Re-scale AdaBoost for Attack Detection in Collaborative Filtering Recommender Systems [☆]

Zhihai Yang¹, Lin Xu^{2*}, Zhongmin Cai¹, Zongben Xu²

1. Ministry of Education Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, 710049, China

2. Institute for Information and System Sciences, Xi'an Jiaotong University, Xi'an, 710049, China

Abstract

Collaborative filtering recommender systems (CFRSs) are the key components of successful E-commerce systems. However, CFRSs are highly vulnerable to attacks due to its openness. Since the size of attackers is usually far smaller than genuine users, conventional supervised learning based detection methods could be too “dull” to handle such imbalanced classification. In this paper, we improve detection performance from following two aspects. Firstly, we extract well-designed features from user profiles based on the statistical properties of the diverse attack models, making hard detection scenarios become easier to perform. Then, refer to the general idea of re-scale Boosting (RBoosting) and AdaBoost, we apply a variant of AdaBoost, called the re-scale AdaBoost (RAdaBoost) as our detection method based on the extracted features. Finally, a series of experiments on the MovieLens-100K dataset are conducted to demonstrate the outperformance of RAdaBoost over other competing techniques such as SVM, kNN and AdaBoost.

Keywords: Recommender system, attack detection, re-scale Boosting, imbalanced classification, detection rate.

[☆]The research was supported by National Basic Research Program (973 Program) (2013CB329404) and Key Project of National Science Foundation of China (Grant No. 11131006, 11401462, 61175039, 61221063, 61375040 and 91330204).

*Corresponding author: xulinshadow@gmail.com

1. Introduction

Personalization recommender systems (RSs) utilize a variety of recommendation methods to suggest products that users may like, such as movies, books etc. Collaborative filtering recommender systems (CFRSs) have been proved to be one of the most successful RSs used by many E-commerce companies such as Amazon, eBay etc. [2, 3, 5, 21]. In practice, CFRSs are prone to manipulation from attackers due to its openness. Typically, attackers carefully inject chosen attack profiles into CFRSs in order to bias the recommendation results for their benefits. It is termed as “shilling” or “profile injection” attacks, which leads to a negative impact on the CFRSs and decreases the trustworthiness of recommendation. Thus, it is crucial to construct an effective method for defending such attackers in CFRSs.

Supervised learning based detection method for “shilling” attackers in CFRSs is an important research direction, which regards the detection attributions as classification features and then distinguishes attack profiles from genuine profiles based on the constructed features. The attack detection problem also can be formulated as an imbalanced classification. Actually, the number of attackers is rare relative to the genuine users in CFRSs when attack size ¹ is small. In this case, the misclassification rate of the rare class maybe much greater than another class. The traditional supervised learning based detection methods (i.e., SVM and kNN) often insensitive to the minority and fail to effectively capture the concerned attackers.

In this paper, we aim to improve detection performance from two aspects. Firstly, we consider the overall statistical signature of attack profiles would significantly differ from genuine profiles. The difference mainly comes from three sources: the distribution of ratings among the selected items, filler items and target items. Based on the statistical properties of the diverse attack models, we apply a series of feature extraction methods on the user profiles (consisting of attack profiles and genuine profiles) to construct a sophisticated features representation for each user. This makes the following task (i.e., clustering or classification) easier to perform. Secondly, refer to the general idea of re-scale Boosting (RBoosting) [19, 31] and AdaBoost [9, 10], we utilize a variant of Boosting algorithm, called the re-scale AdaBoost (RAdaBoost) as our detection method. RBoosting is theoretically and experimentally proved to be better than the classical Boosting algorithm [19, 31]. Furthermore,

¹The ratio between the number of attackers and genuine users.

the theoretical near optimality of the numerical convergence of RBoosting among all the variants of the Boosting-type algorithms was also specified. This means that if the parameter is appropriately selected, RBoosting is comparable to the optimal Boosting-type algorithm. And AdaBoost [9, 10] is one of the most popular ensemble techniques paradigm which introduces a sensitive quality measure function (i.e., re-weighted loss) in the optimization process and has been shown to be very effective in practice [14]. Actually, AdaBoost employs a re-weighted loss function for gradually increasing emphasis (or weights) on misclassifications (i.e., concerned attackers in CFRSs) and can distinctly improve the predictive performance on a difficult data set. Thus, with the help of the re-scale operator, our developed RAdaBoost can be used in conjunction with many other types of learning algorithms (or weak learners) to improve the performance in “shilling” attacks detection. Finally, a series of experiments on the MovieLens-100K dataset are conducted to demonstrate the outperformance (i.e., classification error, detection rate and false alarm rate) of RAdaBoost over conventional classification techniques such as SVM, kNN and the original non-rescale AdaBoost version. The experimental results show that RAdaBoost can effectively improve the performance.

The rest of paper is organized as follows. In the next section, we present a brief introduction to the related work. In Section 3, we introduce the attack profiles and corresponding attack models. In Section 4, based on the feature extraction and RAdaBoost, our approach are described in details. In Section 5, experimental results are reported and analyzed. In the last section, we conclude the paper with a brief summary and predict the direction of the future work.

2. Related work

Existing work in this area have focused on detecting and preventing the “shilling” (or “profile injection”) attacks. Burke et al. [3] proposed and studied several attributions derived from user profiles for their utility in attack detection. They employed the kNN classifier as their detection approach. But it is unsuccessful when detecting attacks with small filler size ². Then,

²The ratio between the number of items rated by user u and the number of entire items in the recommender system.

Williams et al. [27, 28] tried to extract features from user profiles and utilized them to detect “shilling” attacks. They also suffered from low detection accuracy and many genuine profiles are misclassified as attack profiles. After that, He et al. [17] introduced the rough set theory into “shilling” attacks detection by means of taking features of user profiles as the condition attributes of the decision table. However, their method faced with the low overall classification rate in some cases, especially for bandwagon attack. Afterwards, Wu et al. [30] proposed a hybrid detection method to detect “shilling” attacks, which combined the naive Bayesian classifiers and augmented expectation maximization based on several selected metrics. Regretfully, their technique also suffered from low F-measure [6] when the filler size is small. Zhang et al. [32] introduced the idea of ensemble learning for improving predictive capability in the attack detection problem. They constructed the base-classifiers (or weaker learner) with the Support Vector Machine (SVM) approach and then integrated them to generate a high predictive ability learner for detection. Their proposed method exhibited better performance than some benchmarked methods. Nevertheless, it still suffered from low precision especially when the attack size is small. In addition, the same authors Zhang et al. [33] also proposed an online method, HHT-SVM, to detect profile injection attacks by combining Hilbert-Huang transform (HHT) and support vector machine (SVM). They created rating series for each user profile based on the novelty and popularity of items in order to provide basic data for feature extraction. The precision of their method shown better than the benchmarked methods, but the precision significantly decreased with the filler size increased.

Generally speaking, previous studies showed that the detection results of “shilling” attacks are dissatisfactory and leave much to be desired, especially when the filler size or attack size is small. In the current work, we intend to improve the detection performance from two aspects. Firstly, we introduce more well-designed features to depict the distinction between attack profiles and genuine profiles to make hard detection scenarios (i.e., with small filler size or attack size) become easier to perform. Secondly, in view of conventional supervised learning based techniques could be inadequate to handle such imbalanced classification, particularly when the attack size is small, we applied a variant of AdaBoost algorithm, called the re-scale AdaBoost (RAdaBoost) as our detection method. RAdaBoost gradually increases emphasis on the concerned attacks and can distinctly improve the performance for the imbalanced classification task.

3. Attack profiles and attack models

The attackers have various attack intents to bias the recommendation results for their benefits. In the literature, “shilling” attacks are classified into two ways: nuke attack and push attack [3, 13, 27]. In nuke attacks, attackers demote the target items by rating the lowest score, whereas in push attacks, attackers promote the target items by rating the highest score. In order to effectively “nuke” or “push” a target item, the attacker should have a priori information of the attack profiles. The general form of attack profiles is shown in Table 1. The set of target items I_T consists of single target item (called single-target attack) or multiple target items (called multiple-target attack). The target item i_t will be given a rating as determined by the function γ , generally it will be either the maximum rating (called push attack) or minimum rating (called nuke attack). For the set of selected items I_S , some attacks require identifying a group of items for special treatment during the attack. The special set I_S receives rating as specified by the function σ . The set of filler items I_F is added as specified by the function ρ . Finally, I_N are those items with no ratings in the profile [3]. The details of the four sets of items are described as follows:

I_T : A set of target items with singleton or multiple items, called single-target attack or multi-target attack. The rating is $\gamma(i_j^T)$, generally rated the maximum or minimum value in the entire profiles.

I_S : A set of selected items with specified rating by the function $\sigma(i_k^S)$ [35];

I_F : A set of filler items, generally received randomly items with random assigned ratings $\rho(i_l^F)$;

I_N : A set of items with no ratings.

In this paper, we utilize 14 attack models to generate corresponding attack profiles. The involved attack profiles and explanations are listed in Table 2. Let $N(r, \sigma^2)$ denotes the Gaussian distribution with mean r and standard deviation σ , the details of aforementioned attack models are described as follows:

Table 1: General form of attack profiles.

I_T			I_S			I_F			I_N		
i_1^T	...	i_j^T	i_1^S	...	i_k^S	i_1^F	...	i_l^F	i_1^N	...	i_v^N
$\gamma(i_1^T)$	...	$\gamma(i_j^T)$	$\sigma(i_1^S)$	...	$\sigma(i_k^S)$	$\rho(i_1^F)$	...	$\rho(i_l^F)$	<i>null</i>	...	<i>null</i>

Table 2: Attack models summary.

Attack Models	I_S		I_F		I_N	I_T push or nuke
	Items	Rating	Items	Rating		
Random	null		randomly chosen	normal dist around system mean.	null	r_{max} or r_{min}
Average	null		randomly chosen	normal dist around item mean.	null	r_{max} or r_{min}
Bandwagon (average)	popular items	r_{max} or r_{min}	randomly chosen	normal dist around item mean.	null	r_{max} or r_{min}
Bandwagon (random)	popular items	r_{max} or r_{min}	randomly chosen	normal dist around system mean.	null	r_{max} or r_{min}
Segment	segmented items	r_{max} or r_{min}	randomly chosen	r_{min} or r_{max}	null	r_{max} or r_{min}
Reverse Bandwagon	unpopular items	r_{min} or r_{max}	randomly chosen	system mean	null	r_{max} or r_{min}
Love/Hate	null		randomly chosen	r_{min} or r_{max}	null	r_{max} or r_{min}
AOP	null		x-% popular items, ratings set with normal dist around item mean.		null	r_{max} or r_{min}
PIA-AS	power items, ratings set with normal dist around item mean.		null		null	r_{max} or r_{min}
PIA-ID	power items, ratings set with normal dist around item mean.		null		null	r_{max} or r_{min}
PIA-NR	power items, ratings set with normal dist around item mean.		null		null	r_{max} or r_{min}
PUA-AS	copy ratings and items from power user profiles.		null		null	r_{max} or r_{min}
PUA-ID	copy ratings and items from power user profiles.		null		null	r_{max} or r_{min}
PUA-NR	copy ratings and items from power user profiles.		null		null	r_{max} or r_{min}

1) Random attack: $I_S = \phi$ and $\rho(i) \sim N(\bar{r}, \bar{\sigma}^2)$, where $\bar{r}$ denotes the average rating over all items and users, and $\bar{\sigma}$ denotes the standard deviation of ratings over all items and users [35].

2) Average attack: $I_S = \phi$ and $\rho(i) \sim N(\bar{r}_i, \bar{\sigma}_i^2)$, where $\bar{r}_i$ denotes the average rating of item i over all the users that rated this item, and $\bar{\sigma}_i$ denotes the standard deviation of ratings of item i over all the users who have rated this item [35].

3) Bandwagon (average) attack: I_S contains a set of popular items. And then, we use these items as I_S , $\sigma(i) = r_{max}$ or r_{min} (push or nuke) and $\rho(i) \sim N(\bar{r}_i, \bar{\sigma}_i^2)$ [29].

4) Bandwagon (random) attack: I_S contains a set of popular items, $\sigma(i) = r_{max}$ or r_{min} (push or nuke) and $\rho(i) \sim N(\bar{r}, \bar{\sigma}^2)$ [29].

5) Segment attack: I_S contains a set of segmented items. And then, we use these items as I_S , $\sigma(i) = r_{max}$ or r_{min} (push or nuke) and $\rho(i) = r_{min}$ or r_{max} (push or nuke) [13].

6) Reverse Bandwagon attack: I_S contains a set of unpopular items, $\sigma(i) = r_{min}$ or r_{max} (push or nuke) and $\rho(i) \sim N(\bar{r}, \bar{\sigma}^2)$ [13].

7) Love/Hate attack: $I_S = \phi$ and $\rho(i) = r_{min} \text{ or } r_{max}$ (push or nuke) [13].

8) AOP attack: A simple and effective strategy to obfuscate the Average attack is to choose filler items with equal probability from the top x% of most popular items rather than from the entire collection of items [25].

Power items in the CFRSs are able to influence the largest group of items. The influence is indicated by the ability of power item i to change the recommender system prediction for other user u , or for user u 's target item to appear in user j 's top-N list [25]. Similarly, power users in the CFRSs context are able to influence the largest group of CFRS users. The influence is indicated by the ability of power user u to change the CFRSs prediction for another user v , or for power user u 's target item to appear in user v 's top-N list [25]. It is noteworthy that "power items" and "power users" appear in other kinds of attacks are not all the same. PIA (power item attack) consists of three kinds of attacks including PIA-ID, PIA-AS and PIA-NR, and PUA (power user attack) also consists of three kinds of attacks such as PUA-ID, PUA-AS and PUA-NR. The details of these attack models are described as follows:

1) PIA-AS attack: The top-N items with the highest aggregate similarity (AS) scores become the selected set of power items. This method requires at least 5 users who have rated the same item i and item j [25].

2) PIA-ID attack: Based on In-Degree centrality, power items participate in the highest number of similarity neighborhoods. For each item i compute similarity with every item j applying significance weighting $\frac{n_{cij}}{50}$, where n_{cij} is the number of users that have rated the same items i and j , then discard all but the top-N neighbors for each item i . Count the number of similarity scores for each item j , and select the top-N item j 's [25].

3) PIA-NR attack: Power items are the items with the highest number of user ratings. We select the top-N items based on the total number of user ratings they have in their profile [25].

4) PUA-AS attack: The top 50 users with the highest Aggregate Similarity scores become the selected set of power users. This method requires at least 5 co-rated items between user u and user v and does not use significance weighting [24].

5) PUA-ID attack: Based on the In-Degree centrality concept from social network analysis, power users are those who participate in the highest number of neighborhoods. For each user u compute its similarity with every other user v applying significance weighting, then discard all but the top 50 neighbors for each user u . Count the number of similarity scores for each

user v and select the top 50 user v 's [24].

6) PUA-NR attack: Power users are the users with the highest number of ratings. We selected the top 50 users based on the total number of ratings they have in their user profile [24].

4. Our approach

In this section, we firstly present an overall introduction of our approach. Then, the two main aspects of our work including features extraction from user profiles and RAdaBoost for attack detection are described in detail.

4.1. The framework of our approach

The framework of our approach incorporates four phases as Figure 1 shown: the phase of constructing training dataset and test datasets; the phase of feature extraction; the phase of training classifier via RAdaBoost and the phase of test for generating detection results.

At the phase of constructing training set and test sets, the data sets are constructed by attack profiles (diverse attack models are injected) and genuine profiles. Concretely, for training data set, we use several representative attack models such as Random, Average attacks etc. to generate mixed attack profiles. Specially, we employ the number of attacks (160 attackers for each attack models) when constructing the training data set. (More details can be found in section 5). And then, combining them with genuine profiles

Figure 1: The framework of our approach.

as the our training data set. For test data sets, attack profiles with different filler sizes and attack sizes are inserted into the genuine profiles to form

the test data sets (see subsection 5.1). At the phase of feature extraction, we employ 18 features extracted from user profiles to characterize a feature representation (or feature vector) for each user in both training data set and test data sets. At the phase of training, we use RAdaBoost to train a strong composite estimator (or classifier) based on training features. Finally, we use features retrieved from test data sets as the input into the obtained trained estimator and generate detection results at the phase of testing.

4.2. Feature extraction from user profiles

Previous researches [3, 23, 27, 28] summarized different metrics to characterize the features extracted from user profiles. These features generally fall into two types: generic and type-specific features. The generic features are basic descriptive statistics that attempt to discriminate between attack profiles and genuine profiles and the type-specific features are implemented to detect characteristics of profiles generated by specific attack models or specific signatures of attacks. In the present work, we employ 10 features from these two types. Besides, we also employ 5 features based on the filler size [33] and propose additional 3 new features which measure the distribution of specific rating such as mean rating, maximum rating and minimum rating in filler items for each user.

4.2.1. Generic features

Attack profiles usually have high deviation from the mean value for the target items and low deviation from the mean value for remaining items. Thus, generic features such as RDMA, WDMA etc., are often used to measure the deviation of rating for user profiles [3, 27, 28, 36].

Rating Deviation from Mean Agreement (RDMA): RDMA is intended to detect attackers through examining the profile’s average deviation per item and weighted by the inverse of the number of rating for that item [3].

$$\text{RDMA}_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{NR_i}}{N_u} \quad (4.1)$$

where N_u is the number of ratings that user u has rated and NR_i is the number of ratings provided for item i . $r_{u,i}$ denotes the rating given by user u to item i , $\bar{r}_i$ denotes the mean rating of item i across all users.

Weighted Deviation from Mean Agreement (WDMA): WDMA is strongly based on RDMA, but uses higher weight on the rating deviations for sparse

items [3].

$$\text{WDMA}_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{NR_i^2}}{N_u} \quad (4.2)$$

Weighted Degree of Agreement (WDA): WDA just use the numerator of the RDMA equation without divided by the size of a profile in terms of the number of ratings [3].

$$\text{WDA}_u = \sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{NR_i} \quad (4.3)$$

Length Variance (LengthVar): LengthVar is based on the number of total ratings in a given profile to capture the different between attack profiles and genuine profiles [3].

$$\text{LengthVar}_u = \frac{|n_u - \bar{n}|}{\sum_{k \in U} (n_k - \bar{n})^2} \quad (4.4)$$

where n_u is the total number of ratings in the system for user u . U is the total number of users in the system. $\bar{n}$ is the average length of a profile in the system.

4.2.2. Type-specific features

Model-based methods assume that we have some prior knowledge about the attack models. Based on an assumed model, ratings can be automatically divided into filler items and selected items [3, 23, 27, 28, 36]. Therefore, the measurements including MeanVar, FMTD etc., can be calculated from each subset to measure the authenticity of profiles.

Mean Variance (MeanVar): MeanVar is to iterate through all the lowly-rated (for nuke attack) or highly-rated (for push attack) items, choosing each in turn as the possible target and computing the average variance between the filler items and the overall average for detecting average attack [3].

$$\text{MeanVar}_u = \frac{\sum_{j \in P_{u,F}} (r_{u,j} - \bar{r}_u)^2}{|P_{u,F}|} \quad (4.5)$$

where $P_{u,F}$ is the rest of the profile: $P_u - P_{u,T}$, $P_{u,T} = \{i \in P_u, \text{ such that } r_{u,i} = r_{max}\}$ (or r_{min} for nuke attack), P_u is the profile of user u .

Filler Mean Target Difference (FMTD): FMTD is a partitioning feature for detecting segment attack that maximizes the attack's effectiveness is the difference in ratings of items rated by user u with the highest or lowest rating compared to all items rated by user u except for the items rated with the highest or lowest rating [3].

$$\text{FMTD}_u = \left| \frac{\sum_{i \in P_{u,T}} r_{u,i}}{|P_{u,T}|} - \frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|} \right| \quad (4.6)$$

Target Model Focus (TMF): TMF is calculate the degree to which the partitioning of a given profile focuses on items common to other attack partitions and measures a consensus of suspicion regarding each profile [3].

$$\text{TMF}_u = \max_{j \in P_T} F_j \quad (4.7)$$

where $F_i = (\sum_{u \in U} \phi_{u,i}) / (\sum_{u \in U} |P_{u,T}|)$, and $\phi_{u,i}$ is 1 if $i \in P_{u,T}$, 0 otherwise. P_T denotes the item set of potential targets [28].

Filler Mean Variance (FMV): FMV measures the variance of a user's ratings in the hypothesized filler partition from the mean rating for each of the items [23].

$$\text{FMV}_u = \frac{1}{|U_u^{F_m}|} \sum_{i \in U_u^{F_m}} (r_{u,i} - \bar{r}_i)^2 \quad (4.8)$$

where $U_u^{F_m}$ is the partition of the profile of user u hypothesized to be the set of filler items F by model m . $|U_u^{F_m}|$ is the number of items in the hypothesized filler partition of profile P_u by model m .

Filler Mean Difference (FMD): FMD calculates the mean of the absolute value of the difference between the user rating and the average rating for the hypothesized filler items [23].

$$\text{FMD}_u = \frac{1}{|U_u|} \sum_{i=1}^{|U_u|} |r_{u,i} - \bar{r}_i| \quad (4.9)$$

where U_u is the partition of the profiles of user u . $|U_u|$ is the number of the profiles of user u .

Filler Average Correlation (FAC): FAC computes the correlation between the filler ratings in the profile and the mean rating for each item [23].

$$\text{FAC}_u = \frac{\sum_{i \in I_u} (r_{u,i} - \bar{r}_i)}{\sqrt{\sum_{i \in I_u} (r_{u,i} - \bar{r}_i)^2}} \quad (4.10)$$

where I_u is the set of items rated by user u .

4.2.3. Features based on the filler size

User profiles with different number of ratings will generate different features. Since an attack profile, whose number of ratings deviates far away from the genuine profile, can be detected easily, especially when the filler size of attack profiles is higher than genuine profiles. Then, the number of filler items with total items can characterize partly difference between attack profiles and genuine profiles such as FSTI (filler size with total items). Similarly, the number of rating on different types of items will also generate different features. For bandwagon (average and random) attacks, a set of popular items is chosen by the attackers as the selected items, which causes the number of ratings on popular items deviates far away from the genuine profiles. Therefore, the number of popular items with entire items in the system and all items of individual user profile will generate different measurements to discriminate bandwagon attack profiles and genuine profiles including FSPI (filler size with popular items) and FSPII (filler size with popular items in itself). In the same way, for reverse bandwagon attack, we also employ two different features FSUI (filler size with unpopular items) and FSUII (filler size with unpopular items in itself) based on unpopular items [33]. The definitions of these features are describes as follows:

FSTI: The ratio between the number of items rated by user u and the number of entire items in the recommender system.

$$\text{FSTI}_u = \frac{\sum_{i=1}^{|I|} O(r_{u,i})}{|I|} \quad (4.11)$$

where I is the set of items in the system. $|I|$ denotes the total number of items in the system. $O(r_{u,i})$ is 1 if user u rated item i , 0 otherwise.

FSPI: The ratio between the number of popular items rated by user u and the number of entire popular items in the recommender system.

$$\text{FSPI}_u = \frac{\sum_{i=1}^K O(r_{u,i})}{K} \quad (4.12)$$

where K denotes the boundary point of popular items and unpopular items.

FSPII: The ratio between the number of popular items rated by user u and the number of entire items rated by user u .

$$\text{FSPII}_u = \frac{\sum_{i=1}^K O(r_{u,i})}{\sum_{j=1}^{|I|} O(r_{u,j})} \quad (4.13)$$

FSUI: The ratio between the number of unpopular items rated by user u and the number of entire unpopular items in the recommender system.

$$\text{FSUI}_u = \frac{\sum_{i=K+1}^{|I|} O(r_{u,i})}{|I| - K} \quad (4.14)$$

FSUII: The ratio between the number of unpopular items rated by user u and the number of entire items rated by user u .

$$\text{FSUII}_u = \frac{\sum_{i=K+1}^{|I|} O(r_{u,i})}{\sum_{k=1}^{|I|} O(r_{u,k})} \quad (4.15)$$

4.2.4. Our proposed features

We propose 3 new features which focus on the number of specific ratings (such as the maximum score, minimum score and average score) on filler or selected items. Due to attackers show different attack intents in CFRSs, the filler or selected set of attack profiles may be filled by specific items (i.e., select popular items for Bandwagon (average and random) attacks, select randomly items in the system for Random attack) with the highest score or the lowest score or average score. Take nuke attacks for example, the selected items or filler items are rated with maximum score in Reverse Bandwagon, Segment and Love/Hate attacks (as shown in Table 2). Similarly, the selected items or filler items are rated with minimum score in Bandwagon (average), Bandwagon (random) and Segment attacks. In Random attack, the filler items are rated with some average score (normal distribution around system mean). Therefore, the number of specific ratings can be used to evaluate partly the difference between genuine profiles and attack profiles.

Filler Size with Maximum Rating in Itself (FSMAXRI): The ratio between the number of items rated by user u with maximum score and the number of entire items rated by user u .

$$\text{FSMAXRI}_u = \frac{\sum_{i=1}^{|I|} O(r_{u,i} = r_{\max})}{\sum_{k=1}^{|I|} O(r_{u,k})} \quad (4.16)$$

where $r_{u,i}$ is the rating given by user u to item i , r_{max} is the maximum score in the system. I denotes the set of items in the whole system. $O(r_{u,i} = r_{max})$ is 1 if user u rated item i with rating r_{max} , 0 otherwise. $O(r_{u,k})$ is 1 if user u rated item k , 0 otherwise.

Filler Size with Minimum Rating in Itself (FSMINRI): The ratio between the number of items rated by user u with minimum score and the number of entire items rated by user u .

$$\text{FSMINRI}_u = \frac{\sum_{i=1}^{|I|} O(r_{u,i} = r_{\min})}{\sum_{k=1}^{|I|} O(r_{u,k})} \quad (4.17)$$

where $r_{\min}$ is the minimum score in the system. $O(r_{u,i} = r_{\min})$ is 1 if user u rated item i with rating $r_{\min}$, 0 otherwise.

Filler Size with Average Rating in Itself (FSARI): The ratio between the number of items rated by user u with average score and the number of entire items rated by user u .

$$\text{FSARI}_u = \frac{\sum_{i=1}^{|I|} O(r_{u,i} = r_{\text{avg}})}{\sum_{k=1}^{|I|} O(r_{u,k})} \quad (4.18)$$

where r_{avg} is the average score in the system. $O(r_{u,i} = r_{\text{avg}})$ is 1 if user u rated item i with rating r_{avg} , 0 otherwise.

4.3. Re-scale AdaBoost for attack detection

After the raw user profiles are transformed to a set of sophisticated features, an effective detection method based on these features for “shilling” attacks is crucial. As is known, the number of attackers is usually far smaller than genuine users in CFRSs, thus the supervised learning based attack detection can be formulated as an imbalanced classification, actually. Conventional supervised learning based detection method (i.e., SVM or kNN) often inevitably have individual weaknesses for handling this kind of issues. Under this circumstance, Boosting comes into our sights as it has been proved to be efficient when faced with some difficult scenarios as imbalanced classification [14]. In Boosting, weak learners are fitted iteratively to the training data, using appropriate methods to gradually increase emphasis on observations modelled poorly by the existing collection of weak learners. More specifically, AdaBoost apply weights to the observations (or samples), emphasising poorly modelled ones and gradually (or iteratively, more precisely)

strengthening the correction of misclassifications. The following Algorithm 1 interpret the main idea of AdaBoost [9].

From a statistical view, AdaBoost also can be viewed as a form of “Gradient Boosting Machine” [11]. Consider a loss function in this case, a measure that represents the loss in predictive performance due to a sub-optimal model. Boosting is a numerical optimisation technique for minimising the loss function by adding at each step a new weak learner that best reduces (steps down the gradient of) the loss function. Original gradient Boosting algorithm was proved to be consistent, which can be easily deduced by applying the method in [1] to [20, Theorem1], however, a number of studies [7, 20, 26] also showed that its approximation rate is far slower. The numerical convergence rate of Boosting lies in $(C_0 k^{-0.1898}, C'_0 k^{-0.182})$, which is much slower than the minimax nonlinear approximation rate $\mathcal{O}(t^{-1/2})$. Here and hereafter, t denotes the number of iterations, and C_0, C'_0 are absolute constants.

Algorithm 1 AdaBoost

Step 1: **(Initialization)**: Given data $\{(x_i, y_i) : i = 1, \dots, m\}$, where $x \in R^d$ and $y \in \{-1, +1\}$, weights $\{(w_i^{(1)}) = \frac{1}{m} : i = 1, \dots, m\}$, dictionary $\mathcal{D}_n = \{g_1, \dots, g_n\}$, iteration number T and $f_0 \in \text{span}(\mathcal{D}_n)$.

Step 2: Find $g_t \in \mathcal{D}_n$ such that minimizes the weighted sum error

$$\epsilon_t = Pr_{i \sim w^{(t)}}[g_t(x_i) \neq y_i] = \sum_{i=1}^m w_i^{(t)} 1_{(g_t(x_i) \neq y_i)}$$

for misclassified samples.

Step 3: Choose

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

and update weights

$$w_i^{(t)} = \frac{w_i^{(t-1)} \exp^{-y_i \alpha_t g_t(x_i)}}{Z_t}$$

for all samples, where $Z_t = 2[\epsilon_t(1 - \epsilon_t)]^{1/2}$ is a normalization factor.

Step 4: Add to ensemble $f_t = f_{t-1} + \alpha_t g_t$.

Step 5: Increase t by one and repeat Step 2 and Step 3 if $t < T$.

Recently, Lin et al. [19] and Xu et al. [31] proposed a re-scale Boosting (RBoosting) to improve the performance of original gradient Boosting. Dif-

ferent from the aforementioned strategies that focus on controlling the step-size of g_t^* such as some existing variants like Regularized shrinkage Boosting [8], Regularized truncated Boosting [34], ε -Boosting [16], they cheered a novel direction to improve the numerical convergence rate and consequently, the generalization capability of Boosting. The core idea is that if the approximation (or learning) effect of the t th iteration is not good, then we regard f_t to be too aggressive and therefore shrink it within a certain extent. By such an interesting modification, the optimal numerical convergence of RBoosting can be guaranteed. This means that, RBoosting is among the almost optimal nonlinear approximant and therefore, RBoosting may possess better learning performance than other Boosting-type algorithms. Based on the general idea of RBoosting, the re-scale AdaBoost (RAdaBoost) can be interpreted as the following Algorithm 2.

Algorithm 2 Re-scale AdaBoost

Step 1: **(Initialization)**: Given data $\{(x_i, y_i) : i = 1, \dots, m\}$, where $x \in R^d$ and $y \in \{-1, +1\}$, weights $\{(w_i^{(1)}) = \frac{1}{m} : i = 1, \dots, m\}$, dictionary $\mathcal{D}_n = \{g_1, \dots, g_n\}$, a set of shrinkage degree $\{s_t\}_{t=1}^{t^*}$ where $s_t = 2/(t + u)$, $u \in \mathbf{N}$ is shrinkage degree parameter (or factor), iteration number T and $f_0 \in \text{span}(\mathcal{D}_n)$.

Step 2: Find $g_t \in \mathcal{D}_n$ such that minimizes the weighted sum error

$$\epsilon_t = Pr_{i \sim w^{(t)}}[g_t(x_i) \neq y_i] = \sum_{i=1}^m w_i^{(t)} 1_{(g_t(x_i) \neq y_i)}$$

for misclassified samples.

Step 3: Choose

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

and update weights

$$w_i^{(t)} = \frac{w_i^{(t-1)} \exp^{-y_i \alpha_t g_t(x_i)}}{Z_t}$$

for all samples, where $Z_t = 2[\epsilon_t(1 - \epsilon_t)]^{1/2}$ is a normalization factor.

Step 4: Add to ensemble $f_t = (1 - s_t)f_{t-1} + \alpha_t g_t$.

Step 5: Increase t by one and repeat Step 2 and Step 3 if $t < T$.

Remark 4.1. *In the current work, refer to the general idea of RBoosting and*

well-known AdaBoost, we just applied the RAdaBoost as our detection method and demonstrated the outperformance of RAdaBoost over other competing techniques. The more details of RBoosting including motivation, theoretical and empirical analysis can be found in [19] and [31].

5. Experiments and analysis

In this part, we firstly introduce the experimental settings, including the data sets, evaluation metrics and computational environment. Secondly, the impact of the extracted features are analyzed. Then, we compare the performance of RAdaBoost with three other benchmarked methods such as SVM, kNN and AdaBoost on diverse 4 attack detection methods to demonstrate the outperformance of RAdaBoost. Finally, the remaining 10 types of attacks are conducted by means of RAdaBoost to further evaluate its performance.

5.1. Experimental settings

In our experiments, we use the MovieLens-100K ³ dataset as the data set describing the behaviors of genuine users in recommender system. MovieLens-100K was collected by the GroupLens Research Project at the University of Minnesota. It is the one of the most popular data sets used by researchers and developers in the field of collaborative filtering and attack detection in recommender systems. It consists of 100,000 ratings on 1682 movies by 943 raters and each rater had to rate at least 20 movies. All ratings are in the form of integral values between minimum value 1 and maximum value 5. The minimum score means the rater distastes the movie, while the maximum score means the rater enjoyed the movie. According to the information derived from MovieLens website, the sparse ratio ⁴ of the rating data approximates to 93.7% and the average rating of all users is around 3.53. Besides, the Average Number of Items Rated (ANIR) by each user is approximately 7%.

Attack profiles are generated according to different attack models (as shown in Table 2). The attack profiles indicate the attacker’s intention that he wishes a particular item can be rated the highest or lowest rating. In this paper, we just detect the nuke attacks and the push attacks can be detected in the analogous manner. For each attack model, we generate nuke attack

³<http://grouplens.org/datasets/movielens/>

⁴The ratio between the number of ratings and entire ratings in the rating matrix.

profiles according to the corresponding attack models with different attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$. To ensure the rationality of the results, the target item is randomly selected for each attack profile. In addition, for bandwagon attacks, we select movies $\{50, 56, 100, 127, 174, 181, 258, 286, 288, 294\}$ as the popular movies which are rated by more than 300 users in the system. In segment attack, we use movies $\{50, 183, 185, 200, 234, 443\}$ as the segmented movies [18]. And for Reverse Bandwagon attack, we randomly choose 10 movies as the selected movies which are rated by one user in the system.

For training set, we use the whole MovieLens-100K dataset to generate a attack profiles by exploiting 7 representative known attack models (random, average, bandwagon (average), segment, reverseBandwagon, PIA-ID and PUA-NR) with 17.0% attack size (160 attackers) and diverse filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$. And then, we combine these 7 attack datasets into MovieLens-100K dataset to construct a mixed user profiles as our training data. Thus, the training dataset consists of 943 genuine users and 1120 (160×7) attackers. For test data sets, based on the whole MovieLens-100K dataset, we generate respectively attack profiles by exploiting 14 attack models with different attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$. And then, the generated attack profiles are respectively inserted into genuine profiles to construct our test datasets. Therefore, we have 504 ($14 \times 6 \times 6$) test datasets including 14 attack models, 6 different attack sizes and 6 different filler sizes.

To measure the effectiveness of the proposed detection methods, we use three metrics such as classification error, detection rate and false alarm rate in the test sets, which are used in similar experiments [5]. Classification error is defined as the number of misclassifications divided by the number of all test user profiles.

$$\text{classification error} = \frac{\# \text{Misclassifications}}{\# \text{User Profiles}} \quad (5.1)$$

Detection rate is defined as the number of detected attack profiles divided by the number of attack profiles.

$$\text{detection rate} = \frac{\# \text{Detection}}{\# \text{Attack Profiles}} \quad (5.2)$$

False alarm rate is the number of genuine profiles that are predicted as attack profiles divided by the number of genuine profiles.

$$\text{false alarm rate} = \frac{\# \text{False alarm}}{\# \text{Genuine Profiles}} \quad (5.3)$$

All numerical studies are implemented using MATLAB R2014a on a Windows personal computer with Core(TM) i7-3770 3.40GHz CPUs and RAM 16.00GB.

5.2. Impact of extracted features

To evaluate the impact of the extracted features, we conduct a list of experiments in several attack models with diverse filler sizes as Figure 2 illustrated. We utilize EM (Expectation-maximization) clustering method (Clustering results and EM clustering method were created using Weka ⁵) to separate attackers from genuine users as far as possible based on 10 features (generic and type-specific features), 15 features (additional 5 features based on filler size) and 18 features (all aforementioned features including 3 our proposed features), respectively, in order to analyze the relationship between the number of extracted features and the performance with respect to the filler size. Just as shown in Figure 2, Bandwagon (average), Segment, Reverse Bandwagon and PIA-AS attacks are taken for examples. It is distinctly observed from the results that the false alarm rate significantly decrease with using more extracted features. Furthermore, we take two diagrams to intuitively show the clustering results as shown in Figure 3 (Bandwagon (average) and Segment attacks are taken for examples). By fixing attack size (17.0%, 160 attackers) and filler size (13.3%, 170 items), the strikingly clustering results illustrate that hard classification task becomes easier to perform with more well-designed features employed.

5.3. Experimental results and analysis

First, we compare the detection performance of RAdaBoost with three benchmarked methods such as SVM, kNN and AdaBoost on 4 test attack profiles described above to validate the outperformance of RAdaBoost. The details of setting of each method is described as follows:

⁵<http://www.cs.waikato.ac.nz/ml/weka/>

- SVM: LibSVM with Gaussian kernel and the default parameters are employed as [4] for training binary profile classifier with $Prediction = +1$ if classified as authentic and $Prediction = -1$ if classified as attack. To classify unseen test data sets, the trained SVM model (or classifier) in the training set are used to determine the class label.

- kNN: Standard kNN algorithm is used as [15]. The k nearest neighbors (k is chosen by 5-folds cross validation on the training data set) in the training set are collected for prediction using one over Pearson correlation distance weighting.

Figure 2: Relationship between the number of extracted features and the performance with respect to the filler size. Attack size is 17.0%.

Figure 3: The diagrams of clustering results with diverse features employed, where red nodes denote attackers and green nodes denote genuine users.

- AdaBoost: We utilize decision stumps (with the number of splits $J = 1$) to build up the weak learners for classification. The number of iterations (or the number of stumps to be fitted) is also selected via 5-folds cross validation on the training data set.

- RAdaBoost: For additional shrinkage degree parameter, $s_k = 2/(k + u)$, $u \in \mathbf{N}$, in RBoosting, we create 20 equally spaced values of u in logarithmic space between 1 to 10^6 and select the appropriate u^* as [31]. The other settings are the same as Boosting.

Fig.4-Fig.7 illustrate the performance surfaces of the RAdaBoost classifier for the aforementioned test sets, which contains 4 attack models (take AOP, Bandwagon (random), PIA-NR and PUA-NR attacks for examples) with different attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$. For comparison, the performance surfaces of SVM, kNN and AdaBoost are also presented. It can be easily observed from these figures that, the classification error of SVM rise with the increase of the attack size, which implies the SVM classifier could not effectively classify and detect attacks generated by these 4 types of attack models when the total number of attacks are far smaller than genuine users. Although SVM can achieve fairly high classification performance within some small attack size areas, the detection rates of SVM are also small. It shows that the high prediction accuracy is almost produced by abundant genuine

users but fail to capture the little concerned attackers. In our 4 sets of experiments, only a few bandwagon (random) attack profiles could be detected by SVM and naturally SVM barely false alarmed. kNN essentially outperforms SVM in our 4 types of attack detection methods with lower classification error, much more higher detection rate and pinging false alarm rate. However, we also notice that the classification performance of kNN is still poor and it may fail for detection within some certain attack and filler size areas. Just as figures showed, kNN fail to detect AOP, PIA-NR attacks when the filler size is too small and bandwagon (random) attacks when the attack size is small. And for PUA-NR attack, kNN is just slightly better than SVM and barely detected. For AOP attack detected by kNN, the results may indicate that some genuine profiles are misclassified as attack profiles since a large number of genuine profiles have the same or similar number of popular items as the AOP attack profiles when filler size is too big and small.

Compare with SVM and kNN, AdaBoost significantly improve the classification performance owing to it iteratively strengthen the correction of the misclassifications. And hence, AdaBoost further enhance detection rate with very low false alarm rate. However, just as figures shown, although AdaBoost can effectively detect attack over a wide range of attack and filler size, we also observe that its failure within some certain areas (i.e., AOP and PIA-NR attacks with small attack size and high filler size, bandwagon (random) attacks with small attack and filler size). Especially, AdaBoost can not effectively detect PUA-NR attacks within a large high attack size area. As figures shown that, RAdaBoost additionally improve the classification performance of AdaBoost by imposing a re-scale operator and consequence enhance detection rate with negligible false alarm rate in the 4 types of attacks. So far, all the comparative experimental results illustrate that the RAdaBoost outperforms Boosting and conventional supervised learning based detection methods including SVM and kNN.

Finally, to further evaluate the effectiveness of RAdaBoost, we also conduct other 10 types of attacks to show the performance surfaces of RAdaBoost just as Fig.8 illustrated. From results, we can distinctly observe that, except for PUA-AS and PUA-ID attacks, RAdaBoost can effectively detect all of the attacks with almost no false alarm. Although it also shows low detection rates for some attacks with small attack and filler size. Comparing with previous research results [3, 28, 32], the detection performance of RAdaBoost is more optimistic. For PUA-AS and PUA-ID attacks, which are recently published attack models and few researchers pay close attention to

Figure 4: The classification error, detection rate and false alarm rate of RAdaBoost on different test sets in comparison with SVM, kNN and AdaBoost. AOP attack with diverse attack sizes {1.1%, 6.4%, 11.7%, 17.0%, 22.3%, 27.6%} and filler sizes {1.2%, 4.2%, 7.3%, 10.3%, 13.3%, 16.4%}.

Bandwagon (random) Model

SVM

kNN

AdaBoost

RAdaBoost

Figure 5: The classification error, detection rate and false alarm rate of RAdaBoost on different test sets in comparison with SVM, kNN and AdaBoost. Bandwagon (random) attack with diverse attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$.

Figure 6: The classification error, detection rate and false alarm rate of RAdaBoost on different test sets in comparison with SVM, kNN and AdaBoost. PIA-NR attack with diverse attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$.

Figure 7: The classification error, detection rate and false alarm rate of RAdaBoost on different test sets in comparison with SVM, kNN and AdaBoost. PUA-NR attack with diverse attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$.

Figure 8: The detection rate and false alarm rate of RAdaBoost in 10 different attack models with diverse attack sizes $\{1.1\%, 6.4\%, 11.7\%, 17.0\%, 22.3\%, 27.6\%\}$ and filler sizes $\{1.2\%, 4.2\%, 7.3\%, 10.3\%, 13.3\%, 16.4\%\}$.

them. Just as figures shown, the RAdaBoost can not effectively detect such attacks mainly because the present extractive features (as described in section 4) are not enough to depict their material characteristics. Therefore, the results indicate the adaptively new classification features are needed for detecting such new attacks as PUA-AS, PUA-NR and PUA-ID.

6. Conclusion and future work

“Shilling” attacks are serious threats to the collaborative filtering recommender systems (CFRSs). Since the number of detected attackers is far smaller than genuine users. Conventional supervised learning based detection methods have the challenges faced with this imbalanced classification. In the present paper, we improved the detection performance in two directions. First, we extracted features from user profiles based on the statistical properties of the diverse attack models to make them much more easily classified. Then, we applied a variant of Boosting algorithm, called the re-scale AdaBoost (RAdaBoost) as our detection method, which gradually increasing emphasis on concerned attacks and could distinctly improve the predictive performance on a difficult classification task. And all our experimental results also demonstrated the outperformance of RAdaBoost in “shilling” attacks detection.

In our future work, we will explore more simpler and effective features to characterize attack profiles from different perspectives. The existing features based on basic description statistics and model-specific are difficult to fully discriminate between attack profiles and genuine profiles in diverse attack models. In addition, some features based on global calculating similarity such as DegSim (Degree of Similarity with Top Neighbors) are unrealistic in mass user profiles, although they are effective to capture the concerned attack profiles in some extent. Therefore, how to extract local and effective features from user profiles is still an open issue.

Remark 6.1. *Collaborative filtering recommendation techniques have been widely used in the existing E-commerce services [2, 3, 5, 21]. Practical collaborative filtering recommendation algorithms including user-based collaborative filtering (UBCF) [38], item-based collaborative filtering (IBCF) [22] and hybrid recommendation [37], have been developed for making high recommendation accuracy. Since CFRSs are highly vulnerable to “shilling” attacks, defending such attacks before recommendations is really crucial. Our*

current work presents an off-line detection method to spot anomalous users on experimentally generated datasets and the empirical results demonstrated its effectiveness. It is noteworthy that discovering abnormal ratings on real-world datasets such as Amazon, TripAdvisor etc., is also interesting and worthwhile [12]. Related concerns can be approximately summarized in two aspects: a) Detecting negative ratings on products to improve the quality or remove anomalous ratings; b) Examining the distribution of ratings over time for services such as hotel, airport etc., to improve the quality of management. In brief, how to develop an on-line detection method and directly dealing with those real-world datasets will be considered in our future work.

7. Acknowledgements

The research was supported by National Basic Research Program (973 Program) (2013CB329404), Key Project of National Science Foundation of China (Grant No. 11131006, 11401462, 61175039, 61375040, 61221063 and 91330204), International Research Collaboration Project of Shaanxi Province (2013KW11) and Fundamental Research Funds for Central Universities (2012jdhz08). Two anonymous reviewers have carefully read this paper and have provided to us numerous constructive suggestions. As a result, the overall quality of the paper has been noticeably enhanced, to which we feel much indebted and are grateful.

References

- [1] A. Barron, A. Cohen, W. Dahmen, and R. DeVore. Approximation and learning by greedy algorithms. *Ann. Stat.*, 36(1):64–94, 2008.
- [2] K. Bryan, M. OMahony, and P. Cunningham. Unsupervised retrieval of attack profiles in collaborative recommender systems. *ACM conference on Recommender Systems*, pages 155–162, 2008.
- [3] R. Burke, B. Mobasher, and C. Williams. Classification features for attack detection in collaborative recommender systems. *International Conference on Knowledge Discovery and Data Mining*, pages 17–20, 2006.
- [4] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

- [5] C. Chung, P. Hsu, and S. Huang. A novel approach to filter out malicious rating profiles from recommender systems. *Journal of Decision Support Systems*, pages 314–325, 2013.
- [6] M. W. David. Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *Journal of Machine Learning Technologies*, 2011.
- [7] R. DeVore and V. Temlyakov. Some remarks on greedy algorithms. *Adv. Comput. Math.*, 5(1):173–187, 1996.
- [8] J. Ehrlinger and H. Ishwaran. Characterizing l_2 boosting. *Ann. Stat.*, 40(2):1074–1101, 2012.
- [9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. pages 23–37, 1995.
- [10] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. *ICML*, 96:148–156, 1996.
- [11] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, 29(5):1189–1232, 2001.
- [12] S. Gnnemann, N. Gnnemann, and C. Faloutsos. Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. *In KDD’2014*, pages 841–850, 2014.
- [13] I. Gunes, C. Kaleli, A. Bilge, and H. Polat. Shilling attacks against recommender systems: A comprehensive survey. *Artificial Intelligence Review*, pages 1–33, 2012.
- [14] P. Harrington. *Machine Learning in Action*. 2012.
- [15] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning. *New York: springer*, 2001.
- [16] T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electron. J. Stat.*, 1:1–29, 2007.
- [17] F. He, X. Wang, and B. Liu. Attack detection by rough set theory in recommendation system. *IEEE International Conference on Granular Computing*, pages 692–695, 2010.

- [18] C. Li and Z. Luo. Detection of shilling attacks in collaborative filtering recommender systems. *International Conference of Soft Computing and Pattern Recognition*, pages 190–193, 2011.
- [19] S. B. Lin, Y. Wang, and L. Xu. Re-scale boosting for regression and classification. *ArXiv:1505.01371*, 2015.
- [20] E. D. Livshits. Lower bounds for the rate of convergence of greedy algorithms. *Izvestiya: Mathematics*, 73(6):1197–1215, 2009.
- [21] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering. *In: IUI07: Proceedings of the 12th International Conference on Intelligent User Interfaces*, pages 14–21, 2007.
- [22] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based effective attack models for shilling item-based. *WebKDD’05*, 2005.
- [23] M. Morid and M. Shajari. Defending recommender systems by influence analysis. *Information Retrieval*, pages 137–152, 2014.
- [24] C. E. Seminario and D. C. Wilson. Assessing impacts of a power user attack on a matrix factorization collaborative recommender system. *Florida Artificial Intelligence Research Society Conference*, 2014.
- [25] C. E. Seminario and D. C. Wilson. Attacking item-based recommender systems with power items. *ACM Conference on Recommender Systems*, pages 57–64, 2014.
- [26] V. Temlyakov. Greedy approximation. *Acta Numer.*, 17:235–409, 2008.
- [27] C. A. Williams, B. Mobasher, and R. Burke. Defending recommender systems: detection of profile injection attacks. *SOCA*, pages 157–170, 2007.
- [28] C. A. Williams, B. Mobasher, R. Burke, and R. Bhaumik. Detecting profile injection attacks in collaborative filtering: a classification-based approach. *Advances in Web Mining and Web Usage Analysis*, pages 167–186, 2007.

- [29] Z. A. Wu, Y. Q. Wang, and J. Cao. A survey on shilling attack models and detection techniques for recommender systems. *Science China*, 59(7):551–560, 2014.
- [30] Z. A. Wu, J. J. Wu, J. Cao, and D. C. Tao. Hysad: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 985–993, 2012.
- [31] L. Xu, S. B. Lin, Y. Wang, and Z. B. Xu. Shrinkage degree in l_2 -re-scale boosting for regression. *Submitted to IEEE Trans. Neural Netw. & Learn. Syst. (ArXiv:1505.04369)*, 2015.
- [32] F. Zhang and Q. Zhou. A meta-learning-based approach for detecting profile injection attacks in collaborative recommender systems. *J. Comput.*, 7(1):226–234, 2012.
- [33] F. Zhang and Q. Zhou. HHT-SVM: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems*, 2014.
- [34] T. Zhang and B. Yu. Boosting with early stopping: convergence and consistency. *Ann. Stat.*, 33(4):1538–1579, 2005.
- [35] Z. Zhang and S. Kulkarni. Graph-based detection of shilling attacks in recommender systems. *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2013.
- [36] Z. Zhang and S. R. Kulkarni. Detection of shilling attacks in recommender systems via spectral clustering. *International Conference on Information Fusion*, pages 1–8, 2014.
- [37] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, and J. Lu. A hybrid fuzzy-based personalized recommender system for telecom products/services. *Information Sciences*, 235:117–129, 2013.
- [38] Z. D. Zhao and M. S. Shang. User-based collaborative-filtering recommendation algorithms on hadoop. *Knowledge Discovery and Data Mining*, pages 478–481, 2010.