PATR: A Novel Poisoning Attack Based on Triangle Relations Against Deep Learning-Based Recommender Systems

Meiling Chao[1,3], Min Gao[1,3](✉), Junwei Zhang[1,3], Zongwei Wang[1,3], Quanwu Zhao[2,3], and Yulin He[3]

[1] School of Big Data and Software Engineering, Chongqing University, Chongqing, China
{meilingchao,gaomin,jw.zhang,zongwei}@cqu.edu.cn

[2] School of Economics and Business Administration, Chongqing University, Chongqing, China
zhaoquanwumx@cqu.edu.cn

[3] Chongqing Aerospace Polytechnic, Chongqing, China

Abstract. Recommender systems (RSs) have emerged as an effective way to deal with information overload and are very popular in e-commerce. However, because of the open nature of collaborative characteristics of the systems, RSs are susceptible to poisoning attacks, which inject fake user profiles into RSs to increase or decrease the recommended frequency of the target item. The traditional poisoning attack methods (such as random attack and average attack) are easy to be detected and lack of generality since they usually use global statistics, e.g., the number of each user's ratings and the average rating for filler items. Moreover, as deep learning (DL) becomes more widely used in RSs, attackers are likely to use related techniques to attack RSs. To explore the robustness of DL-based RSs under the possible attacks, we propose a novel poisoning attack with triangle relations (PATR). The triangle relations refer to the balance among a fake user and two real users, aiming to improve attack performance. We also present a novel fake & real sampling strategy, i.e., sampling a set of fake users from the real users, to decrease the possibility of being detected. Comprehensive experiments on three public datasets show that PATR outperforms traditional poisoning attacks on attack effectiveness and anti-detection capability.

Keywords: Deep learning · Poisoning attack · Recommender system · Triangle relation

1 Introduction

Recommender systems (RSs) are prevalent in e-commerce since they provide users with a critical discovery mode to mitigate the difficulties in finding items that users are interested in [1,2]. The ability to solve information overload has

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 435–450, 2021.
https://doi.org/10.1007/978-3-030-92638-0_26

driven RSs be widely used in industries (e.g., Amazon, Netflix, and Facebook) [3]. RSs assist users in finding items and help merchants promote new products and increase retail sales. Unfortunately, due to the openness of the rating systems in RSs, malicious users unscrupulously attack RSs to achieve nefarious goals [4,5]. Since RSs have a profound impact on the e-commerce industry, researchers should take the initiative to consider the possible damages to RSs to protect customers' rights and interests better.

There is much effort has been devoted to studying how to spoof RSs to defend against malicious attacks. A variety of attacks such as sybil attack (i.e., illegally infer a user's preference) [6], unorganized attack (i.e., different attackers attack the RS without organization), and powerful user attack (i.e., select most powerful users who can impact RSs) [7] have been studied. In this paper, we focus on the poisoning attacks [8], which were initially referred to as shilling attacks [10,11], where malicious users inject fake user profiles (i.e., carefully crafted ratings) into RSs based on the statistical rating information during the training time. For example, the average attack is one of the poisoning attacks that assigns the highest rating to a target item to be promoted and assigns an average rating to a randomly sampled group of items [11]. Furthermore, we can divide the existing poisoning attacks into push attacks and nuke attacks according to the purpose. The push attacks assign the highest rating on the target item to improve the recommended frequency, and the nuke attacks do the opposite [8,11–13]. The poisoning attacks can be beneficial to unscrupulous merchants for increasing their retail sales and reducing their competitors' retail sales. Since the two types are similar, we only consider push attacks in this paper. Researchers have experimented successfully with poisoning attacks on real-world RSs, such as YouTube, Google search, Amazon, and Yelp [14]. Moreover, Large companies such as Sony, Amazon, and eBay have been attacked in real life [11]. Although all of these existing approaches have proved effective in some cases, they still have the following challenges:

(1) **Easy to be detected:** The generated user profiles lack personalized behavior patterns of real users, which are easily detected [15,16].
(2) **Low attack effectiveness:** According to the way the statistics are calculated, the traditional poisoning attacks are effective on some traditional collaborative filtering (CF) methods but do not do well on deep learning (DL) based RSs, which also means lack of generality [17,18].
(3) **Lack of effective metrics:** In the field of ranking-based recommender algorithms, the hit ratio (HK) is generally used to calculate the number of the target item recommended to real users, which cannot measure the disorder of top-K recommendation lists [10,12].

To address the above challenges and explore the potential security problems such as DL-based attacks of RSs, we propose a novel poisoning attack based on triangle relations (PATR), which includes two parts, a pre-training module, and a reconstruction module. For the pre-training module, we design triangle relations to generate more informative user embeddings to improve the anti-detection capability. As shown in Fig. 1, two cases are considered according to the

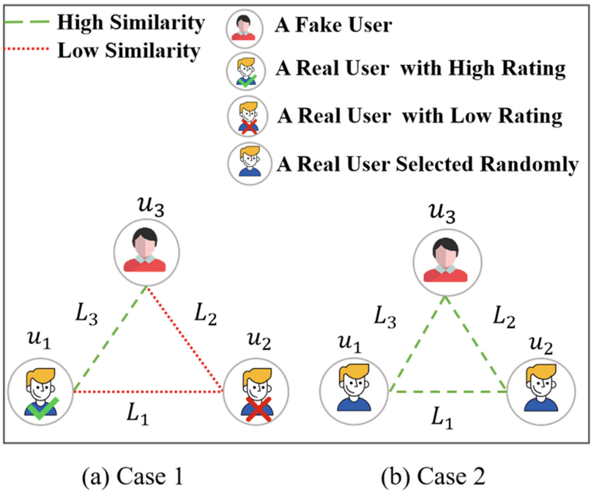

Fig. 1. Triangle relations are designed for the balance of three users. Two cases are considered according to the target item.

target item, where we focus on the balance of one fake user and two real users. We use Graph Convolutional Matrix Commissions (GCMC) [19] designed for recommendation scenarios to implement the pre-training module. We creatively propose a fake & real sampling strategy for the reconstruction module to generate the initial fake user representations. Then we use the convolutional auto-encoder (CAE) [20], which is easy to train and has a lower time cost, to reconstruct the enhanced fake user representations with the output of the pre-training module. We consider these deep learning (DL) techniques can help our model attack DL-based RSs. Our contributions are as follows:

(1) We propose a pre-training module based on triangle relations to assist in attacking RSs. The pre-training module can generate user embeddings with real user features, which reduce the probability of being detected; moreover, we apply CAE to the reconstruction module and combine the outputs of the pre-training module to reconstruct a set of enhanced fake users.
(2) According to heuristic learning, we present a novel fake & real sampling strategy to initialize fake user profiles. In addition to directly injecting fake users, we creatively sample a group of active users directly from real users. The ablation experiment proves that our sampling strategy is helpful for anti-detection capability.
(3) We present a new metric named top-K shift (TKS) to measure the disorder of the top-K recommendation lists affected. Our experimental results show that PATR can effectively attack DL-based RSs, and the anti-detection capability against two detectors is better.

2 Related Work

Generally speaking, the traditional RSs mainly refers to the RSs based on collaborative filtering (CFRSs) [21], which has been successfully applied to practical scenarios filter out unwanted resources. Among various CFRSs, matrix factor factorization (MF) [2] is the most popular one. It utilizes potential feature vectors to represent users and items and projects them into the shared potential space. In recent years, DL develops rapidly, which has been applied to RSs. The DL-based RSs have better performance than the traditional ones because the DL-based models are more consistent with the user-item interaction to improve the recommendation accuracy [17]. For example, adversarial networks (AN), CAE, and deep reinforcement learning (DRL) have been applied to recommender systems to improve recommendation performance [17,18].

As far as we know, O'Mahony et al. [4] first research on poisoning attacks (a.k.a shilling attacks). They define the robustness of RSs and demonstrate several vulnerabilities of poisoning attacks against CFRSs to facilitate specific advice [4,11]. Furthermore, Burke et al. [22] and Mobasher et al. [23] investigate some low-knowledge attack methods for pushing and reducing items, such as random, average, bandwagon, and segment attacks. They find that rating-based and ranking-based CFRSs are vulnerable to attack. Given more knowledge and budget, Wilson et al. [7] propose a powerful attack model that selects the most influential users or items to attack RSs. Fang et al. [24] study the poisoning attacks against graph-based RSs. Besides, Zhang et al. [25] utilize DRL to train the attack agent, which can generate user profiles for data poisoning. Xing et al. [14] conduct experiments on YouTube, Google, and Yelp. The experimental results show that manipulating RSs is possible.

Influenced by the popularity of the generative model in the field of image, some papers are using generative adversarial networks (GAN) [15,16]. Since the GAN mainly define the mini-max problem without loss function, which cannot fit well with the research of this paper, and the issues of long training time and difficult adjustment of parameters [26,27], we choose another generative model CAE [20]. As far as we know, we are the first to apply CAE to poisoning attacks. With the triangle relations and fake & real sampling strategy, our model has destructive attack effectiveness and good anti-detection capability.

3 Proposed Model

This section describes our proposed PATR, which includes a pre-training module and a reconstruction module.

3.1 Problem Formulation

We use $X \in \mathcal{R}^{N \times M}$ to represent the user-item rating matrix, where N is the number of all users, including real users and fake users, and M is the number of items. The user sets and item sets are denoted as $\mathcal{U}$ and $\mathcal{V}$, respectively. u_i

represents each user vector in $\mathcal{U}$, and v_j represents each item vector in $\mathcal{V}$, where $i, j \in \{1, 2, ..., N\}, \{1, 2, ..., M\}$, respectively. r_{ij} represents u_i's rating on v_j, and r_{max} is the highest rating. The fake user sets denote as $\mathcal{F}$, and $|\mathcal{F}|$ is the number of fake users. f_i and $\hat{f}_i$ denote each initial fake user vector and reconstructed fake user vector, where $i \in \{1, 2, ..., |\mathcal{F}|\}$. We use $\mathbf{u}_{iPre}$ and $\mathbf{v}_{jPre}$ to denote each fake user embedding and each item embedding generated by the pre-training module, where $j \in \{1, 2, ..., M\}$. $\mathbf{u}_{iE}$ denotes each fake user embedding generated by the reconstruction module.

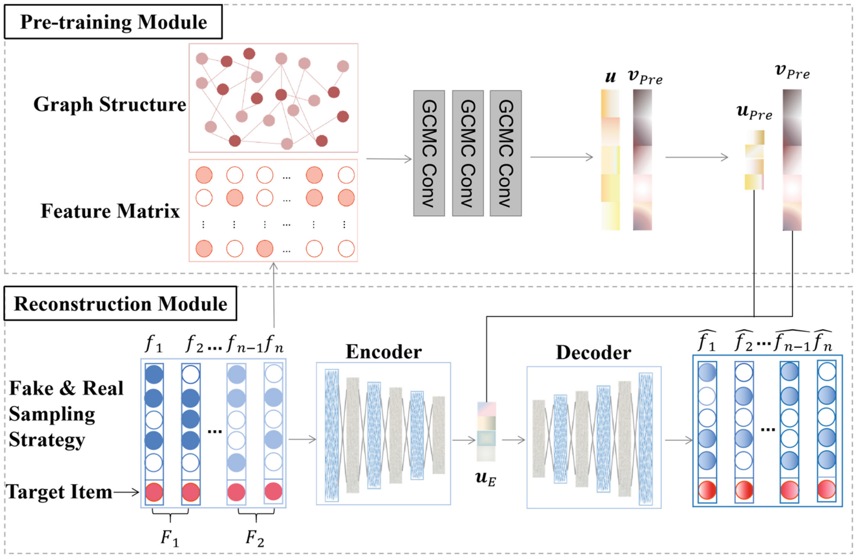

Fig. 2. The overall framework of PATR (Color figure online)

3.2 Pre-training Module

Problem Hypothesis. The traditional poisoning attack methods are easily detected and have a narrow application range due to simply using rating statistics without special designs. To enhance fake users' anti-detection capability and destructiveness, we first propose a pre-training module based on our elaborately designed triangle relations according to heuristic learning. The three nodes in the triangle relations represent three user embeddings of K dimensions, which are denoted as $\boldsymbol{u}_1 = [x_{11}, x_{12}, \ldots, x_{1K}]$, $\boldsymbol{u}_2 = [x_{21}, x_{22}, \ldots, x_{2K}]$, $\boldsymbol{u}_3 = [x_{31}, x_{32}, \ldots, x_{3K}]$, where x_i represents the i-th dimensional data. Each edge represents the similarity of two adjacent users, which is denoted as $L1$, $L2$, $L3$, where $L_1 = \sqrt{\sum_{i=1}^{K}(x_{1i} - x_{2i})^2}$, $L_2 = \sqrt{\sum_{i=1}^{K}(x_{2i} - x_{3i})^2}$, and $L_3 = \sqrt{\sum_{i=1}^{K}(x_{3i} - x_{1i})^2}$. As shown in Fig. 1, we consider the following two cases and use euclidean distance to calculate the similarity.

Case 1: If there are both positive ratings and negative ratings about the target item, we assume that fake users should be similar to real users who purchased the target item and rated it highly and stay away from users that rated it lowly. Moreover, the similarity of the two types of real users should be small. In Fig. 1 (a), $\boldsymbol{u}_1$ is a real user embedding who rated high rating, and $\boldsymbol{u}_2$ is a real user embedding who rated low rating, and $\boldsymbol{u}_3$ is a fake user embedding. Hence, the corresponding optimization problem can be formulated as

$$\min L = \lambda_1 L_3 - \lambda_2 L_2 - \lambda_3 L_1 \tag{1}$$

where λ_1, λ_2 and λ_3 are hyperparameters, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Case 2: If the target item is a cold item, we consider the second triangle relation. In this case, we assume that fake users can learn the commonality of real users and improve the anti-detection capability by minimizing L_1, L_2, and L_3. As shown in Fig. 1 (b), $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are two real users selected randomly. Hence, the optimization is minimizing the sum of three edges. The formula is

$$\min L = \lambda_4 L_1 + \lambda_5 L_2 + \lambda_6 L_3, \tag{2}$$

where λ_4, λ_5 and λ_6 are hyperparameters, and $\lambda_4 + \lambda_5 + \lambda_6 = 1$.

Graph Convolutional Matrix Completion. We use GCMC, a special graph neural network designed for recommendation scenarios, to generate user and item embeddings. As shown in Fig. 2, the input of the pre-training module is a graph structure generated by the user-item interaction matrix, which includes both real users and sampled fake users. The sampling strategy for the fake users is described in Sect. 3.3. In each iteration, we combine the triangle relations with GCMC for training. Specifically, we consider the user-item interaction matrix a weighted undirected graph $G = (\mathcal{W}, \mathcal{E}, \mathcal{R})$. The nodes consist of a collection of user nodes $u_i \in \mathcal{U}$ with $i \in \{1, \ldots, N\}$ and item nodes $v_j \in \mathcal{V}$ with $j \in \{1, \ldots, M\}$, and $\mathcal{U} \cup \mathcal{V} = \mathcal{W}$. The edges $(u_i, r_{ij}, v_j) \in \mathcal{E}$ tagged with labels represent ratings, where $r_{ij} \in \{r_1, \ldots, r_n\} = \mathcal{R}$.

GCMC assigns a specific transformation for each rating, resulting in edge-type specific messages $\delta_{j \to i,r}$, from items j to users i of the following form:

$$\delta_{j \to i,r} = \frac{1}{\varphi_{i,j}} W_r x_j^v, \tag{3}$$

where $\varphi_{i,j}$ is a normalization constant. W_r is an edge-type specific parameter matrix. x_j^v is the initial feature vector of item node j. Messages $\delta_{i \to j,r}$ from users to items are processed analogously. After the message passing step, we accumulate incoming messages at every node by summing over all neighbors $\mathcal{N}_i(u_i)$ connected by a specific edge-type r, and by accumulating the results for each edge type into a single vector representation:

$$h_i^u = \sigma \left[accum \left(\sum_{j \in \mathcal{N}_i(u_i)} \delta_{j \to i,1}, \ldots, \sum_{j \in \mathcal{N}_R(u_i)} \delta_{j \to i,R} \right) \right], \tag{4}$$

where $accum()$ denotes an accumulation operation such as $sum()$, i.e., summation of all messages. $\sigma()$ represents an activation function such as ReLU. To arrive at the final embedding of user node i, we transform the intermediate output h_i as

$$z_i^u = \sigma \left(W h_i^u \right), \tag{5}$$

where W is the same parameter matrix.

The outputs of GCMC are user embeddings and item embeddings. Because only the fake user embeddings are required, we pick them out from all the user embeddings according to the labels.

Algorithm 1. Pre-training Module

```
Input: The user-item interaction matrix (including fake users and real users) $\tilde{X}$.
Output: Powerful fake user embeddings $u_{Pre}$ and item embeddings $v_{Pre}$.
 1: Generate initialized user and item embeddings using Eq. (3) to Eq. (5).
 2: for the number of training epochs do
 3:     for the number of iterations do
 4:         if the target item belongs to Case 1 then
 5:             Randomly select a fake user embedding $u_{iPre} \in \mathcal{F}$, a real user embedding
                $u_{jPre} \in \mathcal{U}_1$ and a real user embedding $u_{kPre} \in \mathcal{U}_2$ from $u_{Pre}$.
 6:             Optimize the loss function according to Eq. (1).
 7:         end if
 8:     else
 9:         Randomly select a fake user embedding $u_{Prei} \in \mathcal{F}$ and two real user embed-
            dings.
10:         Optimize the loss function according to Eq. (2).
11:     end for
12: end for
```

3.3 Reconstruction Module

In this part, we mainly describe the fake & real sampling strategy and the reconstruction module. With the outputs of the pre-training module, we apply CAE to reconstruct enhanced fake users.

Fake & Real Sampling Strategy. Since our goal is to reconstruct enhanced fake users, we first need to sample a batch of initial fake users. To improving the anti-detection capability, we design a unique sampling strategy including two steps, as shown in Fig. 2. In the first step, we consider the existing sampling strategies. In addition to the target item, a fake user needs ratings on other items (the dark blue circles in Fig. 2) to disguise, and these items are called filler items [10–12]. The sampling strategies are used for filler items. Among many sampling strategies, popularity sampling (i.e., sample filler items based on their popularity) performs well [12], so we consider generating a set of fake users $\mathcal{F}_1$ by popularity. In the second step, we directly sample a part of users $\mathcal{F}_2$ from real active users to enhance the anti-detection capability, and the light blue circles in Fig. 2 are the sampled real users' original ratings. Therefore, the initial fake users are denoted as $\mathcal{F}$, where $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. We set the initial fake users' ratings on the target item (the red circles in Fig. 2) as r_{max} for attacking performance. The number of F1 and F2 is explained in parameter sensitivity experiments in Sect. 3.2.

Reconstructing Enhanced Fake Users. We use CAE to reconstruct the enhanced fake users. CAE is a fusion of an encoder and a decoder composed of convolutional networks and pooling layers. Convolution is a dot product between the filter and input data, described as

$$y_i^{l+1}(j) = K_i^l * x^l(j) + b_i^l, \tag{6}$$

where K_i^l and b_i^l denote the weights and bias of the i-th layer, and $x^l(j)$ is the j-th local region of the layer l.$*$ denotes a dot product operation, and $y_i^{l+1}(j)$ is the output of convolution operation, respectively. The activation function of the hidden layer is ReLU, and the activation function of the output layer is Sigmoid. Max pooling reduces the dimension of feature maps by taking the maximum among every window. By reducing the number of parameters, the feature dimension becomes smaller and manageable.

As shown in Fig. 2, the input is the sampled fake users vectors (i.e., the fake user-item rating matrix), and the encoder ultimately reduces input data into latent user embeddings $\boldsymbol{u}_E$. The user embeddings represent the lowest level space in which the input is reduced, with essential information preserved with a strong correlation between input features. To enhance the anti-detection capability of the fake users, we consider $\boldsymbol{u}_E$ generated by the encoder is similar to the fake user embeddings $\boldsymbol{u}_{Pre}$ generated by the pre-training module. Further, to maintain the destructiveness of fake users, we keep the ratings of fake users for the target item as high as possible. The optimization is described as

$$\mathcal{L}_{Encode} = \min_{\theta} \sum_{i=1}^{|F|} (Dist(u_{iE}, u_{iPre}) + \|r_{max} - u_{iE} \odot v_{tPre}\|), \tag{7}$$

where $\boldsymbol{u}_{iE}$ and $\boldsymbol{u}_{iPre}$ represent the fake user $\boldsymbol{u}_i$'s embedding generated by the encoder and pre-training module, respectively. $|F|$ is the number of fake users. $Dist()$ is the euclidean metric, $\boldsymbol{v}_{tPre}$ represents the target item embedding generated by the pre-training module, and $\boldsymbol{u}_{iE} \odot \boldsymbol{v}_{tPre}$ is $\boldsymbol{u}_i$'s rating on the target item. θ defines the parameters of the encoder.

The decoder acts as the mirror image of the encoder. The number of nodes in every layer increases and reconstructs the user embeddings to output as a similar input via transposed convolution. The process is described as

$$\mathcal{L}_{Decode} = \min_{\phi} \sum_{i=1}^{|F|} Dist(\hat{f}_i, f_i), \tag{8}$$

where $\hat{f}_i$ and f_i are the reconstructed and initialized fake user u_i's vector, respectively. ϕ represents the parameters of the decoder.

We combine two loss function as the final loss, which is

$$\mathcal{L} = \alpha \mathcal{L}_{Encode} + \beta \mathcal{L}_{Decode}, \tag{9}$$

where α and β are the hyperparameters, and $\alpha + \beta = 1$.

Algorithm 2. Reconstruction Module

```
Input: Initial fake users' rating matrix $X_F$ (i.e., initial fake users' vectors).
Output: Reconstructed fake users' rating matrix $\hat{X_F}$, the parameter $\theta$ for the encoder.
    $E$ and the parameter $\phi$ for the decoder $D$.
1: for number of training epochs do
2:     for number of iterations do
3:         Uniformly sample a minibatch of fake users $\mathcal{F}'$.
4:         for each fake user $f_i' \in \mathcal{F}'$ do
5:             Let the fake user embeddings $u_{iE}'$ generated by the encoder be similar to
               the fake user embeddings $u_{iPre}'$ of the pre-training module using Eq. (7).
               Set the rating of $u_{iE}'$ for the target item to $r_{max}$ using Eq. (7).
6:             Let the reconstructed fake user vector $\hat{f}_i'$ generated by the decoder be similar
               to the input $f'$ using Eq. (8).
7:         end for
8:     end for
9: end for
```

4 Experiments and Analysis

In this section, Experiments are conducted to verify the effectiveness of our model. We mainly focus on the following questions.

- **Q1:** Does our proposed model PATR have a significant attack effectiveness on DL-based RSs?
- **Q2:** Is PATR more likely to evade detection?
- **Q3:** Are our hypothesis and designs (the pre-training module and sampling strategy) conducive to PATR?

4.1 Experimental Setup

We use three benchmark datasets in our experiments: FilmTrust,[1] Ciao,[2] and ML-100K.[3] Each dataset is randomly split by 9:1 as training set and test set, respectively. Table 1 shows the details of the datasets. To avoid the cold start problem, we filter out users with fewer than ten ratings (They are too sensitive to attacks). Three layers of convolution with 512, 256, and 128 neurons are used in GCMC and CAE. The number of injecting fake users account for 5% of the number of real users, and the number of filler items for each fake user profile in $\mathcal{F}_1$ accounts for 1% of the number of real users.

Table 1. Dataset statistics

Dataset	Users	Items	Ratings	Sparsity
FilmTrust	1,058	2,071	35,497	98.86%
ML-100K	934	1,682	100,000	93,70%
Ciao	7,375	105,114	284,086	99.96%

Attack Models. In this paper, we choose the following four traditional poisoning attack methods as baseline methods.

(1) **Random Attack** [11]**:** Random attack is a naive attack model. The set of filler items are assigned to random ratings with a normal distribution around the mean rating value across the whole dataset, and the target item is given the maximum rating value r_{max}.
(2) **Average Attack** [11]**:** Average attack is a somewhat more sophisticated attack than random attack and requires knowledge of each item's average rating in the system. Each introduced user rates items not in the target set randomly on a normal distribution with a mean equal to the average rating of the rated item. The target item is assigned r_{max}.
(3) **Bandwagon Attack** [10]**:** Bandwagon attack, also known as popular attack, takes advantage of the items with high popularity in the dataset and calls these items selected items. These selected items and the target item are assigned the maximum rating value r_{max}. The ratings on the filler items are determined randomly in a similar manner as in average attack.
(4) **Unorganized Attack** [9]**:** unorganized malicious attacks allow the concurrence of various attack strategies, and the number of rated items, the target item, and the rating functions can be different. Each attacker produces a small number of attack profiles with their own strategies and preference.

[1] https://www.librec.net/datasets/filmtrust.zip.
[2] https://guoguibing.github.io/librec/datasets/CiaoDVD.zip.
[3] https://grouplens.org/datasets/movielens/100k/.

Recommender Algorithms. In this paper, we focus on two DL-based RSs, i.e., neural matrix factorization (NeuMF) [28] and deep matrix factorization model (DMF) [29]. NeuMF is a fusion of MF and multilayer perceptron (MLP), allowing two models to learn individual embeddings and combine them by connecting their final hidden layers. DMF takes the user-item interaction matrix as input and extracts the features of the users and items into a low-dimensional space through the novel loss function based on binary cross-entropy.

Table 2. Attack effectiveness against NeuMF.

RS	NeuMF					
Metric	TKS@10			HR@10		
Dataset	FilmTrust	Ciao	ML-100K	FilmTrust	Ciao	ML-100K
PATR	**0.542**	0.708	**0.627**	**0.0362**	**0.0823**	**0.0513**
Random	0.334	0.571	0.443	0.0072	0.0326	0.0027
Average	0.337	0.643	0.357	0.0064	0.0637	0.0025
Bandwagon	0.425	**0.750**	0.425	0.0141	0.0248	0.0033
Unorganized	0.375	0.689	0.378	0.0076	0.0523	0.0025
PATRt	0.418	0.530	0.569	0.0232	0.0635	0.0183
PATRs	0.501	0.665	0.424	0.0137	0.0687	0.0258

Table 3. Attack effectiveness against and DMF.

RS	DMF					
Metric	TKS@10			HR@10		
Dataset	FilmTrust	Ciao	ML-100K	FilmTrust	Ciao	ML-100K
PATR	**0.574**	**0.682**	**0.776**	0.164	**0.054**	**0.0479**
Random	0.457	0.486	0.329	0.0071	0.0025	0.0024
Average	0.407	0.535	0.305	0.0069	0.0025	0.0019
Bandwagon	0.519	0.682	0.563	0.0875	0.0208	0.0037
Unorganized	0.530	0.498	0.489	**0.191**	0.0231	0.0035
PATRt	0.389	0.635	0.403	0.087	0.042	0.0094
PATRs	0.530	0.678	0.730	0.073	0.0512	0.0154

Metrics. We propose a new metrics top-K shift (TKS) to measure how much the top-K recommendation lists affected after the attack. We assume that each user recommendation list without the attack is L, and the recommendation list after the attack is $\tilde{\mathrm{L}}$. TKS calculates the number of items not in L after the attack. For example, if the top-K recommendation list of u_1 is $\mathrm{L}_1 = \{23, 1, 5, 7, 34\}$ before attack and $\tilde{\mathrm{L}}_1 = \{5, 1, 18, 22, 34\}$ after the attack, then the item 23 and item 7 that should have been recommended to user u_1 are missing in $\tilde{L}_1$. The bigger TKS indicates better attack effectiveness. Because there are some differences in the order of items among the top-K recommendation lists generated by the same RS, we do not consider the specific order bias for each item. TKS is defined as:

$$\text{TKS} = \frac{\sum_{i=1}^{N} |N_{absent}|_i}{K * N} \tag{10}$$

Where K is the length of each user's recommendation list, N is the number of the recommendation lists, N_{absent} is the number of items not in L after the attack.

Another metric is HR [10–12]. Let R_u be the set of top-K recommendations for user u and $H_{u,i}$ denotes whether the target item i is in the recommendation list of user u. For each target item i, $H_{u,i}$ is assigned to 1, where $i \in R_u$, otherwise $H_{u,i}$ is assigned to 0. As with TKS, the bigger HR indicates the better attack effectiveness. The metric is defined as:

$$\text{HR} = \frac{\sum_{u,i} H_{u,i}}{|U| * N} \tag{11}$$

In this work, the K in the top-K recommendation lists is set to 10, which means the metrics are TKS@10 and HR@10.

4.2 Experimental Results and Analysis

Attack Effectiveness. The researchers prefer to deliberately choose the long tail items in previous literature because the long tail items are more sensitive to attack methods, and reflect better attack effectiveness. To accurately reflect the effectiveness of our attack, we do not especially choose the long tail items and randomly select ten items that do not exist in top-K recommendation lists when RSs are not attacked. To verify the contribution of our proposed pre-training module and sampling strategy, we remove these two parts respectively for comparison. The way without the pre-training module (triangle relations) is denoted as PATR$_t$, and the method without fake & real sampling strategy (only use regular widespread sampling) is denoted as PATR$_s$. Table 2 and Table 3 show the average performance of attacking NeuMF algorithm and DMF algorithm, respectively, and we bold the data with the best performance. It can be seen PATR can achieve the best performance in most cases and the second-best occasionally. The average increases(compared to the second-best attack method and a negative growth if PATR is the second-best) of TKS@10 in NeuMF and DMF are 4.26% and 5.06%, respectively, proving that PATR can make recommendation lists more disordered. Meanwhile, the average increases of HR@10 are 98.84% and 67.46%, which means PATR pushes the target item to more real users. Neither PATR$_t$ nor PATR$_s$ performs as well as PATR, which proves that the triangle relations and fake & real sampling strategy are helpful to attack effectiveness. The experiments of attack effectiveness can answer the **Q1** and **Q3**.

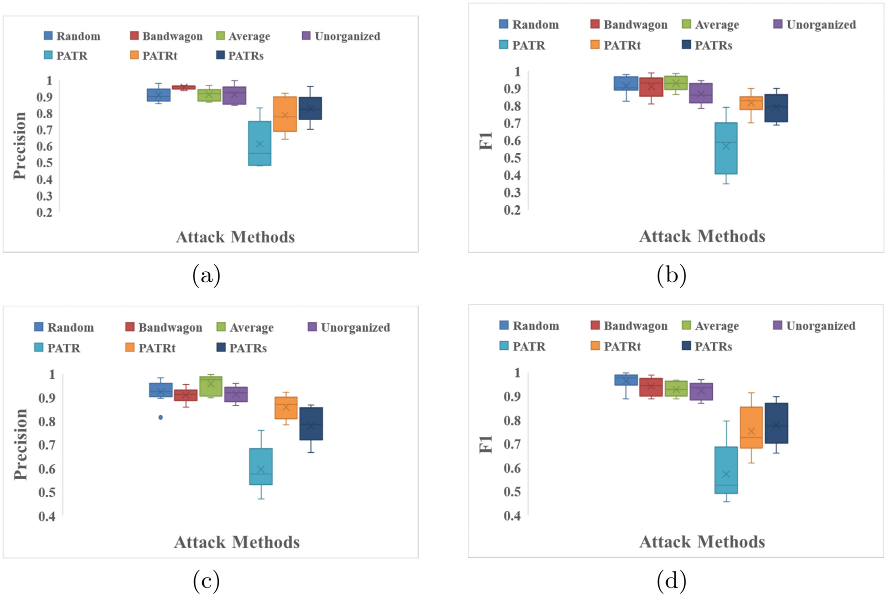

Fig. 3. Anti-detection capability against DLDRA and DegreeSAD. (a) and (b) are the results of DLDRA on the FilmTrust dataset. (c) and (d) are the results of DegreeSAD on the ML-100K dataset.

Anti-detection Capability. We apply two detectors, i.e., a state-of-the-art recommendation attack detector based on deep learning (DLDRA) [30] and a classic detector in recommender systems via selecting patterns analysis (DegreeSAD) [31] to verify the anti-detection capability of our attack. The metrics are precision and F1. The datasets are FilmTrust and ML-100K. The baseline attack models and the selection method of target items are the same as the attack effectiveness experiments. As shown in Fig. 3, we can observe clearly that the traditional poisoning attacks perform poorly, which means these attacks result in mission failure. The anti-detection capability of PATR is better than other methods. Specifically, both the precision and F1 are the smallest in the two datasets. Moreover, PATR makes the detector be the most unstable. Therefore, PATR has better anti-detection capability than baseline methods (answer the **Q2**). In ablation experiments, PATR$_t$ and PATR$_s$ are also better than the traditional poisoning attacks but less than PATR, which indicates that our proposed triangle relations and fake & real sampling strategy are conducive (answer the **Q3**).

Sensitivity Analysis. In this paper, we use some hyper-parameters and conduct experiments to determine the value of these hyper-parameters. We set λ_1 through λ_6 mentioned in Sect. 3.2 to 0.45, 0.45, 0.1, 0.35, 0.35, 0.3, respectively. In the sensitivity analysis, we focus on the encoder loss α, the decoder loss β,

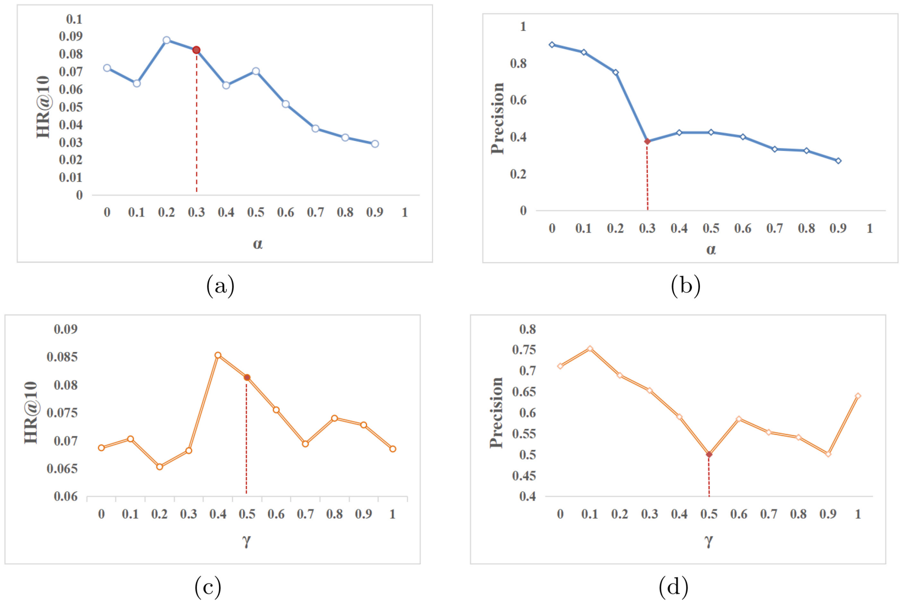

Fig. 4. Sensitivity analysis for β and γ. HR@10 measures the attack effectiveness, and precision measures the anti-detection capability.

and the ratio of user groups $\mathcal{F}_1$ and $\mathcal{F}_2$ in the sampling strategy for balancing our model's attack effectiveness and anti-detection capability. We use the Ciao dataset, and the metrics are HR@10 and precision, respectively. The higher the value of HR@10, the better the attack effectiveness. The smaller value of precision means the better anti-detection capability. We denote the ratio of $\mathcal{F}_1$ as γ. As shown in Fig. 4 (a) and (b), they are the sensitivity analysis of α. When α is 0.2, HR@10 is the best, while the anti-detection is poor. When α is 0.3, the attack effectiveness is the second best, and the anti-detection is the best. Therefore, we compromise by setting α to 0.3, while β is 0.7. Similarly, we set γ to 0.5, i.e., the number of $\mathcal{F}_1$ and $\mathcal{F}_2$ is the same.

5 Conclusion

In this paper, we focus on poisoning attacks. To reduce the probability of being detected, we optimize the sampling strategy for fake users by directly sampling a set of users from the real users. Furthermore, we adopt triangle relations and design a pre-training module. Finally, we propose a reconstruction module that combines CAE with the pre-training module's outputs to generate enhanced fake users. Our experiments on three real-world datasets show that our proposed model PATR outperforms baselines in attack effectiveness and anti-detection capability.

Acknowledgments. This study was supported by the National Key R&D Program of China (2018YFB1403602), Natural Science Foundation of Chongqing, China (cstc2020jcyj-msxmX0690), the Technological Innovation and Application Program of Chongqing (cstc2019jscx-mbdxX0008), the Fundamental Research Funds for the Central Universities of Chongqing University (2020CDJ-LHZZ-039), and the Overseas Returnees Innovation and Entrepreneurship Support Program of Chongqing (cx2020097).

References

1. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–58 (1997)
2. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
3. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_1
4. O'Mahony, M., Hurley, N., Kushmerick, N., et al.: Collaborative recommendation: a robustness analysis. ACM Trans. Internet Technol. (TOIT) **4**(4), 344–377 (2004)
5. Hurley, N.J.: Robustness of recommender systems. In: Proceedings of the fifth ACM Conference on Recommender Systems, pp. 9–10 (2011)
6. Douceur, J.R.: The Sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_24
7. Wilson, D.C., Seminario, C.E.: When power users attack: assessing impacts in collaborative recommender systems. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 427–430 (2013)
8. Li, B., Wang, Y., Singh, A., et al.: Data poisoning attacks on factorization-based collaborative filtering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 1893–1901 (2016)
9. Pang, M., Gao, W., Tao, M., et al.: Unorganized malicious attacks detection. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6976–6985 (2018)
10. Gunes, I., Kaleli, C., Bilge, A., et al.: Shilling attacks against recommender systems: a comprehensive survey. Artif. Intell. Rev. **42**(4), 767–799 (2014)
11. Lam, S.K., Riedl, J.: Shilling recommender systems for fun and profit. In: Proceedings of the 13th International Conference on World Wide Web, pp. 393–402 (2004)
12. Si, M., Li, Q.: Shilling attacks against collaborative recommender systems: a review. Artif. Intell. Rev. **53**(1), 291–319 (2020)
13. Williams, C.A., Mobasher, B., Burke, R.: Defending recommender systems: detection of profile injection attacks. Serv. Oriented Comput. Appl. **1**(3), 157–170 (2007)
14. Meng, W., Xing, X., Sheth, A., et al.: Your online interests: Pwned! A pollution attack against targeted advertising. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, vol. 2014, pp. 129–140 (2014)
15. Lin, C., Chen, S., Li, H., et al.: Attacking recommender systems with augmented user profiles. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 855–864 (2020)

16. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: reliable attacks against black-box machine learning models. In: International Conference on Learning Representations (2018)
17. Zhang, S., Yao, L., Sun, A., et al.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. (CSUR) **52**(1), 1–38 (2019)
18. Sahoo, A.K., Pradhan, C., Barik, R.K., et al.: DeepReco: deep learning based health recommender system using collaborative filtering. Computation **7**(2), 25 (2019)
19. van den Berg, R., Kipf, T.N., Welling, M.: Graph convolutional matrix completion (2017)
20. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_7
21. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, vol. 2000, pp. 241–250 (2000)
22. Burke, R., Mobasher, B., Bhaumik, R., et al.: Segment-based injection attacks against collaborative filtering recommender systems. In: Fifth IEEE International Conference on Data Mining (ICDM'05). IEEE (2005)
23. Mobasher, B., Burke, R., Bhaumik, R., et al.: Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness. ACM Trans. Internet Technol. (TOIT) **7**, 23-es (2007)
24. Fang, M., Yang, G., Gong, N.Z., et al.: Poisoning attacks to graph-based recommender systems. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 381–392 (2018)
25. Zhang, H., Li, Y., Ding, B., et al.: Practical data poisoning attack against next-item recommendation. In: Proceedings of The Web Conference 2020, pp. 2458–2464 (2020)
26. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for GANs do actually converge? In: International Conference on Machine Learning, PMLR, pp. 3481–3490 (2018)
27. Hong, Y., Hwang, U., Yoo, J., et al.: How generative adversarial networks and their variants work: an overview. ACM Comput. Surv. (CSUR) **52**(1), 1–43 (2019)
28. He, X., Liao, L., Zhang, H., et al.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
29. Xue, H.J., Dai, X., Zhang, J., et al.: Deep matrix factorization models for recommender systems. In: International Joint Conference on Artificial Intelligence, vol. 17, pp. 3203–3209 (2017)
30. Zhou, Q., Wu, J., Duan, L.: Recommendation attack detection based on deep learning. J. Inf. Secur. Appl. **52**, 102493 (2020)
31. Li, W., Gao, M., Li, H., et al.: Shilling attack detection in recommender systems via selecting patterns analysis. IEICE Trans. Inf. Syst. **99**(10), 2600–2611 (2016)