Research Article

Multiview Ensemble Method for Detecting Shilling Attacks in Collaborative Recommender Systems

Yaojun Hao ^{1,2,3}, Peng Zhang,⁴ and Fuzhi Zhang ^{1,2}

¹School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei Province, China

²The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao, Hebei Province, China

³Department of Computer Science and Technology, Xinzhou Teachers University, Xinzhou, Shanxi Province, China

⁴School of Software Engineering, Beijing Jiaotong University, Beijing, China

Correspondence should be addressed to Fuzhi Zhang; xjzfz@ysu.edu.cn

Received 10 May 2018; Revised 8 August 2018; Accepted 19 September 2018; Published 11 October 2018

Academic Editor: Jesús Díaz-Verdejo

Copyright © 2018 Yaojun Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Faced with the evolving attacks in collaborative recommender systems, the conventional shilling detection methods rely mainly on one kind of user-generated information (i.e., single view) such as rating values, rating time, and item popularity. However, these methods often suffer from poor precision when detecting different attacks due to ignoring other potentially relevant information. To address this limitation, in this paper we propose a multiview ensemble method to detect shilling attacks in collaborative recommender systems. Firstly, we extract 17 user features by considering the temporal effects of item popularity and rating values in different popular item sets. Secondly, we devise a multiview ensemble detection framework by integrating base classifiers from different classification views. Particularly, we use a feature set partition algorithm to divide the features into several subsets to construct multiple optimal classification views. We introduce a repartition strategy to increase the diversity of views and reduce the influence of feature order. Finally, the experimental results on the Netflix and Amazon review datasets indicate that the proposed method has better performance than benchmark methods when detecting various synthetic attacks and real-world attacks.

1. Introduction

Collaborative recommender systems are widely used in e-commerce websites to handle the information overload problem by providing personalized recommendations for their users. However, due to the openness of such systems, the attackers are likely to inject a large number of fake profiles in order to increase/decrease the recommendation frequency of particular items (e.g., movies and products). This behaviour is often referred to as shilling attacks or profile injection attacks. According to the purpose of attacks, shilling attacks can be categorized as either push attacks or nuke attacks, which promote or demote a particular item to be recommended. The fake profiles are called attack profiles or shilling profiles, which have a negative impact on the prediction quality of collaborative recommender system and make the users lose trust in the system. According to the experimental observations in practical systems, an attack with 3% shilling profiles would result in a prediction shift of around 1.5 points

on a five-point scale [1]. Therefore, in the face of shilling attacks, how to ensure the recommendation quality of system has become a problem that cannot be ignored in the research of recommender system.

To defend collaborative recommender systems, a number of methods have been put forward to detect the shilling profiles and most of them mainly rely on one kind of user-generated information (i.e., single view) such as rating values [2–18], rating time [19–23], and item popularity [24–28]. Due to the diversity and variability of attackers' strategies, it is very difficult to fully characterize the shilling profiles from single view information [29], and many conventional detection methods suffer from poor precision when detecting different types of attacks [30].

By integrating a set of classifiers, some ensemble detection methods have further improved detection precision. However, the base classifiers are trained in the same feature space and the detection precision cannot meet the actual

needs, especially for detecting attacks with low filler sizes and attack sizes. Obviously, not all the features are relevant and effective when detecting different types of attacks. This means that the correlated errors between base classifiers cannot be adequately reduced by traditional ensemble methods in the same feature space [31, 32].

To tackle the aforementioned challenges, we extract the user features by considering the temporal effects of item popularity and rating values. These features can offer us a more comprehensive perspective for the detection of shilling profiles. Moreover, we use an optimal feature set partition algorithm to divide user features into several subsets in order to construct multiple optimal classification views. Since the ensemble method has the potential to improve the detection performance, we design a multiview ensemble detection framework, called MV-EDM, which integrates multiple classification views and base classifiers into a classification model to detect various shilling profiles.

The main contributions of the paper are summarized as follows:

(1) By analysing the item popularity from timestamp, we define the item temporal popularity and deal with it using the wavelet transform method. Based on it, we construct the temporal popularity vector of user ratings and extract 5 user features.

(2) From the observation of item rating mean changing with time, 2 user features are extracted based on dynamic mean of item ratings. Moreover, we also extract 10 user features by analysing item temporal popularity and dynamic mean of item ratings in different popular item sets.

(3) With the repartition strategy, base classifiers can be trained not only from different classification views but also from different partitioning results. Based on the qualitatively different classifiers, we propose a multiview ensemble detection framework. Experiments on the Netflix and Amazon review datasets show that the proposed method can effectively detect various synthetic attacks and real-world attacks.

The remaining parts of this paper are organized as follows. In Section 2, we introduce the background and related work on shilling attack detection. Section 3 presents the proposed method in detail. In Section 4, we present and discuss the experimental results. Finally, we conclude the paper.

2. Background and Related Work

2.1. Shilling Attack Models. Typically, an attack is realized by inserting several shilling profiles into a recommender system database to cause bias on selected target items [30]. A shilling profile can be defined as four sets of items [3–6]. Initially, a set of items, I_S , is the set of selected items to form the characteristics of the attack. Also, another set of items, I_F , is the set of filler items usually chosen randomly to obstruct detection [30]. A unique item i_t is the target item, and I_ϕ is the set of unrated items. Table 1 illustrates the common attack models in the literature.

Random attack and average attack are basic attack models, which generate shilling profiles with ratings to randomly

chosen empty cells around system overall mean or around each item's mean, respectively [3–6].

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 [15].

User shifting and target shifting strategies are used to obfuscate the basic attacks to evade detection [5]. User shifting is designed to reduce the similarity between shilling profiles. The goal of target shifting is to reduce the extreme ratings of shilling profiles [30]. User shifting attacks include user random shifting attack and user average shifting attack. Target shifting attacks include target random shifting attack and target average shifting attack. To facilitate the representation, we define the shifting attack as a collection with the same number of user random shifting attack, user average shifting attack, target random shifting attack, and target average shifting attack.

Power items and power users are able to influence the largest group of items and users, respectively [33, 34]. In power item attack and power user attack, the strategies for selecting power items and power users include the top-N highest aggregation similarity scores, in-degree centrality, and the highest number of ratings [33, 34].

The above attacks can be used as either push attacks or nuke attacks. However, love/hate attack and reverse bandwagon attack are specially used for nuke attacks. Especially, love/hate attack is an extremely effective nuke attack in which randomly chosen filler items are rated with the highest possible rating while the target item is given the lowest one in shilling profiles [30].

When the shilling profiles are generated by the attack models and injected into the system, in the reported experimental results, 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 [12].

2.2. Related Work. In collaborative recommender systems, shilling attack detection has attracted much attention. To distinguish shilling profiles from genuine ones, the existing detection methods mainly rely on single view information, such as rating values [2–18], rating time [19–23], and item popularity [24–28].

Based on the rating values, machine learning methods are primarily used to detect shilling profiles, which include supervised and unsupervised methods [2–18]. In supervised methods, Williams et al. [6] proposed several features based on user ratings and trained three classifiers to detect shilling profiles. Wu et al. [7, 8] proposed algorithms to select the effective features and utilized the supervised and semisupervised classifiers to detect attacks. Zhou [9] proposed a supervised approach for detecting AoP attacks, in which the term frequency-inverse document frequency was used to extract the features. Zhou et al. [10] proposed a two-phase shilling attack detection method, SVM-TIA. This method first uses SVM-based shilling detection technique to obtain a set of suspicious profiles and then applies target item analysis

TABLE I: Common attack models.

Attack model	I_s Items	Rating	I_F Items	Rating	i_t 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	Ratings with normal distribution around item mean and standard deviation			$r_{\max}/r_{\min}$
Power user attack	Copy of the power user's items and ratings				$r_{\max}/r_{\min}$
Bandwagon attack	Popular items	$r_{\max}$	Randomly chosen	System mean	$r_{\max}$
Love/Hate attack	Not used		Randomly chosen	$r_{\max}$	$r_{\min}$
Reverse Bandwagon attack	Unpopular items	$r_{\max}$	Randomly chosen	System mean	$r_{\min}$

method to remove genuine profiles from this set in order to reduce the false positives of detection results. Yang et al. [11] proposed 3 user ratings-based features and presented a supervised detection method based on 18 features, which used a rescale AdaBoost method to perform classification. Yang et al. [12] formulated the problem as finding a mapping model between rating behaviour and item distribution and developed a detection method by combining max-margin matrix factorization with Bayesian nonparametrics. In unsupervised methods, Bhaumik et al. [13] applied k-means clustering approach for attack detection based on several classification attributes. Hurley et al. [14] used Neyman-Pearson statistical detection to identify shilling profiles and proposed statistical detectors for the standard attacks and AoP attack. Mehta et al. [15] proposed a variable selection method based on principal component analysis (PCA) to filter out the attacks. This method is effective in detecting standard attacks, but it performs poorly in detecting AoP attack. Moreover, it needs to know the attack size in advance. Yang et al. [16] utilized an adaptive structure learning method to select features of user and item. Based on the selected features, they proposed a two-stage detection method. In [17], a soft coclustering with propensity similarity model was presented to detect shilling attacks. Unfortunately, the above methods assume that the signature or distribution for rating values of shilling profiles will differ significantly from those of genuine profiles. However, this hypothesis does not always hold because the attackers are likely to look for ways to beat the detector by fabricating the rating values. Therefore, the detection methods based only on rating values suffer from poor precision in detecting attacks, especially for detecting attacks with low filler sizes and attack sizes.

Based on rating time information, some detectors are proposed. Tang and Tang [19] used the span, frequency, and mount factor based on user rating time intervals to prevent the attacked item from the top-N list. Zhang et al. [20] presented a detector by calculating the mean and entropy of the samples in time windows when rating time series was constructed. Xia et al. [21] dynamically divided time intervals and used hypothesis test detection-based framework to identify the anomaly items. Zhou et al. [22] reorganized each item ratings by time series to examine rating segments and then used statistical metrics and target item analysis to detect the anomaly items. However, these methods assume that the attacks are injected in short periods, which cannot effectively detect the long-term decentralized attacks. Neal et al. [23] monitored the ratings at any time and detected the attacks according to statistical attributes of all the ratings, user ratings, and item ratings. Nevertheless, this method assumes that the shilling profiles have constant attack parameters, which is difficult to meet in practice.

Based on item popularity information, Zhang et al. [24] constructed a rating series from item novelty and used Hilbert-Huang transform method to extract user features to detect the attacks. In [25], the user features were extracted by mutual information and statistical methods based on the item popularity series, then C4.5-based ensemble classification was used to detect the attacks. Karthikeyan et al. [26] utilized discrete wavelet transform to get the feature set based on

the rating series of items' popularity and novelty. And then the user features were used by SVM for detection. Li et al. [27, 28] extracted the user features according to the item popularity distributions. Unfortunately, in these methods, the item popularity is simply computed and observed as a static value, which can be easily influenced by noise and manipulated by attacks.

In practice, the shilling profiles are usually far fewer than genuine ones, and the detection method based on supervised classification can be regarded as the imbalanced classification problem [11]. A large number of experiment results [2–10, 26–28] showed that the single classifier became worse in low filler sizes and attack sizes, so ensemble methods were used to alleviate such problems. Zhang et al. [18, 25] used ensemble methods to detect the attacks based on the bagging method. Yang et al. [11] used the improved rescale AdaBoost method to detect the attacks. However, in these ensemble methods, the base classifiers were trained in the same feature space without consideration of the feature's effectiveness. This means that the correlated errors between the base classifiers cannot be adequately reduced.

In [26], a discrete wavelet transform method was used to extract user features in offline training and online detection, which would consume a lot of running time. However, in this paper, we use discrete wavelet transform method only once for the item time series in the offline preprocessing.

In [7, 8], the feature selection algorithms were proposed to estimate the quality of ratings-based feature to distinguish the user profiles that were near to each other. And then a single classifier was used to detect shilling profiles. In [16], the features of user and item were selected by using adaptive structure learning, and a two-stage detection method was proposed. These methods focused on the feature's ability to determinate neighborhood relationship. However, in this paper, we select the effective features by directly evaluating their detection performance and further construct multiple optimal classification views.

In [35], to construct the classification views, the feature set is divided only once. Obviously, the constructed views were affected by the initial order of features and the multiple view ensemble method was only integrated from the fixed views. Therefore, the ensemble result is likely to be unstable. Unlike the work in [35], to construct the views with great diversity and reduce the influence of feature order, we optimally repartition the proposed features at regular intervals based on the random order of features to obtain the stable ensemble results in MV-EDM. Therefore, we integrate base classifiers not only from different views, but also from different partitioning results in MV-EDM.

3. The Proposed Method

The framework of our method is depicted in Figure 1, which consists of four stages: feature extraction, view construction, base classifier generation, and detection. At the stage of feature extraction, the features of users in training and test sets are extracted by using the proposed feature extraction method. At the stage of view construction, the user features

FIGURE 1: The framework of ensemble detection model.

are partitioned into some optimal feature subsets to construct multiple views. At the stage of base classifier generation, the kNN-based classifiers are trained on the training sets from different views and different partition results. At the stage of detection, the base classifiers are integrated to detect the attacks. The final predictive result is obtained by weight voting on the base classifiers and views.

To facilitate the discussions, in Table 2 we give the descriptions of notations used in this paper.

3.1. Feature Extraction. In collaborative recommender systems, the genuine users rate items according to their preferences, which can be represented by the rating values and item temporal popularity at rating time. By contrast, the attackers rate items to cause bias on selected target items [30]. They pay more attention to fabricating sophisticated rating values to manipulate the system's output. Thus, most existing detection methods are devoted to finding the rating differences between genuine and shilling profiles. However, the attackers are likely to imitate the ratings of genuine users to evade the detection. Therefore, the attackers should be tracked by fusing other information into ratings, such as rating time and item popularity.

With consideration of the temporal effects for item popularity, the related definitions are presented in Section 3.1.1. Then, the discrete wavelet transform method is introduced to deal with the item temporal popularity in Section 3.1.2. Based on the above preprocessing, 5 user features are extracted in Section 3.1.3. From the observation that the mean of item ratings changes with time, 2 user features are extracted in Section 3.1.4. By analysing item temporal popularity and dynamic mean of item ratings in different popular item sets, 10 user features are extracted in Section 3.1.5.

3.1.1. Definitions

Definition 1. Time Interval (TI). The time interval refers to the time bin [20], which is obtained by partitioning the timeline from the beginning time to the detection time with the same width.

When the timeline from the beginning time S to the detection time is partitioned with width T , the j th time interval TI_j can be denoted by the following.

$$TI_j = (S + (j - 1) * T, S + j * T) \quad (1)$$

Definition 2. Popularity of Item (PI). The popularity of item refers to the popular degree of the item [24], which is

TABLE 2: Notations and their descriptions.

Notation	Description
U	set of the entire users
U_g	set of genuine users in the training set
I	set of the entire items
I_u	set of items rated by user u
D	rating database, including UserID, ItemID, ratings, and timestamp.
$r_{u,i,t}$	rating of user u to item i at t moment
$r_{u,i,t} \neq \perp$	item i is rated by user u at t moment
$r_{u,i,t} = \perp$	item i is not rated by user u at t moment
$ \cdot $	the cardinality of the set
$\Gamma(r_{u,i,t})$	discriminate function, if $r_{u,i,t} \neq \perp$, the value is 1; otherwise the value is 0

measured by the number of item ratings given by the genuine users. The PI of item i can be computed as follows.

$$PI_i = \sum_{u \in U_g} \Gamma(r_{u,i,t}) \quad (2)$$

Definition 3. Temporal Popularity of Item (TPI). The temporal popularity of item refers to the popular degree of the item in different time intervals, which is measured by the number of item ratings given by the genuine users during different time intervals. The TPI of item i in time interval TI_j can be computed as follows.

$$TPI_{i, TI_j} = \sum_{t \in TI_j, u \in U_g} \Gamma(r_{u,i,t}), \quad j = 1, 2 \dots N \quad (3)$$

Definition 4. Novelty of Item (NI). The novelty of item refers to the degree of difference between the item and other items [36], which is computed as follows:

$$NI_i = \frac{1}{|U_i|} \sum_{u \in U_i, r_{u,i} \neq 0} NI_{u,i} \quad (4)$$

where $NI_{u,i}$ denotes the novelty of item i to user u , which is computed as follows [36]:

$$NI_{u,i} = \frac{1}{|I_u|} \sum_{j \in I_u, r_{u,j} \neq 0} (1 - \text{sim}(i, j)) \quad (5)$$

where $\text{sim}(i, j)$ denotes the similarity between item i and item j , which is computed as follows.

$$\text{sim}(i, j) = \frac{\sum_{u \in U_g} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U_g} r_{u,i}^2} \sqrt{\sum_{u \in U_g} r_{u,j}^2}} \quad (6)$$

In general, the item with greater novelty is less popular.

Definition 5. User Rating Temporal Popularity Vector ($URTPV$). The user rating temporal popularity vector refers to the vector constituted by TPI based on the item novelty series in the user profile. The $URTPV$ of user u can be written as follows:

$$URTPV_u = (urtpv_{u,1}, urtpv_{u,2}, \dots, urtpv_{u,n}) \quad (7)$$

where each component of $URTPV_u$, $urtpv_{u,i}$, can be computed as follows.

$$urtpv_{u,i} = \Gamma(r_{u,i,t}) \cdot TPI_{i, TI_j}, \quad t \in TI_j \quad (8)$$

Figure 2 illustrates the TPI differences between genuine and shilling profiles in the Netflix dataset. The horizontal axis represents the item novelty series in ascending order, and the vertical axis represents the TPI of ratings. In Figure 2, 10 genuine user profiles are randomly selected from the Netflix dataset. 10 shilling profiles are generated by random and average attack models, respectively.

As shown in Figure 2, most ratings of genuine profiles are concentrated on the left side of horizontal axis. This indicates that the genuine users are likely to rate the popular items, which is consistent with the observation in [28]. By contrast, the ratings of shilling profiles are uniformly distributed along the horizontal axis. This is because the filler items are randomly selected in random and average attack models. Also, when these shilling profiles are injected at attack time, there are different distributions of TPI between genuine and shilling profiles.

3.1.2. Wavelet Decomposition of TPI Series. The TPI offers us valuable insights into the user profiles. However, if we directly measure TPI by counting the number of item ratings on a discrete time series, TPI will be disturbed by many factors such as noises, burstiness of item [37], resulting in a nonstationary time series.

Figure 3(a) illustrates the original TPI signal of an item, which is randomly selected from the Netflix dataset. The horizontal axis represents the rating days from January 6, 2000, and the vertical axis represents the TPI .

As shown in Figure 3(a), there is high-frequency information in the original signal, i.e., the item popularity fluctuates violently with time.

According to the theory of wavelet analysis, the time series signal can be decomposed into different frequency channels, in which the signals with lower complexity than the original signal can be obtained and the noise-like high-frequency patterns can be filtered out. Therefore, the discrete wavelet transform is used to obtain the stable trend of TPI in time series.

FIGURE 2: The *TPI* difference between genuine and shilling profiles. FIGURE 3: The original signal and reconstruction signal of *TPI* for an item.

In general, the higher the complexity of the time series is, the more layers should be obtained from the decomposition [38]. In this paper, the original *TPI* signal is decomposed into 6 layers based on the daily intervals in the Netflix dataset and Amazon review dataset. Because we aim to obtain the stable popularity trends of the item, we are mainly interested in the slowest dynamics of original signal and reconstruct the signal at the 6th layer after the original signal is decomposed in six levels.

The reconstruction signal of *TPI* is shown in Figure 3(b); compared with Figure 3(a), the reconstruction signal becomes smooth after the transformation of wavelet. Thus, we adopt the reconstruction signal to reflect the main and stable trends of the *TPI* changes.

3.1.3. Extracting Features from User Rating Temporal Popularity Vector. In genuine profiles, each item is rated with the temporal tastes for this item according to the preferences of genuine users. By contrast, in shilling profiles, the items are selected and rated by attackers according to specific strategy. Therefore, the information transformed by *URTPV* is different between genuine and shilling profiles. Also, due to the different *TPI* distributions in *URTPV* of genuine and shilling profiles, we can use statistical methods to extract the detection features.

(1) Information Entropy of User Rating Temporal Popularity Vector (*IE_URTPV*). The sample space formed by the components of *URTPV*, $\{Y = urtpv_{u,i}\}$, can be regarded as a random

process, and $E(Y_1, \dots, Y_n)$ denotes the information entropy of random process, which can be calculated as follows:

$$E(Y_1, \dots, Y_n) = - \sum_{y_1, \dots, y_n} P(y_1, \dots, y_n) \log P(y_1, \dots, y_n) \quad (9)$$

where $P(y_1, \dots, y_n)$ denotes the joint probability $P(Y_1 = y_1, \dots, Y_n = y_n)$.

The empirical probability is used to evaluate $P(Y_1 = y_1, \dots, Y_n = y_n)$; i.e., the components of *URTPV* are divided into Q bins between the minimum and maximum values, and then the probabilities are determined by the proportions of bin number in all data [39].

(2) Corrected Conditional Entropy of User Rating Temporal Popularity Vector (*CCE_URTPV*). Based on the item novelty series, the sample space formed by the components of *URTPV*, $\{Y = urtpv_{u,i}\}$, can be regarded as a random process, in which the previous state affects the latter one. The conditional entropy is often used to measure the complexity of random process. Given the previous state, the conditional entropy can be calculated as follows:

$$CE(Y_m | Y_1, \dots, Y_{m-1}) = E(Y_1, \dots, Y_m) - E(Y_1, \dots, Y_{m-1}) \quad (10)$$

where $E(Y_1, \dots, Y_m)$ denotes the entropy at a state, and it can be calculated by (9).

However, in recommender systems, most users are inclined to rate a small amount of items; that is, the random process $\{Y = urtpv_{u,i}\}$ is the short series. Thus, we use *CCE* (the corrected conditional entropy) to measure the complexity of the short series, which can be calculated as follows [40]:

$$\begin{aligned} CCE(Y_m | Y_1, \dots, Y_{m-1}) \\ = CE(Y_m | Y_1, \dots, Y_{m-1}) + perc(Y_m) EN(Y_1) \end{aligned} \quad (11)$$

where $perc(Y_m)$ is the percentage of unique sequences of length m and $EN(Y_1)$ is the entropy estimated value with m fixed at 1. The *CCE* minimum over different states is used to quantify the regularity level of random process, which can be defined as follows [40].

$$CCE_URTPV_u = \min(CCE(Y_m | Y_1, \dots, Y_{m-1})) \quad (12)$$

(3) *Range of User Rating Temporal Popularity Vector (R_URTPV)*. The *R_URTPV* represents the difference between the maximum and minimum values of the components in *URTPV*, which can be calculated as follows.

$$R_URTPV_u = \max(URTPV_u) - \min(URTPV_u) \quad (13)$$

(4) *Mean of User Rating Temporal Popularity Vector (M_URTPV)*. The *M_URTPV* is used to measure the overall statistical characteristics of *URTPV*, which can be calculated as follows.

$$M_URTPV_u = \frac{\sum_{i \in I_u} urtpv_{u,i}}{|I_u|} \quad (14)$$

(5) *Variance of User Rating Temporal Popularity Vector (V_URTPV)*. The *V_URTPV* is used to measure the fluctuation degree of *URTPV*, which can be calculated as follows.

$$V_URTPV_u = \frac{1}{|I_u|} \sum_{i \in I_u} (urtpv_{u,i} - M_URTPV_u)^2 \quad (15)$$

To intuitively demonstrate the effects of the above detection features, we take 500 genuine profiles and 480 shilling profiles as examples and illustrate their differences on detection features in Figure 4. The genuine profiles are randomly selected from the Netflix contest dataset. The shilling profiles are generated by random, average, AoP, shifting, power item, power user, love/hate, and reverse bandwagon attack models with {3%, 5%} filler sizes. We randomly select 60 shilling profiles from each type of attacks and finally obtain 480 shilling profiles.

As shown in Figure 4, the features of genuine profiles differ greatly from those of shilling profiles. Take Figure 4(b) for an example, most of the *CCE_URTPV* values in genuine profiles are greater than 2. By contrast, most of the *CCE_URTPV* values in shilling profiles are below 2 except for power item attack. This means that the rating behaviours of genuine profiles are more complicated than those of shilling profiles. Take Figure 4(d) as another example, there exists relatively high mean in *URTPV* of genuine profiles. This indicates that the ratings of genuine profiles have relatively high *TPI* from statistics view. Therefore, based on the definition of *TPI*, the extracted features can be used to separate the shilling profiles from genuine ones.

3.1.4. Extracting Features from Dynamic Mean of Item Ratings. In the previous studies, some generic features are designed to find the differences between genuine and shilling profiles in rating deviations, such as *RDMA* (rating deviation from mean agreement) [4], *WDMA* (weighted deviation from mean agreement) [5], in which the mean of item ratings is a fixed value and is the base to measure the deviations. However, the mean of item ratings is affected by many factors and changes with time [41]. Therefore, when the item is rated with item mean or system mean in shilling profiles according to the attack models in Table 1, the rating deviations in *RDMA* and *WDMA* can be further expanded by using the dynamic mean of item ratings.

(1) *Rating Deviation from Dynamic Mean Agreement (RDDMA)*

$$RDDMA_u = \frac{1}{|I_u|} \sum_{i \in I_u} \frac{|r_{u,i,t} - \bar{r}_{i,TI}|}{|U_i|} \quad (16)$$

where $\bar{r}_{i,TI}$ denotes the mean of ratings for item i in time interval TI .

(2) *Weighted Deviation from Dynamic Mean Agreement (WDDMA)*

$$WDDMA_u = \frac{1}{|I_u|} \sum_{i \in I_u} \frac{|r_{u,i,t} - \bar{r}_{i,TI}|}{|U_i|^2} \quad (17)$$

Figure 5 illustrates the comparisons of *RDMA* and *RDDMA*, *WDMA* and *WDDMA* for genuine and shilling profiles, where the genuine and shilling profiles are the same as those in Figure 4.

As shown in Figure 5, *RDDMA* values of shilling profiles in Figure 5(b) are greater than those of genuine profiles. Moreover, the difference is more obvious than that of *RDMA* in Figure 5(a). Similar results can be found from the comparison of *WDDMA* and *WDMA*. These results intuitively demonstrate that the distinguishing abilities of *RDMA* and *WDMA* have been, respectively, improved in *RDDMA* and *WDDMA* by using the dynamic mean of item ratings. Furthermore, we will also quantitatively evaluate the performance of these features in Section 4.3.4.

3.1.5. Extracting Features from Different Novelty Item Sets. The genuine users have obvious preferences for different novelty items. However, the attackers hardly transfer their preferences by selecting different novelty items. For example, as listed in Table 1, the filler items are randomly chosen in average attack, random attack, user random shifting attack, user average shifting attack, target random shifting attack, target average shifting attack, love/hate attack, bandwagon attack, and reverse bandwagon attack.

To reflect the user's preferences to choose the different novelty items, based on the Zipf's law, the items are divided into popular and novel item sets according to their novelty. Sorted by their novelty in ascending order, the top 20% items are taken as popular item set (*PIS*) and the remaining items

FIGURE 4: Continued.

FIGURE 4: The feature differences between the genuine and shilling profiles.

are used as novel item set (*NIS*). Based on the features in Sections 3.1.3 and 3.1.4, the user features can be extracted from *PIS* and *NIS*, respectively, which are shown in Tables 3 and 4. Figures 6 and 7 illustrate the effects of these detection features, where the genuine and shilling profiles are the same as those in Figure 4.

As shown in Figures 6 and 7, these features have different discrimination abilities for different attacks. For example, *IE_URTPVP* and *M_URN* have the ability to detect power item attack and love/hate attack, respectively. However, they cannot effectively detect other attacks. For *IE_URTPVP*, when the power item selection method is based on the number of ratings in our experiments, most of the rated items are in popular item set. Therefore, with the uncertainty distribution of *TPI*, the profiles of power item attacks have relatively high *IE_URTPVP* in popular item set. For *M_URN*, because love/hate attacks randomly select items and give them the maximum rating values, the profiles of love/hate have the highest mean of user ratings in *NIS*.

Let D denote the rating database, M denote the user feature matrix, US denote the rating timestamp matrix, UT denote the user rating temporal popularity matrix, and UR denote the user rating value matrix. The proposed feature extraction algorithm is described as follows.

Algorithm 1 contains two parts. The first part (lines 1-14) is to preprocess the rating data. Particularly, the UR and US matrices are obtained from the raw rating database D (lines 2-3). Then the item novelty series is constructed to divide *PIS* and *NIS* (lines 4-7), based on which, the wavelet decomposition is used to determine the

item daily popularity (lines 8-14). The second part (lines 15-19) is to convert the raw rating database D to the user rating temporal popularity matrix UT (line 15) and extract the user features (lines 16-18). Finally, the user feature matrix is obtained (line 19).

3.2. Construction of Multiple Optimal Classification Views. As discussed in Section 3.1, the extracted features exhibit different abilities in detecting different attacks. Therefore, we have to select and combine those effective features for shilling attack detection. In feature selection, one main challenge is the trade-off between directly removing the irrelevant detection features and keeping useful features. For this reason, we adopt wrapper methods instead of the filter methods used in [7, 8, 16]. Moreover, the proposed features are extracted via constructing the corresponding relationship among item popularity, ratings, and timestamps, so they can be regarded as heterogeneous data with the same source. That is, with the same source of user behaviour data, there are multiple views (feature subsets) to be used for separating the shilling profiles.

Based on the above analysis, we propose an optimal feature set partitioning method to construct the views. To eliminate the impact of feature order as much as possible and construct the views with great diversities, the feature set is repartitioned with random order of features at regular intervals.

Suppose that the nonempty feature set $X = \{x_1, x_2, \dots, x_n\}$ can be partitioned into k feature subsets to construct the views. Let X_i ($i=1, 2, \dots, k$) denote the i th view

TABLE 3: The features in popular item set.

Feature	Formula	Formula#
Information entropy of $URPTV$ in PIS (IE_URTPVP)	$IE_URTPVP_u = - \sum_{i \in PIS} P(y_1, \dots, y_i) \log P(y_1, \dots, y_i)$	(18)
Mean of $URTPV$ in PIS (M_URTPVP)	$M_URTPVP_u = \frac{\sum_{i \in PIS} urtpv_{u,i}}{ I_u \cap PIS }$	(19)
Variance of $URTPV$ in PIS (V_URTPVP)	$V_URTPVP_u = \frac{1}{ I_u \cap PIS } \sum_{i \in PIS} (urtpv_{u,i} - M_URTPVP_u)^2$	(20)
Mean of user ratings in PIS (M_URP)	$M_URP_u = \frac{\sum_{i \in PIS} r_{u,i,t}}{ I_u \cap PIS }$	(21)
Variance of user ratings in PIS (V_URP)	$V_URP_u = \frac{1}{ I_u \cap PIS } \sum_{i \in PIS} (r_{u,i} - M_URP_u)^2$	(22)

TABLE 4: The features in novelty item set.

Feature	Formula	
Information entropy of $URTPV$ in NIS (IE_URTPVN)	$IE_URTPVN_u = - \sum_{i \in NIS} P(y_1, \dots, y_i) \log P(y_1, \dots, y_i)$	(23)
Mean of $URTPV$ in NIS (M_URTPVN)	$M_URTPVN_u = \frac{\sum_{i \in NIS} urtpv_{ui}}{ I_u \cap NIS }$	(24)
Variance of $URTPV$ in NIS (V_URTPVN)	$V_URTPVN = \frac{1}{ I_u \cap NIS } \sum_{i \in NIS} (urtpv_{ui} - M_URTPVN_u)^2$	(25)
Mean of user ratings in NIS (M_URN)	$M_URN_u = \frac{\sum_{i \in NIS} r_{ui,t}}{ I_u \cap NIS }$	(26)
Variance of user ratings in NIS (V_URN)	$V_URN_u = \frac{1}{ I_u \cap NIS } \sum_{i \in NIS} (r_{ui,t} - M_URN_u)^2$	(27)

FIGURE 5: Comparisons of *RDMA* and *RDDMA*, *WDMA* and *WDDMA*.

FIGURE 6: Continued.

FIGURE 6: The feature differences between the genuine and shilling profiles in popular item set.

in the feature set partitioning result. Let $f(X_i)$ denote the evaluation function for the classification result from view X_i . The optimal feature set partitioning method can be seen as the following optimization problem:

$$\begin{aligned}
 \max \quad & z = \sum_{i=1}^k f(X_i) \\
 \text{s.t.} \quad & |X_i| \geq 1, \quad i = 1, 2, \dots, k \\
 & f(X_i^m) - f(X_i^{m-1}) > 0, \\
 & i = 1, 2, \dots, k, \quad m = 2, 3, \dots, n \\
 & \sum_{j=1}^n \sum_{i=1}^k x_{j,i} \leq n
 \end{aligned} \tag{28}$$

where $|X_i|$ denotes the cardinality of view X_i and X_i^m denotes a subset of X_i with cardinality m (i.e., $X_i^m \in \{X_i \mid |X_i| = m\}$).

$x_{j,i}$ is the flag variable, which means feature x_j whether or not belonging to view X_i .

$$x_{j,i} = \begin{cases} 1 & x_j \in X_i \\ 0 & x_j \notin X_i \end{cases} \quad j = 1, 2, \dots, n, \quad i = 1, 2, \dots, k \tag{29}$$

Let $BsClassifier$ denote the base classifier, X be the feature set, k denote the number of views, $Vdata$ denote the validation dataset, and X_{opt} denote the feature set partitioning result. The proposed optimal feature set partitioning algorithm is described as Algorithm 2.

Algorithm 2 includes two parts. The first part (lines 1-9) is to initialize the views (line 5), accuracy of views (line 6), and accuracy difference of views (line 8). The second part (lines 10-24) is to generate the optimal feature set partition. Firstly, a distinct and unevaluated feature is randomly selected (line 11). Secondly, the selected item is temporarily added into each view (line 13), and the accuracy difference of each view is updated (lines 14-15). Thus, the feature is permanently added into the view with maximum accuracy difference (lines 18-19) and the accuracy of this view is updated (line 20). Finally, after each feature is temporarily added into each view to decide whether or not to be added permanently, the optimal feature set partitioning result is obtained and then returned (line 24).

3.3. Generation of Base Classifiers. In supervised ensemble method, a predictive classifier is generated by integrating multiple classifiers, which are trained on diverse training sets. Figure 8 illustrates the framework of our method to generate the base training sets. As shown in Figure 8, from the horizontal direction, the various base training sets are

FIGURE 7: Continued.

FIGURE 7: The feature differences between the genuine and shilling profiles in novelty item set.

```

Input:  $D$ 
Output:  $M$ 
1  Initializing  $M$ 
2   $UR = \text{conver}(D)$ 
3   $US = \text{converS}(D)$ 
4  if ( $PIS$  and  $NIS$  are not constructed) then
5      Calculate the novelty of items and construct the item novelty series
6      Construct the popular item set  $PIS$  and novel item set  $NIS$ 
7  end if
8  if ( $TPI$  is not calculated) then
9      Calculate the TPI of everyday
10     for each  $i \in I$  do
11         decompose TPI by DWT
12         using wavelet reconstruction to obtain TPI of everyday
13     end for
14 end if
15  $UT = \text{converT}(D)$ 
16 for each  $u \in U$  do
17     Calculate user features  $M_u$  according to formula (9),(12)-(27)
18 end for
19 Return  $M$ 

```

ALGORITHM 1: Feature extraction algorithm.

FIGURE 8: The framework to generate the base training sets.

Input: $Bsclassifier, X, k, Vdata$

Output: X_{opt}

```

1   $n \leftarrow \text{getsize}(X)$ 
2   $FA_i \leftarrow 0$ 
3  for  $i=1$  to  $k$  do
4    While  $FA_i \leq 0.5$  do
5       $X_i \leftarrow \text{getDistinctFea}(X, 1)$ 
6       $FA_i \leftarrow fclassifier(X_i, Vdata)$ 
7    end
8     $Fdiff_i \leftarrow 0$ 
9  end for
10 for  $fea=k+1$  to  $n$  do
11    $Xcur \leftarrow \text{getDistinctFea}(X, 1)$ 
12   for  $i=1$  to  $k$  do
13      $Xtemp \leftarrow X_i \cup Xcur$ 
14      $Fval_i \leftarrow fclassifier(Xtemp, Vdata)$ 
15      $Fdiff_i \leftarrow Fval_i - FA_i$ 
16   end for
17   if  $\max(Fdiff_i) > 0$  then
18      $t \leftarrow \arg \max(Fdiff_i)$ 
19      $X_t \leftarrow X_t \cup Xcur$ 
20      $FA_t \leftarrow Fval_t$ 
21   end if
22 end for
23  $X_{opt} \leftarrow \{X_1, X_2, \dots, X_k\}$ 
24 Return  $X_{opt}$ 

```

ALGORITHM 2: Optimal feature set partitioning algorithm.

generated according to different attack sizes and filler sizes. Furthermore, unlike the traditional ensemble methods, these base training sets are also partitioned by the constructed

views from the vertical direction. Because Algorithm 2 directly supports diversity of views through the order of features, we repartition the feature set at regular intervals in ensemble method and generate a group of optimal feature set partitioning results, such as feature set partition₁, feature set partition₂ in Figure 8.

In this paper, we select kNN as the base classifier due to the following considerations:

(1) Retraining cost: As a simple and nonparametric classification method, kNN classifier can learn from a small set of samples and incrementally add new types of attacks with lower retraining cost. This is very important for real-world applications.

(2) Computational cost: kNN is our first choice because its computational complexity is $O(N \times fd)$, where N is the number of training samples and fd is the number of features. Although C4.5 and SVM are more powerful, the computational costs of them are higher than that of kNN [42]. The time complexity of C4.5 is $O(N \times fd \times \log fd)$, and the time complexity of SVM for training and testing is $O(N_{sv}^3 + N \times N_{sv}^2)$ and $O(N_{sv})$, respectively, where N_{sv} denotes the number of support vectors [42]. It is worth noting that N_{sv} grows as a linear function of N .

Theoretical and empirical results [31, 32] suggest that combining classifiers can derive an optimal improvement in accuracy if the classifiers are not correlated with each other. In our work, feature partitioning methods can be used to further improve the classification performance due to the reduced correlation among the classifiers [43]. Therefore, MV-EDM has the potential to perturb the training set and enable kNN classifiers to avoid some correlated errors.

Let U denote the rating database, I denote the set of items, X denote the feature set, k denote the number of views


```

Input:  $U, I, X, k, d, e, bs, q, Vdata, BsClassifier$ 
Output:  $TR, X_{allopt}, Fpre, Vpre$ 
1   $TR \leftarrow \emptyset, Fpre \leftarrow \emptyset, Vpre \leftarrow \emptyset, X_{allopt} \leftarrow \emptyset$ 
2  for  $p=1$  to  $bs$  do
3     $U_p \leftarrow selectProfiles(U, d)$ 
4     $TR_p \leftarrow getfea(U_p)$ 
5     $Fpre_p \leftarrow getClassifiersPre(X_{opt}, Vdata, TR_p, BsClassifier)$ 
6     $Fpre \leftarrow Fpre \cup \{Fpre_p\}$ 
7     $Vpre_p \leftarrow getViewPre(X_{opt}, Vdata, TR_p, BsClassifier)$ 
8     $Vpre \leftarrow Vpre \cup Vpre_p$ 
9     $X_{allopt} \leftarrow X_{allopt} \cup X_{opt}$ 
10   if  $p \bmod q = 0$  then
11      $X_{opt} \leftarrow constructViews(BsClassifier, X, k, Vdata)$ 
12   end if
13    $TR \leftarrow TR \cup TR_p$ 
14 end for
15 return  $TR, X_{allopt}, Fpre, Vpre$ 

```

ALGORITHM 3: The algorithm for training base classifiers.

in partition method, and d denote the randomly selected profiles in base training set. Let bs denote the number of the base training sets and q denote the interval of repartitioning the feature set. Let $Vdata$ denote the validation dataset, $BsClassifier$ denote the base classifier, and TR denote the set of base training sets. Let $Fpre$ denote the set of different classifiers' accuracy on the validation dataset, X_{allopt} denote the set of all the feature set partitioning results, and $Vpre$ denote the set of different classifiers' accuracy from different views. The algorithm for training base classifiers is described as Algorithm 3.

Algorithm 3 includes two parts. The first part (lines 1-4) is to generate the base training sets. In particular, a certain number user profiles are first selected (line 3) and then the detection features are exacted from these profiles to generate base training sets (line 4) according to Algorithm 1. The second part (lines 5-15) mainly vertically partitions the base training sets and trains the base classifiers from different views in different base training sets. Firstly, the classifier's accuracy on each base training set is calculated (lines 5-6). Then each classifier is trained on a base training set from different views, in which the accuracy of different views is calculated (lines 7-8). In lines 9-12, the feature set is repartitioned with q intervals according to Algorithm 2. Finally, the base training sets TR , feature set partitioning results X_{allopt} , base classifiers' accuracy set $Fpre$, and different views' accuracy set $Vpre$ are obtained.

3.4. Ensemble Detection. In this paper, according to the accuracy, top 15% base classifiers are selected to yield ensemble result. The weight of each base classifier is determined by its accuracy on the validation dataset. Similarly, for every base classifier, the weight of each view is determined by the accuracy of base classifier from this view on the validation dataset. Therefore, the final decision can be obtained by weighting the base classifiers, which can be described as follows:

$$L(u_{ul}) = \begin{cases} \text{attack}, & \sum_{p=1}^{bs_{top}} BsClassifier_p(u_{ul}) > 0 \\ \text{genine}, & \sum_{p=1}^{bs_{top}} BsClassifier_p(u_{ul}) \leq 0 \end{cases} \quad (30)$$

where u_{ul} is the unlabeled user in feature space, $L(u_{ul})$ is the predictive result of u_{ul} , bs_{top} is the number of selected base classifiers, and $BsClassifier_p(u_{ul})$ is the predictive result of base classifier p to user u_{ul} . The decision-making process of each base classifier can be expressed as follows:

$$BsClassifier_p(u_{ul}) = \begin{cases} 1, & \sum_{i=1}^k (BsClassifier_p(u_{ul}(X_i)) \times Vweight_{p,i}) > 0 \\ -1, & \sum_{i=1}^k (BsClassifier_p(u_{ul}(X_i)) \times Vweight_{p,i}) \leq 0 \end{cases} \quad (31)$$

where $u_{ul}(X_i)$ denotes the projection of u_{ul} from view X_i and $Vweight_{p,i}$ denotes the weight of view X_i in base classifier p , which can be calculated as follows.

$$Vweight_{p,i} = \frac{Vpre_{p,i}}{\sum_i Vpre_{p,i}} \quad (i = 1, 2, \dots, k) \quad (32)$$

Let $Ttest$ denote the test set, $BsClassifier$ denote the base classifier, X_{allopt} denote the set of all optimal feature set partitioning results, k denote the number of views in each partitioning result, $Fpre$ denote the set of different classifiers' accuracy, $Vpre$ denote the set of different classifiers' accuracy from different views, and $PLabels$ denote the final detection result. The ensemble detection algorithm is described as Algorithm 4.

```

Input:  $Ttest, BsClassifier, X_{allop}, k, Fpre, Vpre$ 
Output:  $PLabels$ 
1   $PLabels \leftarrow \emptyset$ 
2   $SetBS \leftarrow selectTopClassifier(Fpre)$ 
3   $U_{ul} \leftarrow getfea(Ttest)$ 
4  for each  $u \in U_{ul}$  do
5     $count \leftarrow 0$ 
6    for each  $p \in BS$  do
7      for  $i=1$  to  $k$  do
8         $l_i \leftarrow BsClassifier_p(u, X_i)$ 
9         $l_p = l_p + l_i * Vweight_{p,i}$ 
10     end for
11     if  $l_p > 0$  then
12        $vote=1$ 
13     else
14        $vote=-1$ 
15     end if
16      $count \leftarrow count + vote$ 
17   end for
18   if  $count > 0$  then
19      $PLabels_u \leftarrow \{attack\}$ 
20   Else
21      $PLabels_u \leftarrow \{genuine\}$ 
22   end if
23    $PLabels \leftarrow PLabels \cup PLabels_u$ 
24 end for
25 return  $PLabels$ 

```

ALGORITHM 4: Ensemble detection algorithm.

In Algorithm 4, the top 15% base classifiers with the largest accuracy values are selected to build a predictive classifier by the weight voting. Firstly, the base classifiers are selected (line 2) and the user features of profiles in the test set are extracted (line 3). Secondly, with the extracted user features, the predictive result of every classifier is obtained by the weight voting from different views (lines 7-16). Thirdly, the final predictive result is generated by voting the base classifiers (lines 18-22). When every user profile is detected in the outer loop, the set of final detection results is obtained.

4. Experimental Evaluation

In this section, we first introduce the experimental settings and evaluation metrics. To compare the detection performance under various attacks, we first conduct experiments on the Netflix dataset. Then we conduct the experiments on the Amazon review dataset to demonstrate the practical value of the proposed method. Finally, the comparison of running time is provided.

4.1. Experimental Data and Setting. The Netflix dataset and Amazon review dataset are used for experimental evaluation.

(1) Netflix dataset (this dataset was constructed to support participants in the Netflix Prize (<http://netflixprize.com>)): A contest dataset is provided for Netflix Prize. We randomly

select 542,182 ratings on 4000 movies by 5000 users between January 6th, 2000, and December 31st, 2005 as our experimental dataset.

(2) Amazon dataset: It contains 1,205,125 reviews on 136,785 products by 645,072 reviewers, which is crawled from Amazon.cn till August 20, 2012, by Xu etc. [44]. Each review includes ReviewerID, ProductID, Product Brand, Rating, Date, and Review Text. In this dataset, 5055 reviewers are labelled. Moreover, the information of 8915 reviewer groups is provided [28, 44].

Since Netflix dataset is provided to train the recommender algorithm for Netflix contest, we assume that the original users are genuine ones. We randomly divide 5000 genuine users into three groups. The first group including 3000 genuine users is used for the training dataset; the second and third group including 1000 genuine users are used for the test datasets and validation dataset, respectively.

Due to the lack of labelled users in Netflix dataset, the shilling profiles are generated according to the attack models in Table 1 by reverse engineer [6]. The rating timestamps of shilling profiles are sampled from the genuine ones in order to ensure the rationality of the generated profiles. For example, for a shilling profile, the rating timestamps are generated as follows. Firstly, we obtain a genuine profile set in which all genuine profiles have more rated items than this shilling profile. Secondly, we randomly select one profile from the genuine profile set and then randomly select the timestamps from this genuine profile as the shilling profile's timestamps.

In the shilling profiles, the target item is randomly selected. Firstly, we detect 6 push attacks including random, average, 30% AoP, shifting, power item, and power user attack models shown in Table 1. These attack models can also be used for nuke attacks with the same detection methods. However, the attack models that are effective for pushing items are not necessarily as effective for nuke attacks. Therefore, secondly, we also detect 2 effective nuke attacks including love/hate and reverse bandwagon attack models.

As to filler size, more than 83% genuine profiles are below or equal to 5% in the Netflix dataset. To simulate most of the genuine users, the filler sizes of shilling profiles are set to {3%, 5%} in training and test sets.

As to attack size, the value should not be too large (usually below 20%); otherwise the shilling profiles would be easily detected and the attack cost would be raised [45]. For the purpose of experiments, the attack sizes of shilling profiles are set to {3%, 5%, 10%, 12%} in training and test sets.

In the validation dataset, to balance the proportion between genuine and shilling profiles, the shilling profiles are generated by random, average, 30% AoP, shifting, power user, power item, love/hate, and reverse bandwagon models in Table 1 with 5% attack sizes and {3%, 5%} filler sizes.

In our experiments, the average values of 20 times experiments are used as the final evaluation values. All of our experiments are implemented using Matlab R2015b and Python 2.7 on a personal computer with Intel i7-5500U 2.40GHz CPU, 8GB memory. In Algorithm 2, the proposed features are partitioned into 3 views. In Algorithm 3, 100 kNN base classifiers are trained, whose neighbors are set to 9 according to cross validations.

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

$$\text{Recall} = \frac{TP}{TP + FN} \quad (33)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (34)$$

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

4.3. Experiment on the Netflix Dataset

4.3.1. Comparison of Recall and Precision. To illustrate the effectiveness of MV-EDM, we compare it with the following six baseline methods.

(1) MEL-once: A multiview ensemble learning method [35]. We use the 17 features proposed in this paper as feature set and partition them into 3 views only once. 100 kNN base classifiers are combined from 3 fixed views to generate the final detection results, whose neighbors are set to 9 according to cross validations.

(2) MV-SVM: A multiview detection method. The 17 features extracted from multiple views in this paper are used by SVM for classification. We call it MV-SVM, in which the Gaussian radial basis function is used as the kernel function, and rbf_sigma is set to 4 according to 5-fold cross validation in the training set.

(3) SVM-TIA: A two-phase shilling attack detection method based on SVM and target item analysis [10]. The 7 rating-based features including $RDMA$, $DegSim$, $WDMA$, WDA , $LengthVar$, $MeanVar$, and $DegSim'$ are used by SVM for classification. In SVM algorithm, Gaussian radial basis function is used as the kernel function, and rbf_sigma is set to 5 according to 5-fold cross validation in the training set. In phase 2 of SVM-TIA, the threshold θ , a number of attack profiles, is set to 20.

(4) RAdaBoost: An improved rescale AdaBoost method [11], 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 $s_k = 2/(k + u)$, $u \in \mathbb{N}$ [11], where u is set to 100-time attack size.

(5) RF-13: A random forest ensemble detection method, which uses the 13 rating-based features in [4]. In the experiments, 100 decision trees are used and other parameters are set by default in Matlab.

(6) DWT-SVM: An item popularity-based method [28]. The 17 user features are extracted from user's rating series based on item popularity, and SVM is used for classification. In SVM algorithm, Gaussian radial basis function is used as the kernel function, and rbf_sigma is set to 4 according to 5-fold cross validation in the training set.

Tables 5 and 6 list the recall and precision of seven methods under eight attacks with various filler sizes and attack sizes on the Netflix dataset.

As shown in Table 5, the recall of MV-EDM is higher than or equal to other methods when detecting various attacks. The rating-based methods (SVM-TIA, RAdaBoost, RF-13) and item popularity-based method (DWT-SVM) show relatively high recall under random, average, AoP, shifting, love/hate, and reverse bandwagon attacks. However, under power user attacks, only MEL-once, MV-SVM, and MV-EDM maintain recall at high levels. This is because SVM-TIA, RAdaBoost, RF-13, and DWT-SVM mainly extract detection features from single view and ignore other useful information. Also, this illustrates the effectiveness of the proposed features when detecting various attacks. In multiview methods (MEL-once, MV-SVM, MV-EDM), MEL-once and MV-EDM have higher recall than MV-SVM. In rating-based methods, RAdaBoost and RF-13 have higher recall than SVM-TIA. This is because the ensemble methods (MEL-once, MV-EDM, RAdaBoost, RF-13) can combine the outputs of multiple classifiers to reduce the generalization errors and benefit to detect as more shilling profiles as possible. The item popularity-based method (DWT-SVM) has higher recall than SVM-TIA when detecting random, average, AoP, shifting, love/hate, and reverse bandwagon attacks. The possible reason is that these attacks select filler items in a simple way, which makes the item popularity-based features more effective than the rating-based features.

It can be seen from Table 6 that the precision of MV-EDM is the highest when detecting various attacks. Compared with the single view methods (SVM-TIA, RAdaBoost, RF-13, DWT-SVM), the multiview methods (MEL-once, MV-SVM, MV-EDM) have higher precision than the single view methods. In the single view methods, RAdaBoost outperforms RF-13 in precision in most cases. This may be attributed to the more effective features and rescale AdaBoost method in RAdaBoost. For SVM-TIA, the suspicious user profiles are first detected by SVM based on some user rating-based features. However, shilling profiles may not be fully detected at the first stage. At the second stage, the suspicious profiles are further determined by analysing target items, which has no effect on the recall but increases the precision. In the multiview methods, although the precision of MEL-once at the same filler size increases generally as the attack size increases, it is always below the precision of MV-EDM. This is because MEL-once depends on the fixed views, and the partitioned views are easily affected by the initial order of features. Thus the precision of MEL-once cannot achieve the best performance. The MV-SVM also uses our features to train classifier. However, since MV-SVM does not consider the effectiveness of features and use ensemble method, it has relatively low precision in multiview methods. Therefore, given the same features, the proposed multiview ensemble method is superior to SVM and the traditional multiview ensemble learning method.

Obviously, MV-EDM outperforms other methods in terms of recall and precision metrics when detecting various attacks. This is because MV-EDM extracts user features from multiple perspectives, which enables it to detect more anomaly rating patterns than other methods. Moreover, MV-EDM learns multiview features and classifier simultaneously,

TABLE 5: Continued.

Filler size	3%	5%	10%	12%	5%	5%	10%	12%
Attack size	3%	5%	10%	12%	3%	5%	10%	12%
Love/Hate attack	MEL-once	0.97	0.98	0.99	0.99	0.98	0.99	0.99
	MV-SVM	0.96	0.97	0.97	1	0.96	0.98	0.98
	SVM-TIA	0.58	0.65	0.79	0.82	0.62	0.69	0.82
	RAdaBoost	0.92	0.94	0.97	0.98	0.95	0.96	0.98
	RF-13	0.89	0.91	0.89	0.992	0.92	0.95	0.95
	DWT-SVM	0.88	0.91	0.94	0.96	0.89	0.9	0.97
Reverse Bandwagon attack	MV-EDM	1	1	1	1	1	1	1
	MEL-once	0.96	0.97	0.98	0.99	0.98	0.98	0.99
	MV-SVM	0.95	0.9	0.97	0.98	0.96	0.96	0.98
	SVM-TIA	0.48	0.55	0.57	0.61	0.49	0.57	0.62
	RAdaBoost	0.98	0.96	0.98	0.98	0.97	0.96	0.98
	RF-13	0.95	0.96	0.94	0.97	0.94	0.97	0.97
	DWT-SVM	0.92	0.94	0.97	0.98	0.92	0.95	0.98
	MV-EDM	1	1	1	1	1	1	1

TABLE 6: Precision of seven methods under eight attacks.

Filler size	3%	5%	10%	12%	5%	3%	5%	10%	12%
Attack size	3%	5%	10%	12%	5%	3%	5%	10%	12%
Random attack	MEL-once	0.63	0.68	0.72	0.81	0.62	0.66	0.77	0.85
	MV-SVM	0.56	0.59	0.68	0.79	0.59	0.65	0.74	0.78
	SVM-TIA	0.54	0.59	0.69	0.75	0.56	0.58	0.72	0.79
	RAdaBoost	0.53	0.6	0.71	0.78	0.58	0.6	0.78	0.8
	RF-13	0.26	0.35	0.52	0.57	0.31	0.36	0.53	0.59
	DWT-SVM	0.41	0.46	0.51	0.55	0.43	0.49	0.52	0.57
	MV-EDM	0.75	0.83	0.87	0.9	0.69	0.78	0.85	0.89
	Average attack	0.57	0.63	0.78	0.82	0.58	0.65	0.74	0.82
AoP attack	MV-SVM	0.56	0.62	0.79	0.8	0.57	0.62	0.79	0.81
	SVM-TIA	0.54	0.58	0.78	0.79	0.55	0.61	0.72	0.78
	RAdaBoost	0.52	0.51	0.69	0.81	0.45	0.68	0.73	0.78
	RF-13	0.26	0.39	0.53	0.55	0.29	0.4	0.56	0.58
	DWT-SVM	0.4	0.45	0.49	0.56	0.43	0.49	0.52	0.57
	MV-EDM	0.71	0.77	0.87	0.91	0.65	0.8	0.9	0.89
	Average attack	0.57	0.6	0.71	0.85	0.59	0.66	0.77	0.84
	MV-SVM	0.53	0.57	0.69	0.84	0.55	0.64	0.73	0.81
Shifting attack	SVM-TIA	0.22	0.28	0.39	0.84	0.29	0.36	0.45	0.59
	RAdaBoost	0.27	0.32	0.62	0.72	0.4	0.55	0.65	0.73
	RF-13	0.15	0.17	0.29	0.52	0.2	0.26	0.32	0.49
	DWT-SVM	0.38	0.41	0.43	0.49	0.39	0.41	0.45	0.53
	MV-EDM	0.68	0.74	0.83	0.91	0.69	0.8	0.84	0.91
	Average attack	0.55	0.71	0.78	0.83	0.6	0.72	0.81	0.85
	MV-SVM	0.56	0.69	0.73	0.81	0.59	0.72	0.78	0.83
	SVM-TIA	0.53	0.61	0.75	0.77	0.52	0.59	0.68	0.81
Power item attack	RAdaBoost	0.48	0.65	0.72	0.8	0.5	0.6	0.72	0.83
	RF-13	0.26	0.35	0.52	0.54	0.29	0.34	0.51	0.59
	DWT-SVM	0.42	0.43	0.48	0.55	0.44	0.46	0.53	0.56
	MV-EDM	0.71	0.82	0.85	0.92	0.63	0.76	0.85	0.9
	Average attack	0.45	0.48	0.59	0.7	0.49	0.51	0.61	0.69
	MV-SVM	0.43	0.46	0.51	0.69	0.49	0.52	0.59	0.62
	SVM-TIA	0.28	0.32	0.43	0.58	0.26	0.35	0.47	0.59
	RAdaBoost	0.18	0.16	0.21	0.32	0.19	0.16	0.23	0.35
Power user attack	RF-13	0.15	0.17	0.19	0.21	0.16	0.19	0.25	0.28
	DWT-SVM	0.16	0.18	0.19	0.2	0.17	0.19	0.24	0.28
	MV-EDM	0.65	0.77	0.82	0.87	0.66	0.79	0.82	0.88
	Average attack	0.44	0.49	0.55	0.63	0.47	0.49	0.58	0.65
	MV-SVM	0.44	0.52	0.53	0.55	0.45	0.54	0.57	0.58
	SVM-TIA	0.18	0.19	0.27	0.32	0.17	0.2	0.29	0.35
	RAdaBoost	0.15	0.16	0.22	0.28	0.16	0.18	0.24	0.29
	RF-13	0.09	0.11	0.14	0.19	0.09	0.15	0.18	0.23
Power user attack	DWT-SVM	0.08	0.1	0.13	0.19	0.11	0.18	0.19	0.22
	MV-EDM	0.67	0.75	0.81	0.87	0.67	0.76	0.83	0.88

TABLE 6: Continued.

Filler size	3%	5%	10%	12%	5%	10%	12%
Attack size	3%	5%	10%	12%	3%	5%	12%
Love/Hate attack	MEL-once	0.53	0.55	0.625	0.7	0.53	0.7
	MV-SVM	0.46	0.51	0.58	0.69	0.49	0.62
	SVM-TIA	0.43	0.52	0.54	0.61	0.42	0.66
	RAdaBoost	0.4	0.49	0.53	0.59	0.41	0.62
	RF-I3	0.36	0.44	0.52	0.541	0.39	0.56
	DWT-SVM	0.39	0.45	0.49	0.53	0.42	0.57
Reverse Bandwagon attack	MV-EDM	0.64	0.67	0.79	0.85	0.65	0.86
	MEL-once	0.54	0.59	0.63	0.7	0.56	0.71
	MV-SVM	0.53	0.56	0.61	0.69	0.55	0.68
	SVM-TIA	0.48	0.52	0.63	0.68	0.48	0.69
	RAdaBoost	0.4	0.49	0.55	0.6	0.48	0.72
	RF-I3	0.39	0.41	0.51	0.558	0.45	0.61
	DWT-SVM	0.41	0.43	0.48	0.52	0.42	0.54
	MV-EDM	0.62	0.65	0.72	0.86	0.63	0.87

FIGURE 9: The F1-measure of MV-EDM under nine attacks with various attack sizes and filler sizes.

which leads to training the classifier with diversity. Thus, MV-EDM can achieve better performance than the benchmark methods.

4.3.2. Analysis of Information Gain. Information gain is widely used to evaluate the importance of a feature for a classification system. In general, larger information gain indicates more importance of the feature. Table 7 lists the information gain of the proposed features when detecting different types of attacks. In our experiments, we calculate the mean of information gain for each type of attacks based on the experiments in Section 4.3.1.

As shown in Table 7, *RDDMA* is the most important feature for detecting eight attacks, whose information gain ranks the top 1. *WDDMA* is also important for random, average, AoP, shifting, power user, love/hate, and reverse bandwagon attacks. For power item attack, *IE_URTPVP* is the second important feature. It is worth noting that *M_URP* and *V_URP* are important for detecting love/hate attack, but they are not important for identifying other attacks.

4.3.3. Generalization Ability of MV-EDM. In [6–8, 10–13, 24–28], the detection model was usually trained for some attacks with fixed filler sizes and attack sizes and tested with the same parameters. Based on the experiments in Section 4.3.1, we conduct experiments to further evaluate the overall performance of the proposed method when the attack parameters are modified.

Since precision and recall are two equally important but mutually contradictory metrics, F1-measure metric is widely used to evaluate the overall performance of the detection method. The larger the F1-measure is, the better the overall performance is.

$$F1\text{-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (35)$$

Figure 9 illustrates the F1-measure of MV-EDM under nine attacks with various attack sizes and filler sizes. The training set is the same as that in Section 4.1. However, in the test sets, since the filler sizes of more than 95% genuine profiles are below 10%, the filler sizes are set to {2%, 4%,

TABLE 7: Information gain of the proposed features.

Feature	Random attack		Average attack		AoP attack		Shifting attack		Power item attack		Power user attack		Love/hate attack		Reverse bandwagon attack	
	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank
<i>IE_URTPV</i>	0.15	7	0.16	7	0.15	7	0.16	6	0.18	5	0.14	8	0.16	11	0.15	8
<i>CCE_URTPV</i>	0.21	3	0.23	3	0.19	5	0.24	3	0.19	3	0.21	3	0.22	6	0.23	3
<i>R_URTPV</i>	0.11	11	0.10	10	0.09	10	0.08	14	0.11	11	0.09	12	0.11	14	0.12	10
<i>M_URTPV</i>	0.17	6	0.18	5	0.18	6	0.16	7	0.14	8	0.16	6	0.18	8	0.17	7
<i>V_URTPV</i>	0.09	13	0.08	13	0.09	11	0.09	12	0.10	12	0.09	11	0.11	13	0.11	11
<i>WDDMA</i>	0.28	2	0.28	2	0.25	2	0.28	2	0.16	7	0.23	2	0.31	2	0.28	2
<i>RDDMA</i>	0.31	1	0.31	1	0.32	1	0.32	1	0.34	1	0.31	1	0.32	1	0.29	1
<i>IE_URTPVP</i>	0.18	5	0.18	6	0.21	3	0.18	4	0.23	2	0.21	4	0.18	10	0.17	6
<i>M_URTPVP</i>	0.19	4	0.20	4	0.19	4	0.18	5	0.16	6	0.20	5	0.18	7	0.19	5
<i>V_URTPVP</i>	0.13	9	0.11	9	0.14	9	0.10	10	0.18	4	0.16	7	0.11	15	0.10	13
<i>M_URP</i>	0.08	14	0.08	14	0.05	15	0.08	13	0.09	13	0.03	17	0.28	3	0.09	14
<i>V_URP</i>	0.04	15	0.03	16	0.06	14	0.04	17	0.09	14	0.09	13	0.26	4	0.04	15
<i>IE_URTPVN</i>	0.13	8	0.14	8	0.15	8	0.15	8	0.11	10	0.14	9	0.15	12	0.22	4
<i>M_URTPVN</i>	0.03	16	0.03	17	0.02	17	0.09	11	0.02	17	0.07	14	0.08	17	0.03	16
<i>V_URTPVN</i>	0.03	17	0.04	15	0.03	16	0.04	16	0.03	16	0.06	16	0.09	16	0.02	17
<i>M_URN</i>	0.09	12	0.08	12	0.07	13	0.06	15	0.04	15	0.07	15	0.24	5	0.11	12
<i>V_URN</i>	0.11	10	0.09	11	0.08	12	0.14	9	0.14	9	0.11	10	0.18	9	0.13	9

FIGURE 10: The F1-measure of two methods under eight attacks at 5% filler size across various attack sizes.

6%, 8%, 10%}. In fact, if the filler size of shilling profiles is more than that of genuine profiles, the attacks can be easily detected by the features based on filler size, such as length variance (*LengthVar*) [5, 6] and filler mean variance (*FMV*) [46]. Therefore, we set the maximum filler size in the test sets to 10%. Also, the attack sizes are set to {2%, 4%, 6%, 8%, 10%, 12%, 14%, 16%, 18%, 20%}. Moreover, we use the trained model to detect the bandwagon attack that does not appear in the training set.

As shown in Figure 9, MV-EDM can effectively detect random, average, shifting, love/hate, reverse bandwagon, and bandwagon attacks with various attack sizes and filler sizes except for AoP, power item, and power user attacks. It is worth noting that bandwagon attack does not appear in the training set, but it can be effectively detected by MV-EDM. These experimental results indicate that MV-EDM can still work under some attacks (random, average, shifting, love/hate, reverse bandwagon, and bandwagon attacks) when the attack sizes and filler sizes are modified. This means the proposed method has relatively strong generalization ability under the attacks with randomly selected filler items. However, since a lot of AoP, power item, and power user shilling profiles have similar or even replicate filler items like the genuine profiles, MV-EDM hardly detects such attacks when filler sizes are significantly modified. As shown in Figures 9(c), 9(e), and 9(f), the F1-measure of MV-EDM stays relatively high at {2%, 4%, 6%} filler sizes and decreases at {8%, 10%} filler sizes under AoP, power item, and power user attacks. Therefore, compared with previous research results in which the training and test sets use the same attack parameters [6–8, 10–13, 24–28], the detection performance of MV-EDM is more optimistic even if the attack parameters are modified.

To further test the generalization ability of MV-EDM, we train MV-EDM by one type of attacks and conduct experiments to evaluate the performance of MV-EDM in detecting another type of attacks. MV-EDM is trained by random, average, AoP, shifting, power item, power user, love/hate, and reverse bandwagon attacks with {3%, 5%} filler sizes and 10% attack size, respectively. Then we take each type of attacks with 4% filler size and 6% attack sizes as

examples to test the F1-measure of MV-EDM. Table 8 lists the experimental results.

As shown in Table 8, under random, average, shifting, love/hate, and reverse bandwagon attacks, once MV-EDM is appropriately trained by one type of attacks, it can effectively detect other attacks. This means MV-EDM has strong generalization ability under the attacks with randomly selected filler items, which is hardly affected by rating values. Also, when MV-EDM is trained by AoP attack or power item attack, besides random, average, shifting, love/hate, and reverse bandwagon, it can detect power item attack or AoP attack. This is because the way to select item to rate in AoP attack is similar to that in power item attack in our experiments. It can be seen from Table 8 that MV-EDM can detect power user attack only when it has been trained by the same attack. This is because the rated items and their ratings are copied from the genuine users in power user attack, whose rating pattern is unique and cannot be learned from other attacks. Therefore, MV-EDM has relatively high generalization ability.

4.3.4. Impact of Dynamic Mean of Item Ratings. To evaluate the impact of dynamic mean of item ratings, we conduct experiments to compare the performance of MV-EDM with traditional *RDMA*, *WDMA* (for the sake of differentiation, we name it as *MV-EDM-β*) and that of MV-EDM with the proposed *RDDMA*, *WDDMA*. Figure 10 illustrates the F1-measure of two methods under eight attacks with attack sizes {3%, 5%, 10%, 12%} at 5% filler size. MV-EDM-β and MV-EDM are trained by the same training set in Section 4.1.

As shown in Figure 10, the F1-measure of MV-EDM is always higher than that of MV-EDM-β. Taking AoP attack as an example, at 5% filler size across 3%, 5%, 10%, and 12% attack sizes, the F1-measure of MV-EDM is 0.82, 0.89, 0.91, and 0.95, which is higher by 39%, 45%, 20%, and 10% than that of MV-EDM-β, respectively. These results indicate that the features *RDDMA* and *WDDMA* can help to improve the detection performance of MV-EDM. This means the dynamic mean of item ratings is useful to detect various attacks.

TABLE 8: The FI-measure of MV-EDM with untrained attacks.

Attack for training	Attacks for test							
	Random attack	Average attack	AoP attack	Shifting attack	Power item attack	Power user attack	Love/hate attack	Reverse bandwagon attack
Random attack	0.97	0.90	0.37	0.88	0	0	0.88	0.90
Average attack	0.89	0.98	0.48	0.90	0	0	0.89	0.89
AoP attack	0.74	0.75	0.98	0.71	0.71	0	0.75	0.78
Shifting attack	0.89	0.92	0.44	0.97	0	0	0.90	0.93
Power item attack	0.69	0.69	0.68	0.67	0.98	0	0.70	0.69
Power user attack	0	0	0	0	0	0.98	0	0
Love/hate attack	0.79	0.77	0.41	0.76	0	0	0.97	0.71
Reverse bandwagon attack	0.98	0.97	0.31	0.97	0	0	0.66	0.98

TABLE 9: Recall, precision, and F1-measure of seven detectors with random and group sampling methods in the Amazon review dataset.

		random sampling method	group sampling method
Recall	MEL-once	0.81	0.72
	MV-SVM	0.78	0.71
	SVM-TIA	0.69	0.55
	RAdaBoost	0.72	0.68
	RF-13	0.70	0.63
	DWT-SVM	0.71	0.69
	MV-EDM	0.89	0.84
Precision	MEL-once	0.71	0.65
	MV-SVM	0.70	0.63
	SVM-TIA	0.68	0.58
	RAdaBoost	0.64	0.57
	RF-13	0.61	0.50
	DWT-SVM	0.63	0.54
	MV-EDM	0.76	0.69
F1-measure	MEL-once	0.76	0.68
	MV-SVM	0.74	0.67
	SVM-TIA	0.68	0.56
	RAdaBoost	0.68	0.62
	RF-13	0.65	0.56
	DWT-SVM	0.67	0.61
	MV-EDM	0.82	0.76

4.4. *Experiment on the Amazon Review Dataset.* To examine the performance of the proposed method in practice, we conduct experiments on Amazon review dataset and list the experiment results.

To better simulate actual detection, we take two methods including random sampling and group sampling to generate the training, validation, and test datasets in Amazon review dataset. In random sampling method, we randomly select 500 reviewers as training set, 300 reviewers as validation set, and 4255 reviewers as test set, respectively. In group sampling method, firstly, we randomly select 1000 reviewer groups, in which all the 1535 distinct reviewers are selected as training set. Secondly, we randomly select 500 reviewer groups and find that 102 groups have no common members with the training set. Thus, we select 196 reviewers of the 102 groups as validation set. Finally, we find that 1295 groups in the remained groups have no common members with the training and validation set, so we select 1970 reviewers of the 1295 groups as test set.

With random and group sampling methods in the Amazon review dataset, Table 9 shows the recall, precision, and F1-measure of seven detectors described in Section 4.3.1

As shown in Table 9, MV-EDM has the highest recall, precision, and F1-measure with two sampling methods in practical detection. For overall performance, the F1-measure of MEL-once and MV-SVM is only next to MV-EDM with

two sampling methods, which again illustrates the effectiveness of our proposed features in real-world dataset. With random sampling method, SVM-TIA has the highest F1-measure except MEL-once, MV-SVM, and MV-EDM. Understandably, as shown in Table 9, although SVM-TIA has the lowest recall, it remains the highest precision in single view methods, which results in the very high F1-measure. With group sampling method, in single view methods, RAdaBoost and DWT-SVM have relative high F1-measure, and SVM-TIA and RF-13 remain having the lowest F1-measure. This may be because the grouping sample method weakens the distinguishing ability of features *DegSim* (degree of similarity with top neighbors) and *DegSim'* (the improved *DegSim*) in SVM-TIA and RF-13. In fact, the features *DegSim* and *DegSim'* identify attackers by capturing the nearest neighbors. However, the grouping samples have no common users in training set and test set and make the neighbors information very different between training and test set. Therefore, the effectiveness of classifiers is reduced in SVM-TIA and RF-13 by grouping sample method. Also, we can observe that the seven detectors have better performance with random sampling method than with grouping sampling method. This may be because there are complex user preferences in different groups and the classifier features cannot be effectively learned only from some separated groups of users. By contrast, with random sampling method, the users are almost uniformly distributed in training and test dataset, and the classifiers can benefit from the uniform distribution samples.

No matter what the sampling methods are, MV-EDM takes into consideration not only multiple information, but also advantage of effectiveness from features and base classifiers to improve the performance significantly. Therefore, MV-EDM always outperforms other methods in Amazon review dataset, which shows the practical value of our proposed detection method.

4.5. *Comparison of Running Time.* To examine time consumption of MV-EDM, we conduct experiments on two datasets and calculate the running time of seven methods, respectively. On the Netflix dataset, we take experiments under AoP attack with 3% filler size and 5% attack size as examples. On the Amazon dataset, the experiment setting is similar to Section 4.4. Then we calculate the offline training time and online detection time, respectively. Table 10 shows the comparison of running time for seven detection methods.

As shown in Table 10, the training time and detection time of MV-EDM are smaller than those of SVM-TIA, RF-13, and DWT-SVM, but greater than those of MEL-once, MV-SVM, and RAdaBoost. In multiview methods, compared to MEL-once, MV-EDM needs extra time to repartition the feature set in regular intervals. Compared to MV-SVM, MV-EDM and MEL-once need additional time to integrate base classifiers. Therefore, MV-EDM has more running time than MEL-once and MV-SVM. Also, MV-EDM shows higher running time compared with RAdaBoost, which may be attributed to the view construction and integration in MV-EDM. For RF-13 and SVM-TIA, their running time ranks top 2. This is because the *DegSim* and *DegSim'* need a lot of time to calculate the similarity between users. Since DWT-SVM utilizes discrete

TABLE 10: Comparison of running time for seven detection methods.

Algorithm	Netflix dataset	Training time (s)		Netflix dataset	Detection time (s)	
		Amazon review dataset random sampling method	Amazon review dataset group sampling method		Amazon review dataset random sampling method	Amazon review dataset group sampling method
MV-EDM	74.59	42.81	75.26	6.38	15.92	7.28
MEL-once	29.58	18.22	44.53	5.72	9.25	6.36
MV-SVM	20.79	12.73	36.28	1.36	7.46	5.10
SVM-TIA	487.58	196.38	529.65	18.75	59.37	32.71
RAdaBoost	25.07	24.19	37.58	2.77	27.15	12.67
RF-13	632.11	304.86	712.68	26.42	90.27	45.83
DWT-SVM	115.31	64.65	104.75	15.44	72.38	28.50

wave transform for each user series to extract features in training and test datasets, it needs a lot of time to handle wavelet decomposition. By contrast, in MV-EDM, discrete wave transform is used only once for the item series in the offline preprocessing. Therefore, with the consideration of the performance in Sections 4.3 and 4.4, it is desirable for MV-EDM to acquire better performance at the expense of a small amount of time.

5. Conclusions

The malicious users always try to bias the recommender systems' outputs. To characterize the shilling profiles, we extract the user features by taking advantage of multiple views information, including rating values, item temporal popularity, and rating timestamps. Based on the proposed features, we improve the features set partition algorithm to combine it with the kNN base classifiers, and design the multiview ensemble method to detect various shilling profiles. The experiment results on Netflix and Amazon review datasets illustrate the higher effectiveness of the proposed features and the better performance of the proposed detection method than those of the baseline methods.

In our future work, we will fuse more implicit information to characterize the genuine and shilling profiles, such as brand, gender, online time, and relationship between users. In experiments, since MV-EDM has no obvious advantage on running time, we will develop a distributed version of MV-EDM to improve the detection efficiency. On the other hand, we will also conduct experiments on more large-scale real-world datasets to further evaluate the practical values of MV-EDM.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [Nos. 61772452, 61379116] and the Natural Science Foundation of Hebei Province, China [No. F2015203046].

References

- [1] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, T.-S. Chua, and S. Ma, "Catch the black sheep: Unified framework for shilling attack detection based on fraudulent action propagation," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, pp. 2408–2414, Argentina, July 2015.
- [2] B. Mehta and W. Nejdl, "Attack resistant collaborative filtering," in *Proceedings of the 31st Annual International Conference on Research and Development in Information Retrieval (SIGIR '08)*, pp. 75–82, July 2008.
- [3] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," in *Proceedings of the Ijcai Workshop in Intelligent Techniques for Personalization*, 2005.
- [4] 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)*, pp. 67–74, November 2005.
- [5] C. Williams, B. Mobasher, and R. Burke, "Detection of obfuscated attacks in collaborative recommender systems," in *Proceedings of the Proc*, pp. 19–23, 2006.
- [6] C. A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: Detection of profile injection attacks," *Service Oriented Computing and Applications*, vol. 1, no. 3, pp. 157–170, 2007.
- [7] Z. Wu, J. Wu, J. Cao, and D. Tao, "HySAD: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pp. 985–993, August 2012.
- [8] Z. Wu, Y. Zhuang et al., "Shilling attack detection based on feature selection for recommendation systems," *Acta Electronica Sinica*, vol. 40, no. 8, pp. 1687–1693, 2012.
- [9] Q. Zhou, "Supervised approach for detecting average over popular items attack in collaborative recommender systems," *IET Information Security*, vol. 10, no. 3, pp. 134–141, 2016.

- [10] 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.
- [11] 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.
- [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] R. Bhaumik, B. Mobasher, and R. D. Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in *Proceedings of the Proc. of the 7th IEEE international conference on data mining*, pp. 181–187, Las Vegas, NV, USA, 2011.
- [14] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys '09)*, pp. 149–156, October 2009.
- [15] 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.
- [16] Z. Yang, Z. Cai, and Y. Yang, "Spotting anomalous ratings for rating systems by analyzing target users and items," *Neurocomputing*, vol. 240, pp. 25–46, 2017.
- [17] L. Yang, W. Huang, and X. Niu, "Defending shilling attacks in recommender systems using soft co-clustering," *IET Information Security*, vol. 11, no. 6, pp. 319–325, 2017.
- [18] F. Zhang and Q. Zhou, "Ensemble detection model for profile injection attacks in collaborative recommender systems based on BP neural network," *IET Information Security*, vol. 9, no. 1, pp. 24–31, 2015.
- [19] T. Tang and Y. Tang, "An effective recommender attack detection method based on time SFM factors," in *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011*, pp. 78–81, China, May 2011.
- [20] S. Zhang, A. Chakrabarti, J. Ford, and F. Makedon, "Attack detection in time series for recommender systems," in *Proceedings of the the 12th ACM SIGKDD international conference*, p. 809, Philadelphia, PA, USA, August 2006.
- [21] 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.
- [22] W. Zhou, J. Wen, M. Gao, H. Ren, and P. Li, "Abnormal profiles detection based on time series and target item analysis for recommender systems," *Mathematical Problems in Engineering*, vol. 2015, 2015.
- [23] N. Lathia, S. Hailes, and L. Capra, "Temporal Defenses for Robust Recommendations," in *Privacy and Security Issues in Data Mining and Machine Learning*, vol. 6549 of *Lecture Notes in Computer Science*, pp. 64–77, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [24] 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.
- [25] F. Zhang and H. Chen, "An ensemble method for detecting shilling attacks based on ordered item sequences," *Security and Communication Networks*, vol. 9, no. 7, pp. 680–696, 2016.
- [26] P. Karthikeyan, S. T. Selvi, G. Neeraja, R. Deepika, A. Vincent, and V. Abinaya, "Prevention of shilling attack in recommender systems using discrete wavelet transform and support vector machine," in *Proceedings of the 8th International Conference on Advanced Computing, ICoAC 2016*, pp. 99–104, India, January 2017.
- [27] W. T. Li, M. Gao, H. Li et al., "An shilling attack detection algorithm based on popularity degree features," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 41, no. 9, pp. 1563–1575, 2015.
- [28] 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. E99D, no. 10, pp. 2600–2611, 2016.
- [29] 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.
- [30] 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, 2012.
- [31] R. Bryll, R. Gutierrez-Osuna, and F. Quek, "Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets," *Pattern Recognition*, vol. 36, no. 6, pp. 1291–1302, 2003.
- [32] M. A. Bagheri and Q. Gao, "An efficient ensemble classification method based on novel classifier selection technique," in *Proceedings of the the 2nd International Conference*, p. 1, Craiova, Romania, June 2012.
- [33] A. B. Wilson, *Modeling, Analysis, and Simulation of ADSorption in Functionalized Membranes*, ProQuest LLC, Ann Arbor, MI, 2016.
- [34] 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*, pp. 57–64, USA, October 2014.
- [35] V. Kumar and S. Minz, "Multi-view ensemble learning: an optimal feature set partitioning for high-dimensional data classification," *Knowledge and Information Systems*, vol. 49, no. 1, 2016.
- [36] P. Castells, S. Vargas, and J. Wang, "Novelty and diversity metrics for recommender systems: choice, discovery and relevance," in *International Workshop on Diversity in Document Retrieval at the 33rd European Conference on Information Retrieval, Macdonald, C. Macdonald, J. Wang, and C. Clarke, Eds.*, pp. 29–36, New York, NY, USA, 2011.
- [37] A. Mukherjee, A. Kumar, B. Liu et al., "Spotting opinion spammers using behavioral footprints," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013*, pp. 632–640, USA, August 2013.
- [38] A. Javari and M. Jalili, "Accurate and novel recommendations: An algorithm based on popularity forecasting," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 4, 2015.
- [39] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: Human, bot, or cyborg?" in *Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC 2010*, pp. 21–30, USA, December 2010.
- [40] A. Porta, G. Baselli, D. Liberati et al., "Measuring regularity by means of a corrected conditional entropy in sympathetic outflow," *Biological Cybernetics*, vol. 78, no. 1, pp. 71–78, 1998.
- [41] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

- [42] 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.
- [43] L. Rokach, "Genetic algorithm-based feature set partitioning for classification problems," *Pattern Recognition*, vol. 41, no. 5, pp. 1676–1700, 2008.
- [44] C. Xu, J. Zhang, K. Chang, and C. Long, "Uncovering collusive spammers in Chinese review websites," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM 2013*, pp. 979–988, USA, November 2013.
- [45] Q. Zhou and F. Zhang, "Detecting unknown recommendation attacks based on bionic pattern recognition," *Journal of Software*, vol. 25, no. 11, pp. 2652–2665, 2014.
- [46] M. A. Morid, M. Shajari, and A. R. Hashemi, "Defending recommender systems by influence analysis," *Information Retrieval*, vol. 17, no. 2, pp. 137–152, 2014.