Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data

Hengtong Zhang[1,5†], Changxin Tian[2,4†], Yaliang Li[6], Lu Su[1], Nan Yang[2], Wayne Xin Zhao[3,4♠], Jing Gao[1♠]
[1]School of Electrical and Computer Engineering, Purdue University, USA
[2]School of Information, Renmin University of China, China
[3]Gaoling School of Artificial Intelligence, Renmin University of China
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods, China
[5]Department of Computer Science and Engineering, State University of New York at Buffalo, USA
[6]Alibaba Group
htzhang.work@gmail.com,{tianchangxin,yangnan}@ruc.edu.cn,
yaliang.li@alibaba-inc.com,{lusu,jinggao}@purdue.edu,batmanfly@gmail.com

ABSTRACT

Recent studies reveal that recommender systems are vulnerable to data poisoning attack due to their openness nature. In data poisoning attack, the attacker typically recruits a group of controlled users to inject well-crafted user-item interaction data into the recommendation model's training set to modify the model parameters as desired. Thus, existing attack approaches usually require full access to the training data to infer items' characteristics and craft the fake interactions for controlled users. However, such attack approaches may not be feasible in practice due to the attacker's limited data collection capability and the restricted access to the training data, which sometimes are even perturbed by the privacy preserving mechanism of the service providers. Such design-reality gap may cause failure of attacks. In this paper, we fill the gap by proposing two novel adversarial attack approaches to handle the incompleteness and perturbations in user-item interaction data. First, we propose a bi-level optimization framework that incorporates a probabilistic generative model to find the users and items whose interaction data is sufficient and has not been significantly perturbed, and leverage these users and items' data to craft fake user-item interactions. Moreover, we reverse the learning process of recommendation models and develop a simple yet effective approach that can incorporate context-specific heuristic rules to handle data incompleteness and perturbations. Extensive experiments on two datasets against three representative recommendation models show that the proposed approaches can achieve better attack performance than existing approaches.

† The first two authors contributed equally to this work.
♠ Corresponding author.

CCS CONCEPTS

• **Information systems** → **Personalization**; • **Security and privacy** → **Web application security**.

KEYWORDS

Adversarial learning, Recommender system, Data poisoning

ACM Reference Format:
Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, Jing Gao. 2021. Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467233

1 INTRODUCTION

In the era of information explosion, how to precisely locate the needed information becomes a challenging task for online service users. To tackle such a challenge, service providers (e.g., YouTube, Amazon, and eBay) deploy recommender systems, which suggest items (e.g., products, movies, etc.) to specific users based on their profiles and historical behaviors. These systems play an important role in helping users to make their decisions and choices.

In normal scenarios, the results from online recommender systems are generally considered to be reasonable and unbiased. However, recommender systems' openness (i.e., recommendation models are learned based on user data which is usually publicly accessible) and the great influence on online users offer both opportunities and incentives for adversarial attackers. Particularly, attackers can inject fake data into the training set of the recommendation model to make the model behave abnormally (e.g., to promote some target items to users). This kind of attacks is named data poisoning attacks [6, 14–16, 22, 26–28]. Specifically, the attacker utilizes controlled users to inject well-crafted fake user-item interaction data into the training set. The objective is to manipulate the representations inferred for the target items so that they are similar to that of the target users.

Though yielding reasonably good performance, almost all these works assume an attacker can obtain the entire training data of the target recommendation model. However, such assumption is often impractical in the real world. First, online service providers may restrict one's access to the full user-item interaction data or even perturb these data [20, 21] to enforce user privacy. Second, the attacker's capability of collecting massive user-item interaction data may be limited by his/her resources and/or the platform's data volume restriction. These practical issues may make the existing attack approaches suffer performance decrease. In data poisoning attacks, the attacker employs controlled users to interact with target items and some selected items (referred to as proxy items). In essence, such a process shapes these controlled users' representations to further influence the representations of target items to accomplish the attack goals. Hence, the attacker needs complete and accurate user-item interaction data to select proper proxy items for controlled users. When part of the training data is removed or modified, the attack approaches cannot correctly find the appropriate proxy items.

In this paper, we conduct a pioneer study on practical attack approaches against recommendation models, based on incomplete and perturbed user behavior data. Since recommender systems are generally blackbox for the public, we utilize a local surrogate model to craft fake user-item interactions. Our first approach *RAPU-G* (**R**ecommendation **A**ttack for **P**artial and Pert**U**rbed Data - **G**lobal View) formulates the attack as a bi-level optimization problem. The upper-level problem defines the overall attack objective, and the lower-level problem specifies the recommendation model's learning objective using both original and injected data. Since solving for the optimal fake data in the bi-level optimization framework involves the optimization over the entire original data set, we regard this approach as a *global-view approach*. As mentioned above, the incomplete and perturbed user data may hinder the attack framework from capturing certain users' and items' characteristics. To handle such issue, we incorporate a probabilistic model into the bi-level optimization framework to: (1) find the users and items whose interaction data are sufficient and have not been significantly perturbed; and (2) leverage these users' and items' data to craft the fake user-item interactions.

To improve the efficiency of the attack, we further backtrack the optimization process of recommendation models in depth and propose a much simpler yet effective greedy approach named *RAPU-R* (**R**ecommendation **A**ttack for **P**artial and Pert**U**rbed Data - **R**everse). The RAPU-R method starts from the attack goal and reverse the optimization process to obtain the proper proxy items for controlled users. In essence, to promote a target item to a specific user, the attacker needs to inject well-crafted user-item interactions into the recommendation model's training set so that the representation of the target item moves towards that of the user. Thus, RAPU-R should first infer the optimal modification of the target item's representation that can accomplish the attack goal. Then from the optimization's point of view, to let the target item's representation move along the optimal modification instead of staying at the original place, the controlled user has to use their own representations to influence that of the target item. Based on such a principle, we can further determine the desired representation of the controlled users. Finally, with the controlled user's desired representation, we can search for appropriate items based on their representations and heuristic rules. Moreover, since the amount of fake data injected by controlled users is very small compared with the entire dataset's size, the representation of an overwhelming majority of users and items is not influenced. Hence, the proposed reversing approach does not have to repeatedly update the recommendation models' parameters as the global-view approach. Therefore, the reversing method is much more efficient.

In the experiments, we use three representative recommendation models as target models and conduct attacks on two real-world datasets to evaluate the proposed poisoning attack approaches. Experimental results show that the proposed RAPU-G and RAPU-R outperform baseline attack approaches and can effectively promote target items, especially when the training data is incomplete or perturbed. We also conduct extra studies on the characteristics of the proposed approaches.

2 THREAT MODEL

In this paper, we study *top-K recommendation via implicit feedback*, one of the most widely adopted recommendation settings. Its formal definition is given below:

Definition 2.1 (Top-K Recommendation via Implicit Feedback). Consider a user-item interaction dataset, in which data records associate a user u and an item i. Under the implicit setting, there are only positive feedback signals, which indicate users' positive interactions (e.g., clicks or purchase) with items. The recommendation system's task is to provide the users with a personalized ranking of size K.

With the recommendation task defined above, now let us detail the threat model of the proposed attack.

Attack Goal: An attacker's goal is to promote a set of target items to as many target users as possible. Specifically, suppose the system recommends K items to each user, *the attacker's goal is to maximize averaged display rate, which denotes the fraction of target users whose top-K recommendations results include the target items.*

The Knowledge and Capability of the Attacker: In this paper, we assume that the attacker has the following knowledge and capability:

(1) The attacker can access only part of the training set of the target recommendation model. In the data collected by the attacker, part of each user's (or item's) historical interactions are removed or perturbed.
(2) The attacker has limited resources, i.e., he/she can control a limited number of users.
(3) Since normal users often interact with a small number of items, we also limit the maximum number of items each controlled user can interact with. Such a restriction is imposed to prevent the controlled users from being detected by simple anomaly detectors.
(4) The attacker does not know the details about the target recommendation system. For instance, the parameters and the architecture of the recommendation model.

Attack Approach: To achieve the attack goal, we consider the most general scenario in which the attacker recruits controlled users who can visit or rate the target and the selected proxy items. Besides, the set of user-item interactions before and after controlled users inject fake data are referred to as *original data* and *manipulated data*, respectively.

Moreover, since the attacker does not know the exact architecture or parameters of the target model, *we leverage a local surrogate recommendation model to craft the fake user-item interactions and directly use them to poison the target system.* The intuition behind such design is that: if two recommendation models can both produce satisfactory recommendation results on a given dataset, then the poisoning samples generated for one of the recommendation models can be used to attack the other. Specifically, we choose to use Weighted Regularized Matrix Factorization (WRMF) [12], a fundamental and representative factorization-based model for recommendations with implicit interaction, as the surrogate model. Due to space limitation, please refer to Appendix A.4 for the detailed formulation. We will discuss the transferability of the poisoning samples in our experiment section in detail.

3 RAPU-G: THE GLOBAL VIEW METHOD

In this section, we first formalize the attack approach via a bi-level optimization problem and then propose a probabilistic generative model (PGM) integrated with the bi-level optimization framework to handle the issues caused by the incomplete and perturbed data.

3.1 Attack as an Optimization Problem

In RAPU-G, we directly learn fake user-item interactions for controlled users by solving a bi-level optimization problem that is similar to [5, 15, 22]. On one hand, the lower-level problem solves for the optimal parameters of the recommendation model given both the original data and the fake data injected by the controlled users. On the other hand, the upper-level problem solves for the optimal fake data to accomplish the attack goal given current model parameters obtained via solving the lower-level problem.

Formally, let $\mathcal{I}$ denote the set of items, $\mathcal{U}$ and $\mathcal{U}'$ denote the set of normal users and controlled users respectively, and $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ and $\mathbf{R}^* \in \mathbb{R}^{|\mathcal{U}'|\times|\mathcal{I}|}$ denote the original data and the fake data. We formulate the bi-level optimization problem in Eq. (1).

$$\begin{aligned} &\min_{\mathbf{R}^*} \ \mathcal{L}(\widehat{\mathbf{R}}_{\theta^*}), \\ \text{s.t.} \quad &\theta^* = \arg\min_{\theta} \left(\mathcal{L}_{train}(\mathbf{R}, \widehat{\mathbf{R}}_\theta) + \mathcal{L}_{train}(\mathbf{R}^*, \widehat{\mathbf{R}}^*_\theta)\right), \end{aligned} \tag{1}$$

where θ denotes the model parameters, $\widehat{\mathbf{R}}_\theta$ and $\widehat{\mathbf{R}}^*_\theta$ denote the recommendation predictions from the models trained on original data and fake data with parameter θ respectively, $\mathcal{L}_{train}$ denotes the training loss of the recommendation model, and $\mathcal{L}$ is the adversarial attack objective, which can be adjusted for different malicious goals. Since solving for the optimal fake data involves the optimization over the entire original dataset, we regard the approach as a *global-view approach.*

In this paper, we aim to promote target items to as many normal users as possible. Thus, the adversarial attack objective function can be defined as: $\mathcal{L} = -\sum_{t\in\mathcal{T}}\left(\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{I}_u^*} g(\hat{r}_{ut}-\hat{r}_{uv})\right)$, where $\hat{r}_{ui}$ is the predicted score that the user u gives to the item i, $\mathcal{T}$ denotes the set of target items, $\mathcal{I}_u^*$ is the set of top-K recommended items for a user u according to the predicted interaction, and $g(x) = \frac{1}{1+\exp(-x/b)}$ denote the Wilcoxon-Mann-Whitney loss [1], where b is a parameter called *width.*

Algorithm 1: Generative Process of User-Item Interactions

```
1  for each user u do
2      Draw a keeping ratio: $\kappa_u^{(U)} \sim U(0,1)$;
3      Draw a decent ratio: $\gamma_u^{(U)} \sim U(0,1)$;
4  end
5  for each item i do
6      Draw a keeping ratio: $\kappa_i^{(I)} \sim U(0,1)$;
7      Draw a decent ratio: $\gamma_i^{(I)} \sim U(0,1)$;
8  end
9  for each possible user-item pair (u, i) do
10     Draw a keeping indicator $\delta_{ui}^{(1)} \sim Bernoulli(\frac{\kappa_u^{(U)}+\kappa_i^{(I)}}{2})$;
11     if $\delta_{ui}^{(1)} = 0$ then
12         Sample the observation as: $P(r_{ui} = 0; \delta_{ui} = 0) = 1$ ;
13     else if $\delta_{ui}^{(1)} = 1$ then
14         Draw a decent indicator $\delta_{ui}^{(2)} \sim Bernoulli(\frac{\gamma_u^{(U)}+\gamma_i^{(I)}}{2})$;
15         if $\delta_{ui}^{(2)} = 0$ then
16             Sample the observation from: $r_{ui} \sim \mathcal{N}(\bar{r}, \sigma_r^2)$ ;
17         else if $\delta_{ui}^{(2)} = 1$ then
18             Sample the observation from: $r_{ui} \sim \mathcal{N}(\hat{r}_{ui}, \sigma_r^2)$ ;
19 end
```

3.2 Handling Incomplete and Perturbed Data via Probabilistic Generative Model

As one can see, in Eq. (1), the lower-level optimization problem mainly relies on the original data collected by the attacker, i.e., **R**, to learn the parameters of the local surrogate recommendation model, i.e., θ. Existing works generally assume that the attacker can access to the full training set. However, as discussed in the introduction, this assumption does always hold in real world. When part of the original data **R** is perturbed or removed, the local recommendation model cannot estimate the normal users' and items' characteristics precisely. In this case, the attack framework might be unable to find the proper proxy items for the controlled users to interact with. To tackle this challenge and estimate users' and items' characteristics more accurately, we propose to integrate the bi-optimization framework with a PGM to handle the incompleteness and perturbations.

3.2.1 Overview. In a benign recommendation dataset, observed records indicate positive user-item interactions (e.g., a user clicks on an item), and the user-item pairs that do not exist in the observed records are treated as negative user-item interactions (e.g., a user does not click on an item). Both positive and negative user-item interactions contribute to inferring the characteristics of users and items. In the proposed PGM, we do not assume that all the user-item interactions (both positive and negative) are genuine. Instead, we consider all the possibilities that may result in the observed positive or negative interactions. An positive user-item interaction may be a real positive interaction, or otherwise, the outcome of possible perturbations. Similarly, a negative user-item interaction may be a real negative interaction, or otherwise, the outcome of the online service provider's removals (e.g., removing a positive interaction) or perturbations. To model these phenomena, in the PGM, we introduce parameters to model the chance of a user's (or an item's) historical interactions being removed or modified. For each possible user-item interaction, the PGM infers the possibility that an interaction is real or the outcome of a modification.

3.2.2 Detailed Description of PGM. The proposed PGM is detailed in Algorithm 1. For each user u and each item i, we define variable $\kappa_u^{(U)}$ and $\kappa_i^{(I)}$, respectively, to estimate the ratio of its interaction records that are not removed (line 2 and 6). Similarly, we use variable $\gamma_u^{(U)}$ and $\gamma_i^{(I)}$, respectively, to define the ratio of its interaction records that are not modified (line 3 and 7). All these four variables are drawn from a uniform distribution between 0 and 1, i.e., $U(0,1)$, since we do not assume any prior knowledge of them.

Then we exhaust all the possible user-item pairs regardless of whether there is an observed positive interaction between them. For each possible user-item interaction pair, we first sample a binary indicator $\delta_{ui}^{(1)}$ from a Bernoulli distribution to infer whether the potential interaction is included in the attacker's data (line 10) or not, i.e.,: $\delta_{ui}^{(1)} \sim Bernoulli(\frac{\kappa_u^{(U)}+\kappa_i^{(I)}}{2})$. Here the parameter of Bernoulli distribution is calculated by taking the average of $\kappa_u^{(U)}$ and $\kappa_i^{(I)}$, since both the user and the item involved in the interaction determine whether a potential positive interaction is kept or removed. When $\delta_{ui}^{(1)} = 0$, the observation is determined to be negative (line 11-12). Otherwise, we sample another binary indicator $\delta_{ui}^{(2)}$ from a Bernoulli distribution, i.e., $\delta_{ui}^{(2)} \sim Bernoulli(\frac{\gamma_u^{(U)}+\gamma_i^{(I)}}{2})$, to infer whether the observed interaction comes from an intentional modification ($\delta_{ui}^{(2)} = 0$, line 15-16) or otherwise, comes from the characteristics of the user/item ($\delta_{ui}^{(2)} = 1$, line 17-18). Formally, if $\delta_{ui}^{(2)} = 0$, we draw the observed interaction from a Gaussian distribution centered at $\bar{r}$, which is the averaged interaction. Otherwise, if $\delta_{ui}^{(2)} = 1$, we draw the observed interaction from a Gaussian distribution centered at $\hat{r}_{ui}$, which is the prediction of the arbitrary surrogate recommendation model.

The PGM specifies a negative log-likelihood of observations, latent variables, and parameters given (1) the hyper-parameters and (2) predictions from the recommendation model:

$$\begin{aligned}
&\mathcal{L}_{pgm} \\
&= -\sum_{(u,i)} \log p(\delta_{ui}^{(1)}, \delta_{ui}^{(2)}, r_{ui} \mid \kappa_u^{(U)}, \kappa_i^{(I)}, \gamma_u^{(U)}, \gamma_i^{(I)}, \hat{r}_{ui}, \sigma_r) \\
&= -\sum_{(u,i)} \log \text{Bernoulli}(\delta_{ui}^{(1)} \mid \kappa_u^{(U)}, \kappa_i^{(I)}) - \log \text{Bernoulli}(\delta_{ui}^{(2)} \mid \gamma_u^{(U)}, \gamma_i^{(I)}) \\
&\quad - \delta_{ui}^{(1)}\delta_{ui}^{(2)} \log \mathcal{N}(r_{ui} \mid \hat{r}_{ui}, \sigma_r^2) - \delta_{ui}^{(1)}(1-\delta_{ui}^{(2)}) \log \mathcal{N}(r_{ui} \mid \bar{r}, \sigma_r^2) \\
&\quad - (1-\delta_{ui}^{(1)}) \log \mathbb{I}[\delta_{ui}^{(1)} = 0],
\end{aligned}$$

where $\mathbb{I}[x]$ is the indicator function that evaluates to 1 when x is true, and 0 otherwise.

The proposed PGM can be viewed as an extension upon an arbitrary model-based recommendation method. To incorporate the PGM into the bi-level optimization framework, we simply replace the training loss on original data, i.e., $\mathcal{L}_{train}(\mathbf{R}, \hat{\mathbf{R}}_\theta)$ in the lower-level of Eq. (1), with the joint negative log likelihood specified by the PGM, i.e., $\mathcal{L}_{pgm}(\mathbf{R}, \hat{\mathbf{R}}_\theta)$. By plugging the negative log-likelihood above into the lower-level, the optimization problem becomes:

$$\begin{aligned}
&\min_{\mathbf{R}^*} \ \mathcal{L}(\hat{\mathbf{R}}_{\theta^*}), \\
\text{s.t.} \quad &\theta^* = \arg\min_\theta \left(\mathcal{L}_{pgm}(\mathbf{R}, \hat{\mathbf{R}}_\theta) + \mathcal{L}_{train}(\mathbf{R}^*, \hat{\mathbf{R}}^*_\theta)\right).
\end{aligned} \tag{2}$$

Remarks: If we look into the formulation of the negative log-likelihood, we can find only the user-item interactions that are inferred as real data (in contrast to the outcomes of removals and perturbations) are used to infer the characteristics/representations of users and items (term: $\delta_{ui}^{(1)}\delta_{ui}^{(2)} \log \mathcal{N}(r_{ui} \mid \hat{r}_{ui}, \sigma_r^2)$). Thus, the attack framework can leverage accurately estimated user/item representations to find proper proxy items for controlled users to launch the attacks.

Algorithm 2: Learning fake user data with EM and Gradient Descent

```
    Input: max iteration for inner and outer objective: L and T;
           learning rate for inner and outer objective: $\alpha$ and $\eta$;
1   for $t$ = 1 to $T$ do
2       for $m$=1 to $M$ do
3           Surrogate model forward:
              $\widehat{\mathbf{R}}_{\theta^{(l-1)}}, \widehat{\mathbf{R}^*}_{\theta^{(l-1)}} = Model_{sur}(\mathcal{U}, \mathcal{I}, \mathcal{U}'; \theta^{(l-1)})$ ;
4           Use EM to compute the PGM latent variables and
              parameters for each user-item pair.;
5           Optimize inner objective with SGD: $\theta^{(m)} \leftarrow \theta^{(m-1)} - \alpha \cdot$
              $\nabla_\theta \left(\mathcal{L}_{pgm}\left(\mathbf{R}, \widehat{\mathbf{R}}_{\theta^{(m-1)}}\right) + \mathcal{L}_{train}\left(\mathbf{R}^*, \widehat{\mathbf{R}}^*_{\theta^{(m-1)}}\right)\right)$. ;
6       end
7       $\theta^* \leftarrow \theta^{(M)}$ ;
8       $\tilde{\mathbf{R}} = \mathbf{R}^* - \eta \cdot \nabla_{\mathbf{R}^*} \mathcal{L}(\widehat{\mathbf{R}}_{\theta^*})$ ;
9       Update fake data: $\mathbf{R}^* \leftarrow Project(\tilde{\mathbf{R}} + \beta \mathbf{R}_p)$;
10  end
```

3.3 Learning Method of RAPU-G

The overall learning algorithm is summarized in Algorithm 2. In each outer iteration $t \in \{1...T\}$, we first re-train the surrogate model by performing parameter updates for L iterations (inner iterations, line 2-6), and then use PGD and projection operator to update fake data $\mathbf{R}^*$ (line 8-9).

Specifically, in each inner iteration, we first perform the forward process to get the current recommendation predictions on original data $\widehat{\mathbf{R}}_{\theta^{(m-1)}}$ (line 3), and then solve for the PGM latent variables and parameters via EM algorithm [4] (line 4) :

E-step: In E-step, we compute expectation of the latent variables for all the possible user and item combinations (u, i). Obviously, a specific pair is surely not removed by the service provider if we observe $r_{ui} = 1$, i.e., $\mathbb{E}[\delta_{ui}^{(1)} \mid r_{ui} = 1] = 1$, and $\delta_{ui}^{(2)}$ does not exist if $\delta_{ui}^{(1)} = 0$. Thus, we need to derive the expectation of $\delta_{ui}^{(1)}$ when $r_{ui} = 0$ and the expectation of $\delta_{ui}^{(2)}$ when $\delta_{ui}^{(1)} = 0$:

$$\begin{aligned}
&\mathbb{E}[\delta_{ui}^{(1)} \mid \kappa_u^{(U)}, \kappa_i^{(I)}, \gamma_u^{(U)}, \gamma_i^{(I)}, \hat{r}_{ui}, r_{ui} = 0] \\
=&\frac{K_{ui} \cdot (\Gamma_{ui}\mathcal{N}(0 \mid \hat{r}_{ui}, \sigma_r^2) + (1-\Gamma_{ui})\mathcal{N}(0 \mid \bar{r}, \sigma_r^2))}{K_{ui} \cdot ((\Gamma_{ui}\mathcal{N}(0 \mid \hat{r}_{ui}, \sigma_r^2) + (1-\Gamma_{ui})\mathcal{N}(0 \mid \bar{r}, \sigma_r^2)) + (1-K_{ui})}, \\
&\mathbb{E}[\delta_{ui}^{(2)} \mid \kappa_u^{(U)}, \kappa_i^{(I)}, \gamma_u^{(U)}, \gamma_i^{(I)}, \hat{r}_{ui}, r_{ui}, \delta_{ui}^{(1)} = 1] \\
=&\frac{\Gamma_{ui}\mathcal{N}(r_{ui} \mid \hat{r}_{ui}, \sigma_r^2)}{\Gamma_{ui}\mathcal{N}(r_{ui} \mid \hat{r}_{ui}, \sigma_r^2) + (1-\Gamma_{ui})\mathcal{N}(r_{ui} \mid \bar{r}, \sigma_r^2)}
\end{aligned} \tag{3}$$

where $K_{ui} = \frac{\kappa_u^{(U)}+\kappa_i^{(I)}}{2}, \Gamma_{ui} = \frac{\gamma_u^{(U)}+\gamma_i^{(I)}}{2}$, $\mathcal{N}(r_{ui} \mid \hat{r}_{ui}, \sigma_r^2)$ stand for the values of $\mathcal{N}(\hat{r}_{ui}, \sigma_r^2)$'s probability density function (PDF) evaluated at r_{ui}.

M-step: We define $\mathbb{E}^{(1)}_{ui} = \mathbb{E}[\delta^{(1)}_{ui} \mid \kappa^{(U)}_u, \kappa^{(I)}_i, \gamma^{(U)}_u, \gamma^{(I)}_i, \hat{r}_{ui}, r_{ui}]$ and $\mathbb{E}^{(2)}_{ui} = \mathbb{E}[\delta^{(2)}_{ui} \mid \kappa^{(U)}_u, \kappa^{(I)}_i, \gamma^{(U)}_u, \gamma^{(I)}_i, \hat{r}_{ui}, r_{ui}]$ for simplicity, which are computed from the E-step. To update the parameters of the generative model, i.e., $\kappa^{(U)}_u, \kappa^{(I)}_i, \gamma^{(U)}_u, \gamma^{(I)}_i$, we directly take the derivative of the negative log likelihood with respect to $\kappa^{(U)}_u, \kappa^{(I)}_i, \gamma^{(U)}_u, \gamma^{(I)}_i$:

$$\kappa^{(U)}_u = \frac{\sum_{i\in\mathcal{I}} \mathbb{E}^{(1)}_{ui}}{|\mathcal{I}|}, \qquad \kappa^{(I)}_i = \frac{\sum_{u\in\mathcal{U}} \mathbb{E}^{(1)}_{ui}}{|\mathcal{U}|},$$
$$\gamma^{(U)}_u = \frac{\sum_{i\in\mathcal{I}} \mathbb{E}^{(2)}_{ui} \cdot \mathbb{I}(\delta^{(1)}_{ui})}{\sum_{i\in\mathcal{I}} \mathbb{I}(\delta^{(1)}_{ui})}, \qquad \gamma^{(I)}_i = \frac{\sum_{u\in\mathcal{U}} \mathbb{E}^{(2)}_{ui} \cdot \mathbb{I}(\delta^{(1)}_{ui})}{\sum_{u\in\mathcal{U}} \mathbb{I}(\delta^{(1)}_{ui})}. \tag{4}$$

On getting the PGM latent variables and parameters via EM, we use stochastic gradient descend (SGD) to solve the bi-level optimization problem defined in Eq. (2). The optimization approach we used here is identical to [22]. Note that in line 9, we smooth the fake data via the popularity of each item to obtain the final fake data. The intuition of this operation is that the popularity is a significant prior knowledge for attacks. Specifically, the popularity matrix can be estimated as: $\mathrm{R}_p[u, i] = \frac{c_i}{c_{max}}$, where c_i is the total number of interactions related to item i and c_{max} is the maximum interaction number among all items. We normalize the fake data into feasible region (i.e., $r_{ui} \in \{0, 1\}$).

4 RAPU-R: THE REVERSING ATTACK APPROACH

In this section, we reverse the optimization process of a recommendation model from a reverse review to propose an efficient yet effective attack approach.

4.1 Attack via Reversing the Optimization Process

For simplicity of discussion, we use the aforementioned WRMF [12, 18], to conduct our analysis. While, our discussion on WRMF can be easily extended to other recommendation models.

Consider a simple case in which the attacker wants to promote the target item i to a user u. We denote the representations of user u and item i as $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$, respectively. Promoting the target item i to a user u means that we want to maximize their *inner product*, i.e.: $\hat{r}_{ui} = \boldsymbol{p}_u \cdot \boldsymbol{q}_i$. Since the attacker cannot modify observed interactions of normal users in the training set, we focus on leveraging the controlled users to manipulate the representation of the target item i and further accomplish the promotion goal. Formally, let $\boldsymbol{\epsilon}$ be the perturbation on $\boldsymbol{q}_i$, the optimal $\boldsymbol{\epsilon}$, i.e., $\boldsymbol{\epsilon}^*$ should satisfy:

$$\boldsymbol{\epsilon}^* = \arg\max_{\boldsymbol{\epsilon}} \boldsymbol{p}_u \cdot (\boldsymbol{q}_i + \boldsymbol{\epsilon}).$$

Obviously, the optimal $\boldsymbol{\epsilon}^* = \boldsymbol{p}_u$ in our case.

Thus, we have transformed the problem of promoting target item i into the problem of shifting $\boldsymbol{q}_i$ by $\boldsymbol{\epsilon}^*$. To achieve this goal, *we need to make the sum of terms involving item i in the training loss decrease after $\boldsymbol{q}_i$ is shifted by $\boldsymbol{\epsilon}^*$*. Formally, we have to let controlled users (e.g., user m) inject fake data to make:

$$\sum_{m\in\mathcal{U}\cup\mathcal{U}'} L(\boldsymbol{p}_u, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*; r_{mi}) < \sum_{m\in\mathcal{U}\cup\mathcal{U}'} L(\boldsymbol{p}_m, \boldsymbol{q}_i; r_{mi}), \tag{5}$$

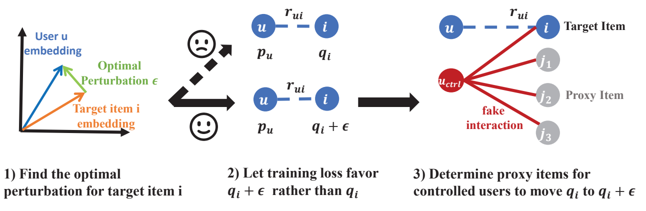

User u embedding
Optimal Perturbation ϵ
Target item i embedding
r_{ui}
u
i
p_u
q_i
r_{ui}
u
i
p_u
$q_i + \epsilon$
r_{ui}
u
i
Target Item
u_{ctrl}
j_1
j_2
j_3
Proxy Item
fake interaction
1) Find the optimal perturbation for target item i
2) Let training loss favor $q_i + \epsilon$ rather than q_i
3) Determine proxy items for controlled users to move q_i to $q_i + \epsilon$

Figure 1: Overview of RAPU-R

Algorithm 3: Workflow of RAPU-R.

```
Input: normal user representations P, item representations Q, set
       of target item $\mathcal{T}$, number of controlled users $M$;
1 for $m$ = 1 to $M$ do
2     Find the optimal direction of perturbation for target items $\mathcal{T}$:
        $\boldsymbol{\epsilon}^* = \sum_{i\in\mathcal{T}} \arg\max_{\boldsymbol{\epsilon}} \boldsymbol{p}_u \cdot (\boldsymbol{q}_i + \boldsymbol{\epsilon})$ ;
3     Get the optimal representations for the controlled user $m$:
        $\boldsymbol{p}^*_m = \arg\min_{\boldsymbol{p}_m} L(\boldsymbol{p}_m, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*; r_{mi})$ ;
4     Find the best proxy item set $\mathcal{I}_{\Updownarrow}$ that should be interacted with,
        based on the similarity between the item representation and
        $\boldsymbol{q}^*$, where: $\boldsymbol{q}^* = \arg\min_{\boldsymbol{q}} (\boldsymbol{p}^*_m \cdot \boldsymbol{q} - 1)$ ;
5     Controlled user $m$ clicks the best proxy item set $\mathcal{I}_{\Updownarrow}$.
6 end
```

where L is the loss function of a single user-item interaction pair[1].

As one can see from Eq. (5), both normal users and controlled users interact with the target item i. However, the attacker can only control the interaction data generated by the controlled users. Hence, the practical surrogate goal is to only consider that the controlled users inject fake data:

$$\sum_{m\in\mathcal{U}'} L(\boldsymbol{p}_u, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*; r_{mi}) < \sum_{m\in\mathcal{U}'} L(\boldsymbol{p}_u, \boldsymbol{q}_i; r_{mi}). \tag{6}$$

Towards the attack goal, the next step is to determine the proxy items for controlled users. From the view of optimization, the optimal representation for a controlled user m, i.e., $\boldsymbol{p}^*_m$, should satisfy:

$$\boldsymbol{p}^*_m = \arg\min_{\boldsymbol{p}_m} L(\boldsymbol{p}_m, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*; r_{mi}),$$

which provides the largest chance to make Eq. (6) hold.

Now, we know the optimal representations for the controlled users. Ideally, the controlled users should have all their representations close to $\boldsymbol{p}^*_m$. By looking into the update of $\boldsymbol{p}_m$ during the training process, we find that the ideal item representation to shape a controlled user's representation as $\boldsymbol{p}^*_m$ should satisfy:

$$\boldsymbol{q}^* = \arg\min_{\boldsymbol{q}} (\boldsymbol{p}^*_m \cdot \boldsymbol{q} - 1). \tag{7}$$

With the ideal representation, we may simply exhaust the items in the item set $\mathcal{I}$ to search for *top items whose representations are close to $\boldsymbol{q}^*$ as proxy items.* Since the injected fake user-item interactions do not significantly influence the representations of most non-target items, we do not explicitly include the model parameter updates like Eq. (2) in RAPU-R. Such an approximation strategy makes the proposed RAPU-R much more efficient than existing

[1]For the recommendation model WRMF specified in Eq. (10) (in Appendix A.4), $L(r_{mi}, \boldsymbol{p}_u, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*)$ is a quadratic loss defined as $L(r_{mi}, \boldsymbol{p}_u, \boldsymbol{q}_i + \boldsymbol{\epsilon}^*) = w_{mi}\,(r_{mi} - \boldsymbol{p}_u \cdot (\boldsymbol{q}_i + \boldsymbol{\epsilon}))^2$.

Table 1: Comparison between the Proposed RAPU-G, RAPU-R and the State-of-the-art Approach RevAdv. Here, the maximum numbers of iterations for the outer objective and EM algorithm are T and E, respectively. τ is the unroll steps constant introduced in [22]. $|\theta|$ is the number of parameters. $\mathcal{U}, \mathcal{I}$ and $\mathcal{U}'$ stand for the set of real users, items and controlled users.

	Time Complexity	Type of Technique	Handle Incompleteness and Perturbations	Performance
RAPU-G	$O(T \cdot (\tau\|\theta\|+E \cdot \|\mathcal{U}\|\|\mathcal{I}\|))$	Bi-level optimization	**Yes** (PGM to identify perturbations or removals)	Best
RAPU-R	$O(\|\mathcal{U}'\|\|\mathcal{I}\|)$	Greedy search algorithm with heuristic rules	**Yes** (Focus on effective and popular proxy items)	Similar to RevAdv
RevAdv	$O(T \cdot \tau\|\theta\|)$	Bi-level optimization	N/A	Similar to RAPU-R

methods based on bi-level optimization frameworks. Finally, we summarize the workflow of the attack approach in Algorithm 3.

Handling Data Incompleteness and Perturbations: There are two mechanisms in RAPU-R that handle the data incompleteness and perturbations issue. First, given a specific target item, RAPU-R greedily forces all the controlled users to interact with the most useful proxy items in terms of reshaping the characteristics of the target item. Such a design can naturally decrease the impact of perturbations and removals since the controlled users can still launch successful attacks with only a small part of uninfluenced useful proxy items. Second, we incorporate context-specific heuristic rules to flexibly reduce the proxy item candidates to find the items that are not influenced by the removals and perturbations. For instance, in this paper, we search the proxy items only within the items with the highest popularity (top 10%). When user-item interaction data suffers removals or perturbations, the popular items have a larger chance of maintaining enough data records so that their representations can be more accurately estimated.

4.2 Comparison of Approaches RAPU-G, RAPU-R and the State-of-the-art Attack

This section presents a comparison of the two proposed approaches and the state-of-the-art attack approach RevAdv [22]. Here, we summarize the key differences between the proposed approaches and RevAdv to highlight our insights and advantages in Table 1.

In a nutshell, the proposed RAPU-G can empirically achieve the best attack performance, since it is with a more elaborately designed probabilistic modeling process. In contrast, the proposed RAPU-R is a simple yet efficient approach, which can achieve comparable performance with RevAdv with a far less time complexity. Thus, it is particularly suitable to launch large scale attacks or conduct vulnerable tests to evaluate the robustness of recommendation models. In terms of handling data incompleteness and perturbations, RAPU-G directly utilizes the proposed PGM to identify the possible perturbations and removals. Only the user-item interactions that are inferred as real data are used to estimate items' characteristics and discover proper proxy items. In contrast, RAPU-R finds proper proxy items by reversing the learning process. It explicitly focuses on using the popular proxy items with the highest chances to accomplish the attack goal. These proxy items are not severely impacted by data removals and perturbations, empirically.

5 EXPERIMENTS

In this section, we conduct the experiments to verify the effectiveness of the proposed two approaches.

5.1 Experiment Settings

5.1.1 Datasets. In this paper, we evaluate our approaches on widely-used two real-world datasets *MovieLens-100k (MovieLens)* [8] and *Amazon Instant Video (Amazon-Video)* [9]. The detailed descriptions of these two dataset are in Appendix A.5.

5.1.2 Baseline Attack Methods. We compare our proposed method to several baseline methods. The parameters of these approaches are set as the original papers suggest.

• **None:** The circumstance when no attack is conducted.

• **Random**: In this attack, controlled users click the target items and some other randomly chosen items.

• **Popular**: Following [6, 16], each controlled user selects 10% popular items and 90% other random items as the proxy items.

• **Projected gradient ascent attack (PGA)** [15]: This approach focuses on attacking matrix factorization based recommendation models. PGA aims to assign high rating scores to the target items and randomly generate filler items for the fake users to rate.

• **Co-visitation attack (CoVis.)** [26]: This baseline attack is designed for association-rule-based recommendation models. In this approach, the attacker finds the proxy items to inject fake co-visitations by solving a standard linear programming problem.

• **Supervised random walks attack (SRWA)** [6]: SRWA also formulates the poisoning attack as an optimization problem and assumes target is a graph-based recommendation model.

• **Revisit Adv.(RevAdv.)** [22]: It also formulates poisoning attack problem as a bi-level optimization problem solved via gradient-based approaches. This is the state-of-the-art attack approach.

5.1.3 Targeted Recommendation Methods. In this section, we consider the following victim methods to attack. These methods are implemented via RecBole [29] library. The parameters of these target methods are shown in Appendix A.7.

• **WRMF** [12]: It is a fundamental and representative factorization-based model for recommendations with implicit feedback (see Section 2 for more details).

• **Neural Collaborative Filtering (NCF)** [11]: It is a popular framework that explores non-linearities in modeling complex user-item interactions. We adopt Generalized Matrix Factorization (GMF) as the instantiation of NCF.

• **LightGCN** [10]: It is the state-of-the-art GNN-based method, which discards the feature transformation and the nonlinear activation functions in the GCN aggregator.

5.1.4 Target Items. Following [22], we uniformly sample 5 items together from the whole item set as a target item set and measure the hit ratio at 50 (HR@50) on the target item set, where it is considered as a hit if one of these items appears in the ranked list.

Table 2: HR@50 of different attacks against different victim models with 90% training data + 10% perturbed data on two datasets. We use bold and underline fonts to denote the best performance and second best performance method, respectively.

Dataset	Method	Percentage of Controlled Users											
		WRMF				NCF				LightGCN			
		0.5%	1%	3%	5%	0.5%	1%	3%	5%	0.5%	1%	3%	5%
MovieLens	None	.0541	.0541	.0541	.0541	.0814	.0814	.0814	.0814	.0679	.0679	.0679	.0679
	Random	.0593	.0604	.0666	.0682	.0917	.1108	.1527	.1386	.0656	.0727	.0709	.0809
	Popular	.0600	.0641	.0725	.0755	.0944	.1132	.1612	.1463	.0648	.0736	.0823	.0960
	PGA	.0612	.0619	.0678	.0737	.0967	.1062	.1222	.1417	.0612	.0703	.0785	.0861
	SRWA	.0594	.0634	.0725	.0722	.0958	.1099	.1209	.1433	.0664	.0715	.0768	.0801
	CoVis.	.0622	.0655	.0732	.0801	.0996	.1124	.1537	.1656	.0653	.0736	.0823	.0960
	RevAdv.	.0642	.0719	.0783	.0856	.1109	.1353	.1841	.1958	.0651	.0775	.0990	.1176
	RAPU-G	.0715	.0779	**.0950**	**.1116**	**.1190**	**.1431**	**.1859**	**.2129**	**.0764**	**.0856**	**.1139**	**.1597**
	RAPU-R	**.0734**	**.0843**	.0945	.0959	.1165	.1413	.1715	.1952	.0721	.0793	.0906	.1082
Amazon-Video	None	.0074	.0074	.0074	.0074	.0369	.0369	.0369	.0369	.0228	.0228	.0228	.0228
	Random	.0571	.0709	.0913	.0998	.0793	.0839	.0935	.0852	.0522	.0674	.1186	.1493
	Popular	.0554	.0769	.0954	.0936	.0807	.0899	.0995	.1198	.0565	.0756	.1518	.2116
	PGA	.0556	.0779	.0927	.0993	.0725	.0879	.0921	.0952	.0480	.0633	.1112	.1504
	SRWA	.0521	.0730	.0971	.1020	.0746	.0849	.0919	.0955	.0532	.0662	.1181	.1609
	CoVis.	.0607	.0789	.0955	.0946	.0866	.0998	.1066	.1164	.0575	.0766	.1523	.2216
	RevAdv.	.0734	.0913	.1297	.1488	.0902	.1169	.1496	.1734	.1066	.1541	.2717	.3386
	RAPU-G	**.0777**	**.1087**	**.1482**	**.1709**	.0945	**.1471**	**.1742**	**.2186**	**.1088**	**.1808**	**.3929**	**.4976**
	RAPU-R	.0770	.1045	.1212	.1065	**.1010**	.1112	.1358	.1475	.0977	.1333	.2639	.3242

5.1.5 General Attack Settings. Unless otherwise stated, we use the following settings for all the attack approaches: The number of proxy items selected by each controlled user are set to 100 and 10 for the MovieLens and the Amazon-Video dataset respectively. The percentage of controlled users is fixed to 3%; the width of Wilcoxon-Mann-Whitney loss, b, is set to 0.1, the initial value of keeping ratio and decent ratio (in the PGM) is set to 0.8, the prior of the prediction feedback's standard deviation σ_r is set to 0.3, β is set to 1.

To simulate the incompleteness and perturbations in real-world data, we build synthetic datasets based on the two evaluation datasets. Specifically, for each dataset, we randomly down-sample the user-item interaction records to 90% of the original data size and add 10% (*w.r.t.* the original data size) randomly perturbed data. The detailed data synthesis protocol is summarized in Appendix A.6.

5.2 Results Analysis

In this section, we present the major comparison results for evaluating our proposed approachs.

5.2.1 Results under Default Setting. Table 2 summarizes the overall results of different attack approaches against three representative target recommendation models.

As shown in Table 2, the proposed global-view approach RAPU-G achieves the best performances in 21 out of 24 scenarios and outperforms the state-of-the-art method RevAdv by a significant margin in all scenarios. The proposed reversing based approach RAPU-R, despite its great efficiency (over 10x faster than RevAdv[2]), still outperforms the state-of-the-art method RevAdv in 11 out of 24 cases. For instance, compared with RevAdv, RAPU-G achieves over 15% performance improvement on the MovieLens dataset facing WRMF. Moreover, when the percentage of controlled users increases, the performance gap between the proposed RAPU-G becomes more significant. For instance, compared with RevAdv, RAPU-G achieves 5.4% and 14.7% performance increases on the Amazon-Video dataset facing WRMF, when the percentage of controlled users rises from 0.5% to 5%. We also observe that the fake data generated by RAPU-G and RAPU-R for WRMF can also be used to attack NCF and LightGCN effectively. This phenomenon demonstrates the strong transfer-ability of the fake data.

[2]This is estimated by running both approaches for 10 times and taking the average on the Movielens dataset.

5.2.2 Analysis. Among the baseline methods, *Random* and *Popular* attacks are heuristics-driven approaches, which are mostly agnostic to recommendation models. Thus, they suffer from poor performance. *PGA*, *SRWA*, *CoVis* and *RevAdv* all formulate the attack as a bi-level or integer optimization problems over the data collected by attacker and directly solve for the optimal fake user-item interactions for controlled users. Since there are no strategies to handle the negative effect caused by the incomplete and perturbed dataset, these methods cannot accurately estimate the items' characteristics and select proper proxy items for controlled users. Consequently, these methods all suffer from performance drop.

In contrast, the proposed RAPU-G integrates a probabilistic model with a bi-level optimization attack framework to infer whether a positive (or negative) user-item interaction is from the original training data or the outcome of removals or perturbations. The interactions inferred as real user-item interactions play a more important role than other data records in learning the model. Such a "denoising" process enhances the robustness of RAPU-G when facing the partial and perturbed data.

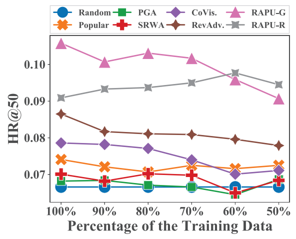
It is somehow surprising that RAPU-R has very good performance and even outperforms the sophisticated RAPU-G in 3 out of 24 scenarios, especially when the percentage of controlled user is below 1%. The reason for RAPU-R's good performance is that it greedily forces all the controlled users to interact with the most influential proxy items in terms of reshaping the representation of the target items. It can also leverage heuristic rules such as item popularity to further narrow down the search scope of finding proper proxy items. Hence, though not directly pursuing a "global optimal" solution, it empirically achieves good performances, especially when controlled user's percentage is low.

5.3 Impact of the Percentage of Data Removal and Perturbations

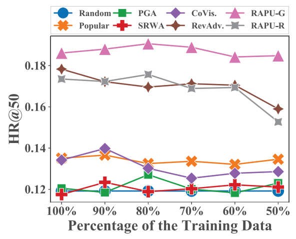
Apart from Table 2, we also conduct extensive experiments to evaluate the robustness of attack methods with different ratios of *data removal and perturbations*.

First, we evaluate the attack performances when different ratios of user-item interaction records are removed from the dataset (Figure 2). Due to space limit, we only show the results on the Movielens dataset. From Figure 2, we can find that the performance of RAPU-G (*pink line*) is consistently better than the state-of-the-art baseline method RevAdv (*brown line*). The proposed reversing based approach RAPU-R (*grey line*) can achieve comparable or even better performance compared with RevAdv. Moreover, its time complexity is far lower. In scenarios like attacking NCF and LightGCN on the Movielens dataset, the performance of RAPU-G is almost unchanged while RevAdv suffers performance decrease.

Next, we evaluate the attack performances in more challenging scenarios when different ratios of perturbed user-item interaction records are added into the dataset (Figure 3). Here we focus on the results of attacking WRMF on the Movielens dataset. From Figure 3, we can find that the performance of RAPU-G (*pink line*) is also much better than the state-of-the-art method RevAdv (*brown line*) in both scenarios. For instance, when the percentage of perturbed data is 20%, the performance of RevAdv is similar to other less sophisticated attack approaches, but RAPU-G can maintain a significantly superior performance. The proposed efficient reversing based approach RAPU-R also performs well, especially for the ratios of 15-20%.



(a) Attack WRMF



(b) Attack NCF

Figure 2: Impact of Training Data on MovieLens. (The results of attacking LightGCN is placed in the Appendix).

5.4 Impact of the Number of Proxy Items per Controlled User

Given a specific percentage of controlled users, the number of proxy items each controlled user interacts with is another important influence factor in these attack methods. In this part, we fix the percentage of training data and controlled users to 90% and 3%, respectively, and vary the number of proxy items. On the Movielens

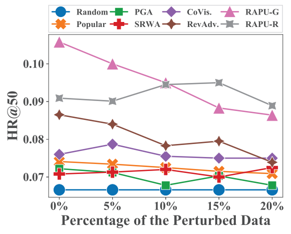


Figure 3: Impact of the Percentage of Data Perturbations on Movielens against WRMF.

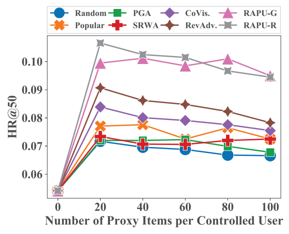


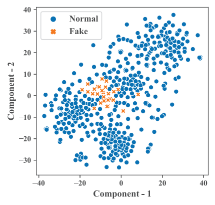
Figure 4: Impact of the Number of Proxy Items per Controlled User on Movielens against WRMF.

dataset, we vary the number of proxy items per controlled user in a range of 0 to 100. The tuning results are shown in Figure 4.

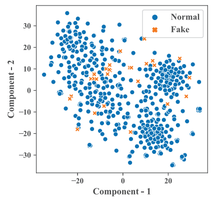
From Figure 4, we can see that the performance of attack baselines decrease as the number of proxy items increases when the number exceeds a certain threshold. This finding is somehow counterintuitive. We note that previous works [6, 16] also have similar observations for the attacks against recommendation models with explicit feedback. A possible reason is that "good" proxy items are limited, and using too many proxy items might introduce extra noise. Instead, compared with baseline attacks, our attacks are very stable and can effectively promote the target items with different numbers of proxy items.

5.5 The Detectability of the Proposed Attacks

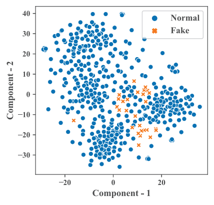
In real-world online platforms, anomaly detectors are usually deployed to detect potential malicious users. In this section, we study the detectability of our attack, i.e., whether the controlled users can be detected as anomalies in the representation space. We extract representations of normal users and controlled users learned by WRMF on manipulated data, and using t-SNE [23] to visualize them.



(a) Popular Attack



(b) RAPU-G



(c) RAPU-R

Figure 5: Controlled Users in the Representation Space.

In Figure 5, we plot the controlled users and 512 (randomly sampled) normal users in the representation space to compare the proposed attack approaches with the heuristic-based *Popular* attack. From Figure 5(a), it can be observed that the controlled users guided by *Popular* attack tend to form clusters in the representation space. Instead, as shown in Figure 5(b) and Figure 5(c), the representations of controlled users guided by RAPU-G and RAPU-R are scattered evenly in the representation space. This means that these controlled users are actually more similar to normal users, so it is difficult to identify the controlled users via distribution discrepancy. Thus, the controlled users guided by our approaches can be well camouflaged as normal users.

6 RELATED WORK

General Data Poisoning Attacks. Data poisoning attacks, in which attackers pollute the training data by injecting well-crafted adversarial samples to force the target model to behave abnormally, have been studied against a wide range of machine learning models, such as SVM [3, 25], neural networks [7, 17], regression methods [13, 24]. However, the majority of the prior work assumes that the attacker observes the whole training set and all the observed data samples are real and unchanged. In contrast, our paper studies a more practical setting, where the attacker has to handle incomplete and even perturbed training data to accomplish the attack goal.

Data Poisoning Attacks against Recommendation Models. The impact of data poisoning attacks has also been recognized in recommendation systems [5, 15]. Earlier work on data poisoning attacks against recommendation models are mostly agnostic to the target methods, e.g., random attack and average attack [14], and thus can not achieve satisfactory performance. Recently, data poisoning attacks [6, 15, 26] are proposed to generate fake behaviors that are optimized according to a particular type of recommendation system. For example, Li et al. [15] proposes data poisoning attacks for matrix-factorization-based recommendation systems. The authors model the attack as an optimization problem to decide the rating scores for the fake users. [26] proposes data poisoning attacks for association-rule-based recommendation systems, where each user injects fake co-visitations between items instead of fake rating scores of items. [6] proposes data poisoning attacks to graph-based recommendation systems. [27] proposes practical poisoning attacks against sequential recommendation models. Tang et al. [22] provides a more precise solution for the general bi-level optimization-based attack framework and relaxes the assumption that the attacker should have full knowledge about the victim model. To the best of our knowledge, there is no existing attack approache that can handle the incompleteness and perturbations in user-item interaction data.

7 CONCLUSIONS

In this work, we identify and address the challenges of data poisoning attack against recommendation systems with incomplete or even perturbed user-item interaction data. We propose two sophisticated data poisoning attack approaches to overcome this issue. The first approach RAPU-G formulate the attack as a bi-level optimization problem. To handle incomplete and untrustworthy user-item interaction data, we propose to incorporate the bi-level optimization problem with a PGM, which considers the possible removals and modifications in the observations. Moreover, we reverse the recommendation model's optimization process and propose an efficient yet effective second approach RAPU-R to conduct the poisoning attack. Experimental results in multiple scenarios clearly demonstrates the effectiveness of the proposed approaches.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Foundation of China under Grant No. 61872369, Beijing Academy of Artificial Intelligence (BAAI), and the National Science Foundation of the United States of America under Grant No. IIS-1747614.

REFERENCES

[1] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proc. of WSDM 2011*.

[2] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of machine learning research* 18 (2018).

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[4] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977).

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

[6] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proc. of ACSAC 2018*.

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

[8] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TiiS* 5, 4 (2015).

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

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

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proc. of WWW 2017*.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proc. of ICDM 2008*.

[13] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proc. of S&P 2018*.

[14] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *Proc. of WWW 2004*.

[15] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Proc. of NIPS 2016*.

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

[17] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*.

[18] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proc. of ICDM 2008*.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. of UAI 2009*.

[20] Yilin Shen and Hongxia Jin. 2014. Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *Proc. of ICDM 2014*.

[21] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy enhanced matrix factorization for recommendation with local differential privacy. *TKDE* 30, 9 (2018).

[22] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *Proc. of RecSys 2020*.

[23] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[24] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *Proc. of ICML 2015*. PMLR.

[25] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial Label Flips Attack on Support Vector Machines.. In *ECAI 2012*.

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

[27] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical Data Poisoning Attack against Next-Item Recommendation. In *Proc. of WWW 2020*.

[28] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. 2019. Data Poisoning Attack against Knowledge Graph Embedding. In *IJCAI 2019*.

[29] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *arXiv preprint arXiv:2011.01731*.

A APPENDIX

A.1 Time Complexity Analysis of RAPU-G

We first analyze the time complexity of the Algorithm 2. If we want to compute the gradient $\nabla_{\mathbf{R}^*}\mathcal{L}$ in Eq.9, we need extra time to compute $\frac{\partial \mathcal{L}}{\partial \theta^{(m+1)}} \cdot \frac{\partial \theta^{(m+1)}}{\partial \theta^{(m)}}$ for each $m \in \{1, ..., M\}$. According to the reverse-mode algorithmic differentiation [2], the time complexity of computing $\nabla_{\mathbf{R}^*}\mathcal{L}$ is proportional to the parameters θ. Thus, $O(M|\theta|)$ time is needed to have all the gradients accumulated for a single update of fake data. To improve computational efficacy, we can adopt approximated technique [22] to unrolling fewer steps when accumulating $\frac{\partial \mathcal{L}}{\partial \theta^{(m+1)}} \cdot \frac{\partial \theta^{(m+1)}}{\partial \theta^{(m)}}$. Unrolling τ steps means back-propagating gradients only within last τ steps. Therefore, the time complexity of computing $\nabla_{\mathbf{R}^*}\mathcal{L}$ can be reduced to $O(\tau|\theta|)$. Besides, we need extra time to perform the EM algorithm. Before iterating, we can cache $\mathcal{N}(0 \mid \hat{r}_{ui}, \sigma_r^2)$ for each user-item pair to reduce duplicate operations. On this basis, $O(|\mathcal{U}||\mathcal{I}|)$ time is needed in each E-step and M-step. As a result, if max iterations for outer objective and EM algorithm are T and E respectively, the time complexity of the Algorithm 2 is $O\big(T \cdot (\tau|\theta| + E \cdot |\mathcal{U}||\mathcal{I}|)\big)$.

A.2 Time Complexity Analysis of RAPU-R

As a contrast, we subsequently analyze the time complexity of the Algorithm 3. If we adopt algorithm 3 to handle partially observed data, the optimal representations for controlled users $\boldsymbol{\epsilon}^{(v)}$ can be obtained directly according to the derivative rules for vector. Because $\boldsymbol{u}_m^{ctrl}$ is only special to target items and target users, we can cache $\boldsymbol{u}_m^{ctrl}$ for speed up. Then we can traversal search 'best' items in the candidate items set, which is constructed according to the popularity of items in the implementation. Thus, the time complexity of the Algorithm 3 is $O(|\mathcal{U}'||\tilde{\mathcal{I}}|)$, where $\tilde{\mathcal{I}}$ is the set of candidate items.

A.3 Details of Gradient Descend

By applying chain rule, the adversarial gradient can be written as:

$$\nabla_{R^*}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{R}^*} + \frac{\partial \mathcal{L}}{\partial \theta^*} \cdot \frac{\partial \theta^*}{\mathbf{R}^*} \tag{8}$$

Without loss of generality, we assume the inner objective is optimized for L times, then $\theta^{(M)}$ is the final parameter used in adversarial objective. under the context of Stochastic Gradient Descent (SGD), $\nabla_{R^*}\mathcal{L}$ will become

$$\begin{aligned}
\nabla_{\mathbf{R}^*}\mathcal{L} &= \frac{\partial \mathcal{L}}{\partial \mathbf{R}^*} + \sum_{m \in [1,M]} \frac{\partial \mathcal{L}}{\partial \theta^{(m)}} \cdot \frac{\partial \theta^{(m)}}{\mathbf{R}^*}, \text{ where} \\
\frac{\partial \mathcal{L}}{\partial \theta^{(m)}} &= \frac{\partial \mathcal{L}}{\partial \theta^{(m+1)}} \cdot \frac{\partial \theta^{(m+1)}}{\partial \theta^{(m)}} = \frac{\partial \mathcal{L}}{\partial \theta^{(m+1)}} \cdot (1 - \alpha \nabla_\theta \nabla_\theta \mathcal{L}_{train}) \\
\frac{\partial \theta^{(m)}}{\mathbf{R}^*} &= -\alpha \nabla_{\mathbf{R}^*} \nabla_\theta \left(\mathcal{L}_{train}(\mathbf{R}, \hat{\mathbf{R}}_{\theta^{(m-1)}}) + \mathcal{L}_{train}(\mathbf{R}^*, \hat{\mathbf{R}}^*_{\theta^{(m-1)}}) \right)
\end{aligned} \tag{9}$$

According to Eq.(9), we can discover that partial data R will significantly influence the poisoning samples injected by controlled users.

A.4 Detailed Formulation of the Surrogate Model Used in this Paper

In WRMF, the user representation matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and the item representation matrix $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}| \times d}$ are used to make predictions $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^\top$ on user-item interaction data $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$, where d is the dimension of latent factor. The loss function of WRMF is:

$$\mathcal{L}_{train} = \sum_{u,i} w_{ui} \left(r_{ui} - \mathbf{P}_u \mathbf{Q}_i^\top \right)^2 + \lambda \left(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2 \right), \tag{10}$$

where w_{ui} is instance weight to differentiate observed and missing interactions. In this section, let us suppose we have trained the model on original data to get the matrices $\mathbf{P}$ and $\mathbf{Q}$.

A.5 Dataset Descriptions

In this paper, we adopt two real-world recommendation datasets to evaluate the effectiveness of the proposed attack approaches.

• **MovieLens-100k (MovieLens)** [8] is a widely used movie recommendation dataset. Following [19], we convert numerical ratings into implicit feedback (1 for positive interaction, 0 for negative).

• **Amazon Instant Video (Amazon-Video)** [9] is one category of the Amazon dataset. In this paper, we transform numerical ratings into implicit feedback and remove cold-start users and items with less than 10 activities.

The statistics of datasets are shown in Table 3.

Table 3: Statistics of the datasets

Datasets	#users	#items	#actions	Avg.	Sparsity
MovieLens	943	1,682	100,000	106.04	93.69%
Amazon-Video	8,049	7,076	58,194	7.23	99.89%

A.6 The Data Synthesis Strategy

To simulate the process of the attacker collecting data, we use a random walk strategy to construct partial and perturbed training data. For simplicity, we describe the details from a user's perspective subsequently.

Suppose we want to get a synthetic dataset with 90% of the original data size real data and add 10% of the original data size randomly perturbed data. With user u and the percentage of the partial data ρ, we will generate a random number ι from 0 to 1, if $\iota < \rho$, we will sample an item i that user u has clicked and add the user-item pair (u, i) to the training dataset. Otherwise, we will jump to a randomly selected item j that user u has not clicked and add the fake user-item pair (u, j) to the training dataset. We will repeat the process until we have collected 90% of the original data size realistic data and 10% of the original data size perturbed data.

A.7 The Configuration of Surrogate Model and Target Models

To raise reproducibility, in Table 4 we report the configuration of each model used in our experiments. Note that we did not tune each model exhaustively but roughly grid search for the hyper-parameters until a reasonable recommendation performance is reached because comparing recommendation performance is not our main focus.

Table 4: The Configuration of Surrogate Model and Target Models.

Dataset	Model	Configuration
MovieLens	Surrogate-WRMF	Latent dimension: 64; Learning rate: 1e-2; Weights for positive feedback: 20; L_2 regularization coefficient: 1e-5; Training epochs: 200;
	WRMF	Latent dimension: 64; Learning rate: 1e-2; Weights for positive feedback: 20; L_2 regularization coefficient: 1e-5; Training epochs: 100;
	NCF	Latent dimension: 64; Learning rate: 1e-2; L_2 regularization coefficient: 1e-5; Training epochs: 100;
	LightGCN	Latent dimension: 64; Learning rate: 1e-2; Number of layers: 3; L_2 regularization coefficient: 1e-5; Training epochs: 100;
Amazon-Video	Surrogate-WRMF	Latent dimension: 32; Learning rate: 1e-2; Weights of positive feedback: 20; Weights of L2 regularization: 1e-5; Training epochs: 200;
	WRMF	Latent dimension: 32; Learning rate: 2e-2; Weights of positive feedback: 20; Weights of L2 regularization: 1e-5; Training epochs: 100;
	NCF	Latent dimension: 64; Learning rate: 1e-2; L_2 regularization coefficient: 1e-5; Training epochs: 200;
	LightGCN	Latent dimension: 64; Learning rate: 1e-2; Number of layers: 3; L_2 regularization coefficient: 1e-5; Training epochs: 100;

A.8 Extra Experimental Results

Analysis on the Vulnerable Items. Note that in the previous subsection, we showed the attack effectiveness only on a set of randomly sampled popular items. In order to analyze the vulnerability of items, the target item sets are sampled to have different popularities in this section. Here we define "Most Pop." items as the items with total clicks (#clicks) above 80 percentile. Similar definitions also apply for "Popular" (60 percentile < #clicks < 80 percentile), "Ordinary" (40 percentile < #clicks < 60 percentile) , "Unpopular" (20 percentile < #clicks < 40 percentile) and "Most Unp." (#clicks < 20 percentile). The result on MovieLens dataset is shown in Table 5. From the table, we can see our attacks, though still successfully promote the target item sets, are less effective for the target items with less popularity. In other words, the cold items are much harder to get promoted. Perhaps this is because cold items are farther away from normal users on the latent space, thus brings more difficulties for the attack.

Table 5: Attack performance on WRMF for target item set with different popularity

Attack	Target Item Popularity				
	Most Pop.	Popular	Ordinary	Unpopular	Most Unp.
None	.5681	.0767	.0456	.0013	.0000
Random	.5737	.0888	.0467	.0074	.0029
Popular	.5996	.0924	.0549	.0070	.0024
PGA	.5737	.0880	.0521	.0052	.0012
SRWA	.5784	.0846	.0512	.0056	.0013
CoVis.	.5871	.0996	.0550	.0072	.0030
RevAdv.	.5880	.1051	.0591	.0069	.0015
RAPU-G	.6080	**.1422**	**.0815**	**.0154**	**.0042**
RAPU-R	**.6101**	.1212	.0716	.0094	.0020

Impact of the Number of Recommended Items. Fig. 6 shows the hit ratios for different numbers of recommended items (i.e., K) when different attacks against WRMF. From the figure, we observe that the proposed attacks are more effective than the existing attacks for different values of K. Moreover, when K is smaller, the hit ratio gains of our attacks over existing attacks are more significant. For instance, when K = 10 and K = 100, the hit ratios of the proposed RAPU-Gimprove upon the best baseline by twice and by 1.5 times, respectively. It is indicated that our attack ranks the target item higher in the recommendation lists than existing attacks.

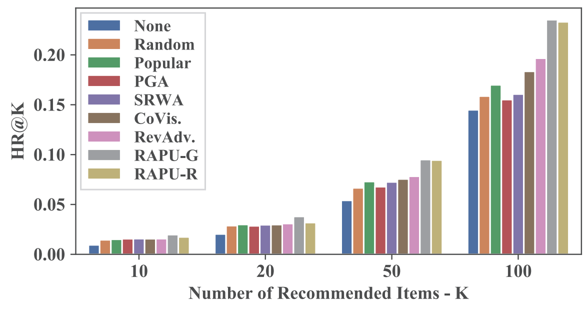


Figure 6: Impact of the Number of Recommended Items.

Impact of the Percentage of the Training Data. Fig. 7 shows the attack performances of different attacks against LightGCN when different ratios of user-item interactions records are removed from the dataset.

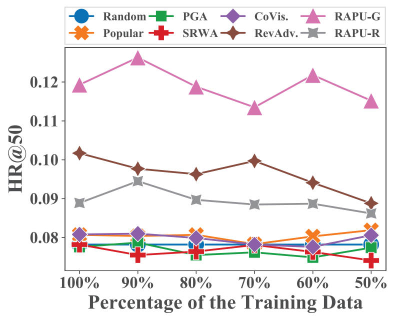


Figure 7: Impact of the Percentage of the Training Data against LightGCN on MovieLens.