Journal of Information Security and Applications 63 (2021) 103051

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Fusing hypergraph spectral features for shilling attack detection

Hao Li [a,b,c], Min Gao [a,b,*], Fengtao Zhou [b], Yueyang Wang [b], Qilin Fan [b], Linda Yang [d]

[a] Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing, 400044, China
[b] School of Big data & Software Engineering, Chongqing University, Chongqing, 401331, China
[c] School of Automotive Engineering, Chongqing University, Chongqing, 400044, China
[d] School of Engineering, University of Portsmouth, Portsmouth, PO1 3AH, UK

ARTICLE INFO

Keywords:
Recommender systems
Shilling attack detection
Spectral feature
User similarity

ABSTRACT

Recommender systems can effectively improve user experience, but they are vulnerable to shilling attacks due to their open nature. Attackers inject fake user profiles to destroy the security and reliability of the recommender systems. Therefore, it is crucial to detect shilling attacks effectively. The primitive detection models are feasible but costly because of the dependence on plenty of hand-engineered explicit features based on statistical measures. Even though the upgraded models based on learning embeddings of the implicit features are more general, they fail to take some distinct features in distinguishing fake users into consideration. Moreover, these primitive and upgraded models are difficult to capture the high order relationships between users and items as the models usually learn the embedding from the first-order interactions. The representation and similarity information learned from the first-order interactions are not comprehensive enough, limiting the detection task. To this end, we propose a novel shilling attack detection model by fusing hypergraph spectral features (SpDetector). The proposed model combines the explicit and implicit features to balance the effectiveness and generality and deal with the high order relationships by hypergraphs-based embedding. From the implicit perspective, SpDetector constructs user hypergraphs and item hypergraphs for the high-order relationships hidden in the interaction and extracts spectral features from hypergraphs to capture high-order similarity for users and items, respectively. From the explicit perspective, it extracts two kinds of explicit features: item similarity offsets (ISO) based on item spectral features and rating prediction errors (RPE), for all users as their distinct capability of distinguishing fake users. Finally, the SpDetector learns to distinguish fake users by training a deep neural network with those features. Experiments conducted on MovieLens and Amazon datasets show that SpDetector outperforms state-of-the-art detection models.

1. Introduction

With the development of Internet information, information overload is becoming more and more serious. Recommender systems can help users screen out the items they are interested in from massive information. High-quality recommendation content improves user experience and user stickiness, but due to the openness of recommender systems, they are vulnerable to shilling attacks [1–3]. In shilling attacks, the attackers inject fake user profiles to influence the recommendation results, which seriously affects the system's reliability and the user experience of recommendation [4]. Shilling attacks seriously threaten the recommender system's security and bring great harm to consumers, businesses, and platforms. Therefore, it is urgent to study the shilling attack and its detection methods to improve the recommender system's security and reliability [5,6].

With the development of online e-commerce platforms, people have become accustomed to purchasing and reviewing items on the Internet. Through these purchasing records and reviews, other people can easily obtain the quality information of these items and make choices based on their preferences [7]. However, the attackers inject fake purchase records and reviews to influence the property of the items, thus affecting people's choices and judgment [8,9]. The injection of these fake user profiles has its specific mode, which is different from real users in some aspects. Thus, how to effectively use this difference and detect attackers has always been the focus of shilling attack detection.

To defend against recent shilling attack proposals, many methods have been proposed to detect various shilling attacks. Most algorithms are designed in statistical-based manners to detect attackers [10–12]. These methods are feasible when detecting known types of attacks but are costly since extracting plenty of effective statistics-based explicit features requires expert labor. After that, researchers leverage user–item interactions from the bipartite graph to obtain the users' rating

* Corresponding author.
E-mail address: gaomin@cqu.edu.cn (M. Gao).

https://doi.org/10.1016/j.jisa.2021.103051

Available online 24 November 2021

preference and structure information to improve the detection accuracy [13,14]. Even though the upgraded models based on learning embeddings of the implicit features are more general, they fail to take some statistical-based features in distinguishing fake users into consideration. Moreover, most of these methods merely focus on the first-order interactions between users and items and pay little attention to the users' higher-order interactions. The high-order user–item interactions contain richer topology information of users [15], which will help to improve the performance of shilling attack detections.

In recent years, to strengthen the detection performance of the models, many features are proposed for shilling attack detection, such as degree of similarity with top neighbors [16], popularity patterns based on items analysis [11], diversity of interest preference [17]. However, the extraction of these features is costly and often requires expert labor. Attackers can also camouflage themselves by using these features to avoid detection. There are also some features are incorporated for additional information, such as textual features from review data [18] and time-series information [19]. These features are not always available and we need extra efforts to process them. To address these gaps, we introduce spectral features extracted from the hypergraphs to shilling attack detection. Compared with the simple graph, the difference of hypergraphs lies in that one hyperedge can connect multiple vertices, and higher-order information between the vertices can be obtained through the connection. Such a structure can better reflect the topological relations between users, items, and their interactions in recommender systems.

To this end, we propose a detection model, SpDetector, that extracts features from both implicit and explicit perspectives. From the implicit perspective, we first construct hypergraphs for users and items, respectively. Then we extract the spectral feature that is the front K eigenvectors of a hypergraph's Laplacian matrix, which contains abundant similarity information. From the explicit perspective, we hold the view that genuine users tend to give close ratings to similar items. When generating attack profiles, attackers often disguise themselves by choosing random items and assigning them with random ratings, which may lead to similar items in the generated profile being assigned with vastly different ratings. If a user gives two very similar items a large difference in ratings, the user is more likely to be a fake user. Thus, we propose a new feature item similarity offset (ISO). We extract the ISO by computing the product of the spectral features of each user's highest-rated items and lowest-rated items. The larger the product is, the more similar the two items are, and the user is more likely to be a fake user. Note that users may rate several items with their highest and lowest ratings, so we choose the highest product to represent the ISO feature. We also adopt another feature: rating prediction error (RPE) [18]. From the Cognitive Psychology perspective, a normal user's preference should be predictable [20]. If a user's predicted rating deviates greatly from the real rating, then the user is likely to be a fake user. We combine these two kinds of features with the user's spectral features and input them into a neural network for training [15]. The experimental results show that our method has a good performance and generalization capacity and significantly outperforms state-of-the-art methods in real scenarios.

The main contributions of this paper are as follows:

- We propose a novel model named SpDetector to detect the shilling attacks, in which hypergraphs are introduced to model the implicit high-order information and statistic features are extracted to capture the explicit information.
- We propose a feature Item Similarity Offset (ISO) to show the similarity difference between items users like and dislike, which is combined with Rating Prediction Error (RPE) and User Spectral Features to train a fake user detection network.
- We evaluate the proposed SpDetector on MovieLens and Amazon datasets. The experimental results demonstrate that SpDetector outperforms state-of-the-art detection models. Ablation experiments also explain why the extracted features can improve the performance of detection.

The rest of the paper is organized as follows. Section 2 introduces the common shilling attack models, shilling attack confusion methods, and shilling attack detection methods. Section 3 shows the structure of the proposed SpDetector. Section 4 reports the experiment results and analysis. In the last section, we conclude our work and discuss our future work.

2. Related work

2.1. Shilling attack models

To avoid detection, malicious users will disguise themselves as genuine users. They use attack models to generate attacker profiles based on knowledge of recommender systems. Based on whether it is improving or reducing the recommended ranking of target items, shilling attacks can be divided into two categories: Push Attack and Nuke Attack [21]. Different attack models adopt different strategies to build attack profiles, simulate user ratings, and generate social relationships to pretend to be normal users. The general profile of an attacker can be divided into four segments:

I_T: the target items set, which is usually set with the highest rating (for a push attack) or the lowest rating (for a nuke attack).

I_S: the selected items set, which is selected by attackers for specific needs.

I_F: the filler items set, which is used to disguise attackers.

I_N: the unrated items set, in which items are all rated empty.

In the recommendation of rating prediction, the basic attack models [22,23] include random attack, average attack, bandwagon attack, segment attack, average over popular items, and love/hate attack, etc (see Table 1). These kinds of attacks rely on the items' historical ratings, the popularity of the items, and similar items to generate fake rating records.

- **Random attack:** Filler items are randomly chosen and are assigned with random ratings (the normal distribution of system item ratings determines the assigned rating ratings).
- **Average attack:** Filler items are randomly chosen and are assigned with the average ratings of items.
- **Bandwagon attack:** Selected items are the most popular items and are assigned with the maximum rating. Filler items are randomly chosen and are assigned with random ratings.
- **Segment attack:** Selected items are very similar to the target item and are assigned with the maximum rating. Filler items are randomly chosen and are assigned with random ratings.
- **Average over popular items (AoP):** Filler items are the most popular items and are assigned with the average ratings of items.
- **Love/Hate attack:** Filler items are randomly chosen and are assigned with the maximum/minimum rating, while the target item is assigned with the minimum/maximum rating.

2.2. Shilling attack confusion methods

The attackers select several filler items and assign the items with well-designed ratings to disguise themselves. However, only using filler items is often not a good disguise strategy, so attackers often use some confusion methods [23,24] to be better disguised.

- **Noise injection:** To mask the characteristics of common attack models, random ratings generated by a particular distribution, such as a Gaussian, are added to each rating in a set of attack profiles.
- **Target offset:** Attackers assign the target item with the highest rating, or the lowest rating is easy to attract the detector's attention. The target offset is to change the target item's rating to the second-highest rating or the second-lowest rating.
- **User offset:** The attacker selects a subset of the rated items to reduce the similarity between the attackers.

Table 1
The features of attack models mentioned in this paper.

Attack model	I^S (Selected item)		I^F (Filler item)		I^T (Target item)
	Items	Ratings	Items	Ratings	
Random attack	–	–	Random choose	r_{random}	r_{max} or r_{min}
Average attack	–	–	Random choose	$r_{average}$	r_{max} or r_{min}
Bandwagon attack	Popular items	r_{max}	Random choose	r_{random}	r_{max} or r_{min}
Segment attack	Similar items to I^T	r_{max}	Random choose	r_{random}	r_{max} or r_{min}
Average over popular items	–	–	The most popular items	r_{random}	r_{max} or r_{min}
Love/Hate attack	–	–	Random choose	r_{max} or r_{min}	r_{min} or r_{max}

- **Popular filling** The filler items are chosen from the top x% popular items with the same probability. The popularity of an item can be measured by the counts of the ratings on the item.

2.3. Shilling attack detection methods

2.3.1. General detection methods

According to whether training labels are needed, shilling detection methods can be divided into three categories [25].

Supervised detection models: Such models require user labels for training. Extracted features and classifiers based on machine learning can be exploited for detection with labeled data. Yang et al. [14] proposed a method that utilizes matrix factorization and user embedding to construct the implicit features and applies latent label information generated by the Bayesian model to update the implicit features for detecting. However, this method only uses first-order interaction, which leads to low detection performance. Zeynep et al. [26] proposed a method that generated additional model-specific properties with generic properties from user profiles to handle the fake user profiles. However, the model-specific attributes proposed in this method are only applicable to specific attack models and are hard to be specified for the segment and love–hate attacks. Zhou et al. [27] designed a model based on deep learning, which can learn directly from the user ratings without extracting the hand-designed features. However, the network used to extract features in this method is too basic, limiting the performance of the model.

Semi-supervised detection models: Such models leverage a large amount of unlabeled data and a small amount of labeled data [28]. Cao et al. [29] proposed a semi-supervised learning method that first trains a classifier with naive Bayes on some labeled user-profiles and then incorporates unlabeled user profiles for EM-λ to improve the Bayes classifier. Wu et al. [30] proposed a spammer detection model based on a semi-supervised hybrid PU-learning, which builds a spammer classifier from both user features and user relations. These methods often do not perform well because of the uneven distribution of training data.

Unsupervised detection models: Such models do not rely on user labels. The classifiers of these methods are built on statistic-based features, which need certain prior knowledge and are biased by human experience. Cai et al. [17] analyze the user rating behavior and calculate the suspicious degree and spot attack users in the set of suspicious users. Liu et al. [19] improved the LSTM model of the LSTM network and used modeling for behavioral time series to identify abnormal behaviors. Zheng et al. [31] utilized the one-class adversarial nets for spammers detection. They first used an LSTM-Autoencoder to learn the implicit user features and then trained a discriminator with a complementary Generative Adversarial Network from the genuine users. Then they used the discriminator to distinguish spammers from genuine users. Cai et al. [32] proposed an approach based on item relationships. They designed an algorithm to calculate the probability of being a target item for each item and identify the target items and attackers in the set of suspicious users. These methods are feasible when detecting known types of attacks but are costly since extracting plenty of effective statistics-based explicit features requires expert labor.

2.3.2. Graph-based detection methods

Graph-based methods often show good performance. Such methods extract abundant structure and similarity information from the graph. Zhang et al. [33] formulate the shilling attack detection problem as finding a maximum submatrix in the similarity matrix and constructing a graph on it to solve the problem. Zhang et al. [34] first use the graph embedding method to obtain the low-dimensional feature representation of nodes, then use the K-means++ clustering algorithm to generate candidate groups. Finally, they use the hierarchical clustering algorithm to cluster the candidate groups and identify the attack groups. Zhang et al. [18] proposed a GCN-based user representation learning framework to perform fraudster detection. This method used prediction error outputted in the recommendation component to enrich the feature in fraudster detection.

These methods ignore the higher-order interactions on graphs or fail to consider statistics-based features, thus showing limitations in shilling attack detection. To extract higher-order information from user–item interactions and reduce the error raised by the statistical feature-based classifier, we propose a supervised method that builds a neural network with features extracted from hypergraphical and statistical perspectives as the detector.

3. Proposed detection model

In this section, we elaborate on our proposed SpDetector for shilling attack detection. The SpDetector framework is shown in Fig. 1, which consists of three crucial features extraction components and a deep neural network for shilling attack detection: (1) The user spectral features component extracts the user spectral features from the user hypergraph. (2) The item similarity offsets component captures the similarity offset between the user's highest-rated items and lowest-rated items for each user. This feature is represented as the product of the item spectral features extracted from the item hypergraph. (3) The rating prediction errors component computes the mean square error between real ratings and predicted ratings for each user and takes it as a crucial feature for shilling attack detection. (4) The deep neural network takes the three features as input and outputs the probability of a user being classified as a genuine user. Such probability is also taken as a weight to control the contribution of this user's predicted rating.

3.1. Hypergraph spectral feature

Compared with the simple binary graph, a hypergraph can capture the high-order correlations between users and items and better represent their similarities. Thus, we introduce hypergraphs for shilling attack detection.

For a traditional hypergraph $G = (V, E, W)$, which can be represented with the incident matrix $H \in \mathcal{R}^{|V|\times|E|}$ (V is the vertex set, E is the hyperedges set, W is the hyperedge weight matrix), $|V| = n$ denotes the number of vertices and $|E| = m$ denotes the number of hyperedges. The information of the incident matrix is only concentrated on whether the hyperedge contains the vertices. The kernel used in the

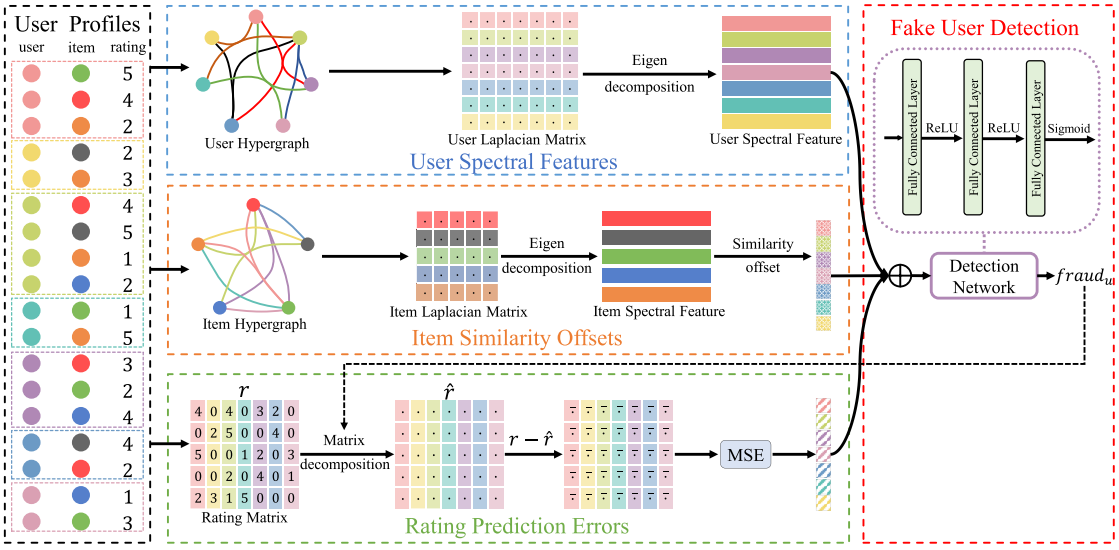


Fig. 1. Illustration of SpDetector, which consists of three crucial features extraction components: user spectral features component, item similarity offsets component, and rating prediction errors component. The deep neural network takes the features extracted by the three components for detection.

traditional hypergraph can be regarded as a flat kernel. The flat kernel K_F is defined as:

$$K_F(v,e) = \begin{cases} 1, & v \text{ is connected with } e \\ 0, & otherwise \end{cases} \tag{1}$$

To obtain more information between vertices and hyperedges, we adopt the weighted incident matrix H_w [35]. The difference between the weighted incident matrix and the normal one is for each nonzero entry $h(v,e)$, a positive value is set to measure the distance between v and e. We adopt the Gaussian Kernel K_G to measure the distance between the vertex v and the hyperedge e.

$$K_G(v,e) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{d(v,e)^2}{\sigma}}, & v \text{ is connected with } e \\ 0, & otherwise \end{cases} \tag{2}$$

In this section, we define a user hypergraph G_u and a item hypergraph G_i to depict the high-order correlations between users and items. For user hypergraph $G_u = (V_u, E_u, W_u)$, V_u is the vertex set containing all n users, E_u is the hyperedge set containing all m items, and $W_u \in \mathcal{R}^{m\times m}$ is the hyperedge weight matrix. We adopt the Gaussian Kernal to form the incidence matrix $H_u \in \mathcal{R}^{n\times m}$, which is defined as: if the user u has interacted with the item i, then $h_w(u,i) = K_G(u,i)$ and $h_w(u,i) = 0$ otherwise. $d(u,i) = r(u,i)$ denotes the rating that u gives to i. For item hypergraph, $G_i = (V_i, E_i, W_i)$, V_i is the vertex set containing all m items, E_i is the hyperedges set containing all n users and $W_i \in \mathcal{R}^{n\times n}$ is the hyperedge weight matrix. The incidence matrix $H_i = H_u^T$.

The Laplacian matrix [36] of the hypergraph can be defined as,

$$L = D^{-\frac{1}{2}} H_w W \Delta^{-1} H^T D^{-\frac{1}{2}}, \tag{3}$$

where $D \in \mathcal{R}^{n\times n}$ denotes the diagonal matrix containing the degrees of vertices, $W \in \mathcal{R}^{m\times m}$ denotes the diagonal matrix containing the weights of hyperedges, $\Delta \in \mathcal{R}^{m\times m}$ denotes the diagonal matrix containing the degrees of hyperedges. In such manner, we can build the user hypergraph Laplacian matrix $L_u \in \mathcal{R}^{n\times n}$ and the item hypergraph Laplacian matrix $L_i \in \mathcal{R}^{m\times m}$.

We factorize the user hypergraph Laplacian matrix L_u with eigen-decomposition: $L_u = \Phi\Lambda\Phi^T$, where $\Phi \in \mathcal{R}^{n\times n}$ is the eigenvector matrix, each row of Φ is an eigenvector, we take the first K eigenvectors to form the spectral feature matrix: $E = \Phi_{*,1:K}$, thus E_i can be represented as the spectral feature of vertex i. The spectral features contain abundant similarity information between vertices. We can calculate $E_i E_j$ to obtain the similarity information between vertex i and j. For similar vertices i and j, $E_i E_j$ is high. The properties of spectral features can be used for classification tasks [37]. Λ denotes the diagonal matrix containing N eigenvalues, and all eigenvalues are in ascending order. Similarly, we can factorize the item hypergraph Laplacian matrix L_i with eigen-decomposition and obtain the item spectral feature F.

3.2. Statistics-based features

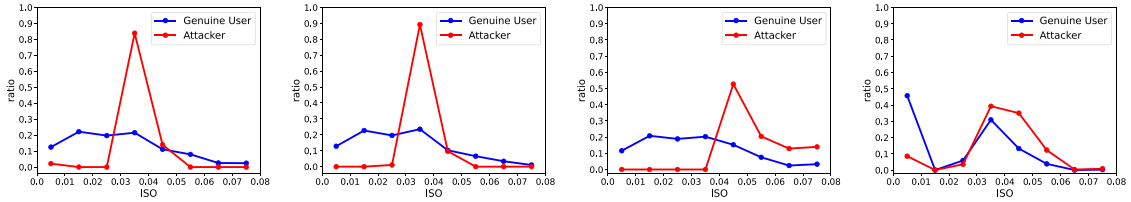
Hand-engineered explicit features based on statistical measures are also effective in shilling attack detection [16,38]. To extract user features from the item aspect, we propose a new feature, Item Similarity Offset (ISO), to capture similarity offset between the user's highest-rated items and lowest-rated items for each user. Another feature, Rating Prediction Error (RPE), represented the error between the real ratings and the predicted ratings for each user is also adopted due to its good performance in fake user detection [18].

3.2.1. Item similarity offset

Considering that strongly connected users/items shall have similar preferences/properties, we assume that users tend to give close ratings to similar items. If a user gives two very similar items a large difference in ratings, the user is more likely to be an attacker. We choose the items that the user assigned the highest rating and the lowest rating and calculate the inner product of the corresponding items' spectral features for each user. The more similar the two items, the larger the spectral features product. Thus, we can use the similarity information between the two types of items to capture this feature u_{iso}:

$$u_{iso} = F_{maxr} F_{minr}, \tag{4}$$

where F_{maxr} denotes the spectral feature of the item which the user gives the highest rating and F_{minr} the lowest rating, respectively. If user u gives item s the highest rating and t the lowest rating, $F_s F_t$ can express the similarity between item s and item t. If s and t are two very similar items, then $F_s F_t$ is high. Note that if a user has several highest ratings or lowest ratings, we will compute all the possibilities for the user and take the maximum u_{iso} as the feature. Fig. 2 shows the ISO distribution of the attackers and genuine users on different datasets. For the three injected MovieLens datasets, we set the attack size to 10% and filler size to 15%. Fig. 3 shows the average value of u_{iso} of attackers and genuine users different datasets. It is obvious that the two kinds of users differ greatly in this feature.



(a) MovieLens(average) (b) MovieLens(random) (c) MovieLens(bandwagon) (d) Amazon

Fig. 2. ISO distribution of attackers and genuine users. The red and blue lines represent the attackers and the genuine users, respectively. Obviously, the distribution of the two types of users is quite different. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

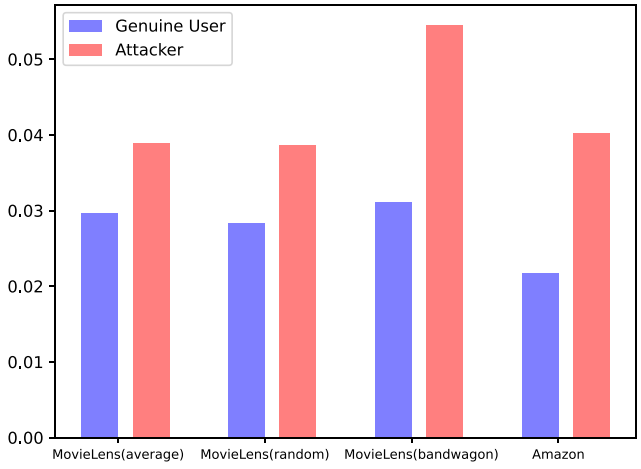


Fig. 3. Item similarity offsets of attackers and genuine users.

3.2.2. Rating prediction error

We adopt the idea that genuine users' preferences are easier to predict than those of attackers. When predicting ratings for all users, if the predicted rating of user u on item i differs too much from the real rating, then user u is likely to be an attacker [20]. This feature can provide a strong signal of whether this user is normal.

To capture this feature, we use matrix factorization to predict ratings for all users. We incorporate the user's authenticity into the loss function to make the prediction more inclined to predict genuine users' ratings. Suppose the authenticity of a user is low. In that case, the framework will reduce this user's contribution to the rating predicting task as this user is more likely to give unfair ratings. We calculate the mean square of all RPEs on the interacted items to get this feature u_{rpe}:

$$L_{rating} = fraud_u \sum_{u,i}(r_{ui} - \hat{r}_{ui})^2, \tag{5}$$

$$\hat{r}_{ui} = p_u^T q_i, \tag{6}$$

$$u_{rpe} = \sum_i (r_{ui} - \hat{r}_{ui})^2, \tag{7}$$

where r_{ui} is the real rating of user u on item i; p_u is the latent vector of user u; q_i is the latent vector of item i; $fraud_u$ is the authenticity of user u, which is initialized to 1 and will be updated on the attacker detection component.

3.3. Fake user detection

We connect u_{rpe}, u_{iso}, with the spectral characteristics of users:

$$e_u = E_{sp} \oplus u_{rpe} \oplus u_{iso}, \tag{8}$$

where $\oplus$ denotes the concatenation of the features; E_{sp} is the spectral feature of user u. Then we put e_u into the three-layer neural network for predicting due to the effectiveness of neural networks in classification work. We adopt the cross-entropy as the loss function.

$$e_1 = \sigma(W_1 e_u + b_1), \tag{9}$$

$$e_2 = \sigma(W_2 e_1 + b_2), \tag{10}$$

$$fraud_u = \sigma(W_3 e_2 + b_3), \tag{11}$$

$$L_{fraud} = -\frac{1}{N}\sum_i [y_i \log(fraud_u) + (1 - y_i)\log(1 - fraud_u)], \tag{12}$$

where $y_i = 1$ if the user u is a genuine user and $y_i = 0$ otherwise, W_1, W_2 and W_3 are the weight terms; b_1, b_2 and b_3 are the bias terms; $\sigma(\cdot)$ is the activation function, and N is the number of training users. Instead of training these two tasks separately, we combine their losses and jointly minimize the following loss function,

$$L = L_{fraud} + \lambda L_{rating}, \tag{13}$$

where L is the total loss of the framework and λ is a hyper-parameter to balance the effect of the two parts. The detailed procedure of SpDetector is shown in Algorithm 1.

Algorithm 1 SpDetector.

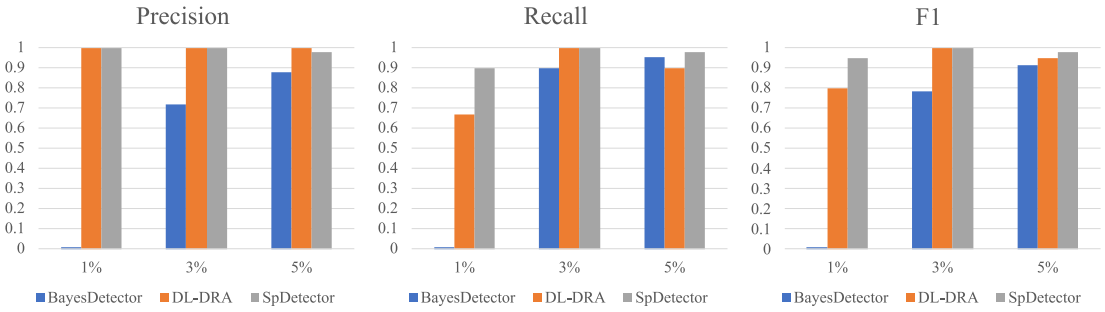
Input: Use-item rating matrix R; User labels for training.
Output: User labels in the testing set.

```
Construct Laplacian matrices of user hypergraph L_u and item hypergraph L_i from R;
Decompose the Laplacian matrices to get the user spectral feature E and item spectral feature F;
Calculate u_iso for each user;
for item = 1 to maxiter do
    Decompose the user–item rating matrix R;
    Calculate u_rpe for each user;
    Concatenate e_u = E_sp ⊕ u_rpe ⊕ u_iso for each user;
    Put feature e_u into a three-layer neural network to calculate fraud_u for each user;
end for
Use e_u to predict user labels (for user u, if fraud_u > 0.5, then user u is a genuine user)
```

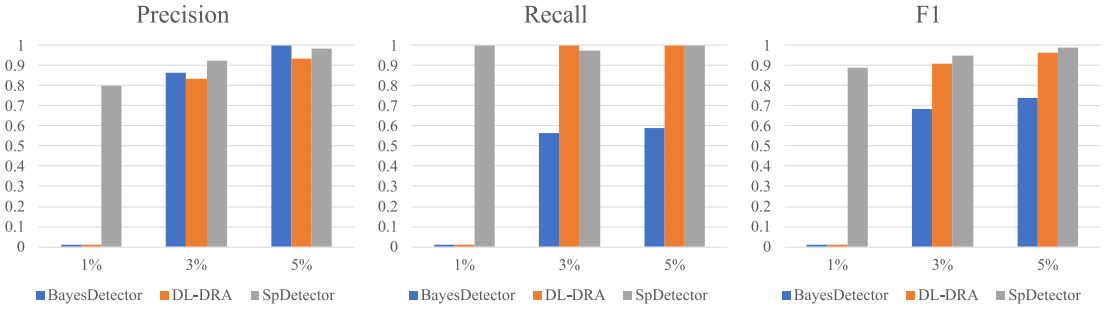
4. Experiments

We conduct extensive experiments on two public datasets to evaluate the performance of SpDetector on shilling attack detection. Particularly, our goal is to answer the following research questions (RQs):

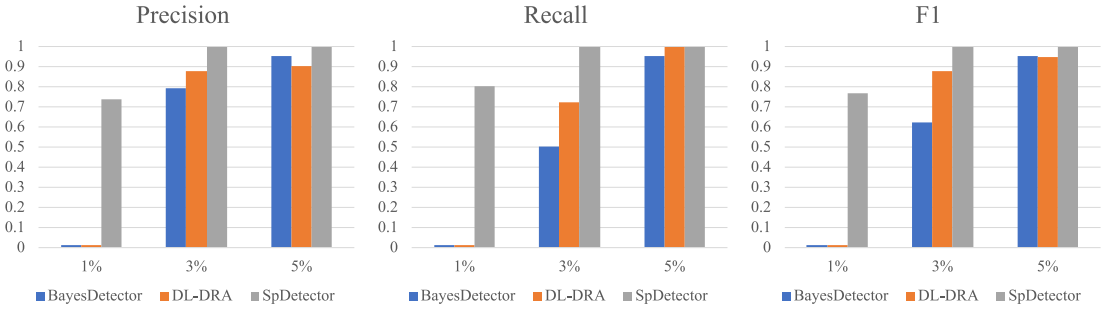
- **RQ1:** How does SpDetector perform when detecting attackers compared with the baseline methods in the real-world condition?
- **RQ2:** How does SpDetector perform when the attack size is small?
- **RQ3:** How does SpDetector perform under different attack sizes and filler sizes?
- **RQ4:** How do the three extracted features affect the shilling attack detection performance?



(a) Comparison experiments under random attack with different attack sizes on the MovieLens dataset



(b) Comparison experiments under bandwagon attack with different attack sizes on the MovieLens dataset



(c) Comparison experiments under average attack with different attack sizes on the MovieLens dataset

Fig. 4. Comparison experiment results under different attack sizes.

- **RQ5:** How does the hyper-parameter K affect the performance of SpDetector?
- **RQ6:** How do the features extracted by SpDetector distributed in low-dimensional space?

4.1. Datasets and evaluation metrix

We use the Amazon and MovieLens datasets to evaluate the proposed model. Amazon dataset [39] contains 60,000 ratings rated by 4902 users on 21,394 items, and MovieLens dataset contains 100,000 ratings rated by 943 users on 1682 movies. The Amazon dataset contains 1937 malicious users and 2965 genuine users. MovieLens dataset does not contain attackers. Without loss of generality, the fake profiles generated by attack models are injected according to the research [11, 40]. We leverage the target offset confusion method during the injection that assigns the target item with the second-highest rating or the second-lowest rating for concealment. The shilling attack detection can be seen as a binary classification task, thus we use precision Eq. (14), recall Eq. (15), and F1-score Eq. (16) to measure the performance. Let N be the number of attackers detected by the algorithms; N^a is the number of attackers detected correctly; N^t is the total number of attackers to be detected.

$$Precision = \frac{N^a}{N}, \tag{14}$$

$$Recall = \frac{N^a}{N^t}, \tag{15}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{16}$$

4.2. Implementation details

We adopt the random attack model, average attack model, and bandwagon attack model in the MovieLens dataset with a 15% attack size and 10% filler size to imitate the real-world dataset. We set the dimension of user spectral feature E to 50, item spectral feature F to 50, user latent vector $\mathbf{p}$ to 10, and item latent vector $\mathbf{q}$ to 10. The hyper-parameter λ, which controls the effect of the rating loss and detection loss, is set to 0.05. The number of neurons in the first layer, the second layer, and the output layer is set to 100, 200, and 1, respectively. The activation function of the two hidden layers is set to ReLU and sigmoid for the output layer. All modules are optimized with Adam optimizer, and the learning rate is set to 0.0002.

4.3. Overall performance comparison (RQ1)

To further analyze the detection effect of SpDetector, we use two datasets mentioned above for experiments. We also compare it with

other state-of-the-art methods such as FAP [41], SemiSAD [29], BayesDetector [14], GraphRFI [18], and DL-DRA [27].

- **FAP**: An unsupervised method based on explicit feature that propagates the degree that a user promotes or degrades a target item.
- **SemiSAD**: A semi-supervised learning method that first trains a classifier with naïve Bayes on some labeled user-profiles and then incorporates unlabeled user profiles for EM-λ to improve the Bayes classifier.
- **BayesDetector**: A supervised method utilizes matrix factorization and user embeddings to construct implicit features for each user and then use the Bayesian model to generate latent label information to update the implicit features for detection.
- **GraphRFI**: A supervised method that uses the graph convolution network and neural random forest to perform robust recommendation and fraudster detection.
- **DL-DRA**: A supervised method that uses the bicubic interpolation algorithm to reduce the sparsity of the rating matrix and uses the structured deep learning network for detection.

The experimental results are shown in Table 2. The bolded data in the table means the best results.

As can be seen from Table 2, all these methods have a good detecting performance on the MovieLens dataset, in which the evaluation metrics can reach above 90%. Compared with the first two statistical-based methods, the latter four methods that leverage the fine-grained interaction between users and items to detect shilling attacks have better performance. This phenomenon suggests that the latter four methods can better capture users' preferences and detect attackers more effectively.

As for the experiment results on the Amazon dataset, which is under complex attack, the latter four methods detect attackers more effectively than the former two methods. When the first two statistical-based methods are used to detect shilling attacks on a real-world dataset, the results can be unsatisfactory because there are limitations when using statistical features to describe a user. SpDetector performs better than BayesDetector and DL-DRA, indicating higher-order interactions along with explicit features can better distinguish attackers from genuine users. As for GraphRFI, its extracted features focus more on performing robust recommendations, showing limitations in shilling attack detection.

4.4. Comparison experiments under different attack sizes (RQ2)

It is often hard to distinguish attackers from genuine users when the attack size is small. To show the robustness of our SpDetector under different attack sizes, we conduct comparison experiments among three supervised detection methods on the MovieLens dataset under small and different attack sizes. We set the filler size to 3%, which is moderate for the small attack sizes. Note that we do not compare with the unsupervised and semi-supervised methods because of their low performance. Fig. 4 shows that under random attack, when the attack size is set to 1%, BayesDetector fails to distinguish attackers from genuine users. The DL-DRA method achieves high precision, but the recall value is only 0.6, which means that only 60% of the attackers have been detected successfully. However, our SpDetector effectively improves the detection performance with over 0.9 precision, recall, and F1. As the attack size increases to 3% and 5%, both DL-DRA and SpDetector can detect the attackers effectively.

Figs. 4(b) and 4(c) show that under average and bandwagon attack, both BayesDetector and DL-DRA fail to distinguish attackers from genuine users with 1% attack size, while SpDetector can still detect 73.3% attackers. As the attack size increases, the three methods' performance also increases, and we can still see the overwhelming superiority of SpDetector.

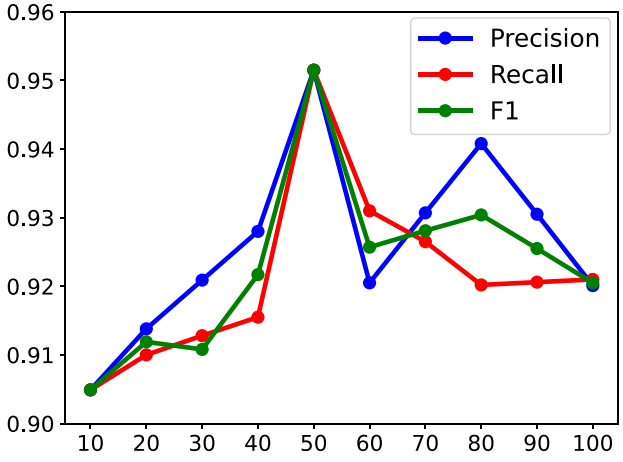


Fig. 5. Impact of parameter K.

4.5. The performance of spdetector under different attacks (RQ3)

To evaluate the performance of SpDetector under different attack models with different attack sizes and filler sizes, we conduct a simulation experiment on the MovieLens dataset. We take the random attack, average attack, and bandwagon attack as the attack models because the three models need rare details about the target recommender systems. The attack size is set to 1%, 3%, 5% and 7% respectively, and the filler size takes 1%, 3%, 5%, 7% 10% and 15%. We assume all the users in the original MovieLens dataset are genuine.

As shown in Table 3, SpDetector has high performance at different attack sizes and filler sizes under the three attack models. The evaluation metrics are almost over 95%, except with a 1% attack size. The effect of shilling attacks with only a 1% attack size is usually weak. Thus, the results are acceptable. When the attack size is only 1%, SpDetector performs extremely well (90% on average), indicating the robustness of our model [42]. With the increase of the attack size, the performance of the SpDetector is getting better. This is due to the increase of the attackers in the training set makes the features of the attackers better learned for the classification task. With the rise of the filler size, the detection performance first decreases and then increases. The performance first decreases because the attackers can better disguise themselves with the help of the filler items but then increases because the high number of filler items reduces the similarity between normal users and attackers, which makes the attackers easier to be detected.

4.6. Ablation studies (RQ4)

To study the effectiveness of the three features, we perform six variant experiments to show that all these features make contribute to shilling attack detection. For $x \in \{I, R, U, IR, IU, RU\}$, SpDetector-$x$ denotes SpDetector without component x where I, R and U denotes the ISO, the RPE, and the user spectral feature, respectively. Table 4 shows the experimental results on the MovieLens dataset (15% attack size and 10% filler size) and the Amazon dataset.

As shown in Table 4, user spectral features play a major role in the detection tasks, and the other two statistical-based features show poor performance when applied for detection separately. However, it is effective to combine the two statistics-based features with the user spectral features to detect attackers, and the ISO is slightly superior to the RPE. The improvement is not obvious on the injected MovieLens dataset, while on the Amazon dataset, the two statistical-based features are effective, showing their contribution to the complex real-world dataset.

Table 2
The experiment results on MovieLens with different attack models and Amazon.

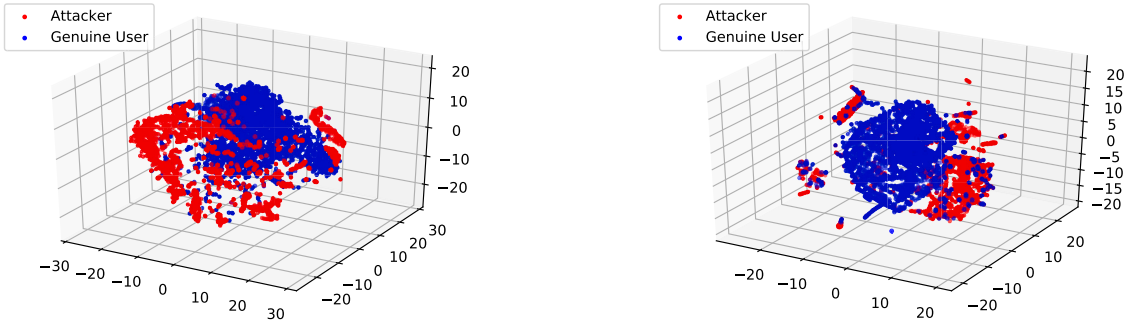
Method		FAP	SemiSAD	BayesDetector	GraphRFI	DL-DRA	SpDetector
MovieLens (random)	*Precision*	0.9631	0.9415	0.9706	0.9845	**0.9951**	0.9920
	Recall	0.9539	0.9181	1.0000	0.9920	0.9985	**1.0000**
	*F*1	0.9654	0.9255	0.9851	0.9882	**0.9968**	0.9960
MovieLens (average)	*Precision*	0.9820	0.9750	0.9500	0.9831	0.9960	**1.0000**
	Recall	0.9899	0.9636	1.0000	1.0000	1.0000	**1.0000**
	*F*1	0.9860	0.9693	0.9744	0.9915	0.9980	**1.0000**
MovieLens (bandwagon)	*Precision*	0.9680	0.9577	0.9618	0.9713	**0.9950**	0.9889
	Recall	0.9805	0.9870	0.9611	0.9862	0.9940	**0.9950**
	*F*1	0.9742	0.9721	0.9615	0.9787	**0.9945**	0.9919
Amazon	*Precision*	0.8943	0.6035	0.8957	0.8892	0.9108	**0.9515**
	Recall	0.7320	0.6208	0.9313	0.9197	0.9097	**0.9515**
	*F*1	0.8050	0.6120	0.9130	0.9042	0.9087	**0.9515**

Table 3
The results of SpDetector under different attack sizes and filler sizes.

Attack size			Filler size				
			3%	5%	7%	10%	15%
Random	1%	*Precision*	1.0000	0.9000	0.9333	0.9000	1.0000
		Recall	0.9000	0.9500	0.8500	0.9000	1.0000
		*F*1	0.9474	0.9243	0.8897	0.9000	1.0000
	3%	*Precision*	1.0000	0.9800	0.9667	0.9667	1.0000
		Recall	1.0000	1.0000	0.9800	0.9800	1.0000
		*F*1	1.0000	0.9899	0.9733	0.9733	1.0000
	5%	*Precision*	0.9778	1.0000	0.9818	1.0000	1.0000
		Recall	0.9778	0.9532	0.9511	0.9800	1.0000
		*F*1	0.9778	0.9761	0.9662	0.9899	1.0000
	7%	*Precision*	0.9867	0.9857	0.9857	1.0000	0.9818
		Recall	1.0000	1.0000	1.0000	0.9846	0.9611
		*F*1	0.9933	0.9928	0.9928	0.9923	0.9714
Average	1%	*Precision*	0.7334	0.8000	0.9000	0.9333	0.9000
		Recall	0.8000	0.8000	1.0000	1.0000	0.9000
		*F*1	0.7653	0.8000	0.9474	0.9655	0.9000
	3%	*Precision*	1.0000	0.9714	1.0000	0.9778	0.9667
		Recall	1.0000	1.0000	1.0000	1.0000	1.0000
		*F*1	1.0000	0.9855	1.0000	0.9888	0.9830
	5%	*Precision*	1.0000	1.0000	1.0000	0.9636	0.9833
		Recall	1.0000	0.9423	1.0000	0.9428	1.0000
		*F*1	1.0000	0.9706	1.0000	0.9531	0.9916
	7%	*Precision*	1.0000	0.9800	0.9846	1.0000	0.9857
		Recall	0.9818	1.0000	1.0000	0.9875	1.0000
		*F*1	0.9908	0.9899	0.9923	0.9937	0.9928
Bandwagon	1%	*Precision*	0.8000	0.8095	0.8333	0.9167	0.9375
		Recall	1.0000	1.0000	1.0000	1.0000	1.0000
		*F*1	0.8889	0.8947	0.9091	0.9565	0.9677
	3%	*Precision*	0.9250	1.0000	0.9750	1.0000	0.9524
		Recall	0.9762	0.9381	1.0000	1.0000	1.0000
		*F*1	0.9499	0.9681	0.9873	1.0000	0.9756
	5%	*Precision*	0.9833	1.0000	0.9870	0.9288	0.9137
		Recall	1.0000	1.0000	0.9841	0.9612	0.9861
		*F*1	0.9916	1.0000	0.9856	0.9447	0.9485
	7%	*Precision*	0.9857	0.9848	0.9985	1.0000	0.9985
		Recall	1.0000	1.0000	1.0000	1.0000	0.9800
		*F*1	0.9928	0.9923	0.9992	1.0000	0.9892

Table 4
Comparison experiment results between SpDetector and its variants.

Method	MovieLens (random)			MovieLens (average)			MovieLens (bandwagon)			Amazon		
	Precision	*Recall*	*F*1	*Precision*	*Recall*	*F*1	*Precision*	*Recall*	*F*1	*Precision*	*Recall*	*F*1
SpDetector-I	0.9905	1.0000	0.9950	1.0000	0.9985	0.9952	0.9861	0.9926	0.9893	0.9358	0.9338	0.9348
SpDetector-R	0.9860	1.0000	0.9930	1.0000	1.0000	1.0000	0.9889	0.9950	0.9919	0.9462	0.9438	0.9450
SpDetector-U	0.5714	0.5000	0.5334	0.6667	0.6000	0.6316	0.7000	0.5385	0.6087	0.7254	0.7231	0.7240
SpDetector-IR	0.9840	1.0000	0.9920	1.0000	0.9985	0.9952	0.9885	0.9920	0.9902	0.9225	0.9313	0.9269
SpDetector-IU	0.4218	0.5120	0.4625	0.5880	0.5293	0.5571	0.5946	0.5223	0.5561	0.6124	0.6232	0.6178
SpDetector-RU	0.5625	0.5000	0.5294	0.5848	0.5520	0.5679	0.6220	0.5143	0.5630	0.6880	0.7023	0.6951
SpDetector	**0.9920**	**1.0000**	**0.9960**	**1.0000**	**1.0000**	**1.0000**	**0.9889**	**0.9950**	**0.9919**	**0.9515**	**0.9515**	**0.9515**



(a) Features extracted by BayesDetector

(b) Features extracted by SpDetector

Fig. 6. Data visualization of features extracted by BayesDetector and SpDetector. The red and blue points represent attackers and genuine users, respectively. Obviously, the number of users overlapping in the clusters is relatively small in the SpDetector. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.7. Parameter settings (RQ5)

In this part, we investigate the impact of the dimension of the spectral feature K. The rest of the parameters are set the same as in Section 4.2. Then we study the influence of K from 10 to 100. Fig. 5 shows the performance of our model under different K values. From the figure, we can see that the performance curve first increases and then decreases, and SpDetector obtains the best performance when K=50. When K changes from 10 to 50, the performance becomes better, which indicates that the first 50 eigenvectors are of great significance for shilling attack detection. When K is greater than 50, the performance becomes worse and finally tends to be stable, which indicates that the eigenvectors after 50 have little effect on the detection and will bring noise, thus reducing the detection performance. However, results on other K values are also acceptable, which validates the stability of our model.

4.8. Analysis of extracted features (RQ6)

The experiments also show that SpDetector is more effective than BayesDetector, which is the state-of-the-art method using implicit features for shilling attack detection. To explore this reason, we adopt data visualization to analyze the characteristics of the features. The enriched spectral features of the hypergraph obtain the users' higher-order interactions, which contains richer topological information of the users.

Data visualization helps to intuitively understand the distribution of original user features and discover patterns between genuine users and attackers. We perform data visualization of the features extracted by BayesDetector and SpDetector on the Amazon dataset, which is under complex attack and is more in tune with real-world attacks. We use the T-SNE proposed by Hinton et al. [43] to visualize the extracted features. The experiment results are shown in Fig. 6.

Figs. 6(a) and 6(b) show that the features of the similar users are distributed more closely, and the boundary of the two kinds of users is more clear in Fig. 6(b) compared with Fig. 6(a) Moreover, the number of users overlapping in the clusters is relatively small in Fig. 6(b) Therefore the features extracted by SpDetector are more effective in detecting attackers in real-world datasets.

5. Conclusion and future work

In this paper, we propose a shilling attack detection method named SpDetector, which combines the implicit and explicit features to balance the effectiveness and generality. From the implicit perspective, this method constructs user hypergraphs and item hypergraphs to capture the high-order relationships hidden in the user–item interactions. From the explicit perspective, this method extracts the item similarity offsets and the rating prediction errors to distinguish attackers. A deep neural network is trained to detect shilling attacks with these features. The experiments show that the features captured by our model outperform several state-of-the-art models significantly, and the ablation studies also prove the effectiveness of all these extracted features in the shilling attack detection task.

Attackers will utilize the social network to disguise themselves, so that combining spectral features and social information to detect spammers effectively will be our future work. Note that a simple three-layer neural network is used in our SpDetector; since more complex deep neural networks may have better performance, we will also try to build a more advanced classifier for future work.

CRediT authorship contribution statement

Hao Li: Conceptualization, Software, Validation, Data curation, Writing – original draft, Methodology. **Min Gao:** Resources, Supervision, Project administration, Funding acquisition, Writing – review & editing. **Fengtao Zhou:** Software, Data curation, Writing – review & editing. **Yueyang Wang:** Formal analysis, Software. **Qilin Fan:** Validation, Investigation. **Linda Yang:** Writing – review & editing, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (62176028), the Natural Science Foundation Project of Chongqing, China (cstc2020jcyj-msxmX0690), and the Fundamental Research Funds for the Central Universities (2020CDJ-LHZZ-039).

References

[1] Gao M, Liu K, Wu Z. Personalisation in web computing and informatics: Theories, techniques, applications, and future research. Inf Syst Front 2010;12(5):607–29.
[2] Gao M, Wu Z, Jiang F. Userrank for item-based collaborative filtering recommendation. Inform Process Lett 2011;111(9):440–6.
[3] Yu J, Gao M, Rong W, Li W, Xiong Q, Wen J. Hybrid attacks on model-based social recommender systems. Physica A 2017;483:171–81.

[4] Anelli VW, Deldjoo Y, Noia TD, Sciascio ED, Merra FA. Sasha: Semantic-aware shilling attacks on recommender systems exploiting knowledge graphs. In: Harth A, Kirrane S, Ngomo AN, Paulheim H, Rula A, Gentile AL, Haase P, Cochez M, editors. The semantic web - 17th international conference, proceedings. Lecture notes in computer science, vol. 12123, Springer; 2020, p. 307–23. http://dx.doi.org/10.1007/978-3-030-49461-2_18.

[5] Alonso S, Bobadilla J, Ortega F, Moya R. Robust model-based reliability approach to tackle shilling attacks in collaborative filtering recommender systems. IEEE Access 2019;7:41782–98. http://dx.doi.org/10.1109/ACCESS.2019.2905862.

[6] Deldjoo Y, Noia TD, Merra FA. Assessing the impact of a user-item collaborative attack on class of users. In: Shalom OS, Jannach D, Guy I, editors. Proceedings of the 1st workshop on the impact of recommender systems co-located with 13th ACM conference on recommender systems. CEUR Workshop Proceedings, vol. 2462, CEUR-WS.org; 2019.

[7] Zheng X, Luo Y, Sun L, Ding X, Zhang J. A novel social network hybrid recommender system based on hypergraph topologic structure. World Wide Web 2018;21(4):985–1013. http://dx.doi.org/10.1007/s11280-017-0494-5.

[8] Deldjoo Y, Noia TD, Merra FA. A survey on adversarial recommender systems: From attack/defense strategies to generative adversarial networks. ACM Comput Surv 2021;54(2):35:1–38. http://dx.doi.org/10.1145/3439729.

[9] Gunes I, Kaleli C, Bilge A, Polat H. Shilling attacks against recommender systems: a comprehensive survey. Artif Intell Rev 2014;42(4):767–99. http://dx.doi.org/10.1007/s10462-012-9364-9.

[10] Burke R, Mobasher B, Williams C, Bhaumik R. Classification features for attack detection in collaborative recommender systems. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2006, p. 542–7.

[11] Li W, Gao M, Li H, Zeng J, Xiong Q, Hirokawa S. Shilling attack detection in recommender systems via selecting patterns analysis. IEICE Trans Inf Syst 2016;99(10):2600–11.

[12] Song Y, Gao M, Yu J, Li W, Wen J, Xiong Q. Pud: Social spammer detection based on pu learning. In: International conference on neural information processing. Springer; 2017, p. 177–85.

[13] Dou T, Yu J, Xiong Q, Gao M, Song Y, Fang Q. Collaborative shilling detection bridging factorization and user embedding. In: International conference on collaborative computing: networking, applications and worksharing. Springer; 2017, p. 459–69.

[14] Yang F, Gao M, Yu J, Song Y, Wang X. Detection of shilling attack based on bayesian model and user embedding. In: 2018 IEEE 30th international conference on tools with artificial intelligence. IEEE; 2018, p. 639–46.

[15] Yu W, Qin Z. Spectrum-enhanced pairwise learning to rank. In: The world wide web conference. 2019, p. 2247–57.

[16] Chirita P-A, Nejdl W, Zamfir C. Preventing shilling attacks in online recommender systems. In: Proceedings of the 7th annual ACM international workshop on web information and data management. 2005, p. 67–74.

[17] Cai H, Zhang F. Detecting shilling attacks in recommender systems based on analysis of user rating behavior. Knowl-Based Syst 2019;177:22–43.

[18] Zhang S, Yin H, Chen T, Hung QVN, Huang Z, Cui L. GCN-based user representation learning for unifying robust recommendation and fraudster detection. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. 2020, p. 689–98.

[19] Guo J, Liu G, Zuo Y, Wu J. Learning sequential behavior representations for fraud detection. In: 2018 IEEE international conference on data mining. IEEE; 2018, p. 127–36.

[20] White CM. Consistency in cognitive social behaviour: an introduction to social psychology. Psychology Press; 2015.

[21] Zhang Z, Kulkarni SR. Detection of shilling attacks in recommender systems via spectral clustering. In: 17th international conference on information fusion. IEEE; 2014, p. 1–8.

[22] Gunes I, Kaleli C, Bilge A, Polat H. Shilling attacks against recommender systems: a comprehensive survey. Artif Intell Rev 2014;42(4):767–99.

[23] Hurley N, Cheng Z, Zhang M. Statistical attack detection. In: Proceedings of the third ACM conference on recommender systems. 2009, p. 149–56.

[24] Williams CA, Mobasher B, Burke R, Bhaumik R. Detecting profile injection attacks in collaborative filtering: a classification-based approach. In: International workshop on knowledge discovery on the web. Springer; 2006, p. 167–86.

[25] Mobasher B, Burke R, Bhaumik R, Williams C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Trans Internet Technol 2007;7(4):23–es.

[26] Batmaz Z, Yilmazel B, Kaleli C. Shilling attack detection in binary data: a classification approach. J Ambient Intell Humaniz Comput 2020;11(6):2601–11.

[27] Zhou Q, Wu J, Duan L. Recommendation attack detection based on deep learning. J Inf Secur Appl 2020;52:102493.

[28] Wu Z, Wu J, Cao J, Tao D. HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. 2012, p. 985–93.

[29] Cao J, Wu Z, Mao B, Zhang Y. Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system. World Wide Web 2013;16(5–6):729–48.

[30] Wu Z, Wang Y, Wang Y, Wu J, Cao J, Zhang L. Spammers detection from product reviews: a hybrid model. In: 2015 IEEE international conference on data mining. IEEE; 2015, p. 1039–44.

[31] Zheng P, Yuan S, Wu X, Li J, Lu A. One-class adversarial nets for fraud detection. In: Proceedings of the AAAI conference on artificial intelligence, vol. 33. 2019, p. 1286–93.

[32] Cai H, Zhang F. An unsupervised method for detecting shilling attacks in recommender systems by mining item relationship and identifying target items. Comput J 2019;62(4):579–97.

[33] Zhang Z, Kulkarni SR. Graph-based detection of shilling attacks in recommender systems. In: IEEE international workshop on machine learning for signal processing. IEEE; 2013, p. 1–6. http://dx.doi.org/10.1109/MLSP.2013.6661953.

[34] Zhang F, Qu Y, Xu Y, Wang S. Graph embedding-based approach for detecting group shilling attacks in collaborative recommender systems. Knowl Based Syst 2020;199:105984. http://dx.doi.org/10.1016/j.knosys.2020.105984.

[35] Yang Y, Deng S, Lu J, Li Y, Gong Z, Hao Z, et al. Graphlshc: Towards large scale spectral hypergraph clustering. Inform Sci 2021;544:117–34.

[36] Zhou D, Huang J, Schölkopf B. Learning with hypergraphs: Clustering, classification, and embedding. In: Advances in neural information processing systems. 2007, p. 1601–8.

[37] Van Lierde H, Chow TW. A hypergraph model for incorporating social interactions in collaborative filtering. In: Proceedings of the 2017 international conference on data mining, communications and information technology. 2017, p. 1–6.

[38] Zhang F, Zhou Q. HHT–SVM: An online method for detecting profile injection attacks in collaborative recommender systems. Knowl-Based Syst 2014;65:96–105.

[39] Xu C, Zhang J, Chang K, Long C. Uncovering collusive spammers in Chinese review websites. In: Proceedings of the 22nd ACM international conference on information & knowledge management. 2013, p. 979–88.

[40] Lin C, Chen S, Li H, Xiao Y, Li L, Yang Q. Attacking recommender systems with augmented user profiles. In: Proceedings of the 29th ACM international conference on information & knowledge management. 2020, p. 855–64.

[41] Zhang Y, Tan Y, Zhang M, Liu Y, Chua T-S, Ma S. Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation. In: Twenty-fourth international joint conference on artificial intelligence. 2015.

[42] Deldjoo Y, Noia TD, Sciascio ED, Merra FA. How dataset characteristics affect the robustness of collaborative recommendation models. In: Huang J, Chang Y, Cheng X, Kamps J, Murdock V, Wen J, Liu Y, editors. Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. ACM; 2020, p. 951–60. http://dx.doi.org/10.1145/3397271.3401046.

[43] Maaten Lvd, Hinton G. Visualizing data using t-SNE. J Mach Learn Res 2008;9(Nov):2579–605.