Hindawi
Security and Communication Networks
Volume 2019, Article ID 6523183, 13 pages
https://doi.org/10.1155/2019/6523183

*Research Article*

# Detecting Shilling Attacks with Automatic Features from Multiple Views

**Yaojun Hao**,[1,2] **Fuzhi Zhang**,[2] **Jian Wang**,[2] **Qingshan Zhao**,[1] **and Jianfang Cao**[1]

[1]*Department of Computer Science and Technology, Xinzhou Teachers University, Xinzhou, Shanxi Province, China*
[2]*School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei Province, China*

Correspondence should be addressed to Yaojun Hao; haoyaojun1@163.com

Received 4 January 2019; Accepted 8 July 2019; Published 5 August 2019

Academic Editor: Jesús Díaz-Verdejo



Due to the openness of the recommender systems, the attackers are likely to inject a large number of fake profiles to bias the prediction of such systems. The traditional detection methods mainly rely on the artificial features, which are often extracted from one kind of user-generated information. In these methods, fine-grained interactions between users and items cannot be captured comprehensively, leading to the degradation of detection accuracy under various types of attacks. In this paper, we propose an ensemble detection method based on the automatic features extracted from multiple views. Firstly, to collaboratively discover the shilling profiles, the users' behaviors are analyzed from multiple views including ratings, item popularity, and user-user graph. Secondly, based on the data preprocessed from multiple views, the stacked denoising autoencoders are used to automatically extract user features with different corruption rates. Moreover, the features extracted from multiple views are effectively combined based on principal component analysis. Finally, according to the features extracted with different corruption rates, the weak classifiers are generated and then integrated to detect attacks. The experimental results on the MovieLens, Netflix, and Amazon datasets indicate that the proposed method can effectively detect various attacks.

## 1. Introduction

Nowadays, information overload has become a prominent problem, which results in the low efficiency for finding products, services, and so on. Collaborative filtering (CF)-based recommender systems can be used to alleviate such problems, which are widely used in e-commerce, social network, IoT services, and so on. However, in some open environments, CF-based recommender systems are vulnerable, in which the attackers are likely to inject a large number of fake profiles in order to increase/decrease the frequency of the target items. The behaviors of injecting a large number of fake profiles are referred to as shilling attacks, which have the negative impacts on the prediction quality of recommender systems and make the users lose trust in the systems [1].

Generally, detection of shilling attacks can be seen as a binary classification problem; that is, for each user profile, the classified result can be a normal user or an attacker [2]. Therefore, a number of detection features have been proposed and some machine learning methods are used to separate the attackers from genuine users. However, the existing features are extracted by human engineering from one piece of user-generated information (e.g., ratings, timestamps, and item popularity), which mainly focus on some specific types of attacks and need high knowledge costs. Due to the diversity and variability of attackers' strategies, it is very difficult to fully characterize the shilling profiles from one piece of user-generated information (i.e., single view) [3, 4]. Moreover, since the number of fake profiles is rare relative to the genuine ones, the detection of shilling attacks can be seen as an imbalanced classification problem [5]. Accordingly, the detection methods often suffer poor performance under various types of attacks with low attack sizes.

To address the above limitations, we propose an ensemble detection method based on the features automatically extracted from multiple views. To collaboratively discover the shilling profiles, the features are extracted from ratings, item popularity, and user-user graph views. Moreover, to reduce the high knowledge costs of feature engineering in the existing methods, we adopt stacked denoising autoencoders

Table 1: The common attack models.

| Attack model | $I_s$ | | $I_F$ | | $i_t$ |
|---|---|---|---|---|---|
| | Items | Rating | Items | Rating | Rating |
| Random attack | Not used | | Randomly chosen | System mean | $r_{\max}/r_{\min}$ |
| Average attack | Not used | | Randomly chosen | Item mean | $r_{\max}/r_{\min}$ |
| AoP attack | Not used | | Top x% of most popular items | Item mean | $r_{\max}/r_{\min}$ |
| User random shifting | Not used | | Randomly chosen | System mean+random | $r_{\max}/r_{\min}$ |
| User average shifting | Not used | | Randomly chosen | Item mean+random | $r_{\max}/r_{\min}$ |
| Target random shifting | Not used | | Randomly chosen | System mean | $r_{\max} - 1/r_{\min} + 1$ |
| Target average shifting | Not used | | Randomly chosen | Item mean | $r_{\max} - 1/r_{\min} + 1$ |
| Power item attack | Power item | Item mean | | | $r_{\max}/r_{\min}$ |
| Power user attack | Copy of the power user's items and ratings | | | | $r_{\max}/r_{\min}$ |

(SDAEs) and principal component analysis (PCA) to automatically extract detection features. Finally, to alleviate the imbalanced classification problem, the detector is realized by integrating weak classifiers with features at different scales.

The main contributions are summarized as follows:

(1) We propose a framework to analyze users' behaviors from multiple views, which can reveal the fine-grained interactions between users and items and interactions between users and users.

(2) We propose a method to automatically extract the user features based on SDAEs and PCA, which can learn the robust and low-dimensional features with low knowledge costs.

(3) The base classifiers are generated with the features at different scales when SDAEs are corrupted with different rates, and then they are integrated to detect various attacks.

The remainder of the paper is organized as follows. Section 2 introduces the background and related work. Section 3 presents the proposed method in detail. In Section 4, we present and discuss the experimental results. Finally, we conclude the paper in Section 5.

## 2. Background and Related Work

*2.1. Shilling Attack Models.* In recommender system, the attack is realized by inserting shilling profiles to cause bias on target items [6]. A shilling profile can be defined as four sets of items [6–8]. Firstly, the set of selected items, $I_S$, is to form the characteristics of the attack. Secondly, the set of filler items, $I_F$, is usually chosen randomly to obstruct detection [6]. Thirdly, a unique item $i_t$ is the target item. Finally, $I_\phi$ is the set of unrated items. Table 1 lists the common attack models, in which $r_{\max}$ and $r_{\min}$ denote the highest and lowest rating values, respectively. The system mean denotes the ratings with normal distribution around system overall mean and standard deviation. The item mean denotes the ratings with normal distribution around each item's mean and standard deviation.

In Table 1, random attack and average attack are basic attack models [1, 6]. AoP (average over popular items) attack is an obfuscated form of average attack, which chooses filler items with equal probability from the top x% of most popular items, rather than from the entire catalogue of items [9].

User shifting and target shifting strategies are used to obfuscate the basic attacks to avoid detection [7]. When the two strategies are applied in the basic attacks, user random shifting attack, user average shifting attack, target random shifting attack, and target average shifting attack can be deployed. To facilitate the representation, we define the shifting attack as a collection with the same portion of user random shifting attack, user average shifting attack, target random shifting attack, and target average shifting attack [3].

Power items and power users are able to influence the largest group of items and users, respectively [10, 11]. In power item attack and power user attack, the strategies for selecting power items and power users include the top-N highest similarity scores, in-degree centrality, and number of ratings [10, 11].

In the experiments, the filler size (fs) refers to the ratio between the number of items rated by a user and the number of entire items. The attack size (as) refers to the ratio between the number of attackers and the number of genuine users [7, 12].

*2.2. Sparse Denoising Autoencoder.* Autoencoder (AE) is a feed-forward neural network which learns features through the hidden structures [13, 14]. In general, a single AE consists of the input layer, the encoding layer, and the decoding layer.

When the AE is fed with the input data $\mathbf{x}$, the encoding $\mathbf{h}$ can be obtained as the hidden representation by the following formula:

$$\mathbf{h} = s_f\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{p}\right) \tag{1}$$

where $s_f$, $\mathbf{W}^{(1)}$, and $\mathbf{p}$ denote the activation function, the weight matrix, and the bias vector, respectively.

In a similar way, the AE tries to reconstruct the input vector from the hidden representation by the following formula:

$$\hat{\mathbf{x}} = s_g\left(\mathbf{W}^{(2)}\mathbf{h} + \mathbf{q}\right) \tag{2}$$

where $\hat{\mathbf{x}}$, $s_g$, $\mathbf{W}^{(2)}$, and $\mathbf{q}$ denote the vector of decoding layer, the activation function, the weight matrix, and the bias vector, respectively.

As a result, the learning problem of the AE is to minimize the reconstruction error between the inputs and the outputs by optimizing the following loss function:

$$J_{AE}(\theta) = \sum L\left(\mathbf{x}, s_g\left(s_f(\mathbf{x})\right)\right) \tag{3}$$

where $\theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{p}, \mathbf{q}\}$ and $L()$ represents the loss function.

A sparse autoencoder (SAE) is an AE that requires that most of the neurons are zero. Usually, $KL$ divergence is selected to add sparsity restrictions. Therefore, the loss function of (3) can be expressed by

$$J_{SAE}(\theta) = \sum L\left(\mathbf{x}, s_g\left(s_f(\mathbf{x})\right)\right) + \beta\sum_{j=1}^{d} KL\left(\rho \parallel \widehat{\rho}_j\right) \tag{4}$$

where $\beta$ is the penalty coefficient. $\rho$ denotes the sparse parameter, and $\widehat{\rho}$ denotes the average activation in the hidden units. The $KL$ divergence is computed by $KL(\rho \parallel \widehat{\rho}_j) = \rho\log(\rho/\widehat{\rho}_j) + (1-\rho)\log((1-\rho)/(1-\widehat{\rho}_j))$.

A denoising autoencoder (DAE) is an AE that can be optimized to reconstruct input data from partial and random corruption [15]. Let $\widetilde{\mathbf{x}}$ denote the corrupted version of $\mathbf{x}$. The reconstruction error of the DA can be expressed by the following.

$$J_{DAE}(\theta) = \sum L\left(\mathbf{x}, s_g\left(s_f(\widetilde{\mathbf{x}})\right)\right) \tag{5}$$

Therefore, based on the sparse restriction in (3), the reconstruction error of the sparse denoising autoencoder (SDAE) can be expressed by the following.

$$J_{SDAE}(\theta) = \sum L\left(\mathbf{x}, s_g\left(s_f(\widetilde{\mathbf{x}})\right)\right) + \beta\sum_{j=1}^{d} KL\left(\rho \parallel \widehat{\rho}_j\right) \tag{6}$$

In general, the deep network in SDAEs is constructed by stacking SDAE.

*2.3. Related Work.* To detect shilling profiles, a number of detection methods have been proposed. Most of them mainly rely on single view information, such as rating values, rating time, and item popularity.

In the methods based on single view, from rating values, Yang et al. [5] proposed 3 features and then combined other detection features to detect shilling attacks. They used rescale AdaBoost algorithm as the classifier algorithm. This method can effectively detect standard attack, but shows low detection rates for some attacks with small attack and filler sizes. Zhou et al. [16] proposed a two-phase shilling attack detection method, SVM-TIA. This method first extracted 6 artificial features, used SVM to generate a set of suspicious profiles, and then removed genuine profiles from this set with target item analysis. However, it is difficult for this method to detect AoP, power item, and power user attacks. In AoP attacks, the fake ratings are more like genuine users and not easy to be filtered out. Hurley et al. [17] proposed statistical detectors for the standard attacks and AoP attack based on Neyman-Pearson statistical method. However, this method suffers from poor performances under power user and power item attacks. Based on rating timestamps, Xia et al. [18] dynamically divided time intervals and used hypothesis test detection-based framework to identify the anomaly items. However, this method assumes that the shilling profiles are injected in short periods, which cannot be always true in practice. Based on item popularity, Zhang et al. [19] constructed a rating series from item novelty and used Hilbert-Huang transform method to extract detection features. Li et al. [20] extracted the user features according to the item popularity distributions. However, these methods cannot effectively detect the power user and power item attacks.

In the methods based on multiple views, from the information of rating values and corated items, Dou et al. [2] proposed a collaborative shilling detection model, which learned the latent factors by decomposing the user-item rating matrix and the user-user cooccurrence matrix. After that, the latent factors were used as features, and decision tree was used as a classifier algorithm to detect attackers. In [3], based on the rating values and item popularity fused with temporal information, 17 artificial detection features were extracted. Then, these features were divided into several subsets by a feature set partition algorithm to construct multiple optimal classification views. Therefore, a multiview ensemble framework was designed as detection method. Similar to the multiple views approaches mentioned above, Shen et al. [4] proposed a social spammer detection method, which jointly integrated multiple views information and a social regularization term into the nonnegative matrix decomposition model. In [4], the multiple views include URLs, Hashtag, text, and social relation. However, in recommender systems, we hardly obtain such information due to the privacy protection. Therefore, based on the simple rating information, it is a challenge to detect various attacks in recommender systems.

Different from [2], we extract the detection features from multiple views including rating values, item popularity, and interaction between users. Unlike the methods based on the human engineering features [3, 5, 7, 16, 19, 20], we use SDAEs to automatically extract detection features under different corruption rates.

## 3. The Proposed Method

The framework of our proposed method is depicted in Figure 1, which consists of three stages: data preprocessing (stage 1), feature extraction with SDAEs and PCA (stage 2), and generation of weak classifier and detection (stage 3). At stage 1, we convert the rating records to the user-item rating matrix and user-item popularity matrix. Also, we construct the weighted user-user graph. At stage 2, we use SDAEs to learn the features from three matrixes, respectively. These features are directly combined and then analyzed by PCA. Moreover, different detection features can be automatically extracted when SDAEs are corrupted with different rates. At stage 3, we use SVM as weak classifier based on the different features and integrate them to generate the detection result.

*3.1. Data Preprocessing.* In collaborative recommender systems, the shilling attacks may cause the abnormalities in the

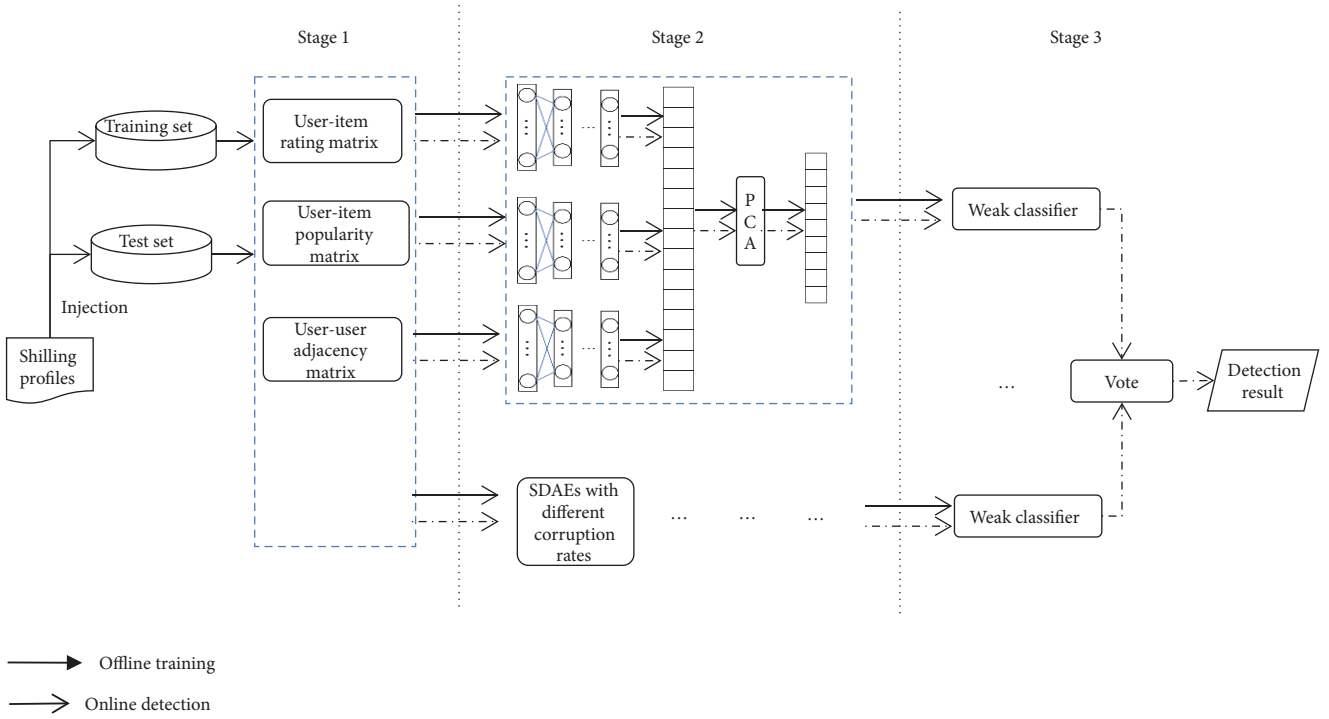



Figure 1: Framework of the proposed detection method.

distribution of the ratings or item popularity. Therefore, some detection features have been extracted from ratings or item popularity by human engineering. Also, some researchers [2, 21] have noticed that the shilling attacks may lead to the abnormal structural information between users. As we all know, attackers' strategies are so diverse and protean that the single view information hardly characterizes their behaviors completely. In fact, the above information has different abilities to characterize attackers' behaviors. Therefore, it is more reasonable to detect shilling profile from multiple views of user behaviors. To further describe the differences between attackers and genuine ones, we synthetically take advantage of ratings, item popularity, and user-user graph information.

According the rating records in recommender system, the user-item matrix can be acquired. For user-item popularity matrix, we first define the novelty of item and then convert the rating records to user-item popularity matrix by replacing rating value with novelty of item. Moreover, since the sequence of items affects the detection results in deep learning methods [22], the items are sorted by novelty of item in order to cluster the similar items and facilitate the learning of SDAEs.

*Definition 1.* Novelty of Item (*NI*). The novelty of item refers to the degree of difference between the item and other items [23], which is computed as follows:

$$NI_i = \frac{1}{|U_i|} \sum_{u \in U_i, r_{u,i} \neq 0} NI_{u,i} \tag{7}$$

where $U_i$ represents the set of users who rated item $i$. $|\bullet|$ represents the cardinality of the set. $NI_{u,i}$ denotes the novelty of item $i$ to user $u$, which is computed as follows:

$$NI_{u,i} = \frac{1}{|I_u|} \sum_{j \in I_u, r_{u,j} \neq 0} (1 - sim(i, j)) \tag{8}$$

where $I_u$ represents the set of items rated by user $u$ and $sim(i, j)$ denotes the cosine similarity between item $i$ and item $j$.

For user-user graph, we assume that the edges exist in the users with corated items. Moreover, different from the unweighted graph in [21], we use the number of corated items to weight the edge between two users. The reason is that attackers tend to promote/demote target items in group [2]. Also, in collaborative filtering recommender system, the attackers are likely to become users' neighbours by the similarity calculation, in which the number of corated items plays an import role.

Based on the above analysis, let *DR* denote rating records and **R** and **P** denote the user-item ratings matrix and user-item popularity matrix, respectively. **M** denotes the adjacency matrix of weighted user-user graph *G*. The function *indexU*(*curi*) returns the user ID corresponding to cursor *curi* in users set. The *rate(DR, tempi, i)* returns the ratings for the item *i* with the user *tempi*; the algorithm for data preprocessing is described as in Algorithm 1.

Algorithm 1 contains two parts. The first part (lines 1-4) is to calculate the novelty of items and construct the item popularity sequence based on the novelty of items, which is executed offline. The second part (lines 8-22) is to generate the matrixes **R, P, M**. In the rating records, for each user who

**Input**: *DR*
**Output**: **R, P, M**
1 **if** (*item sequence is not updated*) **then**
2 *Calculate the novelty of items* $NI_i$ *according to formula (7)*
3 *Construct the item popularity sequence SQI*
4 **end if**
8 **for each** $i \in I$ **do**
9 **for** *curi*=1 *to* $|U_i|$
10 *ui*=*indexU*(*curi*)
11 *R(ui,i)=rate(DR,ui,i)*
12 *P(ui,i)*=$NI_i$
13 **for** *curj= curi*+1 *to* $|U_i|$
14 *uj*=*indexU*(*curj*)
15 **if** *ui!=uj* **then**
16 *M*(*ui, uj*)=*M*(*ui, uj*)+1
17 *M*(*uj, ui*)=*M*(*ui, uj*)
18 **end if**
19 **end for**
20 **end for**
21 **end for**
22 **Return** R, P, M

Algorithm 1: Data preprocessing.

rated the current item, the user's rating is transformed into the element of matrix **R** (line 11). Also, the item popularity is transformed into the element of matrix **P** (line 11) according the novelty of item. Furthermore, the weight of edge between two users is accumulated (lines 13-19). After all the items are traversed, the matrixes **R, P, M** are returned (line 22).

In the first part, the time complexity for construction of item popularity sequence (lines 2-3) is at most $O(|U| \times |I|^3 + |I|^2)$. In practice, this part can be performed at idle time.

In the second part, the time complexity for generation of **R, P** (lines 10-12) is at most $O(|U| \times |I|)$. The time complexity for generation of **M** (lines 13-19) is at most $O(|I| \times |U|^2)$.

Therefore, the time complexity of Algorithm 1 can be described as $O(|U| \times |I|^3 + |I|^2) + O(|I| \times |\mathrm{U}|^2)$.

*3.2. Feature Extraction by SDAEs and PCA.* Although some detection features have been extracted by human engineering, these artificial features are not highly nonlinear and many features are designed for specific attack models. Moreover, because the data in training and test sets is not identically distributed and the traditional detection features are not robust enough, the adaptation of detection method is challenging. The goal of domain adaptation is to generalize a classifier that is trained on a source domain to a target domain. SDAE-learned features have been demonstrated to be very robust and effective for cross-domain generalization [15, 24]. Therefore, we use SDAEs to automatically extract the detection features from multiple views of user behaviors.

Based on the different representations in SDAEs, we directly combine the automatic features from multiple views and then use PCA to eliminate the dependence of features. With the dimension reduction by a nonlinear way in SDAEs, PCA can further reduce the dimension by a linear way. Thus, we can acquire the main composition of representations and more effective detection features.

Let $Set_{md} = \{md_1, \ldots, md_a\}$ denote the set of attack models, $Set_{fs} = \{fs_1\%, \ldots, fs_b\%\}$ denote the set of attack sizes, and $Set_{as} = \{as_1\%, \ldots, as_c\%\}$ denote the set of filler sizes. Let *DRg* denote the rating records of genuine users and *DR* denote the rating records including shilling and genuine profiles. *SDAEopts* denote the parameters of the SDAEs, and *crate* denotes the corruption rate.

The function *getAttackProfiles(DR, md, fs%, as%)* is used to generate the shilling profiles according to attack parameters. The function *preprocessingData(DR)* is used to preprocess the rating records based on Algorithm 1. The function *SDAElearn(SDAE,R,crate)* is used to learn the representations by SDAEs under corruption rate *crate*. The function *FeaUnion(Fea_R,Fea_P,Fea_M)* is used to combine the features. The *function PCA(Fea_SDAE)* is used to extract the principal component of features. The algorithm for feature extraction is described as in Algorithm 2.

In Algorithm 2, firstly, the shilling profiles are generated according to attack parameters and injected into the rating records of genuine users (lines 1-8). Based on rating records, three matrixes are acquired by Algorithm 1. Then SDAEs are set up according to the parameters (line 10). Based on the different corruption rates, SDAEs learn the representations from multiple views (lines 11-13). Finally, the features are combined and dealt with by PCA method (lines 14-15).

In Algorithm 2, denoted the time complexity of generating shilling profiles (line 4) as $O(z)$; the time complexity of injecting shilling profiles (lines 1-8) under different attack parameters is at most $O(a \times b \times c \times z)$. The process for the learning of SDAEs can be completed within the polynomial, whose time complexity can be denoted as $O(y)$ [25]. Therefore, the time complexity of extraction features by

**Input**:$Set_{md}$, $Set_{fs}$, $Set_{as}$, *DRg, SDAEopts, crate*
**output**: **fea**
1 **for** each $md \in Set_{md}$ **do**
2 **for** each $fs\% \in Set_{fs}$ **do**
3 **for** each $as\% \in Set_{as}$ **do**
4 $DR_a$ = getAttackProfiles(*DRg*, *md*, *fs*%, *as*%)
5 $DR = DRg \cup DR_a$
6 **end for**
7 **end for**
8 **end for**
9 [*R, P, M*]=preprocessingData(*DR*)
10 *SDAE*=SDAESetup(*SDAEopts*)
11 *Fea_R*=SDAElearn(*SDAE,R,crate*)
12 *Fea_P*=SDAElearn(*SDAE,P,crate*)
13 *Fea_M*=SDAElearn(*SDAE,M,crate*)
14 Fea_SDAE=FeaUnion(Fea_R,Fea_P,Fea_M)
15 Fea=PCA(Fea_SDAE)
16 **return** *Fea, DR*

Algorithm 2: Feature extraction.

**Input**: *DRtrain, DRtest, SDAEopts, Lcrate, UcrateSet*$_{md}$, $Set_{fs}$, $Set_{as}$
**Output**: *Ylabels*
1 **for** *i=Lcrate to Ucrate* **do**
2 $Featrain_i$=*FeaExtra(Setmd, Setfs, Setas, DRtrain,i)*
3 *Featesti=FeaExtra(Setmd, Setfs, Setas, DRtest,i)*
4 $Ylabel^{(i)}$=*fSVM(Feataini,DRtrain,DRtest)*
5 **end for**
6 *Ylabels=voteLabel(Ylabel)*
7 **Return** *Ylabels*

Algorithm 3: Detection.

SDAEs from different views (lines 11-13) can be expressed as $O(3 \times y)$. The time complexity of PCA method (line 16) can be expressed as $O(N \times p^2)$, where $N$ is the number of samples and $p$ is the number of dimension of features [26]. Therefore, except the preprocessing of Algorithm 1, the time complexity of the Algorithm 2 can be described as $O(a \times b \times c \times z) + O(3 \times y) + O(N \times p^2)$.

*3.3. Generation of Weak Classifiers and Detection.* When attack size is small, the number of shilling profiles is far less than that of genuine ones. Under this imbalanced classification, to improve the performance of detection method, the ensemble detection method is proposed based on the DAE-learned features with different corruption rates.

The DAE is based on the concept that a good representation should contain enough information to reconstruct corrupted versions of the original input [15]. The noise level plays an important role to affect the final representation. In general, at low noise levels, the DAE is able to learn features for reconstructing finer details of the data. At high noise levels, the data is highly corrupted, which forces the DAE to learn more global, coarse-grained features of the data [27, 28]. Therefore, in SDAEs, we use different levels of noise to corrupt the user-item rating matrix, user-item popularity matrix, and adjacent matrix of user-user graph. And then the features can be extracted at different levels of scale or at different levels of granularity. With diverse features, we use weak classifiers to build a strong classifier to derive an optimal detection performance.

For the powerful classification algorithms C4.5, KNN, and SVM, the time complexity is $O(N \times fd \times lgfd)$, $O(N \times fd)$, and $O(N_{sv}^3 + N \times N_{sv}^2)$, respectively [29] (*N, fd, and* $N_{sv}$ denote the number of samples, the number of features, and the number of support vectors, respectively). With the small number of support vectors, SVM has lower time complexity. Faced with the high dimensional and complex features, we select SVM as weak classifier. Based on the results of weak classifier, the detection result is acquired by voting method.

Let *DRtrain* and *DRtest* denote the training and test sets, respectively. *SDAEopts* denote the parameters of the *SDAEs*. *Lcrate* and *Ucrate* denote the lower bound and upper bound of corruption rate, respectively. $Set_{md}$ denotes the set of attack models. $Set_{fs}$ and $Set_{as}$ denote the sets of filler sizes and attack sizes, respectively. The function *fSVM(Feataini,DRtrain,DRtest)* is used in SVM for classification. The function *voteLabel(Ylabel)* is used to generate the detection result based on the voting method. The algorithm for detection is described as follows in Algorithm 3.

In Algorithm 3, firstly, with the different corruption rates, features of profiles in training and test sets are extracted according to Algorithm 2 (lines 2-3). Secondly, based on the extracted features, the predictive result of every classifier is obtained (line 4), whose time complexity can be expressed as $O(t \times (N_{sv}^3 + N \times N_{sv}^2))$ ($t$, $N$, *and* $N_{sv}$ denote the number of corruption rates, the number of samples, and the number of support vectors, respectively). Finally, the final predictive result is generated by the voting method (lines 6), whose time complexity can be express as $O(t \times N)$.

## 4. Experimental Evaluation

*4.1. Experimental Data and Setting.* The following three datasets are used for the experiments:

(a) Netflix dataset (this dataset was published to support participants in the Netflix prize (http://netflixprize.com)): Netflix provides the contest dataset for Netflix prize, which includes 100,480,507 ratings given by 480,189 users on 17,770 movies. Each training rating is a quadruplet of the form <user, movie, date of grade, grade>. We randomly select 542,182 ratings on 4000 movies by 5000 users between January 6th, 2000, and December 31st, 2005, as our experimental dataset. We randomly divide 5000 genuine profiles into two groups. The first group including 3000 genuine profiles is used for the training dataset; the second group including 2000 genuine profiles is used for the test dataset.

(b) MovieLens-1M (https://grouplens.org/datasets/movielens/1m/): It contains 1,000,209 ratings on 3,952 movies by 6,040 users and each user has at least 20 ratings. We randomly divide 6040 genuine profiles into two groups. The first group including 4000 genuine profiles is used for the training dataset; the second group including 2040 genuine profiles is used for test dataset.

(c) Amazon dataset: It is crawled from Amazon.cn and dealt with by Xu etc. [30], which includes 1,205,125 reviews on 136,785 products by 645,072 reviewers. In this dataset, 5055 reviewers are labeled. We select these labeled reviewers as the samples data.

In the Netflix dataset and MovieLens dataset, the ratings are usually used to train the recommendation algorithm and we assume that they should not contain fake ratings. For the purpose of experiment in Netflix dataset and MovieLens dataset, the shilling profiles are generated by random, average, 30% AoP, shifting attack, power user, and power item models. In power user attack, we select the top 5% users according to the total number of ratings they provide in their user profiles as power users [10]. In power item attack, we select the top 5% items according to the total number of user ratings they have in their profiles as power items [11]. In the shilling profiles, the target item is randomly selected and suffered push attacks.

As to filler size, more than 83% and 74% genuine profiles are below or equal to 5% in the Netflix and MovieLens-1M datasets, respectively. To simulate most of the genuine users, the filler sizes of shilling profiles are set to {3%, 5%} in training and test sets. For attack size, to balance the proportion between genuine and shilling profiles in the training set, the shilling profiles are generated by random, average, 30% AoP, shifting, power item, and power user attacks with 8% attack size. To evaluate the performance under various attack sizes, in test sets, the attack sizes of shilling profiles are set to {3%, 5%, 10%, 12%}.

In SDAEs, the number of input nodes is equal to the number of items, and the number of nodes in hidden layer is set to {2000, 500, 200}, respectively. Following the suggestion in [27], the corruption rate is set to $\{0.7, 0.6, 0.5, 0.4, 0.3\}$. According to the cross validation, the sparse parameter$\rho$ = 0.05, and the sparse penalty $\beta$ = 0.9. When the features are combined from multiple views, the dimension of features is reduced to 300 by PCA. In our experiments, the average values of 10 times of experiments are used as the final evaluation values. Our experiments are implemented using Matlab R2015b and Python 2.7 on a PC with Intel i7-5500U 2.40GHz CPU, 8GB memory.

*4.2. Evaluation Metrics.* To evaluate the performance of the proposed method, we use precision and recall metrics. The precision and recall metrics are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

where $TP$ denotes the number of shilling profiles correctly classified, $FN$ denotes the number of shilling profiles misclassified as genuine profiles, and $FP$ denotes the number of genuine profiles misclassified as shilling profiles.

*4.3. Experiment on the MovieLens 1M Dataset.* To illustrate the performance of the proposed method (named as SDAEs-PCA), we compare it with the following baseline methods.

(1) SVM-13: AN SVM-based supervised method, which uses the 13 rating-based features in [1].

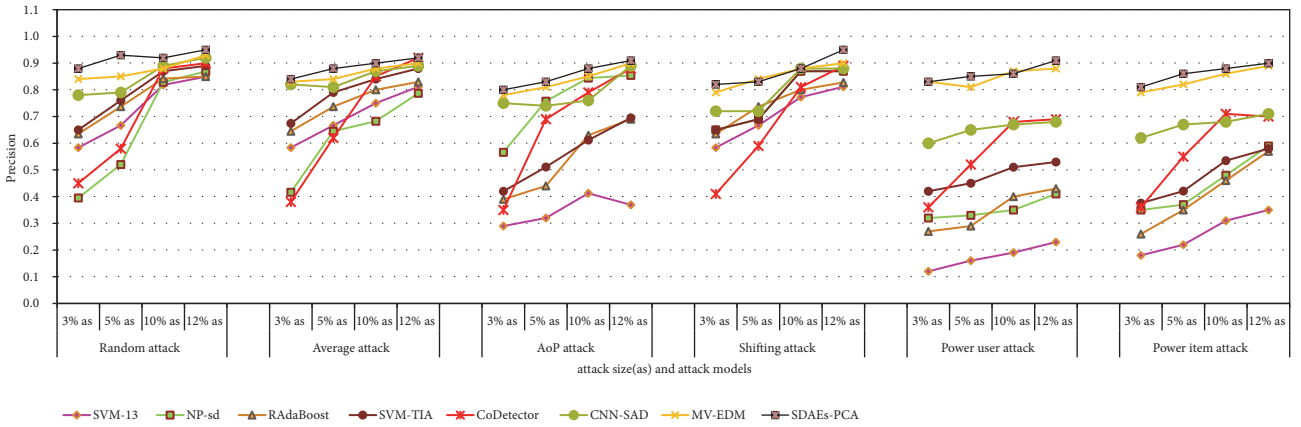
(2) NP-sd: A Neyman-Pearson statistical detection method [17], which we call NP-sd. This method uses supervised Neyman-Pearson detectors to detect attacks.

(3) RAdaBoost: An improved rescale AdaBoost method [5], which uses the 18 rating-based features. In RAdaBoost, 100 decision stumps are used as weak classifiers and the number of iteration times is set to 50. The shrinkage degree parameter is calculated by $s_k = 2/(k + u), u \in \mathbb{N}$ [5], where $u$ is set to 100 times attack size.

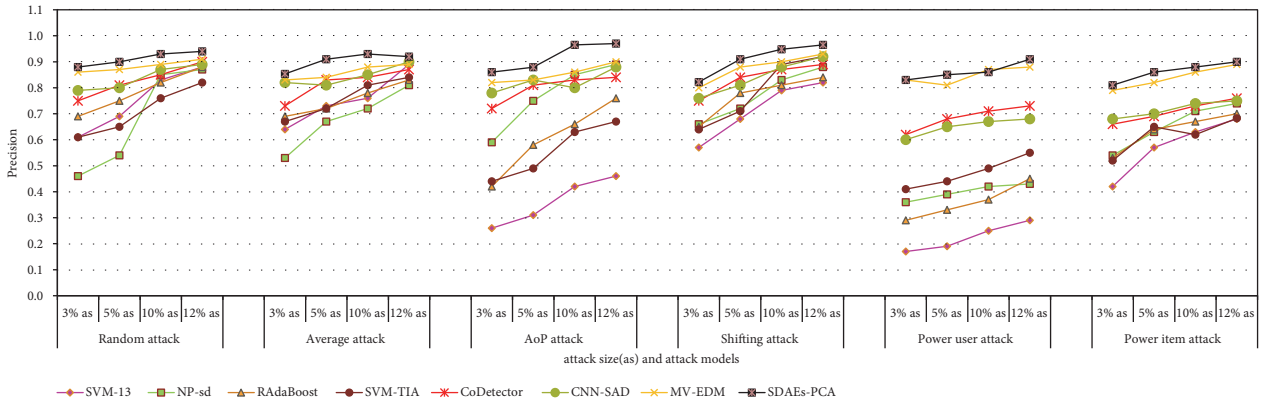
(4) SVM-TIA: A two-phase shilling attack detection method based on SVM and target item analysis [16]. The 7 rating-based features, namely, RDMA, DegSim, WDMA, WDA, LengthVar, MeanVar, DegSim$'$, are used by SVM for classification.

(5) CoDetector: A method based on decision tree and latent factors. The latent factors are obtained by collaboratively decomposing the user-item rating matrix and the user-user cooccurrence matrix [2]. In CoDetector, the dimensionality of latent factors is set to 10 and the negative samples count is set to 25.

(6) CNN-SAD: A method based on deep-level features, which are extracted from users rating profiles by convolutional neural network [22]. According to the cross validation,




(a) 3% filler size




(b) 5% filler size

Figure 2: Precision of eight methods with six types of attacks at various filler sizes across various attack sizes on the MovieLens dataset.

the vector is reshaped into rectangles with the short side length 20. The similarity of items is measured with the number of ratings. In convolutional layer, the number of feature maps is set to 6.

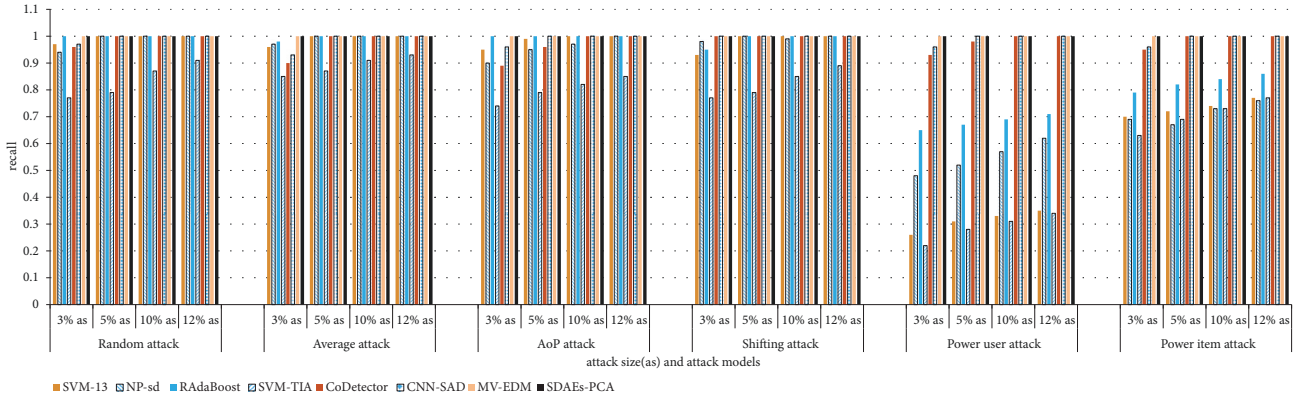
(7) MV-EDM: A method based on 17 artificially designed features, which take the temporal effects of item popularity and rating values into consideration [3]. Moreover, a multiview ensemble algorithm is used as a classifier method. In MV-EDM, we randomly separate 10% from training dataset as validation dataset.

In SVM-13 and SVM-TIA, Gaussian radial basis function is used as the kernel function, and rbf_sigma is set by 5-fold cross validation in the training set. In phase 2 of SVM-TIA, the threshold $\theta$, a prior knowledge for the number of attack profiles, is set to 50.

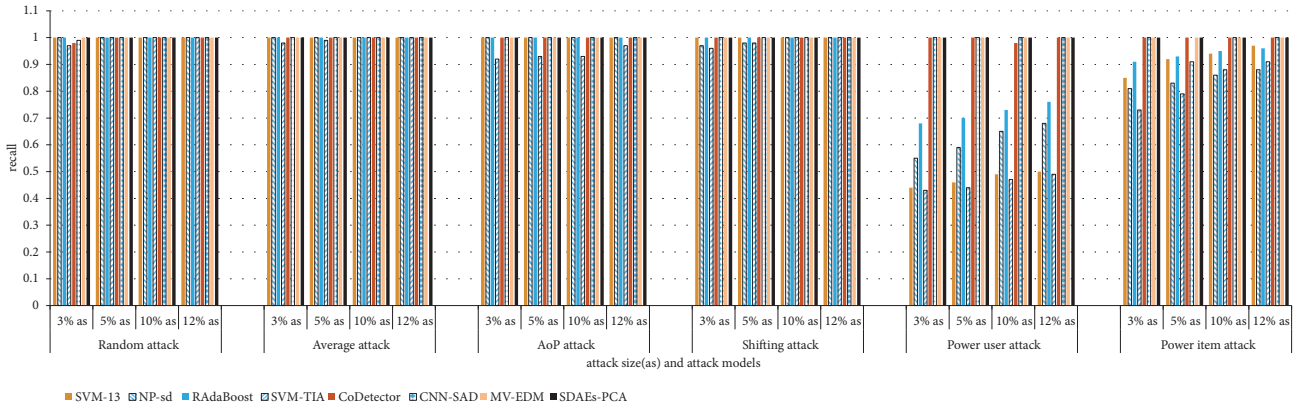
*4.3.1. Comparison of Precision.* Figure 2 shows the precision of eight methods under six types of attacks with different filler sizes and attack sizes on the MovieLens dataset.

As shown in Figure 2, the precision of SDAEs-PCA is significantly higher than that of SVM-13, NP-sd, RAdaBoost, SVM-TIA, CoDetector, CNN-SAD, and MV-EDM under six types of attacks. When detecting random, average, and shifting attacks, the precision of SVM-13, RAdaBoost, and SVM-TIA continues at high levels. However, when detecting AoP, power user, and power item attacks, the precision of SVM-13, RAdaBoost, and SVM-TIA has obvious decline. These results indicate that SVM-13, RAdaBoost, and SVM-TIA cannot effectively detect various attacks. This is because the features based on rating values are aimed at the specific types of attacks and have the poor adaptability under various attacks. Although the precision of NP-sd is high under AoP attacks, this superiority has disappeared under other attacks. This is because NP-sd based on the statistical method is easily affected by the distribution of data in different attacks. CoDetector and MV-EDM can effectively detect various attacks. Their adaptability may be attributed to the multi-perspective analysis for users' behaviors. However, since CoDetector relies on the ratings number when decomposing the matrix, it suffers poor precision under attacks with lower attack sizes and filler sizes. For CNN-SAD and SDAEs-PCA, with the features extracted by deep learning methods, their performance is better than most of traditional detection methods.

Compared with baseline methods, SDAEs-PCA not only takes advantages of multiple views of rating information, but also automatically extracts detection features from different scales. Therefore, SDAEs-PCA significantly outperforms other methods in terms of precision.




(a) 3% filler size




(b) 5% filler size

Figure 3: Recall of eight methods with six attack types at various filler sizes across various attack sizes on the MovieLens dataset.

*4.3.2. Comparison of Recall.* Figure 3 shows the recall of eight methods under six attacks with different filler sizes and attack sizes on the MovieLens dataset.

As shown in Figure 3, under random, average, AoP, and shifting attacks, the recall of SVM-TIA is the lowest. This may be because SVM-TIA only uses six detection features. Under power item and power user attacks, only CoDetector, CNN-SAD, MV-EDM, and SDAEs-PCA maintain high recall as they do under other attacks. This means these four methods can detect various shilling profiles as many as possible.

By taking the precision and recall of SDAEs-PCA into consideration, all the experiment results on MovieLens dataset indicate that SDAEs-PCA can effectively detect various shilling profiles and performs quite better than baseline methods.

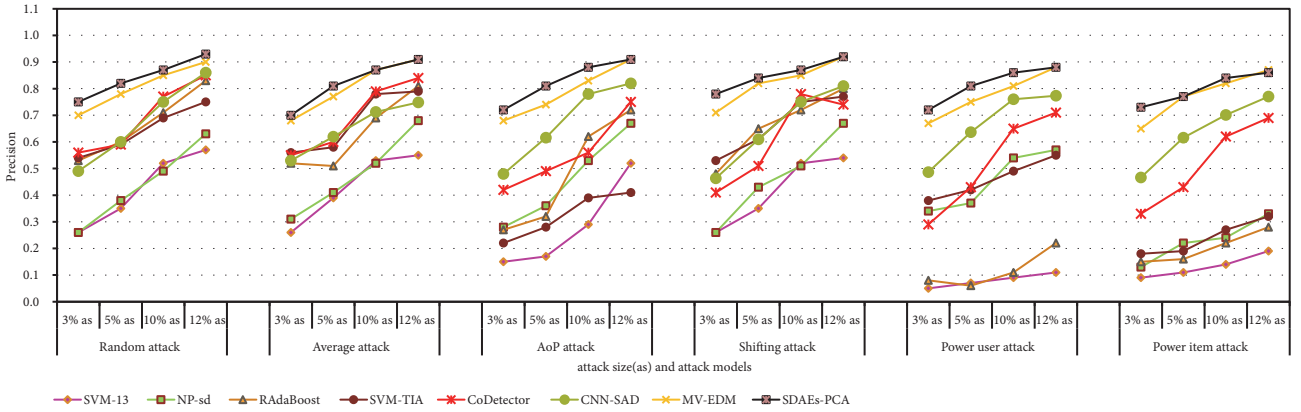
### *4.4. Experiment on the Netflix Dataset*

*4.4.1. Comparison of Precision.* Figure 4 shows the precision of eight methods under six attacks with different filler sizes and attack sizes on the Netflix dataset.

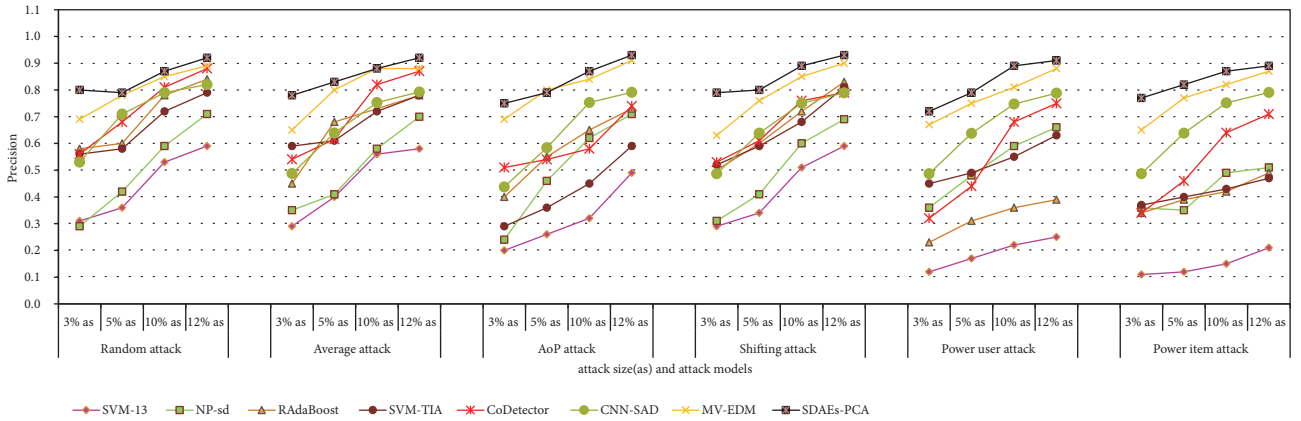
As shown in Figure 4, the precision of SDAEs-PCA is significantly higher than that of baseline methods, which again illustrates that SDAEs-PCA can detect various attacks accurately. Under random, average, and shifting attacks, RAdaBoost and SVM-TIA have high precision. For RAdaBoost, this is because it uses more detection features and effective ensemble methods. For SVM-TIA, this is attributed to the phases of target item analysis. However, since RAdaBoost and SVM-TIA only use the single view of rating values, the classifiers cannot be strong enough for comprehensive identification of shilling profiles and they are inferior to SDAEs-PCA. For SVM-13, although it uses more features than SVM-TIA, without the phases of target item analysis, SVM-13 suffers poor precision. Under power user attack and power item attack, the precision of SVM-13, RAdaBoost, and SVM-TIA has a different level of decrease. As to NP-sd, although the precision is slightly higher than SVM-13, it is always lower than SDAEs-PCA. Under six types of attacks, CoDetector, CNN-SAD, MV-EDM, and SDAEs-PCA keep their own consistency. In these four methods, the precision of MV-EDM is only next to that of SDAEs-PCA. For CoDetector and CNN-SAD, their precision is relatively low under attacks with small attack sizes. This can in part be attributed to the fact that they use single classifier to deal with the unbalance problem in small-scale attacks.

Therefore, SDAEs-PCA has relatively higher precision than baseline methods do.

*4.4.2. Comparison of Recall.* Figure 5 shows the recall of eight methods under six attacks with different filler sizes and attack sizes on the Netflix dataset.




(a) 3% filler size




(b) 5% filler size

Figure 4: Precision of eight methods with six types of attacks at various filler sizes across various attack sizes on the Netflix dataset.

As shown in Figure 5, the recall of SDAEs-PCA is higher than or equal to that of baseline methods under various attacks. SVM-13, RAdaBoost, and SVM-TIA have relatively high recall under random, average, and shifting attacks. However, under power user and power item attacks, the recall of them cannot always continue at high levels. Although the recall of NP-sd increases with the growth of the attack size, it is always below that of SDAEs-PCA. This may be because the statistics-based method (NP-sd) is easily affected by the number of the attack samples. The recall of CoDetector, CNN-SAD, MV-EDM, and SDAEs-PCA is almost 1 under various attacks.

These experimental results again illustrate the better performance of our proposed methods on Netflix dataset.

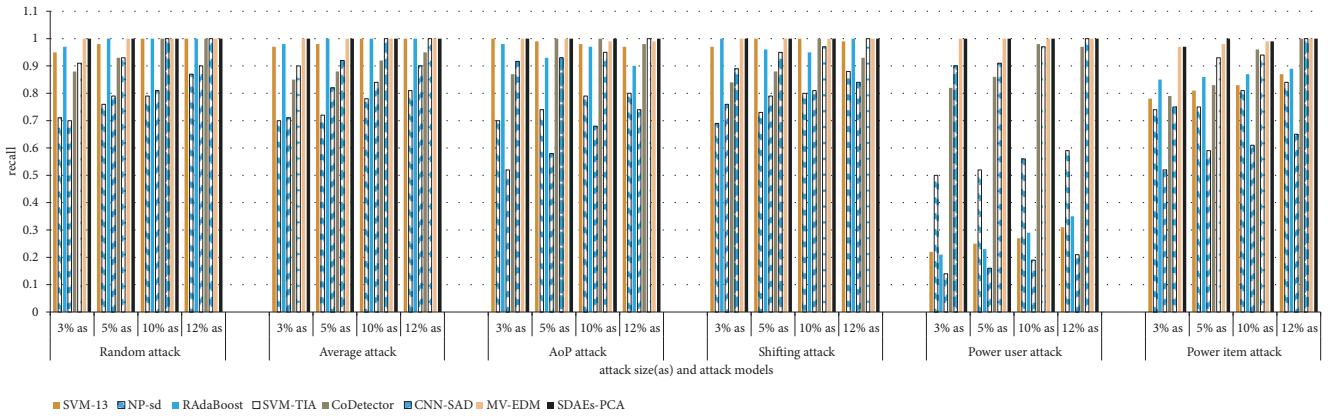
*4.5. Experiment on the Amazon Dataset.* To further evaluate the performance of the proposed method, we conduct experiments on Amazon review dataset, in which 3055 and 2000 reviewers are randomly selected as training set and test set, respectively. Other parameters in the eight detection methods are the same as in Section 4.3.

Since the eight detection methods show the inconsistent trends in terms of recall and precision on Amazon dataset, we use F1-measure metric to evaluate the overall performance, which can be calculated as follows.

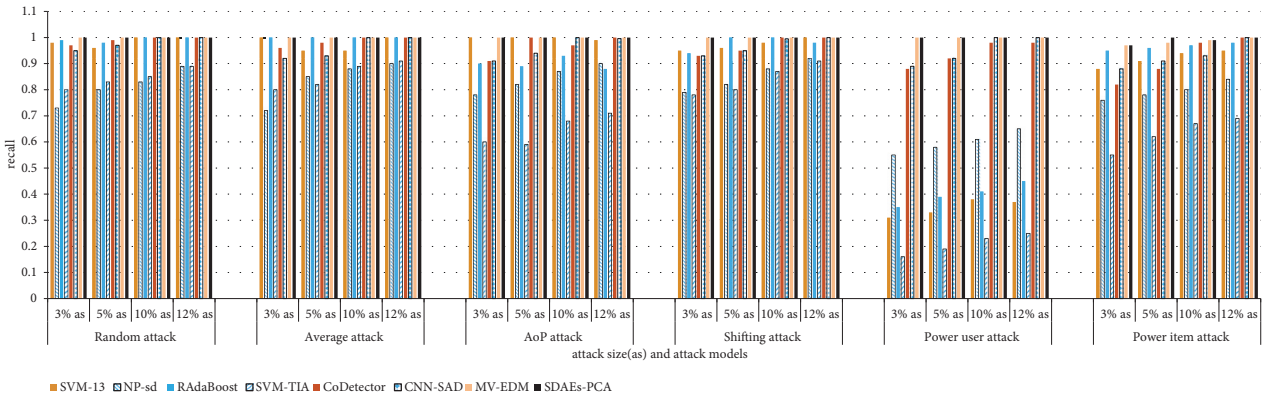
$$\text{F1-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{11}$$

Figure 6 shows the recall, precision, and F1-measure of eight methods described in previous sections.

As shown in Figure 6, for overall performance, MV-EDM and SDAEs-PCA have higher F1-measure. In the artificial features-based methods (SVM-13, RAdaBoost, SVM-TIA, and MV-EDM), MV-EDM has the highest F1-measure. This may be because MV-EDM uses more types of information including time, rating values, and item popularity. In practice, the rating time may provide efficient information for separating attackers. As a statistical detection method, NP-sd may discover more attackers leading high recall. However, the sparse ratings may influence its precision. In the automatic features-based methods (CoDetector, CNN-SAD, and SDAEs-PCA), SDAEs-PCA has the highest F1-measure. Since CNN-SAD only uses the rating value information, its overall performance is not as high as those of other automatic features-based methods. Although CoDetector utilizes the ratings and cooccurrence matrixes, it is likely to be affected




(a) 3% filler size




(b) 5% filler size

Figure 5: Recall of eight methods with six attack types at various filler sizes across various attack sizes on the Netflix dataset.

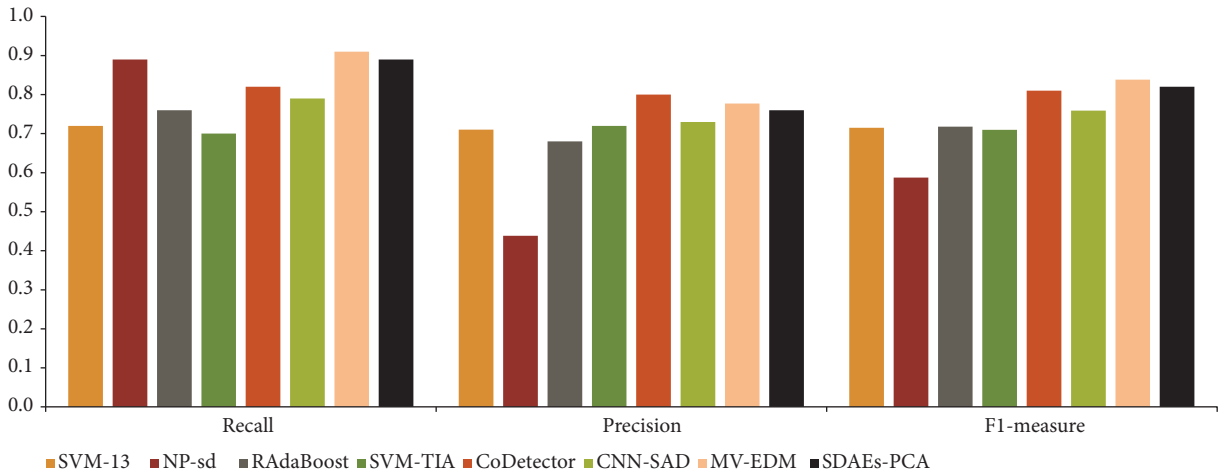



Figure 6: Recall, precision, and F1-measure of eight detection methods on the Amazon dataset.

by the noise in the practical rating behaviors. Therefore, with more rating information, SDAEs-PCA outperforms other automatic features-based methods in the metric of F1-measure. However, without the time information, the overall performance of SDAEs-PCA is slightly weaker than that of MV-EDM.

In consideration of the performance in MovieLens and Netflix datasets and knowledge cost, SDAEs-PCA is a good detection framework to integrate more information and reduce the artificial feature work.

## 5. Conclusion and Future Work

Since the strategies for generation of shilling profiles are varying and evolving, the adaptation of detection is a challenging problem. In this paper, we propose a detection method based

on the multiple views information, which offers us valuable and comprehensive insights into the users' rating behaviors. Moreover, we use SDAEs and PCA method to automatically extract effective detection features. With the weak classifiers at the different scales of features, we integrate the detection results based on voting method. The experiment results on MovieLens, Netflix, and Amazon datasets illustrate the effectiveness of the proposed method.

Our proposed method can be seen as a generalized detection framework and can be deployed in recommender systems for online mall, social network, and so on. In the future work, we will conduct experiments on other areas datasets and capture the information from more views. Also, we will further optimize the SDAEs to improve the efficiency.

## Data Availability

The experimental data in Netflix and MovieLens-1M datasets are available at https://github.com/haoyaojun1/detector_CNN-PCA. Other detailed data are currently under embargo while the research findings are commercialized. Requests for other data, 12 months after publication of this article, will be considered by the corresponding author.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Acknowledgments

This research was funded by the National Natural Science Foundation of China (Nos. 61772452, 61379116), the Technology Innovation Program of Higher Education Institutions of Shanxi Province (No. 201804008), the Innovation Project for Graduate of Qinhuangdao (Nos. 2018110, 2018032), the Open Project of Intelligent Information Processing Key Laboratory of Shanxi Province (No. 2016002), and the Natural Science Foundation of Shanxi Province (No. 201701D21059).

## References

[1] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th ACM International Workshop on Web Information and Data Management (WIDM '05)*, November 2005.

[2] T. Dou, J. Yu, Q. Xiong, M. Gao, Y. Song, and Q. Fang, "Collaborative shilling detection bridging factorization and user embedding," in *Proceedings of the 13th EAI International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Guangzhou, China, 2017.

[3] Y. Hao, P. Zhang, and F. Zhang, "Multiview ensemble method for detecting shilling attacks in collaborative recommender systems," *Security and Communication Networks*, vol. 2018, pp. 1–33, 2018.

[4] H. Shen, F. Ma, X. Zhang, L. Zong, X. Liu, and W. Liang, "Discovering social spammers from multiple views," *Neurocomputing*, vol. 225, pp. 49–57, 2017.

[5] Z. Yang, L. Xu, Z. Cai, and Z. Xu, "Re-scale AdaBoost for attack detection in collaborative filtering recommender systems," *Knowledge-Based Systems*, vol. 100, pp. 74–88, 2016.

[6] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.

[7] C. A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: detection of profile injection attacks," *Service Oriented Computing and Applications*, vol. 1, pp. 157–170, 2007.

[8] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pp. 542–547, 2006.

[9] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *User Modeling and User-Adapted Interaction*, vol. 19, no. 1-2, pp. 65–97, 2009.

[10] D. C. Wilson and C. E. Seminario, "When power users attack: Assessing impacts in collaborative recommender systems," in *Proceedings of the 7th ACM Conference on Recommender Systems, RecSys 2013*, China, 2013.

[11] C. E. Seminario and D. C. Wilson, "Attacking item-based recommender systems with power items," in *Proceedings of the 8th ACM Conference on Recommender Systems, (RecSys 2014)*, ACM, 2014.

[12] Z. Yang and Z. Cai, "Detecting abnormal profiles in collaborative filtering recommender systems," *Journal of Intelligent Information Systems*, vol. 48, no. 3, pp. 499–518, 2017.

[13] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[16] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems," *Neurocomputing*, vol. 210, pp. 197–205, 2016.

[17] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys '09)*, New York, NY, USA, 2009.

[18] H. Xia, B. Fang, M. Gao, H. Ma, Y. Tang, and J. Wen, "A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique," *Information Sciences*, vol. 306, pp. 150–165, 2015.

[19] F. Zhang and Q. Zhou, "HHT-SVM: An online method for detecting profile injection attacks in collaborative recommender systems," *Knowledge-Based Systems*, vol. 65, pp. 96–105, 2014.

[20] W. Li, M. Gao, H. Li, J. Zeng, Q. Xiong, and S. Hirokawa, "Shilling attack detection in recommender systems via selecting patterns analysis," *IEICE Transaction on Information and Systems*, vol. 99, pp. 2600–2611, 2016.

[21] Z. Yang, Z. Cai, and X. Guan, "Estimating user behavior toward detecting anomalous ratings in rating systems," *Knowledge-Based Systems*, vol. 111, pp. 144–158, 2016.

[22] C. Tong, X. Yin, and J. Li, "A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network," *The Computer Journal*, vol. 61, pp. 949–958, 2018.

[23] P. Castells, S. Vargas, and J. Wang, *Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance*, 2011.

[24] P. Vincent, H. Larochelle, I. Lajoie, and P. A. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[25] S. Arora, A. Bhaskara, R. Ge, and T. Ma, *Provable Bounds for Learning Some Deep Representations*, 2013.

[26] N. Shahid, N. Perraudin, V. Kalofolias, and P. Vandergheynst, "Fast robust pca on graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, pp. 740–756, 2016.

[27] K. J. Geras and C. Sutton, "Scheduled denoising autoencoders," *Computer Science*, 2014.

[28] M. Chen, Z. Xu, K. Weinberger, and S. Fei, "Marginalized denoising autoencoders for domain adaptation," *Computer Science*, 2012.

[29] G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, "BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification," *Engineering Applications of Artificial Intelligence*, vol. 49, pp. 176–193, 2016.

[30] C. Xu, J. Zhang, K. Chang, and C. Long, "Uncovering Collusive Spammers in Chinese Review Websites," 2013.