329	0.8539	0.8392	**0.8433**	0.7387	0.8396	0.8319
ML-100K (HR@50)	Average	0.2106	0.5080	**0.3552**	0.1863	0.1920	0.2201***	0.2106	0.2054	**0.3567**	0.1787	0.1958	0.2201**
	Random	0.2106	0.2180	**0.3507**	0.1940	0.2056	0.2251**	0.2106	0.5123	**0.3579**	0.1924	0.2017	0.2275***
	AUSH	0.2106	0.2097	**0.3550**	0.1816	0.1917	0.2168	0.2106	0.2002	**0.3514**	0.1840	0.1938	0.2196**
	PGA	0.2106	0.0208	**0.3528**	0.1897	0.2009	0.2302***	0.2106	0.2040	**0.3524**	0.1830	0.1987	0.2194*
	TNA	0.2106	0.2036	**0.3508**	0.1830	0.1974	0.2222**	0.2106	0.1986	**0.3535**	0.1815	0.2000	0.2209***
ML-1M (HR@50)	Average	0.0933	0.0903	0.0959	0.0775	0.0706	**0.1013*****	0.0933	0.0848	0.0963	0.0697	0.0700	**0.1008*****
	Random	0.0933	0.0990	0.0960	0.0938	0.0759	**0.1052*****	0.0933	0.0933	0.0969	0.0899	0.0762	**0.1045*****
	AUSH	0.0933	0.0909	0.0996	0.0784	0.0713	**0.1016*****	0.0933	0.0860	0.0984	0.0753	0.0705	**0.1005*****
	PGA	0.0933	0.0942	0.0964	0.0853	0.0738	**0.1016*****	0.0933	0.0887	0.0953	0.0793	0.0742	**0.1006*****
	TNA	0.0933	0.0874	0.0953	0.0840	0.0712	**0.1018*****	0.0933	0.0849	0.0976	0.0712	0.0703	**0.1009*****
Yelp (HR@1000)	Average	0.4093	0.4050	0.2997	0.3047	0.3697	**0.4239*****	0.4093	0.4000	0.2998	0.3057	0.3698	**0.4241*****
	Random	0.4093	0.4022	0.2992	0.2931	0.3711	**0.4221*****	0.4093	0.3985	0.2970	0.2946	0.3650	**0.4216*****
	AUSH	0.4093	0.3954	0.3121	0.2859	0.3576	**0.4197*****	0.4093	0.3965	0.3146	0.2871	0.3570	**0.4216*****
	PGA	0.4093	0.3987	0.3087	0.2922	0.3654	**0.4203*****	0.4093	0.3998	0.3093	0.2956	0.3685	**0.4257*****
	TNA	0.4093	0.3745	0.2936	0.2786	0.3587	**0.4161*****	0.4093	0.3723	0.2894	0.2730	0.3583	**0.4180*****

research on the recommender system's security should draw more attention to the poisoning data's imperceptibility.

6 RELATED WORK

6.1 Security of Recommender Systems

Recent studies have shown that recommender systems are vulnerable [9, 22, 35]. Attackers can easily introduce bias into the recommender system by injecting some fake profiles. Earlier attacks [4, 44] manually designed malicious profiles based on simple heuristics, so such model-independent attack's performance was limited. With the development of optimization algorithms, attacks based on specific models have received increasing attention[12, 13, 22, 39, 42, 46]. The training of model-based recommendation algorithms usually uses backpropagation [14, 16], so perturbations were added along the gradient direction to perform the attack [12, 13, 22, 39]. Inspired by the GAN's application [18] in the recommendation, some work [8, 26] uses GAN to generate real-like fake ratings to bypass the detection. Besides, the items' ratings can naturally be modeled as actions, which encourages researchers to use reinforcement learning to generate poisoning data [11, 37, 46].

6.2 Defense against Poisoning Attacks

The widespread applications of recommender systems make their security issues increasingly prominent. The extant defense can be divided into proactive robust model construction and reactive attack detection [9], which will be listed below.

Noting that the variance of noise in collaborative filtering is generally non-Gaussian and heteroscedastic, Student-t prior was used for latent features to improve recommendation performance [20]. Traditional matrix factorization learns two matrices U and V. Bampis et al. [2] proposed that using a user-item matrix with fewer parameters can produce stable recommendations. Inspired by network distillation, Chen et al. [7] used knowledge distillation to learn a student model that is more robust to perturbations. Liu et al. [27] calculated the robust bound of the FM model by relaxation and maximized the bound to enhance robustness. More recently, many works [6, 10, 15, 23, 32, 38, 43, 51] have focused on adversarial training. Assuming that each instance may be the target of attacks [9], adversarial training adds perturbations to the inputs or model parameters that force the model to learn fragile perturbations.

Traditional detection [4, 31] uses statistical methods to extract the user-rating matrix's potential features and uses machine learning algorithms, e.g., SVM and clustering, to identify malicious profiles. Besides, considering that fake users' diversity leads to the lack of prior knowledge, unsupervised learning [30, 45, 50] and semi-supervised learning [5, 41] were used to detect attacks. Since attacks are often completed instantaneously, using time as additional information as a detection consideration can improve the detection performance [1, 48, 52]. Kumar et al. [52] combined fairness, reliability, and goodness to qualify users. A recent work [49] combined robust optimization with fraudster detection to enhance robustness against attacks.

7 CONCLUSION

This paper proposes adversarial poisoning training (APT), a new adversarial training method to resist poisoning attacks in the recommender system. Specifically, we utilize the influence function to find ERM users who minimize the empirical risk, and then the model minimizes the training loss to learn these fake users. The ERM users and model collaboratively improve the model robustness in such dynamic training. Through a simple statistical model, we prove that the positive-sum game strategy of APT increases the upper bound of poisoning users tolerated by the model. Also, we derive the first theoretical proof that adversarial training can improve recommendation robustness. Finally, extensive experiments show that the proposed APT improves the robustness by over 88% on average, which is superior to the highly competitive robustness methods. Since our method only interacts with the model when calculating the ERM user's influence, theoretically, our method can

be applied as long as the recommendation model is second-order differentiable. In the future, we plan to extend APT to more models. In addition, its applications in non-recommendation fields are also worth studying.

ACKNOWLEDGMENTS

The work was supported by grants from the National Key R&D Program of China under Grant No. 2020AAA0103800, the National Natural Science Foundation of China (No. 61976198 and 62022077), JD AI Research and the Fundamental Research Funds for the Central Universities (No. WK2150110017).

A DETAILED PROOFS

First, we provide two related lemmas provided from [33].

Lemma A.1 ([33]). *Let $z \in \mathbb{R}^d$ be drawn from a spherical Gaussian, i.e., $z \sim \mathcal{N}_d\left(\mu, \sigma^2 I\right)$ where $\mu \in \mathbb{R}^d$ and $\sigma > 0$. Moreover, let $w \in \mathbb{R}^d$ be an arbitrary unit vector with $\langle w, \mu\rangle \geq \rho$ where $\rho \geq 0$. Then we have*

$$\mathbb{P}[\langle w, z\rangle \leq \rho] \leq \exp\left(-\frac{(\langle w, \mu\rangle - \rho)^2}{2\sigma^2}\right).$$

Lemma A.2 ([33]). *Let $z_1, \ldots, z_n \in \mathbb{R}^d$ be drawn i.i.d. from a spherical Gaussian with mean norm $\sqrt{d}$, i.e., $z_i \sim \mathcal{N}_d\left(\mu, \sigma^2 I\right)$ where $\mu \in \mathbb{R}^d$, $\|\mu\|_2 = \sqrt{d}$, and $\sigma > 0$. Let $\bar{z} \in \mathbb{R}^d$ be the sample mean vector $\bar{z} = \frac{1}{n}\sum_{i=1}^n z_i$ and let $\widehat{w} \in \mathbb{R}^d$ be the unit vector in the direction of $\bar{z}$, i.e., $\widehat{w} = \bar{z}/\|\bar{z}\|_2$. Then we have*

$$\mathbb{P}\left[\langle \widehat{w}, \mu\rangle \leq \frac{2\sqrt{n}-1}{2\sqrt{n}+4\sigma}\sqrt{d}\right] \leq 2\exp\left(-\frac{d}{8\left(\sigma^2+1\right)}\right).$$

Proof of Theorem 4.3. Assume that the set of users embeddings is $\mathcal{B}(u) = \{u \in \mathbb{R}^d | \, \|u\| \leq \gamma\}$, and $\mathcal{U}'$ is n' poisoning users' embedding selected from $\mathcal{B}(u)$. $\hat{v}$ and $\hat{v}'$ respectively represent the item embedding before and after poisoning. For the recommendation model defined in Definition 4.2, we need to bound the recommendation error β

$$\begin{aligned}
&\mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : f_{\hat{v}'}(u) \neq r] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \langle ru, \hat{v}'\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \langle ru, \frac{1}{n+n'}(n\hat{v} + \sum\nolimits_{(u',r')\in U'} r'u')\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle + \min \sum\nolimits_{(u',r')\in\mathcal{U}'} rr'uu' \leq 0] \\
&\leq \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle - n'd\gamma^2 \leq 0] = \mathop{\mathbb{P}}_{(u,r)}[\langle ru, \frac{\hat{v}}{\|\hat{v}\|_2}\rangle \leq \frac{n'd\gamma^2}{n\alpha}].
\end{aligned}$$

The inequality in the last line is based on $\max \|u\|_\infty = \gamma$ and $r \in \{\pm 1\}$. Invoking Lemma A.1 with $\mu = \overline{u}$ and $\rho = \frac{n'd}{n\alpha}\gamma^2$ to get

$$\mathop{\mathbb{P}}_{(u,r)}[\exists U' \subseteq \mathcal{B}(u) : f_{\hat{v}'}(u) \neq r] \leq \exp\left(-\frac{\left(n\langle \hat{v}, \overline{u}\rangle - n'd\gamma^2\right)^2}{2n^2\alpha^2\sigma^2}\right). \tag{6}$$

Besides, invoking Lemma A.2, we can derive

$$\langle \hat{v}, \overline{u}\rangle \geq \frac{2\sqrt{n}-1}{2\sqrt{n}+4\sigma}\alpha\sqrt{d}. \tag{7}$$

with probability at least $1 - 2\exp(-\frac{d}{8(\sigma^2+1)})$.

Combining Formula 7 and Formula 6 and setting the right part of Eq. 6 equal to β, the theorem can be proved. □

Proof of Theorem 4.4. The inner objective of adversarial training maximizes the recommendation error, that is

$$\Delta_{adv} = \mathop{\arg\min}_{\Delta, \|\Delta\| \leq \epsilon} \langle ru, \hat{v} + \Delta_{adv}\rangle = -sign(ru) \cdot \epsilon.$$

The outer objective is optimized through the standard training. Suppose that the item embedding of adversarial training is $\hat{v}_{AT}$. We have

$$\hat{v}_{AT} + \Delta_{adv} = \sum\nolimits_{i=1}^n r_i u_i = \hat{v}.$$

So $\hat{v}_{AT} = \hat{v} + sgn(ru) \cdot \epsilon$, and we need to bound the quantity

$$\begin{aligned}
&\mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : f_{\hat{v}'_{AT}}(u) \neq r] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \langle ru, \hat{v}' + sgn(ru) \cdot \epsilon\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \\
&\quad \langle ru, \frac{1}{n+n'}(n\hat{v} + \sum\nolimits_{(u',r')\in\mathcal{U}'} r'u') + sgn(ru) \cdot \epsilon\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle + \min \sum\nolimits_{(u',r')\in\mathcal{U}'} rr'uu' + (n+n')\|u\|_1 \cdot \epsilon \leq 0] \\
&\leq \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle - n'd\gamma^2 + (n+n')\epsilon\tau \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\langle ru, \frac{\hat{v}}{\|\hat{v}\|_2}\rangle \leq \frac{n'd}{n\alpha}\gamma^2 - \frac{n+n'}{n\alpha}\epsilon\tau].
\end{aligned}$$

Invoking Lemma A.1, we can get the bound of the recommendation error:

$$\exp\left(-\frac{\left(n\langle \hat{v}, \overline{u}\rangle - n'd\gamma^2 + (n+n')\epsilon\tau\right)^2}{2n^2\alpha^2\sigma^2}\right). \tag{8}$$

Putting Formula 7 into Formula 8 and setting Formula 8 to β, the theorem is proved. □

Proof of Theorem 4.5. Suppose $\mathcal{U}^* = \{(u_1^*, r_1^*), \ldots, (u_{n^*}^*, r_{n^*}^*)\}$ is the set of injected ERM users' embedding. Similar to the proof of Theorem 4.3, we have

$$\begin{aligned}
&\mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : f_{\hat{v}'}(u) \neq r] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \langle ru, \hat{v}'\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\exists \mathcal{U}' \subseteq \mathcal{B}(u) : \\
&\quad \langle ru, \frac{1}{n+n'+n^*}(n\hat{v} + \sum\nolimits_{(u',r')\in\mathcal{U}'} r'u' + \sum\nolimits_{(u^*,r^*)\in\mathcal{U}^*} r^*u^*)\rangle \leq 0] \\
&= \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle + \min \sum\nolimits_{(u',r')\in\mathcal{U}'} rr'uu' + n^*\|u\|_1\gamma \leq 0] \\
&\leq \mathop{\mathbb{P}}_{(u,r)}[\langle nru, \hat{v}\rangle \leq n'd\gamma^2 - n^*\gamma\tau] = \mathop{\mathbb{P}}_{(u,r)}[\langle ru, \frac{\hat{v}}{\|\hat{v}\|_2}\rangle \leq \frac{n'd\gamma^2}{n\alpha} - \frac{n^*\gamma\tau}{n\alpha}].
\end{aligned}$$

Invoking Lemma A.1, we can get the upper bound of recommendation error β:

$$\exp\left(-\frac{\left(n\langle \hat{v}, \overline{u}\rangle - n'd\gamma^2 + n^*\gamma\tau\right)^2}{2n^2\alpha^2\sigma^2}\right). \tag{9}$$

By combining Formula 7 and Formula 9 and setting Formula 9 to β, the theorem is proved. □

REFERENCES

[1] Mehmet Aktukmak, Yasin Yilmaz, and Ismail Uysal. 2019. Quick and accurate attack detection in recommender systems through user attributes. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 348–352.

[2] Christos G Bampis, Cristian Rusu, Hazem Hajj, and Alan C Bovik. 2017. Robust matrix factorization for collaborative filtering in recommender systems. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 415–419.

[3] Akhilan Boopathy, Sijia Liu, Gaoyuan Zhang, Cynthia Liu, Pin-Yu Chen, Shiyu Chang, and Luca Daniel. 2020. Proper Network Interpretability Helps Adversarial Robustness in Classification. In *International Conference on Machine Learning*. PMLR, 1014–1023.

[4] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. 2006. Classification features for attack detection in collaborative recommender systems. In *In Proceedings of KDD'06*. 542–547.

[5] Jie Cao, Zhiang Wu, Bo Mao, and Yanchun Zhang. 2013. Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system. *World Wide Web* 16, 5-6 (2013), 729–748.

[6] Huiyuan Chen and Jing Li. 2019. Adversarial tensor factorization for context-aware recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 363–367.

[7] Xu Chen, Yongfeng Zhang, Hongteng Xu, Zheng Qin, and Hongyuan Zha. 2018. Adversarial distillation for efficient recommendation with external knowledge. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–28.

[8] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 322–330.

[9] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.

[10] Yali Du, Meng Fang, Jinfeng Yi, Chang Xu, Jun Cheng, and Dacheng Tao. 2018. Enhancing the robustness of neural collaborative filtering systems under malicious attacks. *IEEE Transactions on Multimedia* 21, 3 (2018), 555–565.

[11] Wenqi Fan, Tyler Derr, Xiangyu Zhao, Yao Ma, Hui Liu, Jianping Wang, Jiliang Tang, and Qing Li. 2020. Attacking Black-box Recommendations via Copying Cross-domain User Profiles. *arXiv preprint arXiv:2005.08147* (2020).

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

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

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *In Proceedings of IJCAI'17*. 1725–1731.

[15] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 355–364.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[17] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*. 125–136.

[18] Binbin Jin, Defu Lian, Zheng Liu, Qi Liu, Jianhui Ma, Xing Xie, and Enhong Chen. 2020. Sampling-Decomposable Generative Adversarial Recommender. *Advances in Neural Information Processing Systems* 33 (2020).

[19] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*. 1885–1894.

[20] Balaji Lakshminarayanan, Guillaume Bouchard, and Cedric Archambeau. 2011. Robust Bayesian matrix factorisation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 425–433.

[21] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *In Proceedings of WWW'04*. 393–402.

[22] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in Neural Information Processing Systems*. 1885–1893.

[23] Ruirui Li, Xian Wu, and Wei Wang. 2020. Adversarial Learning to Compare: Self-Attentive Prospective Customer Recommendation in Location based Social Networks. In *In Proceedings of WSDM'20*. 349–357.

[24] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized ranking with importance sampling. In *Proceedings of The Web Conference 2020*. 1093–1103.

[25] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. 2020. Geography-Aware Sequential Location Recommendation. In *In Proceedings of KDD'20*. 2009–2019.

[26] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking Recommender Systems with Augmented User Profiles. *arXiv preprint arXiv:2005.08164* (2020).

[27] Yang Liu, Xianzhuo Xia, Liang Chen, Xiangnan He, Carl Yang, and Zibin Zheng. 2020. Certifiable robustness to discrete adversarial perturbations for factorization machines. In *In Proceedings of SIGIR'20*. ACM, 419–428.

[28] Gabriel Resende Machado, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. 2020. Adversarial Machine Learning in Image Classification: A Survey Towards the Defender's Perspective. *arXiv preprint arXiv:2009.03728* (2020).

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[30] Bhaskar Mehta. 2007. Unsupervised shilling detection for collaborative filtering. In *AAAI*. 1402–1407.

[31] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. 2007. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM TOIT* 7, 4 (2007), 23–es.

[32] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *The World Wide Web Conference*. 1443–1453.

[33] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*. 5014–5026.

[34] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based adaptive model to unify warm and cold starts recommendation. In *In Proceedings of CIKM'18*. 127–136.

[35] Mingdan Si and Qingshan Li. 2020. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review* 53, 1 (2020), 291–319.

[36] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon. com. *Ieee Internet Computing* 21, 3 (2017), 12–18.

[37] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. PoisonRec: An Adaptive Data Poisoning Framework for Attacking Black-box Recommender Systems. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 157–168.

[38] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. 2019. Adversarial training towards robust multimedia recommender system. *IEEE Transactions on Knowledge and Data Engineering* 32, 5 (2019), 855–867.

[39] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems*. 318–327.

[40] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. *Advances in Neural Information Processing Systems* 26 (2013), 2643–2651.

[41] Zhiang Wu, Junjie Wu, Jie Cao, and Dacheng Tao. 2012. HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In *In Proceedings of KDD'12*. ACM, 985–993.

[42] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. 2017. Fake Co-visitation Injection Attacks to Recommender Systems.. In *NDSS*.

[43] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial collaborative neural network for robust recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1065–1068.

[44] Fuguo Zhang. 2009. A survey of shilling attacks in collaborative filtering recommender systems. In *2009 International Conference on Computational Intelligence and Software Engineering*. IEEE, 1–4.

[45] Fuzhi Zhang, Zening Zhang, Peng Zhang, and Shilei Wang. 2018. UD-HMM: An unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering. *Knowledge-Based Systems* 148 (2018), 146–166.

[46] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical Data Poisoning Attack against Next-Item Recommendation. In *Proceedings of The Web Conference 2020*. 2458–2464.

[47] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. 2020. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*. PMLR, 11278–11287.

[48] Sheng Zhang, Amit Chakrabarti, James Ford, and Fillia Makedon. 2006. Attack detection in time series for recommender systems. In *In Proceedings of KDD'06*. ACM, 809–814.

[49] 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. *arXiv preprint arXiv:2005.10150* (2020).

[50] Zhuo Zhang and Sanjeev R Kulkarni. 2014. Detection of shilling attacks in recommender systems via spectral clustering. In *17th International Conference on Information Fusion (FUSION)*. IEEE, 1–8.

[51] Fan Zhou, Ruiyang Yin, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Jin Wu. 2019. Adversarial point-of-interest recommendation. In *The World Wide Web Conference*. 3462–34618.

[52] Wei Zhou, Junhao Wen, Qiang Qu, Jun Zeng, and Tian Cheng. 2018. Shilling attack detection for recommender systems based on credibility of group users and rating time series. *PloS one* 13, 5 (2018), e0196533.