Accelerating the Surrogate Retraining for Poisoning Attacks against Recommender Systems

Yunfan Wu
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences, Beijing, China
wuyunfan19b@ict.ac.cn

Qi Cao*
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
Beijing, China
caoqi@ict.ac.cn

Shuchang Tao
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences, Beijing, China
taoshuchang18z@ict.ac.cn

Kaike Zhang
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences, Beijing, China
zhangkaike21s@ict.ac.cn

Fei Sun
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
Beijing, China
sunfei@ict.ac.cn

Huawei Shen
Key Laboratory of AI Safety,
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences, Beijing, China
shenhuawei@ict.ac.cn

ABSTRACT

Recent studies have demonstrated the vulnerability of recommender systems to data poisoning attacks, where adversaries inject carefully crafted fake user interactions into the training data of recommenders to promote target items. Current attack methods involve iteratively retraining a surrogate recommender on the poisoned data with the latest fake users to optimize the attack. However, this repetitive retraining is highly time-consuming, hindering the efficient assessment and optimization of fake users. To mitigate this computational bottleneck and develop a more effective attack in an affordable time, we analyze the retraining process and find that a change in the representation of one user/item will cause a cascading effect through the user-item interaction graph. Under theoretical guidance, we introduce *Gradient Passing* (GP), a novel technique that explicitly passes gradients between interacted user-item pairs during backpropagation, thereby approximating the cascading effect and accelerating retraining. With just a single update, GP can achieve effects comparable to multiple original training iterations. Under the same number of retraining epochs, GP enables a closer approximation of the surrogate recommender to the victim. This more accurate approximation provides better guidance for optimizing fake users, ultimately leading to enhanced data poisoning attacks. Extensive experiments on real-world datasets demonstrate the efficiency and effectiveness of our proposed GP.

*Corresponding author.

RecSys '24, October 14–18, 2024, Bari, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688148

CCS CONCEPTS

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

KEYWORDS

Poisoning Attacks, Recommender Systems, Adversarial Learning

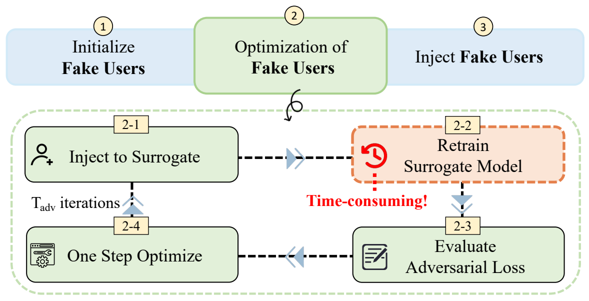
ACM Reference Format:
Yunfan Wu, Qi Cao, Shuchang Tao, Kaike Zhang, Fei Sun, and Huawei Shen. 2024. Accelerating the Surrogate Retraining for Poisoning Attacks against Recommender Systems. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3640457.3688148

1 INTRODUCTION

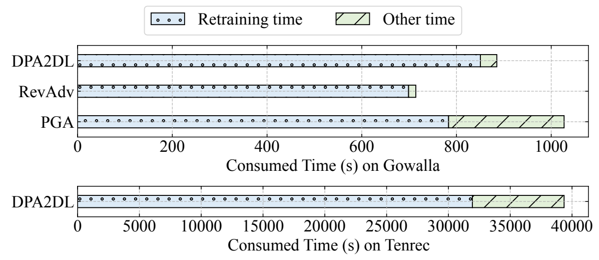
Recommender systems have become an essential component of modern online platforms, providing personalized recommendations that enhance user experience and engagement across various domains [7, 12, 49]. Collaborative filtering (CF) is a widely adopted recommendation scenario, receiving extensive research attention [41]. While the openness and collaborative nature of recommender systems offer convenience to users, they also render these systems vulnerable to adversarial attacks and manipulations [30], emphasizing the need for reliable and secure systems.

Adversaries conduct poisoning attacks by injecting crafted fake users into the training data of recommender systems [1, 24, 48, 57]. In practice, fake accounts are registered for such manipulations, resulting in manipulated recommendations [54]. The business of selling fake YouTube views has been reported, highlighting the prevalence of adversary practices[1]. Given the severe impacts, it is crucial to investigate poisoning attacks against recommender systems. Such research provides a foundation for developing robust defense and improving the trustworthiness of recommenders [54].

[1]https://www.nytimes.com/interactive/2018/08/11/technology/youtube-fake-view-sellers.html



(a) General framework of optimization-based attacks.



(b) Time consumed by existing attack methods.

Figure 1: Retraining surrogate model is an important and time-consuming part of poisoning attacks.

Evolving from heuristic attack strategies, recent research has shifted its focus to optimization-based attacks [24]. These attacks iteratively optimize fake users by utilizing a surrogate recommender and an adversarial loss function. The surrogate recommender can evaluate fake users' attack effectiveness and guide the optimization to minimize the adversarial loss. After each update of fake users, the surrogate recommender has to be retrained on the poisoned data with the latest fake users (Figure 1a). This repetitive surrogate retraining is the most time-consuming part of existing attack methods (Figure 1b). It motivates us to investigate accelerating the surrogate retraining for more efficient and effective attacks.

Existing works mitigate this issue by restricting the retraining time [18, 24, 42]. However, these approaches may reduce the overall attack effectiveness due to underdeveloped surrogate recommenders that behave differently from the victim. Alternatively, some efforts have been made to avoid the retraining process by leveraging the influence function [21, 45, 46, 52]. Nevertheless, the influence function is originally designed to compute a data sample's impact on model training, assuming that the sample has been encountered during training. So it is inaccurate to calculate the influence of a newly crafted adversarial sample without retraining.

In this study, we analyze the retraining process of CF models and find that the recommendation loss requires the representation similarity between interacted user-item pairs. As a result, updating a node's representation in current iteration triggers a cascading effect, affecting the representations of its connected nodes through the user-item interaction graph **in subsequent iterations**.

Inspired by this cascading dynamic, we propose *Gradient Passing (GP)* to accelerate the surrogate retraining and enhance poisoning attacks. During retraining, GP captures the changes of representations via gradients and explicitly passes them between interacted user-item pairs **within one iteration**, thereby approximating the cascading effect and accelerating model convergence. Unlike the practice of message-passing in GNN-based recommenders during forward to improve expressiveness [43], we innovatively leverage gradients as messages in backward to enable faster retraining.

Both theoretical analysis and experiments demonstrate that one training iteration with GP can approximate the effects of multiple original iterations, significantly accelerating the surrogate retraining. It allows for a closer approximation of the surrogate recommender to the victim, enhancing its accuracy in evaluating attack effectiveness and improving the optimization of fake users, which ultimately strengthens poisoning attacks. Experiments on three real-world datasets verify that integrating GP into the state-of-the-art attack method can increase its average effectiveness by 29.57%, 18.02%, and 177.21% while reducing the time cost by 43.27%, 40.54%, and 26.67%.

In this paper, we make the following contributions:

- We introduce a novel method Gradient Passing (GP) based on both intuitive and theoretical analyses, accelerating the retraining process of surrogate recommenders.
- We present the use of GP to enhance data poisoning attacks. It can be integrated into state-of-the-art attack methods and combined with other techniques.
- Extensive experiments on three real-world datasets and six victim recommenders validate the efficiency and effectiveness of GP.

2 RELATED WORK

2.1 Recommender System

Recommender systems have become ubiquitous in online applications in recent years, providing users with personalized suggestions [7, 12, 49]. Collaborative filtering (CF) is one of the most widely adopted recommendation tasks [41]. Its main objective is the top-k recommendation, which aims to generate a personalized ranking list of k items for each user. Early CF methods relied on similarity measures like Pearson correlation [36, 37], while more sophisticated latent factor models were later developed [16, 22, 28]. Recent advancements in deep learning have led to the development of neural CF models, such as autoencoders [26, 38], convolutional neural networks [14], and graph neural networks [13, 43].

This paper focuses on popular two-tower CF models, where two separate towers independently learn user and item representations to effectively capture complex user preferences and item characteristics [15, 43]. Once the latent representations are learned, user-item preference scores can be efficiently calculated using a similarity function like the dot product [49].

2.2 Attack against Recommender System

Early works on poisoning attacks focused on heuristic shilling attacks, such as random attacks [23], bandwagon attacks [1], and others [29, 39]. These attacks rely on the fundamental assumption of CF and generate fake users by heuristic rules. However, they are not specifically optimized for a recommendation model or an adversarial loss, leading to limited attack effectiveness. Recently, optimization-based attacks leveraged a selected surrogate model to obtain the attack feedback and minimize an adversarial loss.

Optimization-based attacks can be divided into two categories based on whether an attack model is utilized. **Model-based** attacks typically leverage Reinforcement Learning (RL) [2, 9, 40, 52] or Generative Adversarial Networks (GANs) [5, 27, 44, 45] to generate fake users. On the other hand, **model-free** approaches commonly employ adversarial gradients [10, 11, 24, 25, 33, 42], influence functions [17, 46], or other priors [3, 18, 47, 51, 53, 57] to optimize fake users without learning an attack model.

The iterative retraining of surrogate recommenders remains the most time-consuming part of current poisoning attacks. In this paper, we investigate our proposed GP technique when integrated into RAPU-R [53] and DPA2DL [18] for two reasons. First, training an attack model like RL or GAN is unstable and may be influenced by irrelevant factors. In contrast, RAPU-R and DPA2DL are model-free attacks that can better demonstrate the effectiveness of GP. Second, they not only achieve state-of-the-art attack performances but also scale well to large datasets. In contrast, other attacks that rely on high-order gradients or influence function can hardly be conducted when dealing with millions of users and items.

2.3 Retraining of Recommender System

In poisoning attacks, a common choice is restricting the time for surrogate retraining [18, 24, 42]. However, it may reduce the overall attack effectiveness due to underdeveloped surrogate recommenders that behave differently from the victim. Alternatively, some efforts have been made to avoid the repeated surrogate retraining by leveraging the influence function [21, 45, 52]. Nevertheless, the influence function is originally designed to compute a data sample's impact on model training, assuming that the sample has been encountered during training. So it is inaccurate to calculate the influence of a newly crafted adversarial sample without retraining.

The efficiency of retraining is also a concern in the field of incremental learning for recommender systems. In this field, researchers study the scenario where new feedback continually arrives. The core challenge lies in efficiently updating a previously trained recommender to maintain high performance on the latest data. Incremental learning methods can be categorized into two main types: sample-based and model-based approaches [55]. Sample-based methods maintain a representative training sample set, circumventing the need to retrain on the entire large dataset and consequently reducing retraining time [8]. Model-based approaches employ a meta-learning model to directly update the parameters of recommendation models without retraining [56]. Drawing inspiration from incremental learning techniques, it has the potential to develop more efficient and effective poisoning attacks.

3 PRELIMINARIES

This section introduces the fundamental concepts of recommender systems and formally defines data poisoning attacks against recommenders. We focus on item promotion attacks and use Hit Ratio as the measure of attack effectiveness. Table 1 summarizes the important mathematical symbols used throughout the paper.

3.1 Recommender System

We formally define the components of a recommender system as follows. $\mathcal{U} = \{\mathrm{u}_1, \mathrm{u}_2, \dots, \mathrm{u}_n\}$ and $\mathcal{I} = \{\mathrm{i}_1, \mathrm{i}_2, \dots, \mathrm{i}_m\}$ denote the sets of n users and m items, respectively. The user-item interaction matrix is represented as $\boldsymbol{I} \in \{0,1\}^{n\times m}$, where $\boldsymbol{I}_{i,j} = 1$ indicates user u_i has interacted with item i_j, and 0 otherwise. We use $\mathcal{I}_{\mathrm{u}_i} = \{\mathrm{i}_j | \boldsymbol{I}_{i,j} = 1\}$ to denote the items interacted by user u_i, and analogously $\mathcal{U}_{\mathrm{i}_j}$ for the users who have interacted with item i_j.

Table 1: Summary of math symbols.

Symbol	Meaning
$\mathcal{U}, \mathcal{I}$	Set of users $\lvert\mathcal{U}\rvert = n$, set of items $\lvert\mathcal{I}\rvert = m$
$\mathrm{u}_i, \mathrm{i}_j$	The user with index i, the item with index j
$\boldsymbol{I}$	User-item interaction matrix, $\boldsymbol{I} \in \{0,1\}^{n\times m}$
$\mathcal{I}_{\mathrm{u}}$	Set of items interacted by user u
$\mathcal{U}_{\mathrm{i}}$	Set of users who interacted with item i
$\boldsymbol{r}_{\mathrm{u}}, \boldsymbol{r}_{\mathrm{i}}$	Representation vectors of user u and item i, $\boldsymbol{r} \in \mathbb{R}^d$
$\boldsymbol{R}$	Representation matrix, $\boldsymbol{R} \in \mathbb{R}^{(n+m)\times d}$
$\boldsymbol{g}_{\mathrm{u}}, \boldsymbol{g}_{\mathrm{i}}$	Original gradient vectors of $\boldsymbol{r}_{\mathrm{u}}$ and $\boldsymbol{r}_{\mathrm{i}}$
$\nabla_{\boldsymbol{R}}\mathcal{L}_{\mathrm{rec}}$	Original gradient matrix of $\boldsymbol{R}$
$(\nabla_{\boldsymbol{R}}\mathcal{L}_{\mathrm{rec}})^{\mathrm{GP}}$	Modified gradient matrix after GP
ξ	Threshold controlling the scope of GP
$\overline{\boldsymbol{A}}^{\mathrm{GP}}$	Normalized GP matrix
l	Number of GP layers
α	Coefficient controlling the weight of GP

Given the user set $\mathcal{U}$, item set $\mathcal{I}$, and interaction matrix $\boldsymbol{I}$, the recommendation model $\mathcal{M}$ learns a preference score $\mathcal{M}(\Theta, \boldsymbol{I}) = \boldsymbol{S} \in \mathbb{R}^{n\times m}$ for each user-item pair. The model parameters Θ are optimized as:

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}_{\mathrm{rec}}(\mathcal{M}(\Theta, \boldsymbol{I}), \boldsymbol{I}), \tag{1}$$

where $\mathcal{L}_{\mathrm{rec}}$ is the recommendation loss.

This paper focuses on the top-k recommendation task. For each user $\mathrm{u}_i \in \mathcal{U}$, the recommender identifies a set of items $\mathcal{T}_{\mathrm{u}_i} \subseteq (\mathcal{I} \setminus \mathcal{I}_{\mathrm{u}_i})$ such that $\lvert\mathcal{T}_{\mathrm{u}_i}\rvert = k$, and for any item $\mathrm{i}_j \in \mathcal{T}_{\mathrm{u}_i}$, $\mathrm{i}_k \in \mathcal{I} \setminus (\mathcal{I}_{\mathrm{u}_i} \cup \mathcal{T}_{\mathrm{u}_i})$, we have $\boldsymbol{S}_{i,j} \geq \boldsymbol{S}_{i,k}$.

3.2 Poisoning Attack against Recommenders

We formalize the item promotion attack as follows. Let $\mathcal{U}^r$ and $\mathcal{U}^f$ denote the sets of real and fake users respectively, with $\lvert\mathcal{U}^r\rvert = n^r$ and $\lvert\mathcal{U}^f\rvert = n^f$. Given the interaction matrix of real users $\boldsymbol{I}^r \in \{0,1\}^{n^r\times m}$, a target item i_t, and the victim recommendation model $\mathcal{M}_v$, data poisoning attacks aim to craft the interactions of n^f fake users $\boldsymbol{I}^f \in \{0,1\}^{n^f\times m}$ under certain budget constraints. Then the generated fake interactions are injected into the training data of the victim recommender to promote the target item. Formally, the attack problem is defined as:

$$\begin{aligned} \max_{\boldsymbol{I}^f} \quad & \mathrm{HR}(\mathcal{M}_v(\Theta^*, \boldsymbol{I}), \mathcal{U}^r, \mathrm{i}_t, k), \\ \text{s.t.} \quad & \Theta^* = \arg\min_{\Theta} \mathcal{L}_{\mathrm{rec}}(\mathcal{M}_v(\Theta, \boldsymbol{I}), \boldsymbol{I}), \\ & \boldsymbol{I} = \mathrm{concatenate}(\boldsymbol{I}^r, \boldsymbol{I}^f), \\ & \forall \mathrm{u} \in \mathcal{U}^f,\ \lvert\mathcal{I}_{\mathrm{u}}\rvert \leq \tau. \end{aligned} \tag{2}$$

The objective is to maximize the Hit Ratio (HR) of the target item i_t among real users $\mathcal{U}^r$ in the victim recommender, which has been retrained using $\boldsymbol{I}$, while ensuring that the interactions of each fake user do not exceed a predefined budget τ. The poisoned interaction matrix $\boldsymbol{I} \in \{0,1\}^{(n^r+n^f)\times m}$ includes both real and fake users.

The HR evaluates the effectiveness of the data poisoning attack on a top-k recommender system. It is defined as:

$$\mathrm{HR}(\mathcal{M}_v(\Theta^*, \boldsymbol{I}), \mathcal{U}^r, \mathrm{i}_t, k) = \frac{1}{|\mathcal{U}^r \setminus \mathcal{U}_{\mathrm{i}_t}|} \sum_{\mathrm{u} \in \mathcal{U}^r \setminus \mathcal{U}_{\mathrm{i}_t}} \mathbb{I}(\mathrm{i}_t \in \mathcal{T}_{\mathrm{u}}), \quad (3)$$

$\mathbb{I}(\cdot)$ is an indicator function, which is 1 if the target item i_t is in $\mathcal{T}_{\mathrm{u}}$, and 0 otherwise.

4 METHODOLOGY

4.1 Intuitive Discussion

The two-tower architecture paired with dot product similarity is well-established in the field of CF [15, 35]. These models maximize the similarity between interacted user-item pairs while minimizing non-interacted pairs. On this basis, we study how injected fake users influence the recommendation of real users. When a fake user u is injected into the recommender system with representation $\boldsymbol{r}_{\mathrm{u}}$, its interacted item $\boldsymbol{r}_{\mathrm{i}}$ will be influenced to maximize their similarity. This change in $\boldsymbol{r}_{\mathrm{i}}$ subsequently affects other users who have interacted with item i, causing a cascading effect.

This cascading dynamic inspires GP, designed to accelerate the retraining of surrogate recommenders. During training, gradients signify the direction and magnitude of changes required to minimize the recommendation loss. Thus, gradients are the core signals in the cascading effect and we propose GP to explicitly pass them between interacted user-item pairs. In this way, a single training iteration with GP could approximate the cascading effect in multiple original iterations and accelerate the convergence of recommenders.

We present an example with a toy dataset and two-dimensional vector representations for users and items, illustrated in Figure 2. Binary Cross Entropy (BCE) is used as the recommendation loss, with one positive and one negative sample per iteration. At the t^{th} iteration, GP passes gradients from $\boldsymbol{r}_{\mathrm{u}_2}$ to $\boldsymbol{r}_{\mathrm{i}_2}$ to preserve their high similarity. The additional gradient information from GP guides $\boldsymbol{r}_{\mathrm{i}_2}$ to optimize towards its convergent cluster, accelerating the retraining. In contrast, with standard Stochastic Gradient Descent (SGD), $\boldsymbol{r}_{\mathrm{i}_2}$ not will be optimized until the positive pair u_2 and i_2 is sampled.

4.2 Theoretical Analysis

BCE loss is a commonly employed point-wise loss function [15], which formulates the recommendation task as a binary classification problem. The BCE loss with dot product similarity is:

$$\mathcal{L}_{\text{rec}} = \sum_{\mathrm{u,i} \in \Omega} \mathrm{softplus}(-\boldsymbol{r}_{\mathrm{u}}^T \boldsymbol{r}_{\mathrm{i}}) + \beta \sum_{\mathrm{u,i} \notin \Omega} \mathrm{softplus}(\boldsymbol{r}_{\mathrm{u}}^T \boldsymbol{r}_{\mathrm{i}}), \quad (4)$$

with $\Omega = \{(\mathrm{u}_i, \mathrm{i}_j) \,|\, \boldsymbol{I}_{i,j} = 1\}$ representing the set of interacting pairs, and β is the coefficient for negative samples. The softplus function is $\mathrm{softplus}(x) = \log(1 + e^x)$ or equivalently $-\log(\sigma(-x))$. Here $\boldsymbol{r}_{\mathrm{u}}, \boldsymbol{r}_{\mathrm{i}} \in \mathbb{R}^d$ are the vector representations of user u and item i, and σ represents the sigmoid function.

Lemma 4.1. *Let $\boldsymbol{R} = vstack(\boldsymbol{r}_{\mathrm{u}_1}, \cdots, \boldsymbol{r}_{\mathrm{u}_n}, \boldsymbol{r}_{\mathrm{i}_1}, \cdots, \boldsymbol{r}_{\mathrm{i}_m}) \in \mathbb{R}^{(n+m) \times d}$ denote the representation matrix for all users and items. The gradient $\nabla_{\boldsymbol{R}} \mathcal{L}_{rec}$ can be derived through **message-passing** on $\boldsymbol{R}$.*

Proof. The gradient vector of $\mathcal{L}_{\text{rec}}$ with regard to the user representation $\boldsymbol{r}_{\mathrm{u}}$ is,

$$\boldsymbol{g}_{\mathrm{u}} = \nabla_{\boldsymbol{r}_{\mathrm{u}}} \mathcal{L}_{\text{rec}} = -\sum_{\mathrm{i} \in \mathcal{I}_{\mathrm{u}}} \sigma(-\boldsymbol{r}_{\mathrm{u}}^T \boldsymbol{r}_{\mathrm{i}}) \boldsymbol{r}_{\mathrm{i}} + \beta \sum_{\mathrm{i} \notin \mathcal{I}_{\mathrm{u}}} \sigma(\boldsymbol{r}_{\mathrm{u}}^T \boldsymbol{r}_{\mathrm{i}}) \boldsymbol{r}_{\mathrm{i}}. \quad (5)$$

For brevity, $\boldsymbol{g}_{\mathrm{u}}$ and $\boldsymbol{g}_{\mathrm{i}}$ are used to denote $\nabla_{\boldsymbol{r}_{\mathrm{u}}} \mathcal{L}_{\text{rec}}$ and $\nabla_{\boldsymbol{r}_{\mathrm{i}}} \mathcal{L}_{\text{rec}}$.

Considering the components of $\boldsymbol{g}_{\mathrm{u}} \in \mathbb{R}^d$ from Equation (5), $\boldsymbol{P}^{\text{grad}} \in \mathbb{R}^{n \times m}$ is constructed as:

$$\boldsymbol{P}^{\text{grad}}_{i,j} = \begin{cases} \sigma(-\boldsymbol{r}_{\mathrm{u}_i}^T \boldsymbol{r}_{\mathrm{i}_j}) & \text{if } \boldsymbol{I}_{i,j} = 1 \\ -\beta\sigma(\boldsymbol{r}_{\mathrm{u}_i}^T \boldsymbol{r}_{\mathrm{i}_j}) & \text{otherwise} \end{cases}. \quad (6)$$

By comparing Equation (5) with (6), the relationship is established:

$$\mathrm{vstack}(\boldsymbol{g}_{\mathrm{u}_1}, \cdots, \boldsymbol{g}_{\mathrm{u}_n}) = -\boldsymbol{P}^{\text{grad}} \mathrm{vstack}(\boldsymbol{r}_{\mathrm{i}_1}, \boldsymbol{r}_{\mathrm{i}_2}, \cdots, \boldsymbol{r}_{\mathrm{i}_m}), \quad (7)$$

where vstack is a function that constructs an $n \times d$ matrix from n vectors, each of size d.

Similarly, the following equation also holds.

$$\mathrm{vstack}(\boldsymbol{g}_{\mathrm{i}_1}, \cdots, \boldsymbol{g}_{\mathrm{i}_m}) = -(\boldsymbol{P}^{\text{grad}})^T \mathrm{vstack}(\boldsymbol{r}_{\mathrm{u}_1}, \boldsymbol{r}_{\mathrm{u}_2}, \cdots, \boldsymbol{r}_{\mathrm{u}_n}), \quad (8)$$

To calculate $\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}} = \mathrm{vstack}(\boldsymbol{g}_{\mathrm{u}_1}, \cdots, \boldsymbol{g}_{\mathrm{u}_n}, \boldsymbol{g}_{\mathrm{i}_1}, \cdots, \boldsymbol{g}_{\mathrm{i}_m})$, we construct $\boldsymbol{A}^{\text{grad}} \in \mathbb{R}^{(n+m) \times (n+m)}$ from $\boldsymbol{P}^{\text{grad}}$,

$$\boldsymbol{A}^{\text{grad}} = \begin{pmatrix} & \boldsymbol{P}^{\text{grad}} \\ (\boldsymbol{P}^{\text{grad}})^T & \end{pmatrix}. \quad (9)$$

Finally, the gradient of $\mathcal{L}_{\text{rec}}$ with regard to $\boldsymbol{R}$ is,

$$\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}} = -\boldsymbol{A}^{\text{grad}} \boldsymbol{R} \in \mathbb{R}^{(n+m) \times d}. \quad (10)$$

□

Next we prove that **Gradient Passing** between user-item pairs could accelerate training for recommender systems.

Proposition 4.2. *There exists a gradient passing matrix $\boldsymbol{A}^{GP} \in \mathbb{R}^{(n+m) \times (n+m)}$. When optimizing a recommender using BCE loss and SGD optimizer, a single iteration using $\boldsymbol{A}^{GP} \nabla_{\boldsymbol{R}} \mathcal{L}_{rec}$ can reach the state after two iterations with the original gradients $\nabla_{\boldsymbol{R}} \mathcal{L}_{rec}$.*

Proof. Assuming that the SGD optimizer for the representation matrix $\boldsymbol{R}$ uses a learning rate α, the update at iteration t is:

$$\boldsymbol{R}_{t+1} = \boldsymbol{R}_t - \alpha \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}(\boldsymbol{R}_t), \quad (11)$$

where $\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}(\boldsymbol{R}_t)$ is the gradient of $\mathcal{L}_{\text{rec}}$ w.r.t. $\boldsymbol{R}$ at iteration t.

Using Equation (10), the update can be rewritten as:

$$\boldsymbol{R}_{t+1} = (\boldsymbol{1} + \alpha \boldsymbol{A}_t^{\text{grad}}) \boldsymbol{R}_t. \quad (12)$$

Here, $\boldsymbol{1}$ represents the identity matrix.

Applying the update rule again for the subsequent step yields:

$$\begin{aligned} \boldsymbol{R}_{t+2} &= (\boldsymbol{1} + \alpha \boldsymbol{A}_{t+1}^{\text{grad}})(\boldsymbol{1} + \alpha \boldsymbol{A}_t^{\text{grad}}) \boldsymbol{R}_t \\ &= \boldsymbol{R}_t + \alpha(\boldsymbol{1} + \alpha \boldsymbol{A}_{t+1}^{\text{grad}}) \boldsymbol{A}_t^{\text{grad}} \boldsymbol{R}_t + \alpha \boldsymbol{A}_{t+1}^{\text{grad}} \boldsymbol{R}_t \\ &= \boldsymbol{R}_t + \alpha \left[(\boldsymbol{2} + \alpha \boldsymbol{A}_{t+1}^{\text{grad}}) \boldsymbol{A}_t^{\text{grad}} \boldsymbol{R}_t + (\boldsymbol{A}_{t+1}^{\text{grad}} - \boldsymbol{A}_t^{\text{grad}}) \boldsymbol{R}_t \right] \\ &= \boldsymbol{R}_t - \alpha \left[\boldsymbol{2} + \alpha \boldsymbol{A}_{t+1}^{\text{grad}} + (\boldsymbol{A}_{t+1}^{\text{grad}} - \boldsymbol{A}_t^{\text{grad}})(\boldsymbol{A}_t^{\text{grad}})^{-1} \right] \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}(\boldsymbol{R}_t). \end{aligned}$$

We define the gradient passing matrix as:

$$\boldsymbol{A}^{\text{GP}} = \boldsymbol{2} + \alpha \boldsymbol{A}_{t+1}^{\text{grad}} + (\boldsymbol{A}_{t+1}^{\text{grad}} - \boldsymbol{A}_t^{\text{grad}})(\boldsymbol{A}_t^{\text{grad}})^{-1}. \quad (13)$$

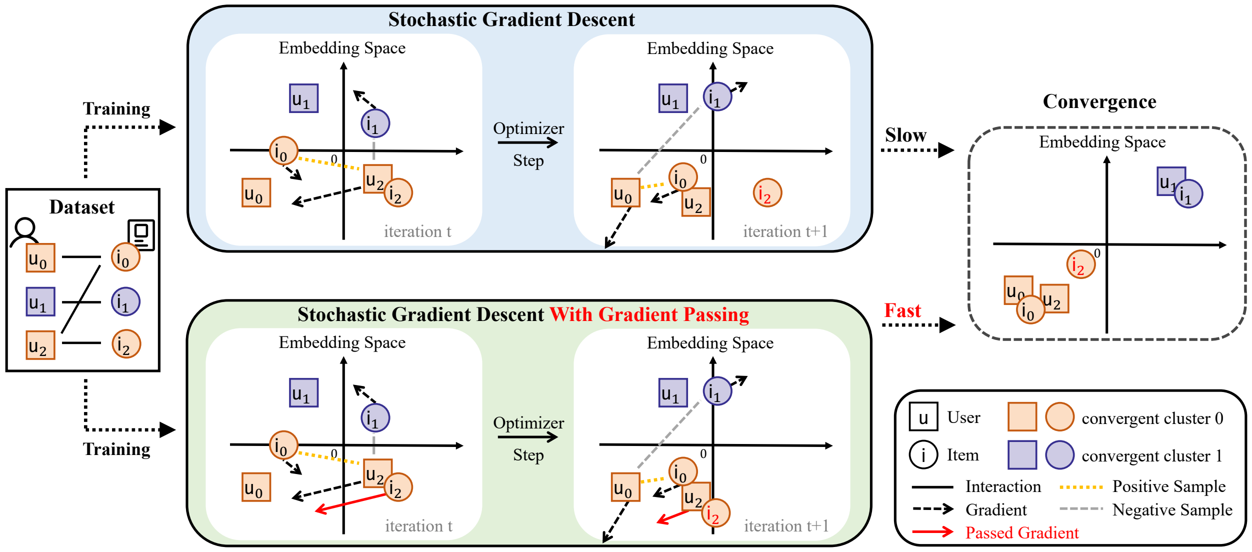


Figure 2: Representation optimization in a surrogate recommender over two iterations: comparing SGD alone (blue) to SGD with GP (green). GP accelerates retraining to the convergence state by passing gradients between interacted user-item pairs.

The resulting update is: $\boldsymbol{R}_{t+2} = \boldsymbol{R}_t - \alpha \boldsymbol{A}^{\text{GP}} \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}(\boldsymbol{R}_t)$. Consequently, a single training iteration with passed gradients $\boldsymbol{A}^{\text{GP}} \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}$ reaches the effect of two standard SGD iterations on $\boldsymbol{R}$. □

4.3 Gradient Passing Strategy

Ideally, the GP matrix $\boldsymbol{A}^{\text{GP}}$ would be defined as in Equation (13). However, directly applying this formula faces several practical challenges. The matrix $\boldsymbol{A}^{\text{grad}}_{t+1}$, required to compute $\boldsymbol{A}^{\text{GP}}$, is unknown at iteration t due to its dependency on future state $\boldsymbol{R}_{t+1}$. Additionally, inverting $\boldsymbol{A}^{\text{grad}}_t$ poses computational difficulties. Moreover, the resultant $\boldsymbol{A}^{\text{GP}}$ is a dense matrix, whose use in GP would be computationally intensive with time complexity $O(nmd)$.

While the exact application of $\boldsymbol{A}^{\text{GP}}$ is impractical, the equation provides valuable **theoretical guidance** for a feasible approach:

- The matrix $\boldsymbol{A}^{\text{GP}}$ is composed of three terms: an identity matrix and two additional terms, carrying distinct weights, with the coefficients 2, α and 1 for three terms.
- $\boldsymbol{A}^{\text{grad}}_{t+1}$ is block anti-diagonal, and $(\boldsymbol{A}^{\text{grad}}_{t+1} - \boldsymbol{A}^{\text{grad}}_t)(\boldsymbol{A}^{\text{grad}}_t)^{-1}$ is block diagonal. The second term facilitates GP *between user-item pairs*, while the third enables GP *within user pairs and item pairs.*

Inspired by these insights, we first introduce a GP matrix $\boldsymbol{A}^{\text{GP-even}}$. Its *even power* can pass gradients *within user pairs and item pairs*, approximating the third term in Equation (13).

$\boldsymbol{A}^{\text{grad}}_{t+1} - \boldsymbol{A}^{\text{grad}}_t$ in the third term suggests, among interacted user-item pairs, the focus should be on those exhibiting a similarity reduction at iteration t. Therefore, we introduce the condition term $\boldsymbol{r}_{\text{u}}^T \boldsymbol{g}_{\text{i}} + \boldsymbol{r}_{\text{i}}^T \boldsymbol{g}_{\text{u}} > \xi_{\text{even}}$, where $\boldsymbol{g}_{\text{u}}$ and $\boldsymbol{g}_{\text{i}}$ represent the original gradients of $\boldsymbol{r}_{\text{u}}$ and $\boldsymbol{r}_{\text{i}}$. This condition is intrinsically interpreted as $\Delta(\boldsymbol{r}_{\text{u}}^T \boldsymbol{r}_{\text{i}}) < -\xi_{\text{even}}$. Specifically,

$$\begin{aligned} -\Delta(\boldsymbol{r}_{\text{u}}^T \boldsymbol{r}_{\text{i}}) &\approx -\boldsymbol{r}_{\text{u}}^T \Delta(\boldsymbol{r}_{\text{i}}) - \boldsymbol{r}_{\text{i}}^T \Delta(\boldsymbol{r}_{\text{u}}) \\ &\approx \boldsymbol{r}_{\text{u}}^T \boldsymbol{g}_{\text{i}} + \boldsymbol{r}_{\text{i}}^T \boldsymbol{g}_{\text{u}}. \end{aligned} \tag{14}$$

It is feasible because GP is performed during backpropagation when the original gradients have already been computed.

Then, the subblock $\boldsymbol{P}^{\text{GP-even}} \in \mathbb{R}^{n \times m}$ of $\boldsymbol{A}^{\text{GP-even}}$ is defined:

$$\boldsymbol{P}^{\text{GP-even}}_{i,j} = \begin{cases} 1 & \text{if } \boldsymbol{I}_{i,j} = 1 \text{ and } \boldsymbol{r}_{\text{u}}^T \boldsymbol{g}_{\text{i}} + \boldsymbol{r}_{\text{i}}^T \boldsymbol{g}_{\text{u}} > \xi_{\text{even}} \\ 0 & \text{otherwise} \end{cases}. \tag{15}$$

$\boldsymbol{A}^{\text{GP-even}}$ is extended from $\boldsymbol{P}^{\text{GP-even}}$ as Equation (9). Next, we normalize it following GCN [20]:

$$\overline{\boldsymbol{A}}^{\text{GP-even}} = \boldsymbol{D}^{-1/2} \boldsymbol{A}^{\text{GP-even}} \boldsymbol{D}^{-1/2}, \tag{16}$$

where $\boldsymbol{D} = \text{diag}(|\mathcal{I}_{\text{u}_1}|, \cdots, |\mathcal{I}_{\text{u}_n}|, |\mathcal{U}_{\text{i}_1}|, \cdots, |\mathcal{U}_{\text{i}_m}|)$ is a diagonal degree matrix, representing the number of interactions.

We apply $2\boldsymbol{l}$ message passing layers to the original gradient $\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}$, which are defined as:

$$\begin{aligned} (\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})_0^{\text{GP-even}} &= \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}}, \\ (\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})_{i+1}^{\text{GP-even}} &= \overline{\boldsymbol{A}}^{\text{GP-even}} (\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})_i^{\text{GP-even}}. \end{aligned} \tag{17}$$

Remember that, $\boldsymbol{A}^{\text{GP-even}}$ is to pass gradients *within user pairs and item pairs.* So we only collect l even index terms and obtain

$$\boldsymbol{G}^{\text{GP-even}} = \sum_{i=1}^{l} (\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})_{2i}^{\text{GP-even}}. \tag{18}$$

To pass gradients *between user-item pairs*, we construct $\boldsymbol{A}^{\text{GP-odd}}$ similarly and collect its odd terms $\boldsymbol{G}^{\text{GP-odd}} = \sum_{i=1}^{l} (\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})_{2i-1}^{\text{GP-odd}}$. The only distinction between $\boldsymbol{A}^{\text{GP-odd}}$ and $\boldsymbol{A}^{\text{GP-even}}$ lies in their different thresholds, ξ_{odd} and ξ_{even}.

Finally, we assign the odd and even terms with different weights $\alpha_{\text{odd}}, \alpha_{\text{even}}$ and modify the gradients of $\boldsymbol{R}$ for the optimizer to perform gradient descent.

$$(\nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}})^{\text{GP}} = \nabla_{\boldsymbol{R}} \mathcal{L}_{\text{rec}} + \alpha^{\text{odd}} \boldsymbol{G}^{\text{GP-odd}} + \alpha^{\text{even}} \boldsymbol{G}^{\text{GP-even}}. \tag{19}$$

Algorithm 1 Enhanced DPA2DL Attack via Gradient Passing

```
Require: Real user interactions $\boldsymbol{I}^r \in \{0,1\}^{n^r \times m}$; target item $\mathrm{i}_t$; surrogate model $\mathcal{M}_s$; recommendation loss $\mathcal{L}_{\text{rec}}$; adversary loss $\mathcal{L}_{\text{adv}}$; attack budgets $n^f$, $\tau$; GP hyperparameters $l$, $\xi$, $\alpha$; DPA2DL hyperparameters.
Ensure: Fake user interactions $\boldsymbol{I}^f \in \{0,1\}^{n^f \times m}$.
 1: Initialize $\boldsymbol{I}^f$ as a zero matrix of size $n^f \times m$.
 2: for $i = 0$ to $n^f - 1$ do
 3:     // The $i^{\text{th}}$ row $\boldsymbol{I}^f_i$ is the interaction vector for the $i^{\text{th}}$ fake user $\mathrm{u}^f_i$.
 4:     Add an interaction between $\mathrm{u}^f_i$ and the target item $\mathrm{i}_t$ to $\boldsymbol{I}^f_i$.
 5:     Initialize parameters $\Theta_s$ for $\mathcal{M}_s$ with user size $n^r + i + 1$.
 6:     Train $\mathcal{M}_s$ on poisoned dataset $[\boldsymbol{I}^r; \boldsymbol{I}^f]$ using $\mathcal{L}_{\text{rec}}$, apply GP.
 7:     Train $\mathcal{M}_s$ with both $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{adv}}$, apply GP on $\mathcal{L}_{\text{rec}}$.
 8:     Predict preference scores $\boldsymbol{S}^f_i$ for $\mathrm{u}^f_i$ using $\mathcal{M}_s$.
 9:     Update $\boldsymbol{I}^f_i$ based on $\boldsymbol{S}^f_i$ with at most $\tau$ interactions.
10: end for
11: return $\boldsymbol{I}^f$.
```

In our proposed GP strategy, four hyperparameters are introduced: ξ_{odd}, ξ_{even}, α_{odd}, and α_{even}. ξ is designed to control the gradient passing scope and α determines the weight.

GP can be incorporated to enhance existing poisoning attacks by enabling a closer approximation of the surrogate recommender to the victim. The specific procedure of a state-of-the-art attack DPA2DL [18] enhanced by GP is shown in Algorithm 1.

Complexity Analysis. The time complexity of GP is $O(\|\boldsymbol{I}\|_0 ld)$. Here, $\|\boldsymbol{I}\|_0$ is the number of interactions. l signifies the number of GP layers, and d is the hidden size of user/item representations.

5 EXPERIMENTS

To thoroughly evaluate the effectiveness of GP in accelerating surrogate retraining and enhancing poisoning attacks, we conduct extensive experiments to analyze the following questions:

- **Q1:** How does GP enhance the efficiency and effectiveness of state-of-the-art poisoning attacks?
- **Q2:** Does GP maintain its effectiveness when pre-training a surrogate model and combined with other techniques?
- **Q3:** Do real-world examples support our motivations and how do hyper-parameters influence the effectiveness of GP?

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on three publicly available benchmark datasets: **Gowalla** [2] [4], **Yelp** [3], and **Tenrec** [4] [50], which represent diverse domains with items corresponding to geographical locations, local businesses, and news articles. To ensure data quality and align with previous work [42], we pre-process the datasets by filtering out users and items with fewer than 15 interactions. For each remaining user, its interactions are chronologically split into training (80%) and validation (20%) sets for training and hyper-parameter tuning of recommenders. For Yelp dataset, we consider ratings above 3 as interactions. For Tenrec dataset, we treat clicks as interactions. Key statistics of the processed datasets are summarized in Table 2.

[2]https://snap.stanford.edu/data/loc-Gowalla.html
[3]https://www.yelp.com/dataset
[4]https://static.qblv.qq.com/qblv/h5/algo-frontend/tenrec_dataset.html

Table 2: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density%
Gowalla	13 149	14 009	535 650	0.290 79
Yelp	35 528	24 573	1 268 345	0.145 28
Tenrec	1 195 207	97 761	40 806 690	0.034 92

5.1.2 Evaluation Protocol. We evaluate the effectiveness of poisoning attacks in a black-box context, utilizing a fixed surrogate model to attack multiple victim recommenders. The selected surrogate models are MF-MSE for PGA and RevAdv, and MF-BCE for DPA2DL and RAPU-R. Each attack generates fake user interactions under certain budgets and injects them into the poisoned training (80%) and validation (20%) sets, which are used to retrain victim recommenders from scratch. For each dataset, we randomly select 5 items from all items as our target item set and repeat this process 5 times, following [25]. The results reported represent the averages and standard deviations across 5 target item sets.

Because the attack targets a set of items $\mathcal{I}_t$, we quantify the attack performance using Recall, which is defined as:

$$\text{Recall@}k = \frac{1}{|\mathcal{U}^r \setminus \mathcal{U}^{\text{all}}_{\mathcal{I}_t}|} \sum_{\mathrm{u} \in \mathcal{U}^r \setminus \mathcal{U}^{\text{all}}_{\mathcal{I}_t}} \frac{|\mathcal{T}_\mathrm{u} \cap \mathcal{I}_t|}{|\mathcal{I}_t \setminus \mathcal{I}_\mathrm{u}|} \tag{20}$$

where $\mathcal{U}^{\text{all}}_{\mathcal{I}_t}$ denotes the set of users who have interacted with all items in $\mathcal{I}_t$, and $\mathcal{T}_\mathrm{u}$ denote the top-k recommendation set for user u. The Recall metric simplifies to HR defined in Equation (3), when the target item set $\mathcal{I}_t$ contains only one item. Consistent with prior research [42], we set $k = 50$, i.e., $|\mathcal{T}_\mathrm{u}| = 50$.

5.1.3 Baseline Attack Methods. The experiments on data poisoning attacks against recommender systems utilize both heuristic (Random, Bandwagon) and optimization-based (PGA, RevAdv, RAPU-R, DPA2DL) attacks as baselines.

- **None**: This refers to scenarios where no attack is executed.
- **Random Attack** [23]: In this attack, fake users interact with the target items along with some random items.
- **Bandwagon Attack** [31]: Building upon Random attack, Bandwagon attack additionally involves some popular items.
- **PGA Attack** [24]: It specifically targets factorization-based recommenders by using an analytic solution of adversarial gradients.
- **RevAdv** [42]: This attack computes higher-order adversarial gradients of retraining by automatic differentiation libraries.
- **RAPU-R** [53]: It reverses the optimization process of recommendation models to construct fake user interactions.
- **DPA2DL** [18]: This attack simulates a deep learning based poisoned recommender, and generates fake users by its predictions.

Optimization-based methods all rely on retraining a surrogate recommender. Among them, PGA and RevAdv require computing adversarial gradients, which can not scale to large datasets. Therefore, we incorporated GP into RAPU-R and DPA2DL to investigate its potential in enhancing poisoning attacks.

5.1.4 Victim Recommender Systems. We evaluated the attack effectiveness using five representative CF methods as the victims, including a robust defense method MF-APR.

- **MF-BPR**: The most basic latent factor model Matrix Factorization (MF) [22] optimized by Bayesian Personalized Ranking loss [34].

Table 3: Data poisoning attack evaluated by Recall@50(%), the most effective attack is highlighted in bold.

Dataset	Attacker	MF-BPR	MF-APR	LightGCN	MultiVAE	NeuMF	MF-BCE	Average[1]
Gowalla	None	0.423 ± 0.344	0.427 ± 0.460	0.409 ± 0.353	0.332 ± 0.295	0.388 ± 0.290	0.304 ± 0.326	0.380 ± 0.339
	Random	0.280 ± 0.204	0.178 ± 0.237	1.131 ± 0.217	0.332 ± 0.360	0.462 ± 0.224	0.265 ± 0.284	0.441 ± 0.247
	Bandwagon	0.386 ± 0.265	0.302 ± 0.316	**1.748 ± 0.162**	0.385 ± 0.404	0.469 ± 0.275	0.233 ± 0.349	0.587 ± 0.284
	PGA	0.528 ± 0.317	0.409 ± 0.186	0.777 ± 0.199	0.544 ± 0.499	0.520 ± 0.231	0.373 ± 0.323	0.525 ± 0.285
	RevAdv	0.521 ± 0.345	0.423 ± 0.260	0.683 ± 0.270	0.472 ± 0.385	0.512 ± 0.277	0.369 ± 0.296	0.496 ± 0.302
	RAPU-R	0.614 ± 0.325	0.588 ± 0.258	0.858 ± 0.287	0.498 ± 0.420	0.617 ± 0.353	0.587 ± 0.546	0.627 ± 0.361
	RAPU-R×2	0.867 ± 0.472	0.909 ± 0.403	0.842 ± 0.384	0.755 ± 0.549	0.766 ± 0.458	0.943 ± 0.583	0.847 ± 0.448
	RAPU-R+GP	0.945 ± 0.513	1.086 ± 0.561	0.656 ± 0.285	0.987 ± 0.508	0.854 ± 0.482	1.086 ± 0.701	0.935 ± 0.487
	DPA2DL	0.574 ± 0.328	0.630 ± 0.374	1.367 ± 0.338	0.793 ± 0.540	0.395 ± 0.343	0.458 ± 0.569	0.703 ± 0.394
	DPA2DL×2	0.884 ± 0.381	1.069 ± 0.497	1.333 ± 0.348	1.451 ± 0.692	0.809 ± 0.415	1.004 ± 0.496	1.092 ± 0.467
	DPA2DL+GP	**1.077 ± 0.476**	**1.450 ± 0.707**	1.493 ± 0.401	**1.699 ± 0.815**	**1.147 ± 0.585**	**1.623 ± 0.989**	**1.415 ± 0.650**
	GP Gain[2]	+21.83% ↑	+35.64% ↑	+12.00% ↑	+17.09% ↑	+41.77% ↑	+61.65% ↑	+29.57% ↑
Yelp	None	0.205 ± 0.194	0.217 ± 0.288	0.220 ± 0.257	0.168 ± 0.204	0.225 ± 0.252	0.342 ± 0.538	0.229 ± 0.287
	Random	0.099 ± 0.071	0.053 ± 0.064	1.529 ± 0.180	0.156 ± 0.169	0.347 ± 0.256	0.260 ± 0.435	0.407 ± 0.184
	Bandwagon	0.211 ± 0.140	0.237 ± 0.120	1.197 ± 0.210	0.326 ± 0.138	0.291 ± 0.231	0.135 ± 0.216	0.399 ± 0.166
	PGA	0.245 ± 0.148	0.461 ± 0.312	0.483 ± 0.313	0.238 ± 0.209	0.343 ± 0.227	0.287 ± 0.444	0.343 ± 0.271
	RAPU-R	0.193 ± 0.192	0.448 ± 0.223	1.193 ± 0.424	0.529 ± 0.249	0.269 ± 0.295	0.198 ± 0.313	0.472 ± 0.252
	RAPU-R×2	0.236 ± 0.227	0.499 ± 0.267	1.140 ± 0.392	0.514 ± 0.206	0.275 ± 0.331	0.229 ± 0.373	0.482 ± 0.283
	RAPU-R+GP	0.533 ± 0.226	0.863 ± 0.309	0.323 ± 0.161	0.398 ± 0.246	0.481 ± 0.300	0.693 ± 0.559	0.549 ± 0.296
	DPA2DL	0.862 ± 0.202	1.676 ± 0.188	1.560 ± 0.431	1.450 ± 0.338	0.761 ± 0.389	1.409 ± 0.579	1.286 ± 0.329
	DPA2DL×2	0.882 ± 0.295	1.538 ± 0.332	1.575 ± 0.436	1.616 ± 0.408	0.705 ± 0.329	1.508 ± 0.789	1.304 ± 0.403
	DPA2DL+GP	**0.992 ± 0.391**	**1.742 ± 0.697**	**1.684 ± 0.565**	**1.786 ± 0.616**	**1.048 ± 0.607**	**1.985 ± 1.381**	**1.539 ± 0.654**
	GP Gain	+12.47% ↑	+13.26% ↑	+6.92% ↑	+10.51% ↑	+48.65% ↑	+31.63% ↑	+18.02% ↑
Tenrec	None	0.014 ± 0.020	0.014 ± 0.011	0.027 ± 0.027	0.010 ± 0.013	OOM[3]	0.012 ± 0.013	0.015 ± 0.017
	Random	0.001 ± 0.000	0.043 ± 0.013	**0.152 ± 0.020**	0.198 ± 0.030	OOM	0.284 ± 0.011	0.135 ± 0.006
	Bandwagon	0.006 ± 0.003	0.032 ± 0.012	0.037 ± 0.006	0.162 ± 0.022	OOM	0.074 ± 0.014	0.062 ± 0.005
	RAPU-R	0.004 ± 0.004	0.009 ± 0.009	0.011 ± 0.003	0.061 ± 0.018	OOM	0.007 ± 0.003	0.018 ± 0.007
	RAPU-R×2	0.004 ± 0.002	0.005 ± 0.003	0.011 ± 0.005	0.012 ± 0.006	OOM	0.022 ± 0.007	0.011 ± 0.004
	RAPU-R+GP	0.006 ± 0.004	0.008 ± 0.010	0.013 ± 0.007	0.011 ± 0.003	OOM	0.032 ± 0.015	0.014 ± 0.006
	DPA2DL	0.070 ± 0.010	0.083 ± 0.019	0.124 ± 0.022	0.137 ± 0.042	OOM	0.068 ± 0.007	0.096 ± 0.015
	DPA2DL×2	0.077 ± 0.021	0.064 ± 0.008	0.077 ± 0.016	0.079 ± 0.059	OOM	0.099 ± 0.008	0.079 ± 0.014
	DPA2DL+GP	**0.099 ± 0.017**	**0.164 ± 0.036**	0.103 ± 0.015	**0.395 ± 0.118**	OOM	**0.335 ± 0.063**	**0.219 ± 0.015**
	GP Gain	+28.57% ↑	+156.25% ↑	+33.76% ↑	+400.00% ↑	OOM	+238.38% ↑	+177.21%

[1]: Average Recall@50(%) across 6 victim recommender systems.
[2]: $(\text{Recall@50}_{\text{DPA2DL+GP}} - \text{Recall@50}_{\text{DPA2DL}\times 2})/\text{Recall@50}_{\text{DPA2DL}\times 2}$.
[3]: PGA, RevAdv attacker and NeuMF recommender cannot be applied to large-scale datasets, due to GPU memory limit.

Table 4: Time (s) Comparison: three DPA2DL variants.

Dataset	DPA2DL	DPA2DL×2	DPA2DL+GP	Reduction
Gowalla	885	1761	999	43.27% ↓
Yelp	2730	5500	3270	40.54% ↓
DPA2DL	39358	72140	52899	26.67% ↓

- **MF-APR**: This variant of MF utilizes the Adversarial Personalized Ranking framework [34], enhancing the robustness of BPR through adversarial training on embedding parameters.
- **Mult-VAE** [26]: It employs the variational autoencoder architecture [19] to encode and decode users' interaction behaviors.
- **NeuMF** [15]: It employs MLP to model the nonlinear interactions between the representations of users and items.
- **LightGCN** [13]: A state-of-the-art recommendation method, using a simplified version of Graph Convolutional Network (GCN).
- **MF-BCE**: MF trained by BCE loss function, which is the surrogate recommender used in our implemented RAPU-R and DPA2DL.

5.1.5 Parameter Settings. We implement GP using PyTorch [32]. The entire source code, including data preparation, hyper-parameter tuning, baseline attack methods, and victim recommenders, is accessible on GitHub[5]. For all attack methods and recommenders, we adjust their hyper-parameters for each dataset on the validation set. Specific to GP, we set the GP layer l to 2 by default. The threshold ξ is tuned across $\{-\infty, 0, \infty\}$, and the weight α across $\{0.1, 1, 10, 100, 1000\}$. We introduce a small number of fake users, amounting to 1% of real users, denoted as $n^f = 0.01n^r$. The interaction budget τ is equal to the average number of interactions across real users in each dataset.

[5]https://github.com/WuYunfan/GradientPassingAttack

5.2 Enhancing Poisoning Attacks (Q1)

This section investigates black-box poisoning attacks, with detailed attack settings described in Sections 5.1.2 and 5.1.5. Notably, hyper-parameters of attackers, including surrogate learning rate and ℓ_2 regularization, are optimized before applying GP. To control the time costs, we limit the number of retraining epochs to 1 for RAPU-R, DPA2DL, and their GP-enhanced variants. The number of training iterations depends on dataset size and batch size. For Yelp and Tenrec, GP is applied with probabilities of 0.5 and 0.25, respectively, rather than at every iteration. Additionally, we evaluate RAPU-R×2 and DPA2DL×2 with 2 retraining epochs to assess GP's efficiency.

Table 3 presents the results using Recall@50(%) = Recall@50 × 100 as the primary metric, while Table 4 compares the time costs of three DPA2DL variants. Among the baselines, DPA2DL emerges as the most scalable and advanced method. DPA2DL×2 generally outperforms DPA2DL, particularly when the victim model is MF-BCE, which aligns with its surrogate model. This suggests that a

Table 5: Data poisoning attack on single unpopular target item, evaluated by HR@50(%).

Dataset	Attacker	MF-BPR	MF-APR	LightGCN	MultiVAE	NeuMF	MF-BCE	Average
Gowalla	DPA2DL×2	0.508 ± 0.092	0.532 ± 0.140	**1.010 ± 0.172**	1.996 ± 0.303	0.468 ± 0.113	0.660 ± 0.193	0.863 ± 0.110
	DPA2DL+GP	**1.304 ± 0.464**	**1.217 ± 0.329**	0.969 ± 0.220	**2.262 ± 0.372**	**0.856 ± 0.160**	**0.913 ± 0.270**	**1.253 ± 0.274**
Yelp	DPA2DL×2	1.450 ± 0.102	2.359 ± 0.115	1.565 ± 0.053	2.846 ± 0.271	1.217 ± 0.241	1.736 ± 0.164	1.862 ± 0.103
	DPA2DL+GP	**1.858 ± 0.168**	**2.513 ± 0.338**	**1.712 ± 0.132**	**3.083 ± 0.195**	**1.434 ± 0.175**	**1.900 ± 0.166**	**2.084 ± 0.110**

Table 6: Attack performances of four DPA2DL variants evaluated by Recall@50(%), when the surrogate is pre-trained.

Dataset	Attacker	MF-BPR	MF-APR	LightGCN	MultiVAE	NeuMF	MF-BCE	Average
Gowalla	Pre-train	1.039 ± 0.568	1.250 ± 0.719	1.376 ± 0.457	**1.438 ± 0.789**	1.084 ± 0.830	1.783 ± 1.516	1.328 ± 0.804
	+Sample	1.779 ± 0.494	2.522 ± 0.450	5.188 ± 2.312	0.850 ± 0.397	**2.863 ± 0.855**	**20.705 ± 1.786**	5.651 ± 0.842
	+Sample×2	1.722 ± 0.443	1.964 ± 0.504	5.230 ± 2.120	0.906 ± 0.464	1.527 ± 0.670	8.646 ± 1.745	3.332 ± 0.799
	+Sample+GP	**4.540 ± 1.791**	**4.981 ± 1.691**	**14.103 ± 5.367**	1.091 ± 0.548	1.870 ± 0.472	17.193 ± 2.685	**7.296 ± 1.896**
Yelp	Pre-train	0.869 ± 0.348	1.619 ± 0.621	1.444 ± 0.342	**1.777 ± 0.392**	1.139 ± 0.549	2.650 ± 1.686	1.583 ± 0.610
	+Sample	1.801 ± 0.348	2.504 ± 0.380	11.038 ± 1.008	0.897 ± 0.214	1.160 ± 0.441	14.086 ± 2.499	5.248 ± 0.594
	+Sample×2	1.223 ± 0.357	1.652 ± 0.483	11.631 ± 1.045	0.801 ± 0.327	1.031 ± 0.385	9.759 ± 2.608	4.349 ± 0.696
	+Sample+GP	**8.024 ± 0.914**	**11.703 ± 0.854**	**13.132 ± 0.960**	1.557 ± 0.753	**2.420 ± 0.639**	**21.190 ± 3.480**	**9.671 ± 0.767**

more accurate surrogate model, closely approximating the victim recommender, yields a more potent attack.

DPA2DL+GP is the most effective attack method, primarily due to GP's ability to accelerate iterative surrogate retraining and provide more accurate feedback for optimizing fake users. Specifically, GP improves the average attack effectiveness by 29.57%, 18.02%, and 177.21% across three datasets while reducing attack time by 43.27%, 40.54%, and 26.67% when comparing DPA2DL+GP with DPA2DL×2. By assigning distinct weights and thresholds for odd and even terms, GP strategically focuses on influential passing paths, surpassing the effects of simply doubling retraining epochs. RAPU-R exhibits similar results to DPA2DL when enhanced by GP, except for its failure on the large Tenrec dataset.

Attack results against LightGCN differ from other victims, possibly due to its graph architecture inadvertently facilitating the propagation of attack influence. However, as none of the baselines consider it as the surrogate recommender, transferability to LightGCN is not guaranteed. The robust framework APR does not consistently enhance resistance against attacks, because it targets parameter perturbation attacks rather than data poisoning. Large standard deviations in Table 3 are attributed to the diverse characteristics of random target item sets. Table 5 provides additional experiments targeting a single unpopular item, further demonstrating GP's effectiveness in promoting less-favored items.

5.3 Generalizability in Pre-training Setting (Q2)

The surrogate recommender is repeatedly retrained on the poisoned dataset with the latest fake users. However, the majority of the poisoned data, i.e., real user interactions, remains unchanged. Therefore, it is feasible to pre-train a surrogate recommender using only real interactions. Then the parameter weights from the pre-trained surrogate can be used to initialize the iterative surrogate retraining during fake users' optimization. It is expected to yield a better surrogate recommender under limited retraining epochs.

We investigate integrating a pre-trained surrogate recommender with the sampling strategy inspired by incremental learning. Specifically, a surrogate is first pre-trained on real interactions with sufficient epochs. During each retraining, a sampled poisoned dataset is constructed, comprising 10% randomly sampled real interactions and all fake ones, emphasizing the attack impact of fake users. The effectiveness and generalizability of GP are assessed when combined with these techniques. Four variants of DPA2DL are compared: **Pre-train**, **Pre-train+Sampling**, **Pre-train+Sampling**×2 and **Pre-train+Sampling+GP**.

The outcomes are summarized in Table 6. A comparison between Pre-train and original DPA2DL (Table 3) shows the efficacy of pre-training a surrogate recommender beforehand. The combination of pre-training and sampling further enhances the attack significantly. However, doubling the retraining epochs decreases the performance when comparing Pre-train+Sampling×2 with Pre-train+Sampling. It may be attributed to potential over-fitting to the biased sampled dataset, as it only contains partial real interactions. Consequently, over-training on it may reduce the accuracy of the surrogate. GP alleviates this by training on the sampled dataset while passing gradients on the whole dataset, further improving the attack performance. While the strategy of constructing a sampled dataset may introduce the over-fitting issue, it is effective in many cases, underlining the under-explored potential of incremental learning techniques in enhancing poisoning attacks.

5.4 More Analyses (Q3)

5.4.1 Gradient Similarity between Interacted User-item Pairs. Our proposed GP is primarily driven by the intuition that gradients between interacted user-item pairs show high similarity during a period. Thus, explicitly passing gradients within every training iteration can bring additional optimization signals for users and items, accelerating the surrogate retraining. To further support this hypothesis, we compute the cosine similarity of average gradients aggregated over one epoch for interacted user-item pairs. The mean and standard deviation across all pairs are recorded. For comparison, we also select an equal number of random user-item pairs.

Figure 3 shows the outcomes on Gowalla and Yelp datasets. There is a clear difference between the similarity of interacted pairs and random ones. During early training, gradient similarity among interacted pairs initially rises exceeding 0.5, then diminishes to 0. This is because the representations of users and items start with

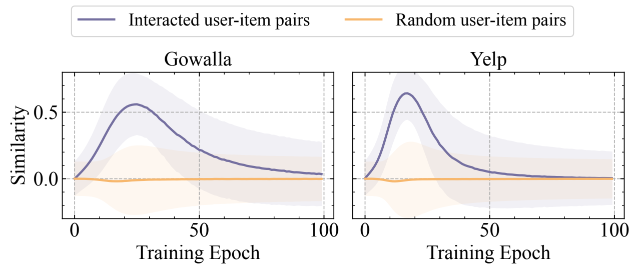


Figure 3: Comparison of gradient similarity between interacted and random user-item pairs.

random initialization at epoch 0 and undergo optimization to find optimal positions in the embedding space. So similarity increases during this optimization process. Towards the end of training, all gradients have small values with decreased similarity.

5.4.2 *Retraining Enhancement of GP.*

Retraining the surrogate recommender is crucial in data poisoning attacks. To demonstrate the effectiveness of GP in accelerating retraining and obtaining a more accurate surrogate, we conduct experiments on Gowalla to evaluate the surrogate's capability in replicating the victim's behavior. Two CF methods, MF and LightGCN, along with two loss functions, BPR and BCE, are employed in the experiments.

For each experiment, a victim recommender is first trained over 1000 epochs. We then train a surrogate recommender with the same architecture, but under different epoch constraints (1, 5, 10, 50, 100). The similarity between the recommendation lists of surrogate and victim recommenders is evaluated by Jaccard Index [6] averaged across all users. The surrogates trained with and without GP are compared, using their optimal learning rate and ℓ_2 regularization specifically for each experiment.

The results of retraining are illustrated in Figure 4 with consumed time annotated by texts. Surrogate recommenders trained with GP consistently achieve higher similarity than the original ones under same epochs, demonstrating the effectiveness and generalizability of GP across different models and loss functions. It underscores the capability of GP to enhance the behavioral similarity between the surrogate and the victim recommender, potentially leading to stronger attacks. Moreover, training with GP for just 5 epochs attains results comparable to or even surpassing the original training for 10 epochs, highlighting the efficiency of GP.

5.4.3 *Hyper-parameter Analysis.*

We conduct experiments to investigate the effectiveness of GP with different hyper-parameters. The results attacking MF-BCE recommender on Gowalla dataset are presented in Figure 5. By default, ξ_{odd} and ξ_{even} are set to 0 and $-\infty$, respectively, while α_{odd} and α_{even} are set to 1 and 10.

The optimal thresholds ξ and weights α for odd and even terms differ, confirming the validity of our GP design. The even terms related to *GP within user pairs and item pairs*, have a more significant impact. It suggests that the gradients of users may exhibit some incongruity with those of items, leading users to prefer adopting gradients from other users rather than items. Furthermore, GP surpasses DPA2DL across most hyper-parameter configurations.

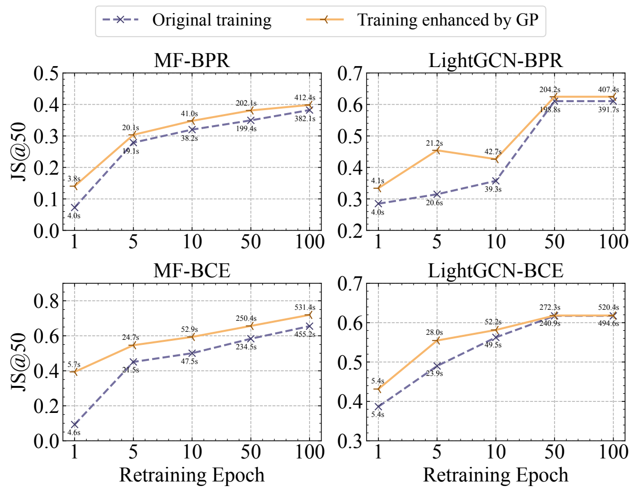


Figure 4: Jaccard Similarity between the surrogate and victim recommenders across various retraining epochs, on Gowalla.

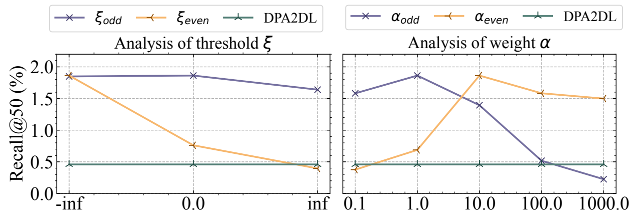


Figure 5: Hyperparameter Analysis on Gowalla.

6 CONCLUSION

We propose Gradient Passing (GP), a novel technique that accelerates the surrogate retraining in poisoning attacks by passing gradients between the representations of interacted user-item pairs during backpropagation. Through theoretical analysis and extensive experiments on real-world datasets, we demonstrate that GP can significantly accelerate the retraining process. When integrated into existing attack methods, GP improves their attack effectiveness by enabling a closer approximation of the surrogate recommender to the victim and providing better attack feedback for optimizing fake users. Since most optimization-based poisoning attacks require the time-consuming surrogate retraining, GP provides a simple yet effective solution to enhance attacks against recommender systems. As for defense, an effective approach to mitigate the risk is preventing the leakage of interaction data to potential attackers. By securing user data, we can significantly reduce the effectiveness of such poisoning attacks. Further research can explore the potential of GP in enhancing general training of recommenders and extend GP from CF to other tasks like sequential recommendation.

ACKNOWLEDGMENTS

This work is funded by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB0680101, and the National Natural Science Foundation of China under Grant Nos. 62272125, 62102402, U21B2046. Huawei Shen is also supported by Beijing Academy of Artificial Intelligence (BAAI).

REFERENCES

[1] Robin Burke, Bamshad Mobasher, and Runa Bhaumik. 2005. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of 3rd international workshop on intelligent techniques for web personalization (ITWP 2005), 19th international joint conference on artificial intelligence (IJCAI 2005)*. 17–24.

[2] Jingfan Chen, Wenqi Fan, Guanghui Zhu, Xiangyu Zhao, Chunfeng Yuan, Qing Li, and Yihua Huang. 2022. Knowledge-enhanced Black-box Attacks for Recommendations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 108–117.

[3] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, and Gabriele Tolomei. 2023. The dark side of explanations: Poisoning recommender systems with counterfactual examples. *arXiv preprint arXiv:2305.00574* (2023).

[4] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '11)*. 1082–1090.

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

[6] Luciano da F Costa. 2021. Further generalizations of the Jaccard index. *arXiv preprint arXiv:2110.09619* (2021).

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 191–198.

[8] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. Real-time top-n recommendation in social streams. In *Proceedings of the sixth ACM conference on Recommender systems*. 59–66.

[9] Wenqi Fan, Tyler Derr, Xiangyu Zhao, Yao Ma, Hui Liu, Jianping Wang, Jiliang Tang, and Qing Li. 2021. Attacking black-box recommendations via copying cross-domain user profiles. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1583–1594.

[10] 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.

[11] 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.

[12] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35 (1992), 61–70.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[14] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2227–2233.

[15] 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.

[16] Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 89–115.

[17] Chengzhi Huang and Hui Li. 2023. Single-User Injection for Invisible Shilling Attack against Recommender Systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 864–873.

[18] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data poisoning attacks to deep learning based recommender systems. In *NDSS*.

[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[23] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*. 393–402.

[24] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* 29 (2016).

[25] Haoyang LI, Shimin DI, and Lei Chen. 2022. Revisiting Injective Attacks on Recommender Systems. *Advances in Neural Information Processing Systems* 35 (2022), 29989–30002.

[26] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[27] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking recommender systems with augmented user profiles. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 855–864.

[28] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[29] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. 2005. Effective attack models for shilling item-based collaborative filtering systems. In *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD*, Vol. 2005.

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

[31] Michael P O'Mahony, Neil J Hurley, and Guénolé CM Silvestre. 2005. Recommender systems: attack types and strategies. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 1*. 334–339.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[33] Fulan Qian, Bei Yuan, Hai Chen, Jie Chen, Defu Lian, and Shu Zhao. 2023. Enhancing the Transferability of Adversarial Examples Based on Nesterov Momentum for Recommendation Systems. *IEEE Transactions on Big Data* (2023).

[34] S. Rendle, C. Freudenthaler, Zeno Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*.

[35] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 240–248.

[36] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[37] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[38] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[39] Carlos E Seminario and David C Wilson. 2014. Attacking item-based recommender systems with power items. In *Proceedings of the 8th ACM Conference on Recommender systems*. 57–64.

[40] 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.

[41] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in artificial intelligence* (Jan. 2009).

[42] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 318–327.

[43] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[44] Yanling Wang, Yuchen Liu, Qian Wang, Cong Wang, and Chenliang Li. 2023. Poisoning Self-supervised Learning Based Sequential Recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 300–310.

[45] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2021. Triple adversarial learning for influence based poisoning attack in recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1830–1840.

[46] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2023. Influence-Driven Data Poisoning for Robust Recommender Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[47] Yiqing Wu, Ruobing Xie, Zhao Zhang, Yongchun Zhu, FuZhen Zhuang, Jie Zhou, Yongjun Xu, and Qing He. 2023. Attacking Pre-trained Recommendation. *arXiv preprint arXiv:2305.03995* (2023).

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

[49] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. 974–983.

[50] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. 2022. Tenrec: A Large-scale Multipurpose

Benchmark Dataset for Recommender Systems. *Advances in Neural Information Processing Systems* 35 (2022), 11480–11493.

[51] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-box attacks on sequential recommenders via data-free model extraction. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 44–54.

[52] 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.

[53] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and 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 & Data Mining*. 2154–2164.

[54] Kaike Zhang, Qi Cao, Fei Sun, Yunfan Wu, Shuchang Tao, Huawei Shen, and Xueqi Cheng. 2023. Robust Recommender System: A Survey and Future Directions. *arXiv preprint arXiv:2309.02057* (2023).

[55] Peiyan Zhang and Sunghun Kim. 2023. A survey on incremental update for neural recommender systems. *arXiv preprint arXiv:2303.02851* (2023).

[56] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1479–1488.

[57] Yihe Zhang, Xu Yuan, Jin Li, Jiadong Lou, Li Chen, and Nian-Feng Tzeng. 2021. Reverse Attack: Black-box Attacks on Collaborative Recommendation. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 51–68.