Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

Semi-supervised recommendation attack detection based on Co-Forest

Quanqiang Zhou[*], *Liangliang Duan*

School of Information and Control Engineering, Qingdao University of Technology, Qingdao, Shandong Province, China

ARTICLE INFO

Article history:
Received 21 December 2020
Revised 3 May 2021
Accepted 24 June 2021
Available online 1 July 2021

Keywords:
Collaborative recommender system
Recommendation attack
Attack detection
Semi-supervised learning
Co-Forest algorithm

ABSTRACT

In recommendation attack, malicious users attempt to bias the recommendation results by injecting fake profiles into the rating database. To detect such attack, three types of methods, i.e., unsupervised, supervised and semi-supervised, have been proposed. Among these works, the advantage of semi-supervised methods is that they can use the unlabeled user profiles to improve the detection performance. However, the existing semi-supervised methods suffer from low precision. Aiming at this problem, in this paper, we propose a semi-supervised detection approach named SSADR-CoF based on the Co-Forest algorithm. Being different from the existing semi-supervised methods which only use a few of features to train a single classifier for the detection, the proposed approach uses a series of features to train an ensemble of classifiers to detect the recommendation attack. We first use the window dividing and rating behavior statistical methods to extract a series of user rating behavior mode features for training the detection model. Then, we use a small number of labeled user profiles to initialize an ensemble of classifiers, and use the ensemble of classifiers to assign labels to the unlabeled user profiles. Finally, we use the labeled and the newly labeled user profiles to iteratively update the classifiers for the detection. Experiments conducted on three benchmark datasets MovieLens 10M, MovieLens 25M, and Amazon show that the proposed approach can effectively improve the precision of the semi-supervised methods under the condition of maintaining high recall and AUC.

1. Introduction

In the era of big data, the problem of information overload is becoming more and more serious. Collaborative recommender systems are effective ways to solve this problem since they can actively recommend the information to users to satisfy the users interest according to the established user profiles (Meng et al., 2020). They have been successfully applied in many practical applications. For example, many e-commerce sites[1],[2] use collaborative recommender systems to recommend products for users from mass items to enhance sales.

Openness is one of the main characteristics of collaborative recommender systems. Due to this fact, collaborative recommender systems are under recommendation attack (Lam and Riedl, 2004; Zhang et al., 2006) in which attackers inject a certain number of fake profiles into a collaborative recommender system to influence or change the recommendation results. Fake profiles are also known as attack profiles.

[*] Corresponding author.
E-mail address: zhouqiang128@126.com (Q. Zhou).

[1] https://www.amazon.com/.
[2] https://www.netflix.com/.

https://doi.org/10.1016/j.cose.2021.102390
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

From the perspective of purposes, recommendation attack can be divided into push attack and nuke attack. The purposes of push and nuke attack are to promote or demote the target item to be recommended, respectively (Mahony et al., 2004). In the early research stage, researchers usually employ attack models (Mahony et al., 2004), which are build based on reverse engineering, to generate attack profiles. Common attack models include random attack, average attack, bandwagon attack, and so on (Burke et al., 2005; Lam and Riedl, 2004; Si and Li, 2020). Besides the target item, some other items will also be rated to make the generated attack profile look like the genuine profile. The strength of recommendation attack is usually measured by attack size and filler size (Mahony et al., 2004). In recent years, the real attack profiles are labeled from the Amazon Review dataset (Xu et al., 2013). This work enriches the user rating behavior modes of recommendation attack, effectively. Recommendation attack seriously reduces the recommendation quality of collaborative recommender systems.

To detect the recommendation attack, three types of methods, i.e., unsupervised (Chirita et al., 2005; Chung et al., 2013; Lee and Zhu, 2012; Mehta et al., 2007; Mehta and Nejdl, 2009; Yang et al., 2018; Zhang et al., 2018), supervised (Burke et al., 2006; Ebrahimian and Kashef, 2020; Tong et al., 2018; Vivekanandan and Praveena, 2020; Williams et al., 2007; Xu and Zhang, 2019; Yang et al., 2016; Zhou et al., 2020; 2016), and semi-supervised (Wu et al., 2011; 2012), have been proposed. Despite the effectiveness of these methods, they still face some shortcomings as follows.

- The benefit of unsupervised methods is that they do not require labeled user profiles for training the detection model. However, most of them require certain prior knowledge, such as attack size, which is usually different to get in real application.
- Supervised methods, especially the ones which are built based on deep learning, usually have good detection performance. However, most of them require certain number of labeled user profiles for training classifiers as the detection model.
- The advantage of semi-supervised methods is that they can use the unlabeled user profiles, whose number is usually large in a rating database, to improve the classifier which is initialized by some labeled user profiles. However, the study on semi-supervised methods is seldom, and even worse, the precision of this type of methods is low.

In this paper, to further explore the semi-supervised methods and improve their detection performance we propose a detection approach SSADR-CoF based on the Co-Forest algorithm (Li and Zhou, 2007). In the proposed approach, a series of features are used to train an ensemble of classifiers to detect the recommendation attack. This is different from the existing semi-supervised methods in which only a few of features are used to train a single classifier for the detection. The proposed approach is devoted to employing more features and more classifiers to improve the detection performance. The main contributions of this paper are summarized as follows.

- We proposed a feature extraction method based on window dividing and rating behavior statistical methods to extract a series of features which can characterize the user rating behavior mode for training the detection model.
- We proposed a semi-supervised detection algorithm based on Co-Forest. In the proposed algorithm, we employed ensemble learning, combining with the extracted features, to improve the precision of the semi-supervised detection methods.
- We conducted comparative experiments on benchmark datasets to verify the effectiveness of the proposed approach.

The rest of this paper is organized as follows. Section 2 introduces the background and reviews the related work. Section 3 describes the details of the proposed approach. Section 4 presents the experiments and discussions. Conclusions are given in Section 5.

2. Background and related work

2.1. *Recommendation attack models*

Attack models (Mahony et al., 2004) are defined as strategies to construct attack profiles based on the knowledge of the recommender system. In an attack model, items are usually divided into four sets which are $I_S, I_F, I_\emptyset$, and I_t.

Items in I_S are selected for special treatment during the attack. For some attack models, I_S is an empty set. A certain number of items are randomly selected as filler items to generate I_F. Filling items can reduce the sparsity of attack profile, effectively. $I_\emptyset$ denotes the set of unrated items. Similar to most of the genuine profiles, most items of the recommender system will be in this set. I_t denotes the set of target items. These items are usually given the maximum or minimum rating values for push or nuke attack, respectively.

Random, average, and bandwagon attack (Burke et al., 2005; Lam and Riedl, 2004; Si and Li, 2020) are three common attack models. Let μ and σ denote the mean and standard deviation of ratings for all items. Let μ_i and σ_i denote the mean and standard deviation of ratings for item i. Let $r(i)$ denote a rating value drawn from a normal distribution $N(\mu, \sigma)$. Let r_{max} and r_{min} denote the maximum and minimum rating values in the recommender system, respectively. Let I denote the set of all items in the recommender system. The strategies for constructing attack profiles in these three attack models are shown in Table 1.

Attack size and filler size (Mahony et al., 2004) are used to measure the strength of recommendation attack. Attack size is defined as the ratio of $|U_a|$ to $|U_g|$, where $|U_a|$ and $|U_g|$ denote the number of attack profiles and the number of genuine profiles in a rating set, respectively. Filler size is defined as the ratio of $|I_u|$ to $|I|$, where $|I_u|$ denotes the number of items rated by user u and $|I|$ denotes the number of entire items in the recommender system.

2.2. *Related work*

In unsupervised methods, (Chirita et al., 2005) first proposed several metrics through analyzing rating patterns of attack

Table 1 – Recommendation attack models.

Attack model	Strategy for constructing attack profiles
Random attack	$I_S = \emptyset$. Items in I_F are randomly selected from $I - I_t$. For each i in I_F, $r(i) \sim N(\mu, \sigma)$. For the target item i in I_t, $r(i) = r_{max}$ or r_{min}.
Average attack	$I_S = \emptyset$. Items in I_F are randomly selected from $I - I_t$. For each i in I_F, $r(i) \sim N(\mu_i, \sigma_i)$. For the target item i in I_t, $r(i) = r_{max}$ or r_{min}.
Bandwagon attack	Items in I_S are the k most popular items. For each i in I_S, $r(i) = r_{max}$. Items in I_F are randomly selected from $I - I_S - I_t$. For each i in I_F, $r(i) \sim N(\mu, \sigma)$. For the target item i in I_t, $r(i) = r_{max}$ or r_{min}.

profiles. This method is successful when detecting high-density attack profiles. (Mehta et al., 2007; Mehta and Nejdl, 2009) proposed a PCA-based detection algorithm in which attack profiles are processed and identified as outliers. Several types of attack profiles can be detected by their algorithm. However, the algorithm requires the total number of attack profiles as a priori knowledge which is difficult to get in real applications. (Chung et al., 2013) used Beta distribution to filter out the recommendation attack. Their method can effectively detect certain types of attack profiles. However, when detecting attack profiles with large attack sizes, this method has poor performance. The clustering-based methods (Lee and Zhu, 2012; Yang et al., 2018; Zhang et al., 2018) try to detect recommendation attack by clustering genuine profiles and attack profiles into different clusters. However, this type of methods may have low detection performance when there is only one type of user profiles in the test set.

In supervised methods, (Burke et al., 2006; Williams et al., 2007) trained three supervised algorithms kNN, C4.5, and SVM to generate classifiers for the detection. Although the recall of this method is high, this method suffers from low precision. (Yang et al., 2016) proposed a detection method by using re-scale Boosting and AdaBoost to improves the detection performance for imbalanced test sets. (Zhou et al., 2016) proposed a SVM-based method for the detection. In this method, Borderline-SMOTE is used to solve the problem of imbalance between genuine profiles and attack profiles when training the classifier. The target item analysis is used to further improve the classification accuracy when detecting the test set. However, in the case that the target item is misidentified, this method will have low detection performance. (Xu and Zhang, 2019) extracted trust features of user profiles and trained a SVM-based classifier for the detection. This method improves the detection performance of the single classifier. However, it is based on the correct identification of the target item which might be a challenging task to determine from a large number of items in the rating database. The deep learning-based methods were proposed to detect the attack profiles (Ebrahimian and Kashef, 2020; Tong et al., 2018; Vivekanandan and Praveena, 2020; Zhou et al., 2020). Although, these methods have good detection effectiveness, many labeled user profiles are required in them.

(Wu et al., 2011; 2012) first introduced the semi-supervised method for the detection. In their method, a Naïve Bayes classifier is first generated. Then, the unlabeled user profiles are used to improve the classifier. This method has high recall for some types of recommendation attack. However, it suffers from low precision as shown in our experiments.

2.3. *Co-Forest algorithm*

Co-Forest (Li and Zhou, 2007) is a semi-supervised learning algorithm based on ensemble learning. It uses the Random Forest algorithm (Leo, 2001) to extend the co-training paradigm. It can easily produce the final better hypothesis with the power of ensemble in semi-supervised learning. Co-Forest can be easily used in many real-world applications, since it requires neither that the data be described by sufficient and redundant attribute subsets nor special learning algorithms which frequently employ time-consuming cross validation in learning.

Let L and U denote the labeled set and unlabeled set, respectively, which are independently drawn from the same data distribution. Co-Forest uses an ensemble of N classifiers, which is denoted by H^*, to estimate the confidence of the given unlabeled example. When determining the most confidently labeled examples for a component classifier of the ensemble h_i (i=1,...,N), all other component classifiers in H^* except for h_i are used. These component classifiers form a new ensemble, which is called the concomitant ensemble of h_i, denoted by H_i. H_i differs from H^* only by the absence of h_i. Now, the confidence for an unlabeled example can be simply estimated by the degree of agreements on the labeling, i.e., the number of classifiers that agree on the label assigned by H_i. By using this method, Co-Forest firstly trains an ensemble of classifiers on L and then refines each component classifier with unlabeled examples selected by its concomitant ensemble.

The Co-Forest algorithm is built based on the following deductions.

According to (Angluin and Laird, 1988), if the size of training data m, the noise rate η, and the hypothesis worst case error rate ϵ satisfy the following relationship:

$$m = \frac{c}{\epsilon^2 (1 - 2\eta)^2}, \tag{1}$$

where, c is a constant, then the learned hypothesis h_i that minimizes the disagreement on a sequence of noisy training examples converges to the true hypothesis h^* with probability equal to 1.

By reforming Eq. (1), the following utility function is obtained:

$$f = \frac{c}{\epsilon^2} = m(1 - 2\eta)^2. \tag{2}$$

In the tth learning iteration, a component classifier h_i (i=1,..., N) is supposed to refine itself on the union of original labeled set L with the size of m_0 and the newly labeled set $L'_{i,t}$, with the size of $m_{i,t}$, where $L'_{i,t}$ is determined and labeled by its concomitant ensemble H_i. Let the error rate of H_i on L be $\hat{e}_{i,t}$, and then the weighted number of examples being mislabeled by H_i in $L'_{i,t}$ is $\hat{e}_{i,t}W_{i,t}$, where $W_{i,t} = \sum_{j=0}^{m_{i,t}} w_{i,t,j}$, and $w_{i,t,j}$ is the predictive confidence of H_i on x_j in $L'_{i,t}$. To uniform the expressions, m_0 is rewritten as the weighted form W_0, where $W_0 = \sum_{j=0}^{m_0} 1$. In the augmented training set $L \cup L'_{i,t}$, the noisy examples consist of the noisy examples in L and the examples in $L'_{i,t}$ that are misclassified by the concomitant ensemble H_i. Thus, the noise rate in $L \cup L'_{i,t}$ is estimated by

$$\eta_{i,t} = \frac{\eta_0 W_0 + \hat{e}_{i,t}W_{i,t}}{W_0 + W_{i,t}}. \tag{3}$$

By replacing η and m in Eq. (2) with Eq. (3) and the weighted size of the augmented training set (W_0+$W_{i,t}$), respectively, the utility of h_i in the tth iteration takes the form of

$$f_{i,t} = (W_0 + W_{i,t})\left(1 - 2\frac{\eta_0 W_0 + \hat{e}_{i,t}W_{i,t}}{W_0 + W_{i,t}}\right)^2. \tag{4}$$

Similarly, the utility of h_i in the (t-1)th iteration is

$$f_{i,t-1} = (W_0 + W_{i,t-1})\left(1 - 2\frac{\eta_0 W_0 + \hat{e}_{i,t-1}W_{i,t-1}}{W_0 + W_{i,t-1}}\right)^2. \tag{5}$$

According to Eq. (2), the utility f is inversely proportional to the squared worse case error rate ϵ^2. Thus, to reduce the worst case error rate of each classifier h_i, the utility of h_i should be increased in the learning iterations, i.e., $f_{i,t} > f_{i,t-1}$. Now, assume that little noise exists in L, and each component classifier h_i meets the requirement of a weak classifier, i.e., $\hat{e}_{i,t} < 0.5$. By comparing the right-hand side of Eq. (4) and Eq. (5), $f_{i,t} > f_{i,t-1}$ holds when $W_{i,t} > W_{i,t-1}$ and $\hat{e}_{i,t}W_{i,t} < \hat{e}_{i,t-1}W_{i,t-1}$, which are further summarized by

$$\frac{\hat{e}_{i,t}}{\hat{e}_{i,t-1}} < \frac{W_{i,t-1}}{W_{i,t}} < 1. \tag{6}$$

According to Eq. (6), $\hat{e}_{i,t} < \hat{e}_{i,t-1}$ and $W_{i,t} > W_{i,t-1}$ should be satisfied at the same time. However, even if this requirement is met, $\hat{e}_{i,t}W_{i,t} < \hat{e}_{i,t-1}W_{i,t-1}$ might still be violated since $W_{i,t}$ might be much larger than $W_{i,t-1}$. To make Eq. (6) hold again in this case, $L'_{i,t}$ must be subsampled so that $W_{i,t}$ is less than $\hat{e}_{i,t-1}W_{i,t-1}/\hat{e}_{i,t}$.

In Co-Forest algorithm, the majority voting method, which is widely used in ensemble learning, is employed to produce the final classification results.

3. Proposed detection approach

In this section, we first introduce the proposed feature extraction method. Then, we describe the proposed detection algorithm.

3.1. *Feature extraction method*

In the rating database of a collaborative recommender system, ratings are produced from the user rating behaviors and reflect the interests and preferences of users. Different users, especially real users and attackers, should have different interests and preferences. Therefore, they may have different rating behaviors, which will result in different user rating behavior modes as shown in Fig. 1.

Based on above analysis, we first divide the items in the rating database into different windows in which we use the rating behavior statistical methods to characterize the user rating behavior modes in the proposed feature extraction method.

Let I denote the set of items in the rating database of a collaborative recommender system. Let J denote the number of windows for dividing. We divide the items in I into J windows, averagely. The order of items in the window is the same as they appear in the rating database.

Then, we can get J windows $\{\omega_1, \omega_2, ..., \omega_j, ..., \omega_J\}$. The number of items in the jth window ω_j can be defined as follows.

$$|\omega_j| = \begin{cases} Q+1, & j \le q, \\ Q, & j > q, \end{cases} \tag{7}$$

where, Q denotes the quotient and q denote the remainder when $|I|$ is divided by J. When q is equal to 0, the number of items in each window is Q. When q is greater than 0, the first q windows have one more item than the latter. To set optimal parameter J for different rating database, J is used as a hyper parameter which will be determined in the experiments.

Let S denote the set of users in the rating database. Let $r_{u,i}$ denote the rating rated by user u for item i where $u \in S$ and $i \in I$. Two rating behavior statistical methods are used to characterize the user rating behavior modes in each window ω as follows.

• Number of Ratings in each Window (*NRW*). *NRW* captures the rating behavior feature of user u in the jth window ω_j. It can be defined as follows.

$$NRW_{u,\omega_j} = \sum_{i\in\omega_j, r_{u,i}\neq 0} 1. \tag{8}$$

• Weighted Number of Ratings in each Window (*WNRW*). *WNRW* captures the ratio feature of the rating behavior in one window to the total rating behavior in the rating database of user u. It can be defined as follows.

$$WNRW_{u,\omega_j} = \frac{\sum_{i\in\omega_j, r_{u,i}\neq 0} 1}{\sum_{i\in I, r_{u,i}\neq 0} 1}. \tag{9}$$

Therefore, the feature vector of user u which consists of a series of features can be defined as follows

$$x_u = (NRW_{u,\omega_1}, WNRW_{u,\omega_1}, \ldots, NRW_{u,\omega_j}, WNRW_{u,\omega_j}, ..., NRW_{u,\omega_J}, WNRW_{u,\omega_J}). \tag{10}$$

According to above analysis, we propose a feature extraction algorithm based on dividing windows and rating behavior statistical methods as shown in Algorithm 1.

Figures 2 and 3 show the feature vectors generated by performing Algorithm 1 on the user profiles described in Fig. 1. As displayed in Figs. 2 and 3, we separate the feature vectors into two figures to show the distribution of the features *NRW* and *WNRW* more clearly.

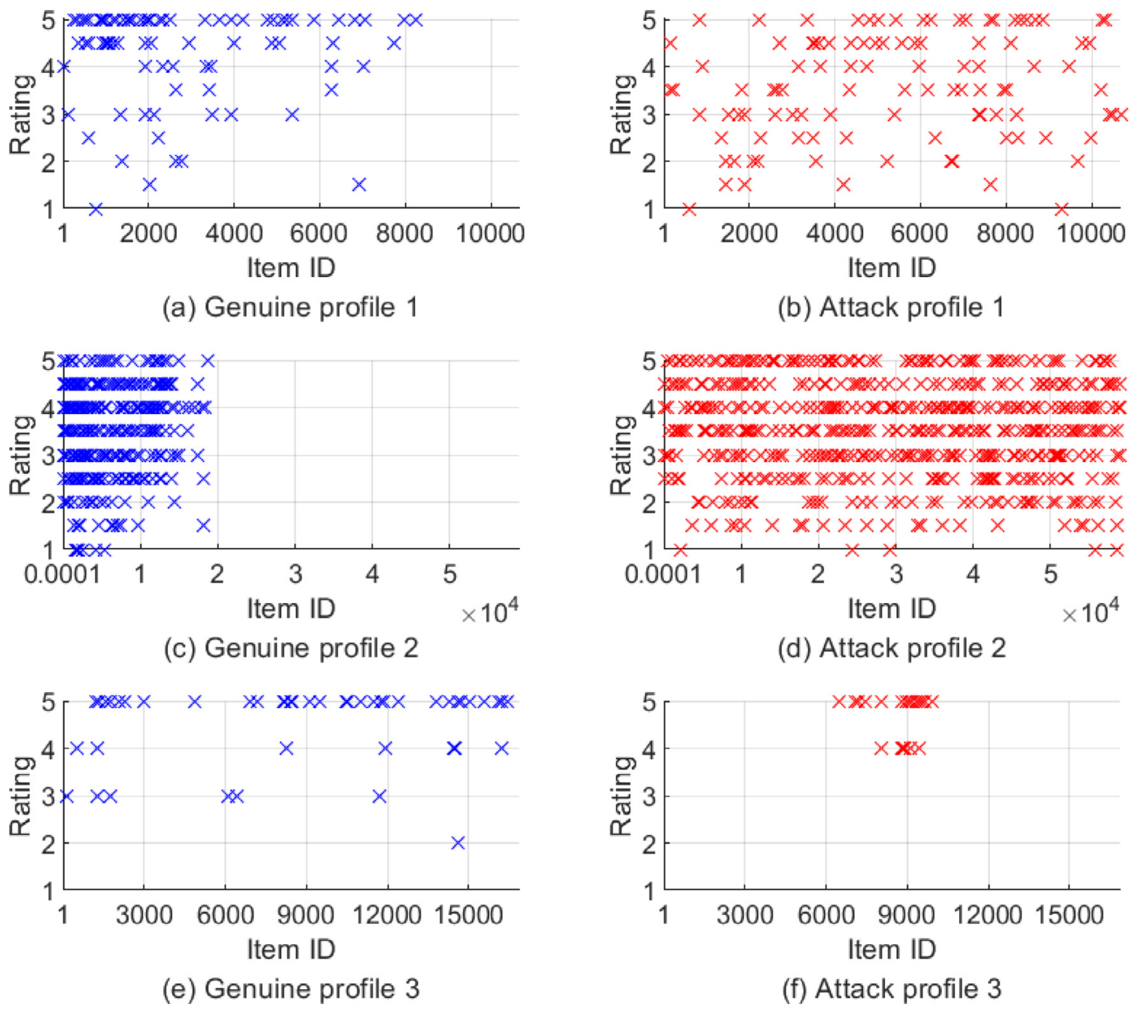

Fig. 1 – Rating vectors of three genuine profiles and three attack profiles. (a) and (b) are randomly selected from the experimental data of MovieLens 10M dataset. (c) and (d) are randomly selected from the experimental data of MovieLens 25M dataset. (e) and (f) are randomly selected from the experimental data of Amazon dataset. The horizontal-axis and vertical-axis denote the item IDs and rating values, respectively. The order of items is the same as they appear in the rating database.

Algorithm 1 Feature extraction algorithm based on dividing windows and rating behavior statistical methods.

```
Require: the set of users S, the number of windows $J$, the set of windows $\{\omega_1, \omega_2, ..., \omega_j, ..., \omega_J\}$
Ensure: the set of feature vectors $X$
 1: $X \leftarrow \emptyset$.
 2: for each user $u \in S$ do
 3:   for $j = 1$ to $J$ do
 4:     Calculate $NRW_{u,\omega_j}$ using Eq. (8).
 5:     Calculate $WNRW_{u,\omega_j}$ using Eq. (9).
 6:   end for
 7:   $x_u \leftarrow (NRW_{u,\omega_1}, WNRW_{u,\omega_1}, ..., NRW_{u,\omega_J}, WNRW_{u,\omega_J})$.
 8:   $X \leftarrow X \cup x_u$
 9: end for
10: return $X$
```

As shown in Figs. 2 and 3, a series of features are extracted from each rating vector. Since these features can capture the details of the user rating behavior mode, effectively, the genuine profiles and attack profiles have different feature vectors as long as they have different user rating behavior modes. Therefore, the proposed feature extraction method can be well used to train the classifiers proposed bellow for the detection.

3.2. *Detection algorithm*

Semi-supervised detection methods require both labeled and unlabeled training samples. They first use some labeled user profiles to initialize a classifier. Then, they use the unlabeled user profiles to improve the classifier.

Let L_{train} and U_{train} denote the labeled and unlabeled training sets of feature vectors which are extracted from the user profiles by Algorithm 1.

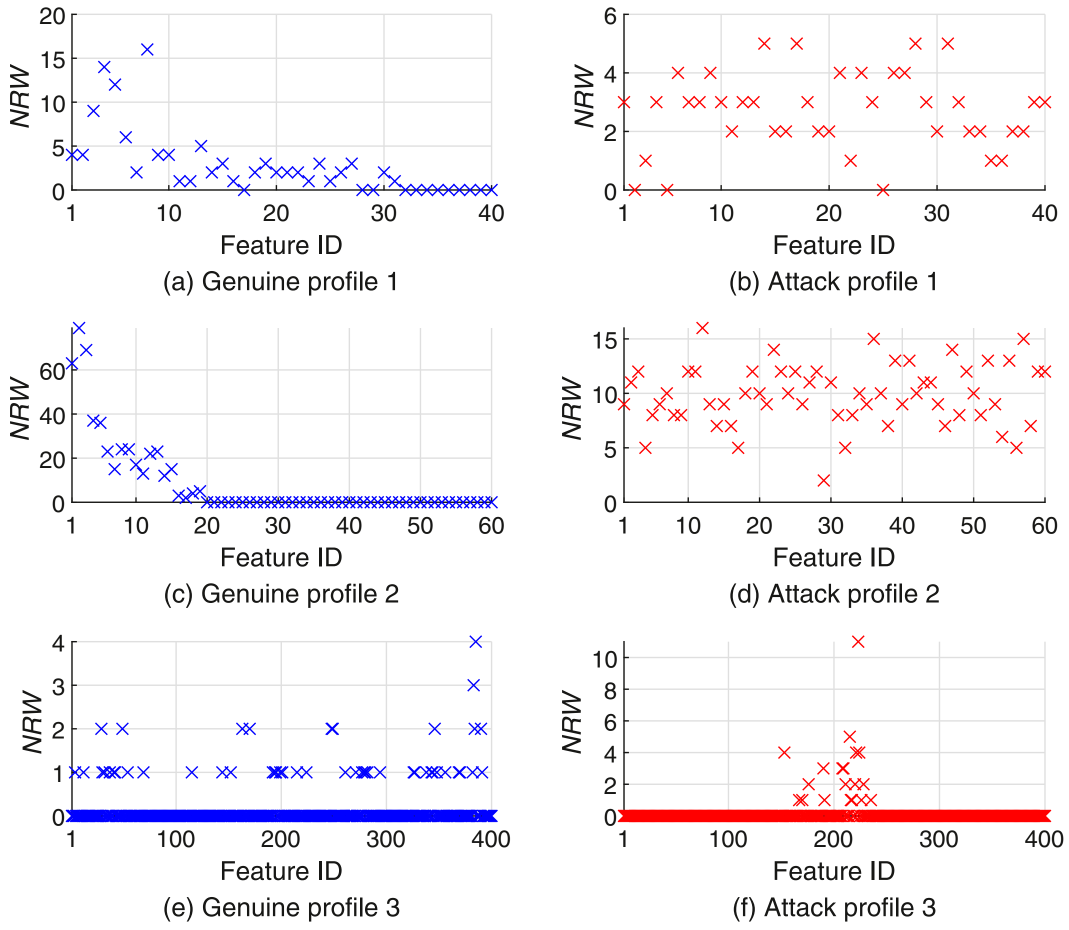

Fig. 2 – Feature *NRW* corresponding to Fig. 1. *J* is set to 40 for (a) and (b). *J* is set to 60 for (c) and (d). *J* is set to 400 for (e) and (f). The horizontal-axis and vertical-axis denote the feature IDs and feature values, respectively.

The basic process of training classifiers for SSADR-CoF can be divided into three steps.

i. L_{train} and the Random Forest algorithm are used to initialize the ensemble of classifiers.
ii. The ensemble of classifiers is used to assign labels to the unlabeled user profiles in U_{train} for producing new labeled user profiles.
iii. L_{train} and the newly labeled user profiles are used to update the classifiers. Repeat steps ii and iii until the stop condition is satisfied.

It becomes simple to assign label to unlabeled user profile after an ensemble of classifiers is generated. To produce new labeled user profiles for updating h_i, the user profiles in U_{train} are classified by the classifiers in H_i. The classification result of each classifier for each user profile consists of two labels, either genuine profile or attack profile. If the ratio of votes for a label exceeds a preset confidence threshold θ, the label is assigned to the user profile. The unlabeled user profile along with the newly assigned label is then copied into the newly labeled set L'_i. The sets $L_{train} \cup L'_i$ are used to update the classifier h_i in this iteration.

If the ratio of votes for a label is above θ, the user profile will be added to L'_i. Under such a condition, the size of L'_i might be very large. All the user profiles in U_{train} might be added to L'_i in an extreme case. However, when the underlying distribution has not been captured by the classifiers, particularly in the previous iterations, using such a huge amount of automatically labeled user profile might reduce the performance of the detection model.

To avoid excessively adding user profile to L'_i, SSADR-CoF uses a weighting mechanism. A user profile is weighted by the predictive confidence of the concomitant ensemble H_i. Here, the confidence is the ratio of votes for a label generated by the classifiers in H_i. Only the user profile satisfying both the confidence threshold and the weight requirement as shown in Eq. (6) might be added to L'_i.

An ensemble of classifiers is expected to have good accurate for the labeling of the unlabeled user profiles. However, it is inevitable to misclassify an unlabeled user profile from time to time. The noisy data might bias the hypothesis of the classifiers in the process of the iterative learning. To compensate the negative influence caused by such noisy data, SSADR-CoF augments the labeled set under the condition of Eq. (6). Satisfying this equation means that both the error rate and the

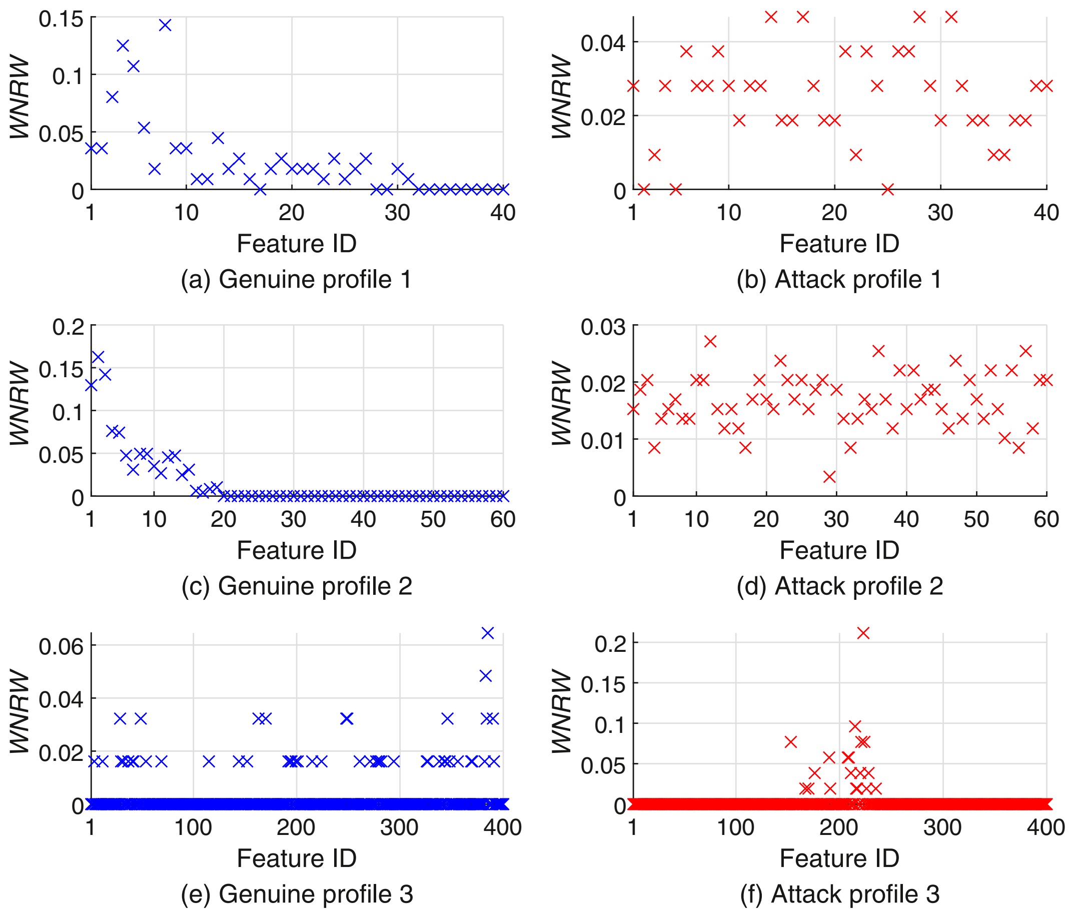

Fig. 3 – Feature *WNRW* corresponding to Fig. 1. *J* is set to 40 for (a) and (b). *J* is set to 60 for (c) and (d). *J* is set to 400 for (e) and (f). The horizontal-axis and vertical-axis denote the feature IDs and feature values, respectively.

weighted number of user profiles being mislabeled by H_i in L'_i are reduced at each iteration.

High diversity among the classifiers is an important factor to improve the generalization power of ensemble learning. To maintain the diversity in the process of learning, SSADR-CoF uses two strategies. On the one hand, Random Forest is used to generate the random trees as the ensemble of classifiers. In this process, certain randomness is injected. Therefore, even for the similar training sets, the random trees could be different from each other. On the other hand, not all the user profiles in U_{train} will be examined by the concomitant ensembles. Instead, a subset of unlabeled user profiles is randomly selected from U_{train}. The total weight of the user profiles in the subset should be less than $\hat{e}_{i,t-1}W_{i,t-1}/\hat{e}_{i,t}$ according to Eq. (6). Then, confident user profiles are further selected from the subset.

According to above analysis, the training process of classifiers based on Co-Forest for SSADR-CoF is described in Algorithm 2.

As shown in Algorithm 2, the Random Forest is used to generate the random trees, i.e., the ensemble of classifiers using the labeled training set L_{train} in Line 1. Certain randomness is injected in this process to maintain the diversity among the classifiers. The error rate and the weight are initialized in Line 2-Line 6. t in Line 7 and Line 9 is used to record the number of iterations.

The training set is updated in Line 10-Line 26. The error rate is estimated on L_{train} using H_i in Line 12. The newly added set is initialized as empty at the beginning in Line 13. To meet the condition of Eq. (6), the error rate of tth iteration should be less than that of $(t-1)$th iteration in Line 15. A subset is randomly selected from the U_{train} in Line 16. The weight of the subset should meet the condition of Eq. (6). Certain randomness is also injected in this process to maintain the diversity among the classifiers. The user profile whose confidence is less than the threshold will be filtered in Line 20. The user profile with the newly added label is added to $L'_{i,t}$ in Line 21. The weight is calculated in Line 22.

The ensemble of classifiers is update under the condition of Eq. (6) in Line 28-Line 33. The classifiers are returned in Line 35.

Let X_{test} denote the test set in the feature space. Let $h_i(i = 1, ..., N)$ denote the ensemble of classifiers generated by Algorithm 2. Let D_{result} denote the set of detection result. The detecting process of SSADR-CoF using the ensemble of classifiers based on Co-Forest is described in Algorithm 3.

As shown in Algorithm 3, the majority voting method is used to generate the detection result in Line 3. The detection

Algorithm 2 Training process of classifiers based on Co-Forest for SSADR-CoF.

```
Input: the labeled set $L_{train}$, the unlabeled set $U_{train}$, the confidence threshold $\theta$, and the number of random trees $N$
Output: $N$ classifiers $h_i(i = 1, ..., N)$
1: Construct a random forest consisting $N$ random trees using $L_{Train}$. /*Initialize the ensemble of classifiers*/
2: /*For each random tree, i.e., each classifier*/
3: for $i \in 1, \dots, N$ do
4:     $\hat{e}_{i,0} \leftarrow 0.5$
5:     $W_{i,0} \leftarrow 0$
6: end for
7: $t \leftarrow 0$
8: repeat
9:     $t \leftarrow t + 1$ /*Record the number of iterations*/
10:    /*For each random tree, i.e., each classifier*/
11:    for $i \in 1, \dots, N$ do
12:        $\hat{e}_{i,t} \leftarrow$ EstimateError $(H_i, L_{Train})$ /*Estimate the error rate*/
13:        $L'_{i,t} \leftarrow \emptyset$
14:        /*If the error rate is reduced*/
15:        if $\hat{e}_{i,t} < \hat{e}_{i,t-1}$ then
16:            $U'_{i,t} \leftarrow$ SubSampled$(U_{train}, \frac{\hat{e}_{i,t-1}W_{i,t-1}}{\hat{e}_{i,t}})$ /*Select a subset from $U_{train}$, randomly, under the condition of Eq.(6)*/
17:            /*For each user profile in the subset*/
18:            for each $x_u \in U'_{i,t}$ do
19:                /*Judge the confidence of each user profile*/
20:                if Confidence $(H_i, x_u) > \theta$ then
21:                    $L'_{i,t} \leftarrow L'_{i,t} \cup (x_u, H_i(x_u))$ /*Merge the user profile with the newly added label */
22:                    $W_{i,t} \leftarrow W_{i,t} + Confidence(H_i, x_u)$ /*Calculate the weight*/
23:                end if
24:            end for
25:        end if
26:    end for
27:    /*For each random tree, i.e., each classifier*/
28:    for $i \in 1, \dots, N$ do
29:        /*Satisfy Eq. (6)*/
30:        if $\hat{e}_{i,t}W_{i,t} < \hat{e}_{i,t-1}W_{i,t-1}$ then
31:            $h_i \leftarrow$ LeanRandomTree$(L \cup L'_{i,t})$ /*Update the classifier*/
32:        end if
33:    end for
34: until none of the trees in Random Forest changes
35: return $h_i(i = 1, ..., N)$ /*Return the ensemble of classifiers*/
```

Algorithm 3 Detecting process of SSADR-CoF using the ensemble of classifiers based on Co-Forest.

```
Input: $X_{test}, h_i(i = 1, ..., N)$
Output: $D_{result}$
1: $D_{result} \leftarrow \emptyset$
2: for $x \in X_{test}$ do
3:     $d \leftarrow \arg\max_{y \in \{g,a\}} \sum_{i:h_i(x)=y} 1$ /*Majority voting*/
4:     $D_{result} \leftarrow D_{result} \cup d$ /*Record the detection result*/
5: end for
6: return $D_{result}$ /*return the detection result*/
```

Table 2 – Number of samples in the training set based on the MovieLens 10M dataset.

Type of data	Number	Filler size			
		1%	3%	5%	10%
Genuine	1000	-	-	-	-
Random attack	-	10	10	10	10
Average attack	-	10	10	10	10
Bandwagon attack	-	10	10	10	10

Table 3 – Number of samples in the validation set based on the MovieLens 10M dataset.

Type of data	Number	Filler size			
		1%	3%	5%	10%
Genuine	200	-	-	-	-
Random attack	-	10	10	10	10
Average attack	-	10	10	10	10
Bandwagon attack	-	10	10	10	10

results consist of g and a which denote genuine profile and attack profile, respectively.

4. Experiments and analysis

In this section, we first introduce the experimental data and settings. Then, we set the optimal hyper parameters for the proposed approach. Finally, we describe the comparison methods and analyze the experimental results.

4.1. *Experimental data and settings*

Three benchmark datasets, i.e., MovieLens 10M (Harper and Konstan, 2015), MovieLens 25M (Harper and Konstan, 2015), and Amazon (Xu et al., 2013), are used in the experiments. MovieLens datasets are the most widely used datasets in the research field of recommendation attack detection. Amazon dataset is from the real application scenario. Different from the MovieLens datasets in which attack profiles are constructed by attack models, Amazon dataset contains real attack profiles. From the perspective of data size, Amazon is a small dataset, MovieLens 10M is a medium-scale dataset, and MovieLens 25M is a large-scale dataset.

MovieLens 10M dataset contains 10,000,054 ratings on 10,681 movies from 71,567 users. Each user profile consists of at least 20 ratings. The rating values are between 0.5 and 5.0 with 0.5 as an increment. All user profiles in this dataset are labeled as genuine profiles. Attack profiles will be generated by attack models.

Table 2 shows the number of samples in the training set. As shown in Table 2, the 1000 genuine profiles are selected from the MovieLens 10M dataset. The 10 attack profiles are constructed by using the random, average, and bandwagon attack models with various filler sizes.

Table 3 shows the number of samples in the validation set. As shown in this table, the 200 genuine profiles are selected

Table 4 – Number of samples in the training set, validation set, and test set based on Amazon dataset.

Experimental set	Genuine profile	Attack profile
Training set	1000	1000
Validation set	200	200
Test set	1075	308

from the remaining profiles of the MovieLens 10M dataset. The attack profiles are constructed by using different attack models with various filler sizes.

To verify the effectiveness of the proposed approach, a large number of test sets are created in this paper. For each test set, we randomly select 1000 profiles from the remaining profiles of the MovieLens 10M dataset as genuine profiles. Attack profiles are generated by random, average, and bandwagon attack models. Attack sizes are set to 1%, 2%, 5%, and 10%, respectively. Filler sizes are set to 1%, 3%, 5%, and 10%, respectively. The target item in each test set is selected randomly. The rating values of the target item is set to the maximum value. The above process of generating test sets is repeated 10 times and the average values of detection results are reported in our experiments.

MovieLens 25M dataset contains 25,000,095 ratings on 62,423 movies from 162,541 users. Each user profile consists of at least 20 ratings. The rating values are between 0.5 and 5.0 with 0.5 as an increment. All user profiles in this dataset are labeled as genuine profiles. Attack profiles will be generated by attack models. The numbers and methods for creating the training, validation, and test sets on MovieLens 25M dataset are just the same as on MovieLens 10M dataset.

Amazon dataset contains 5055 users consisting of 3118 genuine profiles and 1937 attack profiles. Since a few ratings cannot provide enough information to infer the taste, we remove the users whose number of ratings is less than 5. The rest of the dataset is used in our experiments which contains 47,408 ratings on 16,885 items from 2275 genuine profiles and 1508 attack profiles. Table 4 shows the experimental sets based on the Amazon dataset.

To evaluate the detection performance, three standard metrics, i.e., recall, precision, and AUC (Fawcett, 2006; Hand and Till, 2001), are used in our experiments. F-measure (Hand and Till, 2001) is the harmonic average of the recall and precision. To find the optimal hyper parameters for the proposed approach, f-measure is used to evaluate the detection results of validation set. The RandomForest package in WEKA3.4[3] is used to generate the random trees.

4.2. *Setting optimal hyper parameters*

The proposed approach contains five hyper parameters which are the number of windows J in Algorithm 1 and the size of L_{train}, the size of U_{train}, the confidence threshold θ, and the number of random trees N in Algorithm 2. To find the optimal hyper parameters for each dataset, we observe the detection performance of the proposed approach on the validation set with different values of hyper parameters.

We use the ratio of labeled user profiles to determine the size of L_{train} and U_{train}. Let D denote the size of the training set. Let $d\%$ denote the ratio of labeled user profiles. The size of L_{train} and U_{train} can be calculated as $D \times d\%$ and $D \times (1 - d)\%$, respectively. The user profiles are randomly selected from the training set to construct L_{train} and U_{train} according to their sizes. The class distributions in L_{train} and U_{train} are kept similar to that in the training set.

The detection results of the proposed approach with different hyper parameters on the validation set of MovieLens 10M dataset are shown in Fig. 4.

As shown in Fig. 4(a), the detection performance is better with small J. When J is set to 40, the f-measure shows the maximum value. Therefore, we set J to 40 for MovieLens 10M dataset. As shown in Fig. 4(b), when $d\%$ is greater than or equal to 50% the detection performance keeps at a high level. The reason for this phenomenon might be that there are enough samples to learn a reasonable hypothesis for the proposed approach when $d\%$ is greater than or equal to 50%. the detection performance is quite good when $d\%$ is set to 30% which means that only 30% user profiles in the training set are used as the labeled samples. Therefore, we set $d\%$ to 30% in the proposed approach for MovieLens 10M dataset. As shown in Fig. 4(c), the f-measure does not achieve the optimal value when θ is set too large or too small. The large values of θ will reduce the recall. The small values of θ will increase the false alarm rate. When θ is set to 0.55, 0.65, 0.70, or 0.75, the detection performance is good. Therefore, we set θ=0.75 in the proposed approach for MovieLens 10M dataset. As shown in Fig. 4(d), the f-measure does not continue to increase with the increasing of N. One possible reason for this phenomenon is that the diversity among the classifiers cannot always increase for the fixed training set. The increasing of same or similar classifiers will not effectively improve the detection performance. The maximum values of f-measure occur when N is set to 6 on MovieLens 10M dataset.

The detection results with different hyper parameters on the validation set of MovieLens 25M dataset are shown in Fig. 5.

As shown in Fig. 5(a), when J is set to 60, 80, 100, 200, 300, or 400, the f-measure shows the maximum value. To reduce the feature dimension of samples which is determined by J, we set J to 60 in the proposed approach for MovieLens 25M dataset. As shown in Fig. 5(b), when $d\%$ is set to 1%, the proposed approach already has good detection performance. Therefore, $d\%$ is set to 1% for MovieLens 25M dataset. As shown in Fig. 5(c), when θ is between 0.5 and 0.75, the performance of the detection method is good. When θ is greater than 0.75, the value of f-measure begins to decrease. Therefore, θ is set to 0.75 for MovieLens 25M dataset. As shown in Fig. 5(d), when N is greater than 6, f-measure shows the maximum value. Therefore, N is set to 6 for MovieLens 25M dataset.

The detection results with different hyper parameters on the validation set of Amazon dataset are shown in Fig. 6.

As shown in Fig. 6(a), when J is set to 400 or 700, the proposed approach achieves the maximum f-measure. Therefore, we set J to 400 in the proposed approach for Amazon dataset. As shown in Fig. 6(b), the detection performance is quite good

[3] https://www.cs.waikato.ac.nz/ml/weka/index.html.

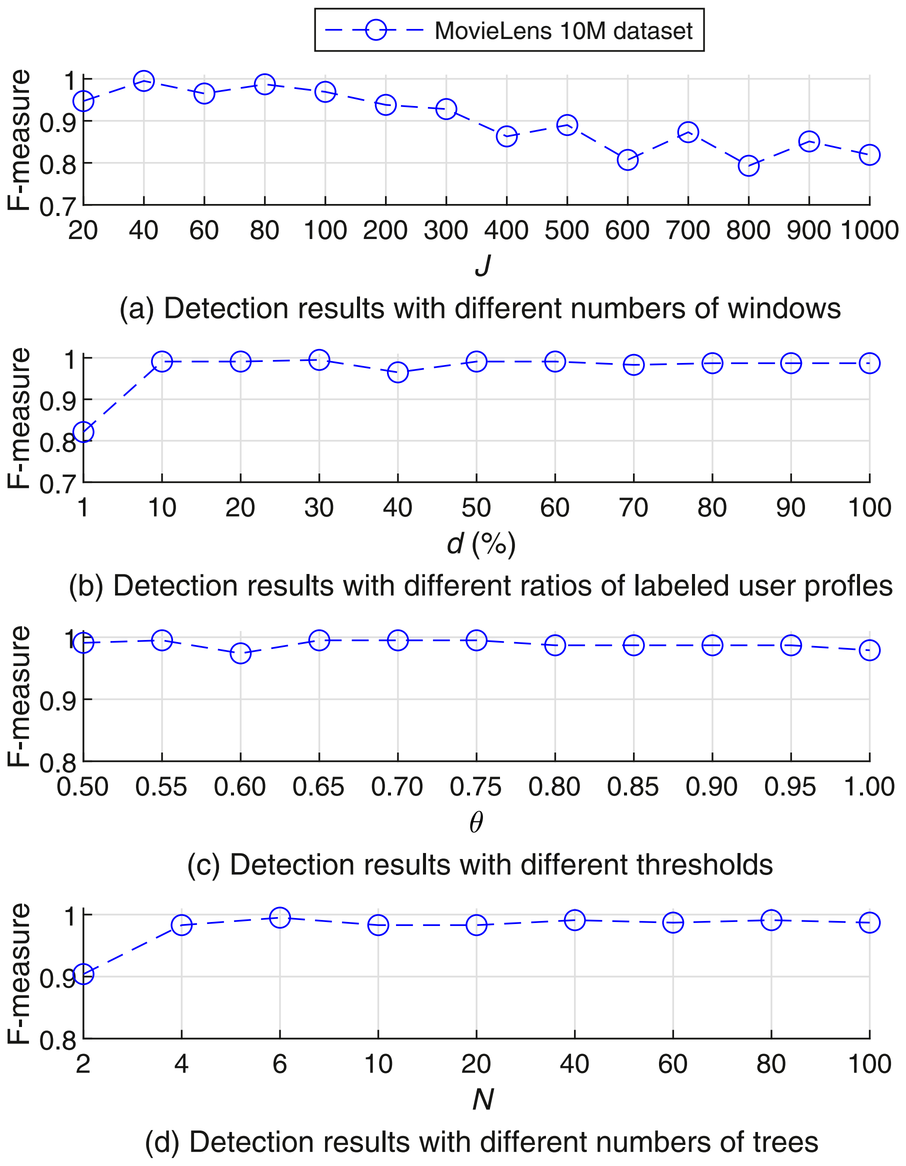

Fig. 4 – Detection results of the proposed approach with different hyper parameters on the validation set of MovieLens 10M dataset. (a) d%=30%, θ=0.75, and N=6. (b) J=40, θ=0.75, and N=6. (c) J=40, d%=30%, and N=6. (d) J=40, d%=30%, and θ=0.75.

when d% is set to 30%. Therefore, we set d% to 30% in the proposed approach for Amazon dataset. As shown in Fig. 6(c), when θ is set to 0.55 or 0.75, the f-measure shows the maximum value. Therefore, we set θ to 0.75 in the proposed approach for the Amazon dataset. As shown in Fig. 6(d), the maximum values of f-measure occur when N is set to 40 on Amazon dataset. Therefore, we use 40 as the hyper parameters in the proposed approach.

The final hyper parameters set in the proposed approach for MovieLens 10M, MovieLens 25M, and Amazon datasets are summarized in Table 5.

Table 5 – Hyper parameters for MovieLens 10M, MovieLens 25M, and Amazon datasets.

Hyper parameter	MovieLens 10M	MovieLens 25M	Amazon
J	40	60	400
d%	30%	1%	30%
θ	0.75	0.75	0.75
N	6	6	40

4.3. *Comparison methods*

The following state-of-the-art detection methods are used to compare with the proposed approach.

- PCA-VarSelect (Mehta et al., 2007): The PCA (Principal Component Analysis) technique is used to filter out the attack profiles through computing covariance among users in this method. This is an unsupervised and early representative method with good detection performance.

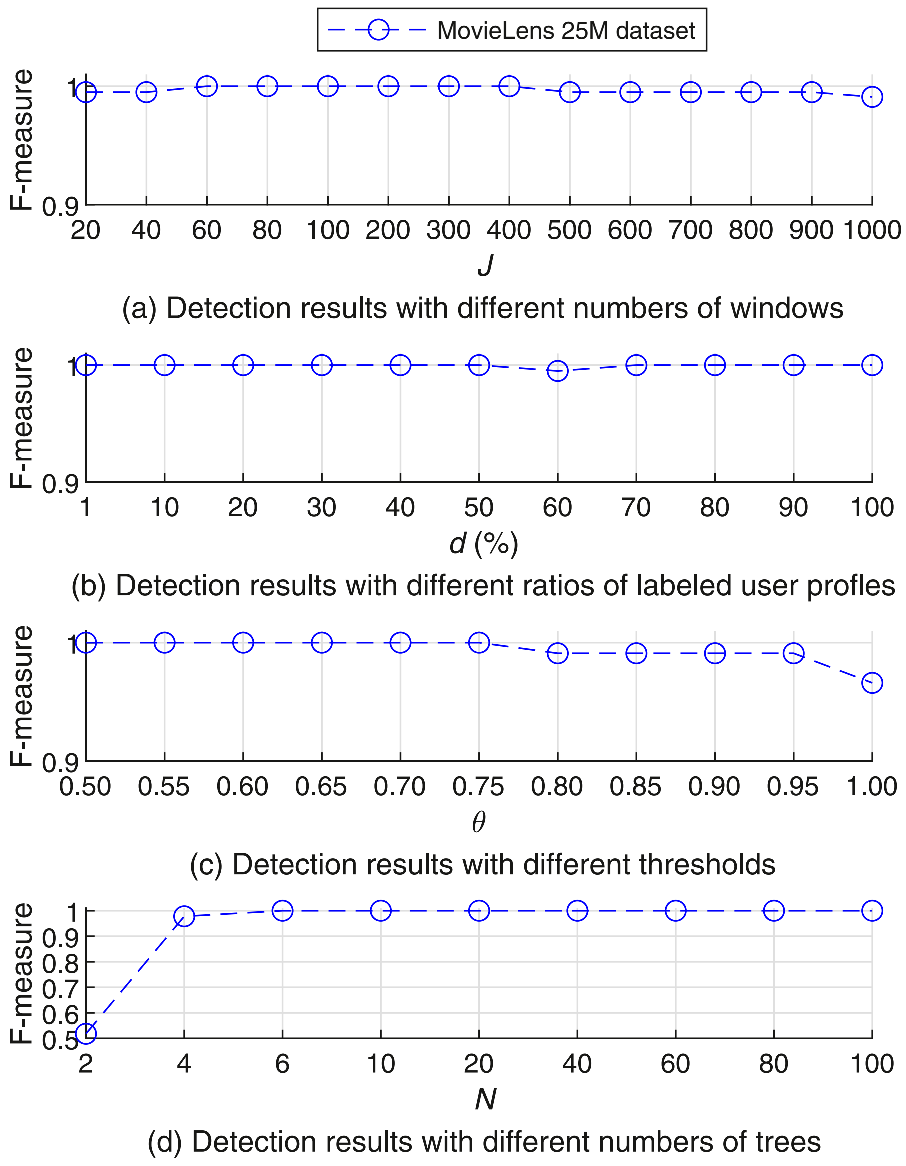

Fig. 5 – Detection results of the proposed approach with different hyper parameters on the validation set of MovieLens 25M dataset. (a) d%=30%, θ=0.75, and N=6. (b) J=60, θ=0.75, and N=6. (c) J=60, d%=1%, and N=6. (d) J=60, d%=1%, θ=0.75.

- CNN-SAD (Tong et al., 2018): A deep learning-based method in which one convolutional and pooling layer is used to generate a classifier for the detection. This is a supervised and representative method with good detection performance.
- DL-DRA (Zhou et al., 2020): A deep learning-based method in which multiple convolutional layers are used to built a classifier. Under the assumption that there are enough training samples, this method will have good detection performance.
- CNN-LSTM (Ebrahimian and Kashef, 2020): A deep learning-based method in which hybrid neural network layers are used to train a detection model.
- HySAD (Wu et al., 2012): This is a semi-supervised and representative method with certain detection performance. A single Naïve Bayes-based classifier is used to detect the attack profiles in this method.

To illustrate the effectiveness of the unlabeled user profiles in terms of improving detection performance in the proposed approach, we compare the proposed approach with the following methods.

- RF-L (Random Forest-Labeled): This method uses the Random Forest algorithm to generate the ensemble of classifiers as the detection model. The same settings as in the proposed approach are used for Random Forest algorithm. Being different from the proposed approach, only the d% selected user profiles in the training set are used to train the Random Forest algorithm, and no unlabeled user profiles are used. This method is expected to have lower detection performance than that of the proposed approach, since no unlabeled user profiles are used to improve the classifiers.

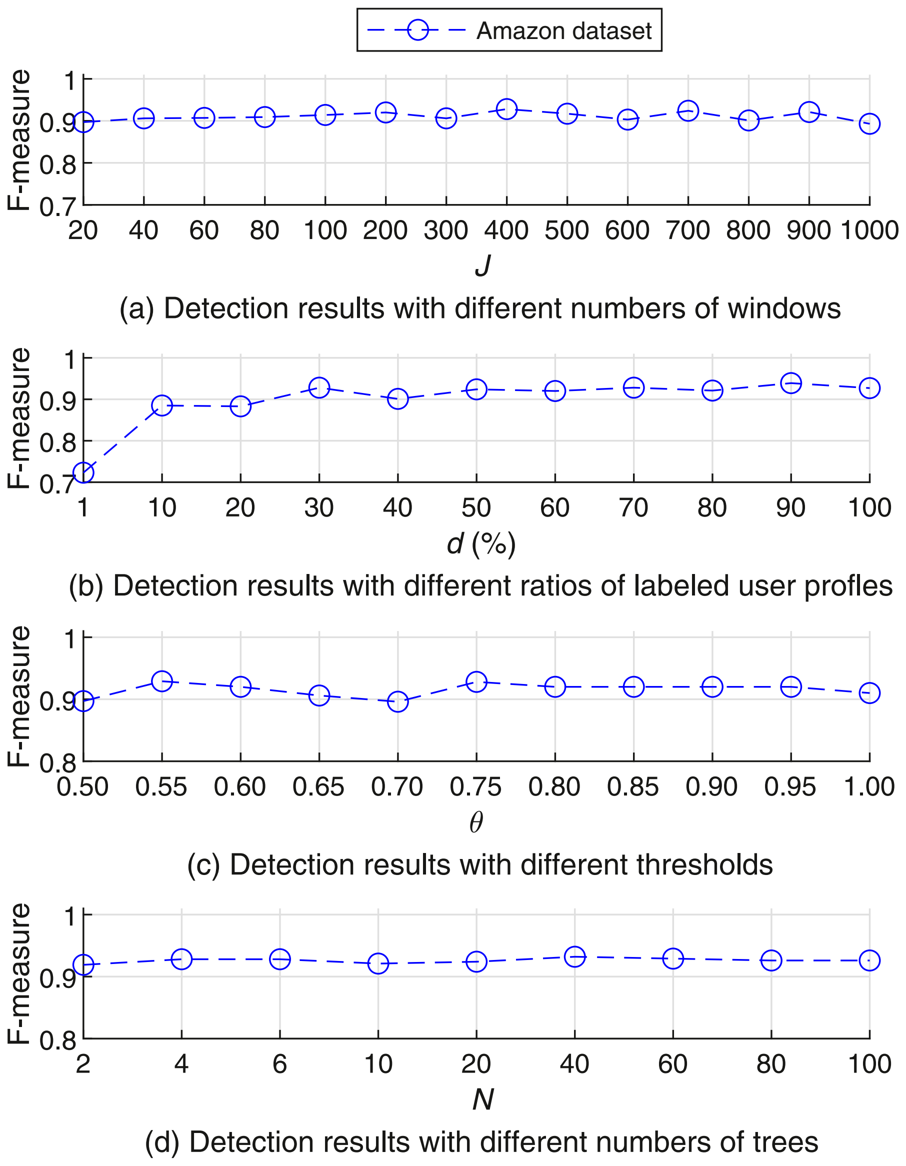

Fig. 6 – Detection results of the proposed approach with different hyper parameters on the validation set of Amazon dataset. (a) *d*%=30%, *θ*=0.75, and N=6. (b) *J*=400, *θ*=0.75, and N=6. (c) *J*=400, *d*%=30%, and N=6. (d) *J*=400, *d*%=30%, and *θ*=0.75.

- RF-A (Random Forest-All): This method uses the same settings as RF-L. The difference is that all the user profiles in the training set are used as labeled samples to train the Random Forest algorithm for generating the ensemble of classifiers. This method is expected to have similar detection performance as that of the proposed approach, since both this method and the proposed approach use the same number of training samples.

4.4. *Experimental results and analysis*

Detection results of test sets for random, average, and bandwagon attack on MovieLens 10M dataset are shown in Figs. 7–9, respectively.

As shown in Figs. 7–9, PCA-VarSelect method has good detection performance when detecting the random and average attack on MovieLens 10M dataset. When detecting the bandwagon attack, however, the detection performance of this method decreases especially for small filler sizes. The reason for the performance degradation might be as follows. In bandwagon attack, the ratings of popular items increase the contribution of attack profiles to the collaborative recommender system. The PCA-VarSelect method takes part of the attack profiles as the principal components and cannot recognize them, effectively.

For most test sets of MovieLens 10M dataset, the HySAD method has high recall but low precision. The results illustrate that although this method can recognize a certain attack profiles, it misclassified too many genuine profiles as attack profiles.

For many test sets of MovieLens 10M dataset, the RF-L method has low recall and precision. The main reason is that the training set for the method contains too small samples. The detection model cannot be trained, effectively.

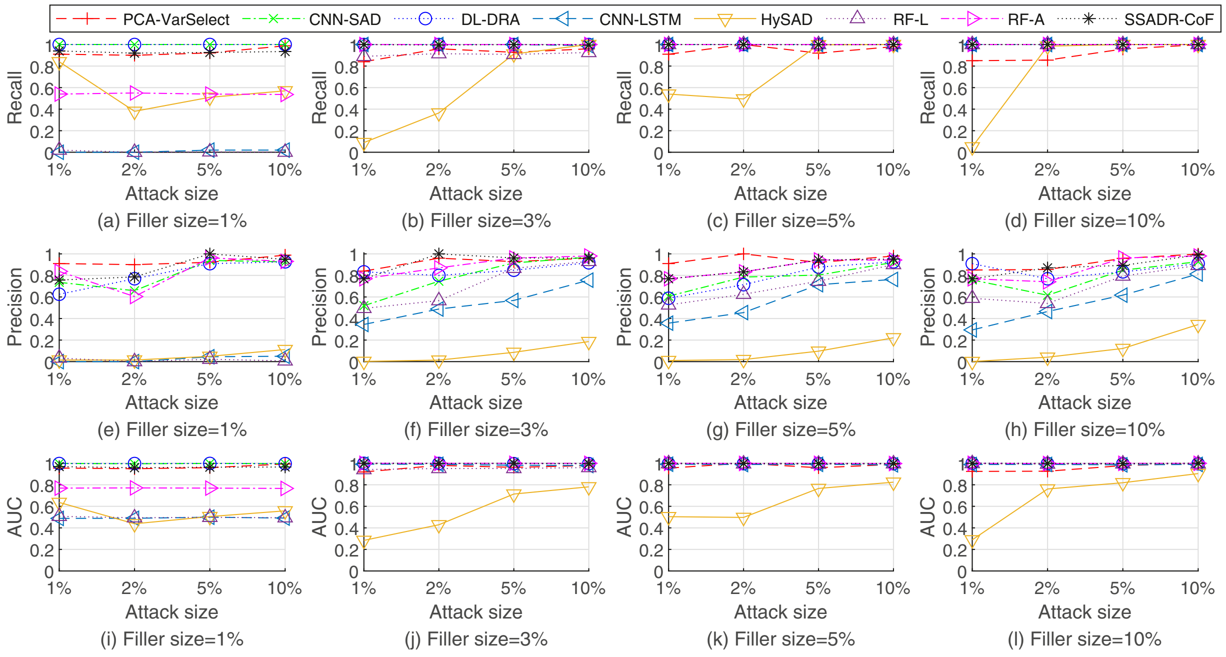

Fig. 7 – Detection results of the eight methods for random attack on the test sets of MovieLens 10M dataset.

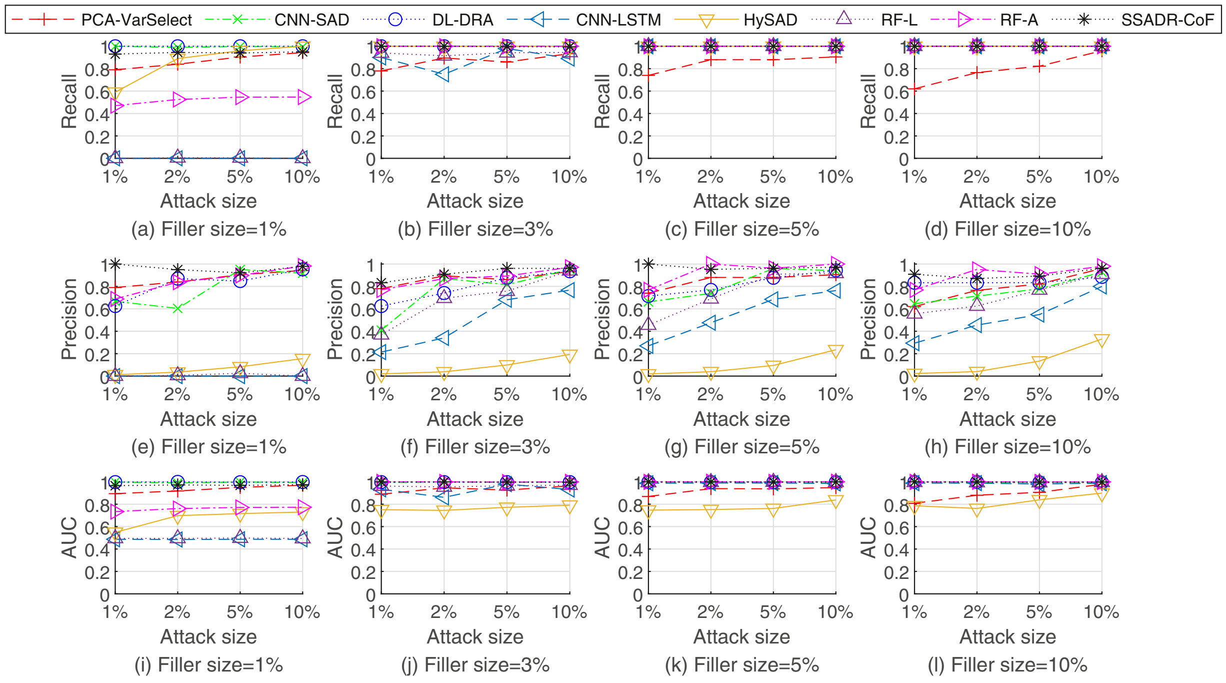

Fig. 8 – Detection results of the eight methods for average attack on the test sets of MovieLens 10M dataset.

Although, CNN-LSTM has high recall and precision, the precision of this method is low. The CNN-SAD, DL-DRA, RF-A, and SSADR-CoF methods have high recall, precision, and AUC when detecting the three types of attack profiles on MovieLens 10M dataset.

As deep learning-based methods, CNN-SAD and DL-DRA effectively displays the ability of accurate identification. The series of features and the ensemble of classifiers are the important basis to improve the detection performance of RF-A and SSADR-CoF. Note that, being different from CNN-SAD, DL-DRA and RF-A in which all user profiles in the training set are used as the labeled samples, only 30% user profiles in the training set are used as the labeled samples in the proposed approach SSADR-CoF.

As a summary, we show the average recall, precision, and AUC of all test sets on MovieLens 10M dataset in Table 6.

As shown in Table 6, the proposed approach SSADR-CoF is the only method whose average recall, precision, and AUC

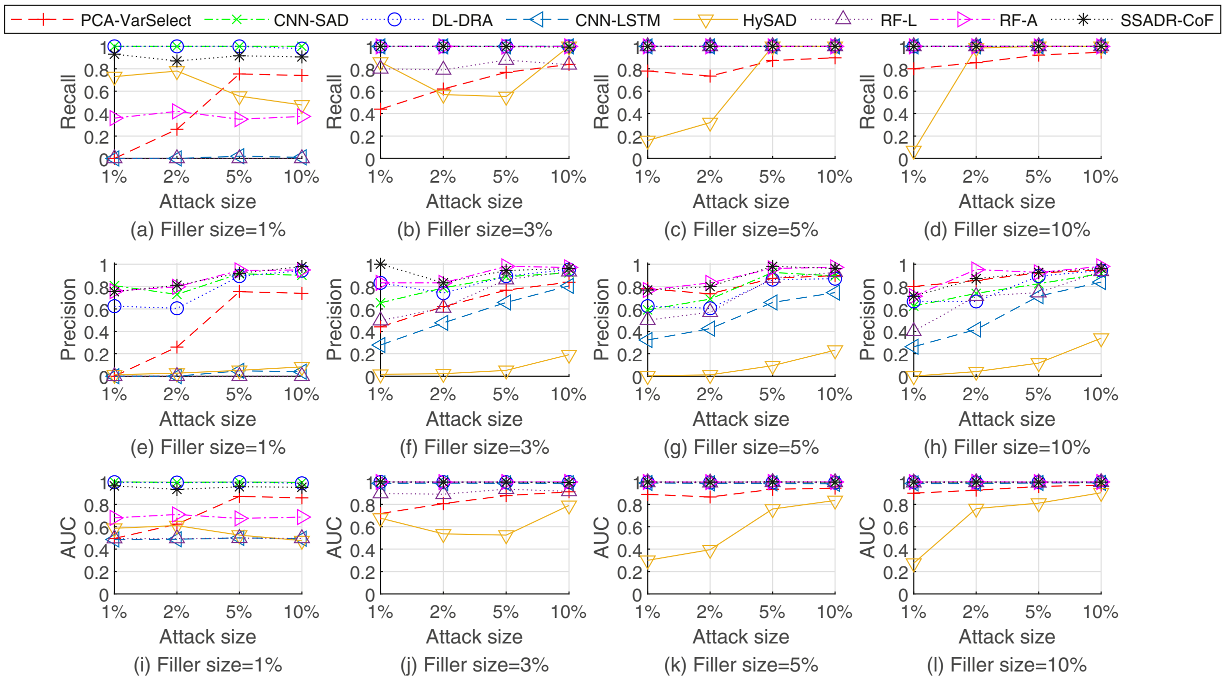

Fig. 9 – Detection results of the eight methods for bandwagon attack on the test sets of MovieLens 10M dataset.

Table 6 – Average recall, precision, and AUC of the eight methods on all test sets of MovieLens 10M dataset. The values which are greater than 0.9 are bold.

Method	Average recall	Average precision	Average AUC
PCA-VarSelect	0.825	0.825	**0.910**
CNN-SAD	**0.999**	0.792	**0.996**
DL-DRA	**0.999**	0.811	**0.996**
CNN-LSTM	0.741	0.410	**0.857**
HySAD	0.775	0.080	0.658
RF-L	0.723	0.535	0.856
RF-A	0.869	0.880	**0.933**
SSADR-CoF	**0.980**	**0.904**	**0.988**

Table 7 – Average recall, precision, and AUC of the eight methods on all test sets of MovieLens 25M dataset. The values which are greater than 0.9 are bold.

Method	Average recall	Average precision	Average AUC
PCA-VarSelect	0.876	0.876	**0.936**
CNN-SAD	**0.998**	**0.943**	**0.998**
DL-DRA	**0.999**	**0.926**	**0.999**
CNN-LSTM	**0.999**	0.793	**0.996**
HySAD	0.025	0.040	0.507
RF-L	0.000	0.000	0.500
RF-A	**0.999**	**0.914**	**0.999**
SSADR-CoF	**0.999**	**0.943**	**0.999**

are larger than 0.9 on the MovieLens 10M dataset. The experimental results demonstrate the effectiveness of the proposed approach.

Detection results of test sets for random, average, and bandwagon attack on MovieLens 25M dataset are shown in Figs. 10–12, respectively.

As shown in Figs. 10–12, the detection performance of HySAD and RF-L drop to low level on the large-scale dataset. However, the other methods, such as PCA-VarSelect, CNN-SAD, DL-DRA, CNN-LSTM, RF-A, and SSADR-CoF, have better detection performance.

As a summary, we show the average recall, precision, and AUC of all test sets on MovieLens 25M dataset in Table 7.

As shown in Table 7, the average recall, precision, and AUC of four detection methods CNN-SAD, DL-DRA, RF-A, and SSADR-CoF are greater than 0.9. It should be pointed out that the proposed approach only uses 1% of the training data and achieves the same good detection results as the state-of-the-art methods.

Detection results of test sets on Amazon dataset are shown in Table 8. Note that, only one test set is built as introduced in Table 4.

As shown in Table 8, the detection performance of PCA-VarSelect method is low on Amazon dataset. One possible reason is that there are a certain number of group attack profiles in the Amazon dataset. The PCA-VarSelect method takes the group attack profiles as the principal components and cannot effectively filter them.

HySAD has high recall and AUC on Amazon dataset. However, the precision of this method is low. The reason might be that a single Naïve Bayes classifier used in HySAD may not be able to effectively recognize multiple user rating behavior modes in the rating database. Therefore, some genuine profiles are misclassified by this method.

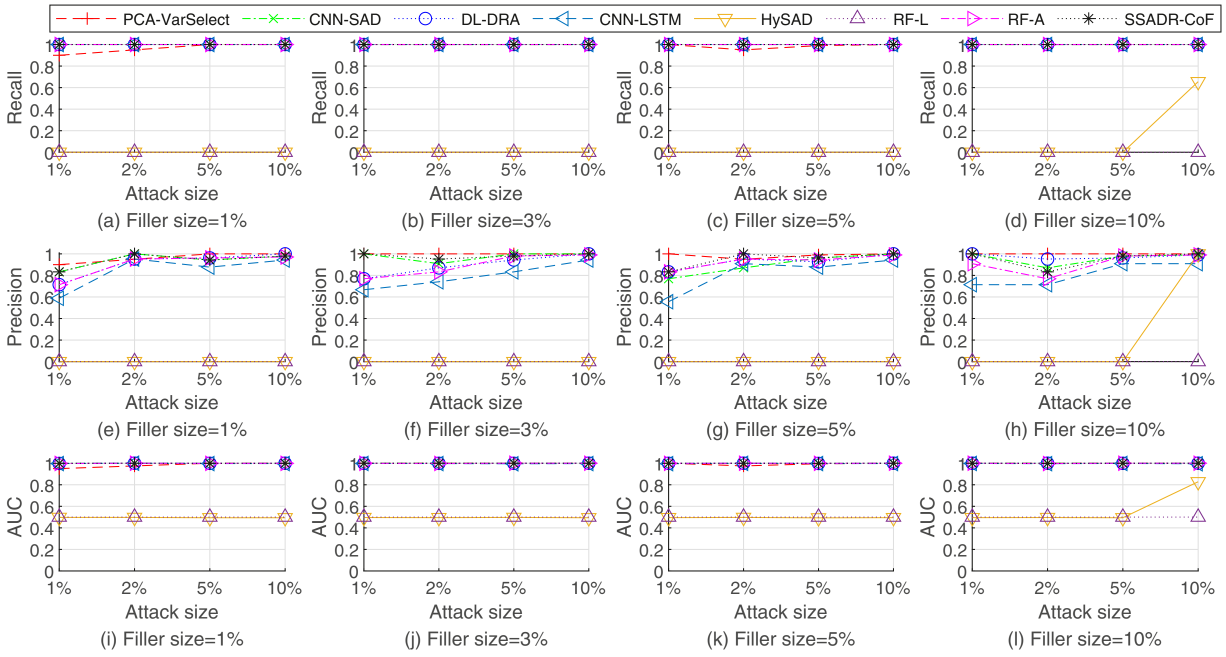

Fig. 10 – Detection results of the eight methods for random attack on the test sets of MovieLens 25M dataset.

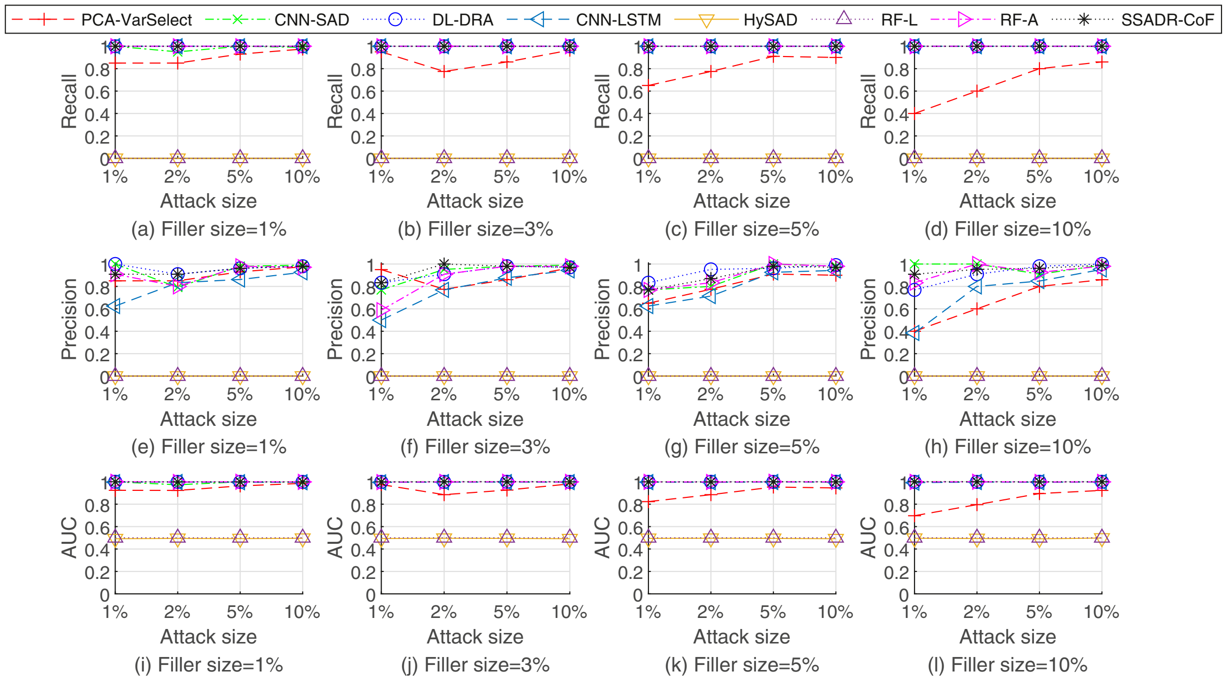

Fig. 11 – Detection results of the eight methods for average attack on the test sets of MovieLens 25M dataset

Due to the lack of training samples, the precision of RF-L is not high. When detecting the test set of Amazon dataset, the deep learning-based methods CNN-SAD, DL-DRA, and CNN-LSTM show good stability.

Both RF-A and SSADR-CoF have high and similar detection performance on Amazon dataset. The experimental results on Amazon dataset demonstrate the effectiveness of the proposed approach.

4.5. *Discussions*

The existing feature extraction methods usually focus on extracting one or several features from a certain angle. For example, the widely used features *WDMA* (weighted deviation from mean agreement) and *RDMA* (rating deviation from mean agreement) (Wu et al., 2012) describe the characteristics of recommendation attack from the perspective of rating deviation. Being different from these works, the proposed feature extrac-

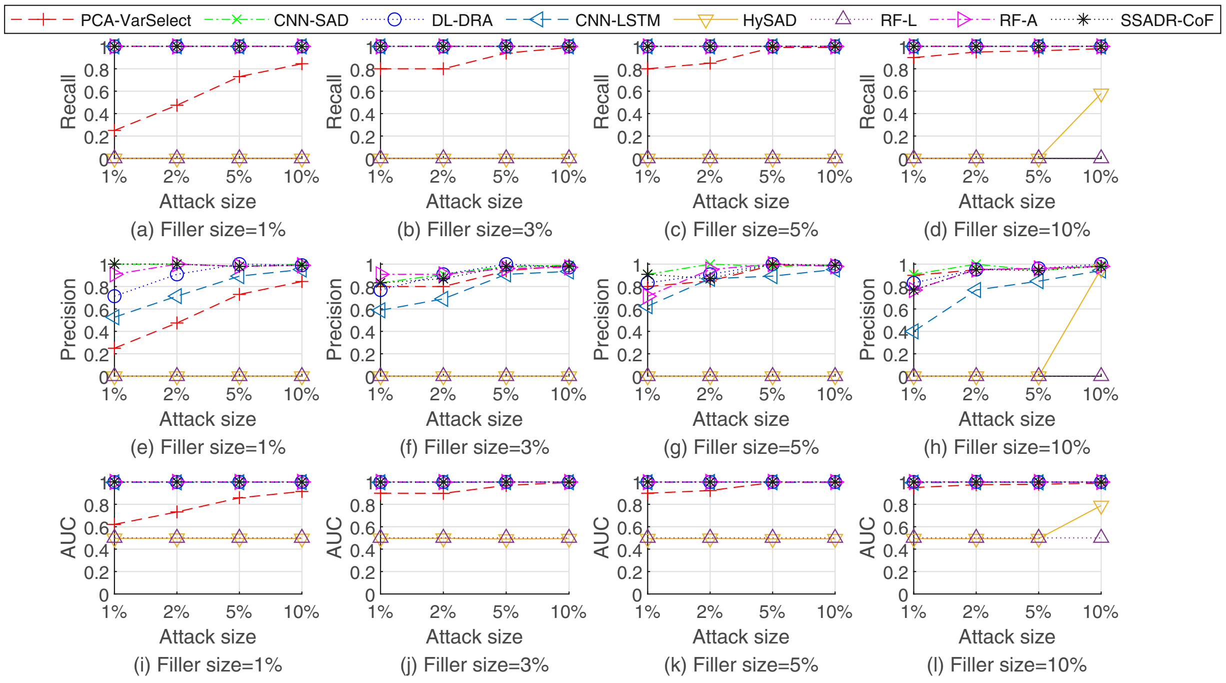

Fig. 12 – Detection results of the eight methods for bandwagon attack on the test sets of MovieLens 25M dataset.

Table 8 – Detection results of the eight methods on the test sets of Amazon dataset. The values which are greater than 0.9 are bold.

Method	Recall	Precision	AUC
PCA-VarSelect	0.207	0.207	0.490
CNN-SAD	0.795	0.819	0.872
DL-DRA	**0.931**	0.797	**0.932**
CNN-LSTM	**0.951**	0.605	0.886
HySAD	0.886	0.460	0.794
RF-L	**0.977**	0.677	**0.922**
RF-A	**0.964**	0.777	**0.942**
SSADR-CoF	**0.961**	0.795	**0.945**

tion method is devoted to extracting a series of, rather than one or a few of, features to capture the details of the user rating behavior mode. The above experimental results illustrate the effectiveness of the proposed feature extraction method.

The existing representative semi-supervised methods use a single classifier for detecting the recommendation attack and suffers from low precision as shown in the experiments. Being different from the existing work, the proposed approach is based on the ensemble learning and improves the precision, effectively.

As shown in Table 6, the precision of the proposed approach SSADR-CoF is greater than 0.9. This is due to the fact that the series of features and the ensemble learning technology effectively improve the detection performance in the proposed approach. As shown in Table 7, the precision of four approaches is greater than 0.9. It shows better detection performance on large-scale dataset than that on the MovieLens 10M dataset. The reason for this phenomenon might be as follows. There is a large number of items in the large-scale dataset. The ratings of genuine users may focus on a segment with certain interests. However, in the attack profiles built by attack models, a number of items are selected, randomly. It may be this strategy that makes the attack profiles easier to be detected on the large-scale dataset. As shown in Table 8, precision of all the approaches is less than 0.9. One possible reason for this phenomenon is that real attack profiles are more diverse and complex, which brings new challenges to the detection. Detection methods for real attack profiles need to be further studied.

As shown in Table 6, the recall of CNN-SAD and DL-DRA is 0.019 larger than the recall of SSADR-CoF. AUC of CNN-SAD and DL-DRA is 0.008 larger than the AUC of SSADR-CoF. However, the precision of the proposed approach is 0.112 and 0.093 larger than the precision of CNN-SAD and DL-DRA, respectively. CNN-SAD and DL-DRA increase the recall at the cost of increasing the misjudgment. As shown in Table 8, the precision of CNN-SAD is 0.024 larger than that of the proposed approach. However, the recall of CNN-SAD is 0.166 less than that of the proposed approach. The proposed approach has better comprehensive detection performance.

As shown in Table 8, both RF-L and RF-A have high recall and AUC. One of the main reasons for this phenomenon is that ensemble learning is used to generate a number of classifiers for the detection in these methods. Therefore, we may conclude that the reasonable use of ensemble learning can improve the detection performance, effectively.

One main difference among the detection results of the three datasets is the precision. As shown in Table 7, on the large-scale dataset the precision of a certain number of approaches is greater than 0.9. Attack profiles injected into large-scale dataset are easier to be identified than these injected into medium-scale dataset as shown in Table 6. One possible reason for this phenomenon is that the random strategy of selecting rating items used in the attack models makes attack profiles more unusual on large-scale dataset, since genuine users may focus on a segment with certain interests. Due to the diversity and complexity of real attack profiles, the precision of all approaches is less than 0.9 as shown in Table 8. Further researches are needed on this type of real datasets.

To set the optimal hyper parameters for different datasets, we require finding them carefully as shown in Section 4.2. The proposed approach cannot find the optimal hyper parameters, automatically. This is where the proposed approach requires further improvement.

5. Conclusions and future work

In this paper, a semi-supervised approach SSADR-CoF is proposed based on the Co-Forest algorithm for detecting the recommendation attack. An effective feature extraction method is proposed through dividing windows and employing rating behavior statistical methods. This method can effectively extract a series of features to capture the user rating behavior mode for the detection. A semi-supervised detection algorithm is proposed for detecting the recommendation attack based on Co-Forest algorithm. This algorithm can effectively improve the detection performance of semi-supervised methods by using the extracted features and ensemble learning. Experiments on three benchmark datasets demonstrate that the proposed approach has good detection performance with only using a small number of labeled user profiles and a certain number of unlabeled user profiles in the training set. In particular, on MovieLens 10M dataset, the recall, precision, and AUC are greater than 0.9. On MovieLens 25M and Amazon datasets, the proposed approach achieves the same detection performance as the state-of-the-art methods. On all the three datasets, the proposed approach significantly improves the precision of the semi-supervised methods under the condition of maintaining high recall and AUC.

In the future, it will be interesting to research the automatic optimization methods for the hyper parameters used in the proposed approach.

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.

CRediT authorship contribution statement

Quanqiang Zhou: Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Visualization, Supervision. **Liangliang Duan:** Software, Validation, Formal analysis, Investigation, Resources, Data curation, Project administration, Funding acquisition.

Acknowledgements

This work was supported by the Shandong Provincial Natural Science Foundation of China (Nos. ZR2020QF044 and ZR2019PF019).

REFERENCES

Angluin D, Laird P. Learning from noisy examples. Mach. Learn. 1988;2(4):343–70. doi:10.1023/A:1022873112823.

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.

Burke R, Mobasher B, Zabicki R, Bhaumik R. Identifying attack models for secure recommendation. In: Beyond Personalization: A Workshop on the Next Generation of Recommender Systems; 2005. p. 347–61. Retrieved from https://www.aminer.cn/pub/53e99cd2b7602d9702586674/beyond-personalization-the-next-stage-of-recommender-systems-research

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. DOI: 10.1145/1097047.1097061

Chung CY, Hsu PY, Huang SH. βp: a novel approach to filter out malicious rating profiles from recommender systems. Decis. Support Syst. 2013;55(1):314–25. doi:10.1016/j.dss.2013.01.020.

Ebrahimian M, Kashef R. Detecting shilling attacks using hybrid deep learning models. Symmetry 2020;12(11):1805. doi:10.3390/sym12111805.

Fawcett T. An introduction to ROC analysis. Pattern Recognit. Lett. 2006;27(8):861–74. doi:10.1016/j.patrec.2005.10.010.

Hand DJ, Till RJ. A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach. Learn. 2001;45(2):171–86. doi:10.1023/A:1010920819831.

Harper FM, Konstan JA. The movielens datasets: history and context. ACM Trans. Interact. Intell.Syst. 2015;5(4). doi:10.1145/2827872. 19:1–19:20

Lam SK, Riedl J. Shilling recommender systems for fun and profit. In: Proceedings of the 13th International Conference on World Wide Web; 2004. p. 393–402. DOI: 10.1145/988672.988726

Lee JS, Zhu D. Shilling attack detection-a new approach for a trustworthy recommender system. INFORMS J. Comput. 2012;24(1):117–31. doi:10.1287/ijoc.1100.0440.

Leo B. Random forests. Mach. Learn. 2001;45(1):5–32. doi:10.1023/A:1010933404324.

Li M, Zhou ZH. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Trans. Syst. Man Cybern.- Part A 2007;37(6):1088–98. doi:10.1109/TSMCA.2007.904745.

Mahony MO, Hurley N, Kushmerick N, Silvestre G. Collaborative recommendation: a robustness analysis. ACM Trans. Internet Technol. 2004;4(4):344–77. doi:10.1145/1031114.1031116.

Mehta B, Hofmann T, Fankhauser P. Lies and propaganda: detecting spam users in collaborative filtering. In: Proceedings of the 12th International Conference on Intelligent User Interfaces; 2007. p. 14–21.

Mehta B, Nejdl W. Unsupervised strategies for shilling detection and robust collaborative filtering. User Model. User-Adapted Interact. 2009;19(1–2):65–79. doi:10.1007/s11257-008-9050-4.

Meng S, Gao Z, Li Q, Wang H, Dai H, Qi L. Security-driven hybrid collaborative recommendation method for cloud-based IoT services. Comput. Secur. 2020. doi:10.1016/j.cose.2020.101950.

Si M, Li Q. Shilling attacks against collaborative recommender systems: a review. Artif. Intell. Rev. 2020;53:291–319. doi:10.1007/s10462-018-9655-x.

Tong C, Yin X, Li J, Zhu T, Lv R, Sun L, Rodrigues J. A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network. Comput. J. 2018;61(7):949–58. doi:10.1093/comjnl/bxy008.

Vivekanandan K, Praveena N. Hybrid convolutional neural network (CNN) and long-short term memory (LSTM) based deep learning model for detecting shilling attack in the social-aware network. J. Ambient Intell. Humanized Comput. 2020. doi:10.1007/s12652-020-02164-y.

Williams CA, Mobasher B, Burke R. Defending recommender systems: detection of profile injection attacks. Serv. Oriented Comput. Appl. 2007;1(3):157–70. doi:10.1007/s11761-007-0013-0.

Wu Z, Gao J, Mao B, Wang Y. Semi-SAD: applying semi-supervised learning to shilling attack detection. In: Proceedings of the 5th ACM Conference on Recommender Systems; 2011. p. 289–92.

Wu Z, Wu J, Cao J, Tao D. HySAD: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2012. p. 985–93.

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. DOI: 10.1145/2505515.2505700

Xu Y, Zhang F. Detecting shilling attacks in social recommender systems based on time series analysis and trust features. Knowl.-Based Syst. 2019;178:25–47. doi:10.1016/j.knosys.2019.04.012.

Yang Z, Sun Q, Zhang Y, Zhang B. Uncovering anomalous rating behaviors for rating systems. Neurocomputing 2018;308:205–26. doi:10.1016/j.neucom.2018.05.001.

Yang Z, Xu L, Cai Z. Re-scale AdaBoost for attack detection in collaborative filtering recommender systems. Knowl.-Based Syst. 2016;100:74–88. doi:10.1016/j.knosys.2016.02.008.

Zhang S, Ouyang Y, Ford J, Makedon F. Analysis of a low-dimensional linear model under recommendation attacks. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; 2006. p. 517–24. DOI: 10.1145/1148170.1148259

Zhang Z, Zhang Z, Zhang P, Wang S. UD-HMM: an unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering. Knowl.-Based Syst. 2018;148:146–66. doi:10.1016/j.knosys.2018.02.032.

Zhou Q, Wu J, Duan L. Recommendation attack detection based on deep learning. J. Inf. Secur. Appl. 2020;52:102493. doi:10.1016/j.jisa.2020.102493.

Zhou W, Wen J, Xiong Q, Gao M, Zeng J. SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems. Neurocomputing 2016;210:197–205. doi:10.1016/j.neucom.2015.12.137.

Quanqiang Zhou received his Ph.D. degree in School of Information Science and Engineering from Yanshan University, China, in 2014. Now, he is an associate professor of School of Information and Control Engineering, Qingdao University of Technology, Qingdao, Shandong Province, China. He has published papers in international journals such as Knowledge-Based Systems, Journal of Information Security and Applications, and IET Information Security. His research interests include recommender systems, machine learning, and information security.

Liangliang Duan received his Ph.D. degree in School of Information Science and Engineering from Yanshan University, China, in 2016. Now, he is a lecturer of School of Information and Control Engineering, Qingdao University of Technology, Qingdao, Shandong Province, China. His research interests include machine learning and deep learning.