A Meta-learning-based Approach for Detecting Profile Injection Attacks in Collaborative Recommender Systems

Fuzhi Zhang
School of Information Science and Engineering, Yanshan University, Qinhuangdao, China
Email: xjzfz@ysu.edu.cn

Quanqiang Zhou
School of Information Science and Engineering, Yanshan University, Qinhuangdao, China
Email: zhouqiang128@126.com

Abstract—**Recent research has shown the significant vulnerabilities of collaborative recommender systems in the face of profile injection attacks, in which malicious users insert fake profiles into the rating database in order to bias the system's output. A single Support Vector Machine (SVM) approach for the detection of profile injection attacks, however, suffers from low precision. With this problem in mind, in this paper we propose a meta-learning-based approach to detect such attacks. In particular, we propose an algorithm to create the diverse base-level training sets through flexible combination of various attack types. Combining the created training sets with SVM, we construct the base-level and meta-level classifiers. Based on these classifiers, we present a meta-learning-based detection algorithm which uses the meta-classifier to integrate the outputs of the base-classifiers and generates the final results of detection. The diversities among the base-classifiers effectively reduce the correlation of the misclassifications and improve the predictive capability of the meta-level. We conduct comparative experiments with a single SVM and the voting-based ensemble method on different-scale MovieLens datasets. The experimental results show that the proposed approach can effectively improve the precision under the condition of holding a high recall.**

Index Terms—**meta-learning, support vector machine, profile injection attacks, attack detection, collaborative filtering recommendation**

I. Introduction

Collaborative filtering recommender systems [1] can filter out the information to satisfy the users' interest according to the established user profiles and actively recommend the information to users. It provides an effective way to solve the information overload problem on the Internet and it has become an important part of many e-commerce sites. Due to its natural openness, however, some malicious users artificially inject a large number of fake profiles into the system in order to bias the recommendation results to their own advantage. These attacks, where a malicious user inserts fake profiles into the rating database to influence the system's recommendation behavior, have been termed "shilling" attacks or "profile injection attacks" [2]. To distinguish the genuine profiles, we usually call the fake profiles as attack profiles. Attack model is an approach for attackers to construct the attack profiles according to the knowledge about the recommender system's rating database, products, users, etc. According to its construction strategy, the attack model can be divided into the random attack, bandwagon attack, average attack, segment attack, etc [3-5]. For the different purposes of attacks, profile injection attacks can be divided into push attack and nuke attack [6], which increase and decrease the recommendation frequency of the target item respectively. The strength of profile injection attacks is measured by attack size and filler size [7]. Attack size is the ratio between the number of attack profiles and the number of genuine profiles in a recommender system. Filler size is the ratio between the number of ratings in an attack profile and the number of all items in the recommender system. In the face of the various attack types, therefore, how to effectively identify and resist profile injection attacks has become an urgent problem to be solved for the well development and extensive application of collaborative filtering recommender systems.

Recently, the detection of profile injection attacks has become a hot research area in collaborative filtering recommender systems. Chirita et al. [8] proposed several statistical features to describe attack profiles for the detection of high-density attack profiles. Su et al. [9] developed a similarity spreading algorithm to detect simple groups of attack profiles. Mehta et al. [10-12] presented a PCA-based detection algorithm by using Principal Component Analysis (PCA) technique to filter out the attack profiles. PCA-based algorithm is effective

Manuscript received May 2, 2011; revised August 2, 2011; accepted August 9, 2011.

Supported by the Natural Science Foundation of Hebei Province, China (F2011203219) and the Special Fund for Fast Sharing of Science Paper in Net Era by CSTD of Ministry of Education, China (20101333110013).

for detecting various attack types. But it needs to know the total number of attack profiles injected into the rating database of the recommender system before detecting, which is difficult to estimate in practice. Bryan et al. [13] proposed a detection algorithm, which is called UnRAP, to identify the attack profiles by introducing the variance adjusted mean square residue. This algorithm can detect random attack and average attack successfully, but it can not detect bandwagon attack. Hurley et al. [14] designed supervised and unsupervised Neyman-Pearson detectors based on statistical detection theory. These detectors can only detect the attack profiles which have the specific distributions. He et al. [15] used rough set theory to detect attack profiles. They simply take the feature values of user profiles as the condition attributes of the decision table to perform the operations of reducing data and generating rules. This method can detect most of the attack profiles, but the precision is low. Burke et al. [16-18] trained three supervised classifiers (i.e. KNN, C4.5 and SVM) to detect the attacks by extracting the features of user profiles. Of the three classifiers, SVM has the best detection performance. To effectively identify the various attack profiles in practice, which are constructed by various attack models at various attack sizes and filler sizes, the training set of SVM needs to contain enough attack samples. Although this operation can ensure that the classifier has the capability of detecting most of the attack profiles in the recommender system, the classifier's precision is poor due to the fact that too many genuine profiles are misclassified as attack profiles. In our previous work, we presented an improved PCA-based detection algorithm by introducing the normal cloud model theory and explored its effectiveness when detecting profile injection attacks with small attack sizes [19][20]. In [21], we proposed an anomaly detection algorithm based on analyzing rating distribution characteristics of item over rating time series and we showed our algorithm to be more effective at detecting target items than the method described in [22].

In this paper, we propose a meta-learning-based approach to improve the precision of detecting profile injection attacks. Our contributions mainly include:

(1) We propose an algorithm to create diverse base-level training sets;
(2) We present a meta-learning-based detection algorithm to improve the overall performance of attack detection;
(3) We conduct simulation experiments on different-scale MovieLens datasets. The experimental results show that the proposed approach can not only hold a high recall, but also effectively improve the precision of the detection.

II. Theoretical Foundation

Meta-learning is an ensemble learning approach. The underlying idea of this approach is based on relearning the existed knowledge to boost overall predictive effectiveness. The meta-classifier (or level-1 model) tries to acquire how the outputs of the base-classifiers (or level-0 models) should be combined to obtain the final classification [23].

Stacked Generalization [24] is one of the most commonly used meta-learning methods. The details of this method are given as follows.

Given a dataset $\mathcal{S} = \{(y_n, \boldsymbol{x}_n) \mid n = 1, 2, ..., N\}$, where y_n is the class value and $\boldsymbol{x}_n$ is a vector representing the attribute values of the nth instance, N is the number of instances in $\mathcal{S}$, this method randomly splits the dataset into J almost equal parts $\{\mathcal{S}_1, ..., \mathcal{S}_J\}$. $\mathcal{S}_j$ and $\mathcal{S}^{(-j)} = \mathcal{S} - \mathcal{S}_j$ are defined to be the test and training set for the jth fold of a J-fold cross-validation. $\mathcal{S}^{(-j)}$ is used to train K classification algorithms $\{\mathcal{A}_1, ..., \mathcal{A}_K\}$ to generate K classification models $\{\mathcal{M}_1, ..., \mathcal{M}_K\}$ respectively, where $\{\mathcal{A}_1, ..., \mathcal{A}_K\}$ are called level-0 algorithms, $\{\mathcal{M}_1, ..., \mathcal{M}_K\}$ are called level-0 models or base-classifiers.

For each instance $\boldsymbol{x}_j$ in $\mathcal{S}_j$, let $\{\mathcal{M}_1(\boldsymbol{x}_j), ..., \mathcal{M}_K(\boldsymbol{x}_j)\}$ denote the predictions of the models $\{\mathcal{M}_1, ..., \mathcal{M}_K\}$ on $\boldsymbol{x}_j$, then, all the predictive results create a new dataset:

$$\mathcal{S}_{meta} = \{(y_j, \mathcal{M}_1(\boldsymbol{x}_j), \ldots, \mathcal{M}_K(\boldsymbol{x}_j)) \mid j = 1, \ldots, L\} \tag{1}$$

where L is the number of instances in $\mathcal{S}_{meta}$. $\mathcal{S}_{meta}$ is called level-1 training set or meta-level training set. $\mathcal{S}_{meta}$ is used to train a learning algorithm to generate a classification model $\mathcal{M}_{meta}$, where $\mathcal{M}_{meta}$ is called level-1 model or meta-classifier, the learning algorithm is called level-1 generalization algorithm.

Given a new instance q, the level-0 models produce a vector $(\mathcal{M}_1(q), ..., \mathcal{M}_K(q))$. This vector is input to the level-1 model $\mathcal{M}_{meta}$, whose output is the final predictive result for q.

III. Meta-learning-based Approach for Attack Detection

To facilitate the discussions in this section, we introduce some symbols: $\mathbb{I}$ is the set of items in the recommender system, $\boldsymbol{x}$ is a rating vector in a user profile and $\boldsymbol{x} = (rating_1, ..., rating_i, ..., rating_{\mathbb{I}})$, where $rating_i \in \varnothing \cup \{rating_{min}, ..., rating_{max}\}$, $y_n \in \{-1, 1\}$ is the class value of the nth instance, where -1 represents a genuine profile and 1 represents an attack profile. $f_k(\boldsymbol{x}) = \mathrm{SVM}_k(\boldsymbol{x})$ denotes the prediction of the kth classifier SVM_k on $\boldsymbol{x}$.

The framework of the meta-learning-based approach for detecting profile injection attacks is depicted in Figure 1. The detection model of meta-learning is in the dashed box. This model contains two training processes which are base-level training and meta-level training. Both levels use SVM as their learning algorithms. The rating database and the attack profiles are input to the proposed algorithm for creating the base-level training sets.

$\{f_1(\boldsymbol{x}),...,f_K(\boldsymbol{x})\}$, which are the predictions of the K base-classifiers $\{\text{SVM}_1,...,\text{SVM}_K\}$ on $\boldsymbol{x}$, are input to SVM_{meta}. The final predictive results of the test set are generated by SVM_{meta}.

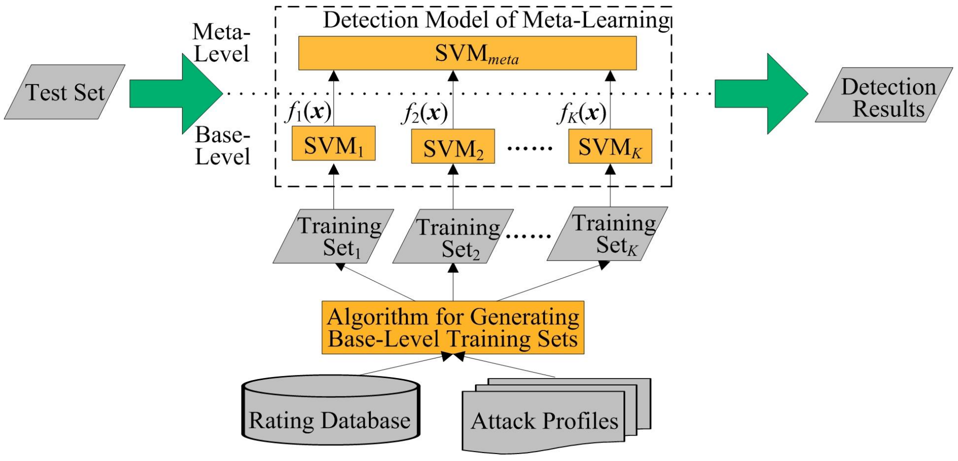

Figure 1. Framework of the meta-learning-based approach for detecting profile injection attacks

A. Construction of Base-level Classifiers

To improve the predictive quality of the meta-level model, the base-classifiers have to be diverse [25]. Stacked Generalization uses the strategy of cross-validation to create the base-training sets. Although this strategy can create different base-training sets, serious skew may be introduced to these sets between the attack profiles and the genuine profiles. In the skewed training sets, the training samples can not accurately reflect the data distributions of the entire space and the classifiers trained from these sets would be weak [26]. To balance the proportion of the training samples between attack profiles and genuine profiles, we propose an algorithm to create the diverse base-level training sets through flexible combinations of various attack models at various attack sizes and filler sizes.

Let $\text{A}=\{a_1\%,...,a_P\%\}$ denote P different attack sizes, $\text{B}=\{b_1\%,...,b_R\%\}$ denote R different filler sizes, $\text{C}=\{c_1,...,c_W\}$ denote W different attack models and i_{target} denote the target item. Let quad $(a\%,b\%,c,i_{target})$ denote the set of attack profiles which are constructed by attack model c at attack size of $a\%$ across filler size of $b\%$ and their target item is i_{target}. Let getTargetItem($\mathbb{I}$) be a function to randomly select an item from $\mathbb{I}$. Let $\text{T}=\{(y_n,\boldsymbol{x}_n)\,|\,n=1,2,\dots,N\}$ denote the rating database of the recommender system, where N is the number of user profiles in T which dose not contain attack profiles. Let $K=P\times R$ denote the number of base-training sets and Z denote the set of base-training sets. The algorithm for creating the base-training sets is described as follows:

Algorithm 1 Flexible method for creating the base-training sets
Input: A, B, C, P ,R ,W, T, $\mathbb{I}$, i_{target}, K
Output: Z

```
1:Z← $\varnothing$
2:Mark← $\varnothing$
3:For $k$ = 1 to $K$ Do
4: $\text{T}_{Attack\ Profile}\leftarrow\varnothing$
5: $\text{T}_{Genuine\ Profile}\leftarrow\varnothing$
6: flag←0
7: For $p$ = 1 to $P$ Do
8:   For $r$ = 1 to $R$ Do
9:     For $w$ = 1 to $W$ Do
10:      $i_{target}$←getTargetItem($\mathbb{I}$)
11:      $\text{T}\leftarrow\text{T}\cup\{(y_e,\boldsymbol{x}_e)|\,\boldsymbol{x}_e\in(a\%, b\%, c, i_{target})\}$
   /* Insert attack profiles into the rating database */
12:      $\text{T}_{Attack\ Profile}\leftarrow\text{T}_{Attack\ Profile}\cup$
          $\{(y_e,\boldsymbol{x}_e')|\,\boldsymbol{x}_e'$ is $\boldsymbol{x}_e$ after features extraction
          of T}
                /* Create samples of attack profiles */
13:      If((($p$,$r$,$w$) $\notin$ Mark) and (flag < $W$)) Then
14:        $\text{T}_{Genuine\ Profile}\leftarrow\text{T}_{Genuine\ Profile}\cup$
            $\{(y_n,\boldsymbol{x}_n')|\,\boldsymbol{x}_n'$ is $\boldsymbol{x}_n$ after features extraction
            of T}
              /* Create samples of genuine profiles */
15:        Mark←Mark ∪ {($p$,$r$,$w$)}
16:        flag←flag+1
17:      End If
18:      $\text{T}\leftarrow\text{T}-\{(y_e,\boldsymbol{x}_e)|\,\boldsymbol{x}_e\in(a\%, b\%, c, i_{target})\}$
19:    End For
20:  End For
21: End For
22: Z←Z ∪ {$\text{T}_{Attack\ Profile}\cup\text{T}_{Genuine\ Profile}$}
23:End For
24:Return Z
```

In the stage of features extraction for user profiles, we use the detection attributes described in [17] to extract 13 features as follows:

(1) Six generic features are WDMA, RDMA, WDA, Length Variance, DegSim (k=450) and DegSim′ (k=2, d=963) respectively. These features attempt to make an attack profile look different from a genuine profile through capturing some of the characteristics.

(2) Seven attack model features are FMD (random attack, push), FAC (random attack, push), FMD

(bandwagon attack, push), FAC (bandwagon attack, push), FMV (average attack, push), FMD (average attack, push) and PV (average attack, push) respectively. These features are designed to recognize the characteristics of known attack models.

Let $\{V_1(\boldsymbol{x}_h),...,V_{13}(\boldsymbol{x}_h)\}$ denote the feature values of $\boldsymbol{x}_h$ in a base-training set. The *k*th base-training set is:

$$\mathrm{z}_k = \{(y_h, V_1(\boldsymbol{x}_h),...,V_{13}(\boldsymbol{x}_h)) \mid h = 1,\ldots,H\} \qquad (2)$$

where H is the number of samples in z_k.

All the base-training sets are as follows:

$$\mathrm{Z} = \{\mathrm{z}_1,\ldots,\mathrm{z}_K\}. \qquad (3)$$

The base-level SVMs learn the base-training sets respectively in Z to generate the base-classifiers $\{\mathrm{SVM}_1,...,\mathrm{SVM}_K\}$.

In Algorithm 1, the base-training sets still contain enough samples of attack profiles in order to effectively identify the various attack profiles. That is to say, we do not attempt to reduce the misclassifications of the base-classifiers, but to create the diverse base-training sets under the condition that the recommender system is attacked by various attack types. These base-training sets can increase the diversities of the base-classifiers, as well as reduce the correlations of the misclassifications. Furthermore, the predictive capability of the meta-level can be improved by using these diverse base-classifiers.

B. Construction of Meta-level Classifier

To create the meta-training set, we set K=1 in Algorithm 1 to create a base-training set:

$$\mathrm{z} = \{(y_l, V_1(\boldsymbol{x}_l),...,V_{13}(\boldsymbol{x}_l)) \mid l = 1,...,L\} \qquad (4)$$

where L is the number of samples in z.

The set z is input to the base-classifiers $\{\mathrm{SVM}_1,...,\mathrm{SVM}_K\}$, whose outputs are used as the meta-training set:

$$\mathrm{z}_{meta} = \{(y_l, f_1(\boldsymbol{x}_l),..., f_K(\boldsymbol{x}_l)) \mid l = 1,...,L\}. \qquad (5)$$

We still select SVM as the learning algorithm of the meta-level. SVM has a good generalization performance and it can find a decision surface that optimally separates the instances into two classes based on the Structural Risk Minimization Principle. The decision function is as follows [27]:

$$f(\boldsymbol{x}) = sign\left(\sum_{i=1}^{M} y_i a_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right) \qquad (6)$$

where, M is the number of training samples, $\boldsymbol{x}$ is the vector of a test sample, $\boldsymbol{x}_i$ and y_i denote the vector and class value of the *i*th training sample, $K(\boldsymbol{x}, \boldsymbol{x}_i)$ is a kernel function, a_i and b are the parameters of the model. a_i can be learned by solving following quadratic programming problem:

$$\max Q(a) = \sum_{i=1}^{M} a_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} a_i a_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$s.t. \sum_{i=1}^{M} a_i y_i = 0, 0 \le a_i \le C, i = 1,\ldots,M. \qquad (7)$$

Since the outputs of the base-classifiers are binary, a lot of input data of the meta-level are reduplicative. The special advantages of SVM are that SVM is only sensitive to the data on the boundary and it only uses the support vectors to predict the class value for a new instance. The existence of the reduplicative data does not affect the classification performance of this algorithm.

In the end, the meta-level SVM learns z_{meta} to generate the meta-classifier SVM_{meta}.

C. Meta-learning-based Algorithm for Attack Detection

Based on the constructed classifiers $\{\mathrm{SVM}_1,...,\mathrm{SVM}_K\}$ and SVM_{meta}, we present an algorithm to detect profile injection attacks.

Let $\mathrm{T}_{test} = \{(y_q, \boldsymbol{x}_q) \mid q = 1,...,Q\}$ denote the test set, where Q is the number of user profiles in T_{test} and y_q is unseen to the classifiers. Let T_{result} denote the set of the final detection results. The algorithm for detecting profile injection attacks is described as follows:

```
Algorithm 2 The meta-learning-based algorithm for
attacks detection
Input: $\mathrm{T}_{test}$
Output: $\mathrm{T}_{result}$
 1:$\mathrm{T}_{result} \leftarrow \varnothing$
 2:$\mathrm{T}_{test}' \leftarrow \varnothing$
 3:$\mathrm{T}_{test}' \leftarrow \{(y_q, \boldsymbol{x}_q') \mid \boldsymbol{x}_q'$ is $\boldsymbol{x}_q$ after features extraction
            of $\mathrm{T}_{test}\}$
 4:For $q$=1 to Q Do
 5:   For $k$=1 to $K$ Do
 6:      $f_k(\boldsymbol{x}_q) \leftarrow \mathrm{SVM}_k(\boldsymbol{x}_q')$
        /*Generate the outputs of the base-classifiers */
 7:   End For
 8:   $f_{meta}(\boldsymbol{x}_q) \leftarrow \mathrm{SVM}_{meta}(f_1(\boldsymbol{x}_q),\ldots,f_K(\boldsymbol{x}_q))$
        /* Generate the output of the meta-classifier */
 9:   $\mathrm{T}_{result} \leftarrow \mathrm{T}_{result} \cup \{f_{meta}(\boldsymbol{x}_q)\}$
10:End For
11:Return $\mathrm{T}_{result}$
```

In Algorithm 2, the base-classifiers firstly generate their classification results. Then, the meta-classifier integrates these results and outputs the final predictions for the test set. Since each base-classifier is obtained using different knowledge, the hypothesis space is explored differently. Thus, the correlations of the misclassifications are effectively reduced. The combination in the meta-level performs better than the base-classifier and it also performs better than the simple voting-based ensemble method. In our experiments below, we show support for this intuition.

IV. Experiments and Evaluations

A. Experimental Data and Settings

We select two different-scale MovieLens[1] datasets as the experimental data in this paper:

(1) MovieLens 100K dataset. This dataset consists of 100,000 ratings on 1,682 movies by 943 users. All ratings are integer values between 1 and 5, where 1 is the lowest (disliked) and 5 is the highest (most liked). Each user in this dataset has rated at least 20 movies. We partition the dataset in half. The first half is used as the parameter T in Algorithm 1 to create base-training sets. To create each test set, the second half is injected with various attack profiles.

(2) MovieLens 1M dataset. This dataset consists of 1,000,209 ratings on 3,900 movies by 6,040 users. All ratings are integer values between 1 and 5, where 1 is the lowest (disliked) and 5 is the highest (most liked). We randomly select 1,000 user profiles from this dataset as the parameter T in Algorithm 1 to create base-training sets. To create each test set, we randomly select 1,000 user profiles to form a new set from the remaining dataset and then inject various attack profiles into this new set.

In the stage of training, we set A={1%, 2%, 5%, 10%}, B={1%, 3%, 5%, 10%, 25%}, C={random attack, bandwagon attack, average attack} in Algorithm 1 to create the base-training sets. Then, we combine these sets with the methods described in section III.*A* and section III.*B* to generate the base-classifiers and the meta-classifier. These classifiers are generated using libsvm[2].

To create the test sets, the attack profiles, which are constructed by the attack models of random, bandwagon and average attack at filler sizes of 3% and 5% across attack sizes of 1%, 2%, 5% and 10%, are individually injected into the second half of MovieLens 100K dataset and the second 1,000 user profiles of MovieLens 1M dataset described above. We randomly select 50 movies as the target items for each test. Each of these movies is attacked individually and the average is reported for all experiments. So, the final metric values in section IV.*C* and section IV.*D* are the average values of these experiments.

We only detect the push attack in this paper. Through changing the ratings of the target item from maximum to minimum, we can use the same method to detect the nuke attack.

B. Evaluation Metrics

To measure the detection performance, we use the standard measurements of recall and precision [18]:

$$\text{Recall} = \frac{TP}{TP+FN}, \tag{8}$$

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

where, *TP* is the number of attack profiles correctly detected, *FP* is the number of genuine profiles misclassified as attack profiles, *FN* is the number of attack profiles misclassified as genuine profiles.

C. Experimental Results and Analysis on MovieLens 100K Test Sets

To verify the effectiveness of the proposed approach, the test sets are detected individually by three detection method which are Algorithm 2 (called Meta SVM), the method of a single SVM (called single SVM) in [17] and the voting method (called Voting SVM) which ensembles the outputs of the base-classifiers through majority voting method [28].

The detection results of these three methods on the MovieLens 100K test sets are shown in Table Ⅰ, Table Ⅱ, Figure 2 and Figure 3.

TABLE Ⅰ.

RECALL ON MOVIELENS 100K TEST SETS FOR VARIOUS ATTACK MODELS AT FILLER SIZE=3% ACROSS VARIOUS ATTACK SIZES

Attack Model	Random Attack				Bandwagon Attack				Average Attack			
Attack Size	1%	2%	5%	10%	1%	2%	5%	10%	1%	2%	5%	10%
Single SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Voting SVM	0.98	0.98	0.99	1.00	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00
Meta SVM	0.99	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.99	0.99	0.99

TABLE Ⅱ.

RECALL ON MOVIELENS 100K TEST SETS FOR VARIOUS ATTACK MODELS AT FILLER SIZE=5% ACROSS VARIOUS ATTACK SIZES

Attack Model	Random Attack				Bandwagon Attack				Average Attack			
Attack Size	1%	2%	5%	10%	1%	2%	5%	10%	1%	2%	5%	10%
Single SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Voting SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
Meta SVM	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99

[1]http://www.grouplens.org/node/73
[2]http://www.csie.ntu.edu.tw/~cjlin/libsvm

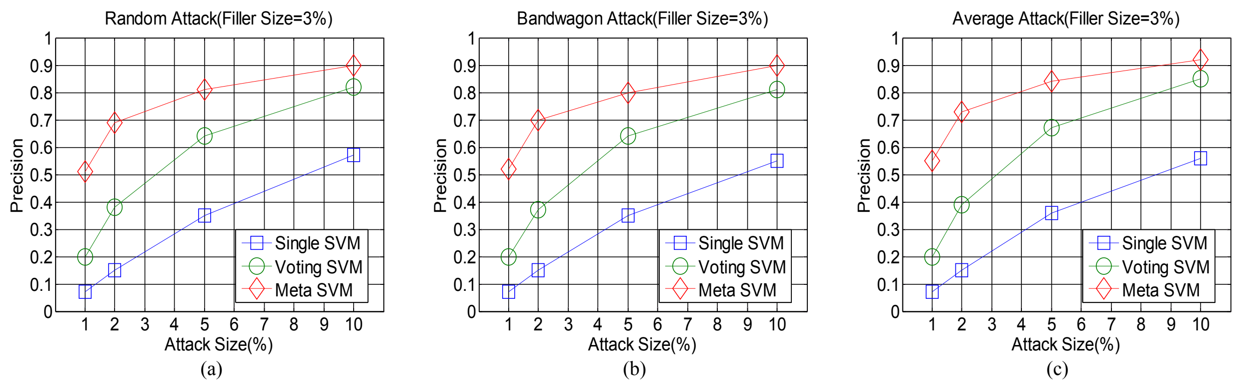

Figure 2. Precision on MovieLens 100K test sets for various attack models at filler size=3% across various attack sizes

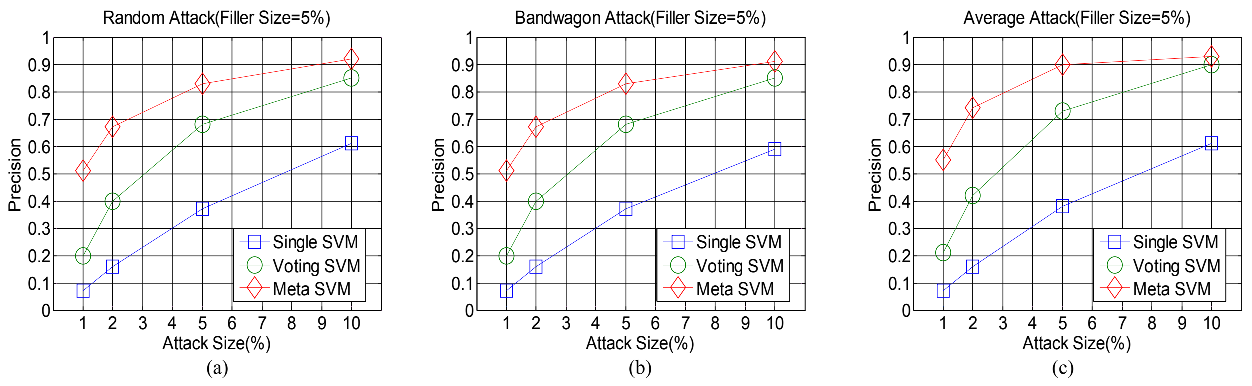

Figure 3. Precision on MovieLens 100K test sets for various attack models at filler size=5% across various attack sizes

As shown in Table Ⅰ and Table Ⅱ, most recalls of Meta SVM are between 0.99 and 1. These results, which are similar to the results of Single SVM and Voting SVM, keep at a high level. This is easy to explain, since each base-classifier itself can detect most of the attack profiles, the integration of these classifiers can still own this capability.

In terms of the attack effects on recommendations, it is discovered that the average attack is more powerful than others [3][4]. However, as shown in Table Ⅰ and Table Ⅱ, Meta SVM is able to effectively identify average attacks at various filler sizes across attack sizes and the recall can reach 0.99. The reason for these high recalls is that the useful classification information of the supervised classifier focuses on the training set where we had injected enough attack samples.

Figure 2 and Figure 3 show that the precision of Meta SVM has been significantly improved compared to the precision of Single SVM and Voting SVM. At 1% attack size, the average precision of Meta SVM increases by 6.6 percentage points and 1.7 percentage points compared to the average precision of Single SVM and Voting SVM respectively. At 10% attack size, the average precision of Meta SVM reaches 0.91 while the average precisions of Single SVM and Voting SVM are 0.58 and 0.85 respectively. These results illustrate the success of Meta SVM in reducing the misclassifications.

D. *Experimental Results and Analysis on MovieLens 1M Test Sets*

The detection results of the three methods (i.e. Single SVM, Voting SVM and Meta SVM) on the MovieLens 1M test sets are shown in Table Ⅲ, Table Ⅳ, Figure 4 and Figure 5.

TABLE III.

RECALL ON MOVIELENS 1M TEST SETS FOR VARIOUS ATTACK MODELS AT FILLER SIZE=3% ACROSS VARIOUS ATTACK SIZES

Attack Model	Random Attack				Bandwagon Attack				Average Attack			
Attack Size	1%	2%	5%	10%	1%	2%	5%	10%	1%	2%	5%	10%
Single SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Voting SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
Meta SVM	0.97	0.98	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99

TABLE Ⅳ.
RECALL ON MOVIELENS 1M TEST SETS FOR VARIOUS ATTACK MODELS AT FILLER SIZE=5% ACROSS VARIOUS ATTACK SIZES

Attack Model	Random Attack				Bandwagon Attack				Average Attack			
Attack Size	1%	2%	5%	10%	1%	2%	5%	10%	1%	2%	5%	10%
Single SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Voting SVM	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Meta SVM	0.98	0.97	0.97	0.95	0.95	0.98	1.00	1.00	0.99	1.00	1.00	0.99

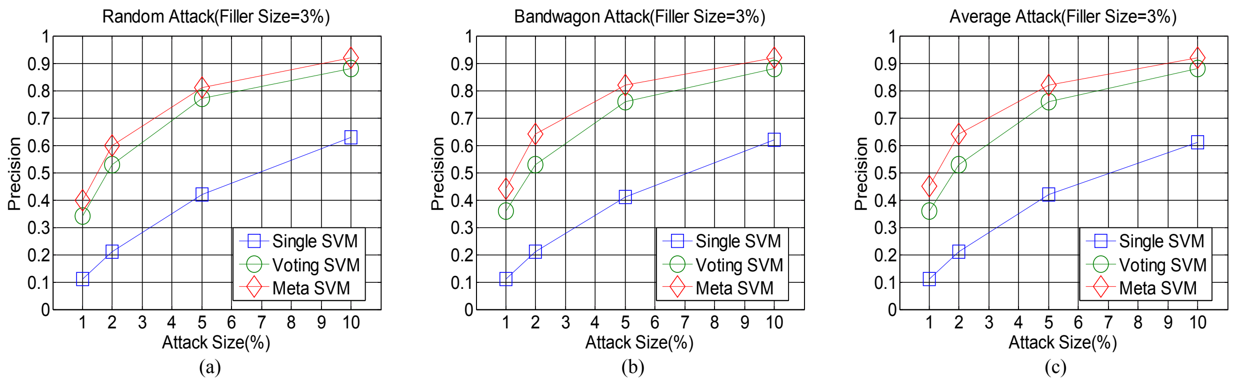

Figure 4. Precision on MovieLens 1M test sets for various attack models at filler size=3% across various attack sizes

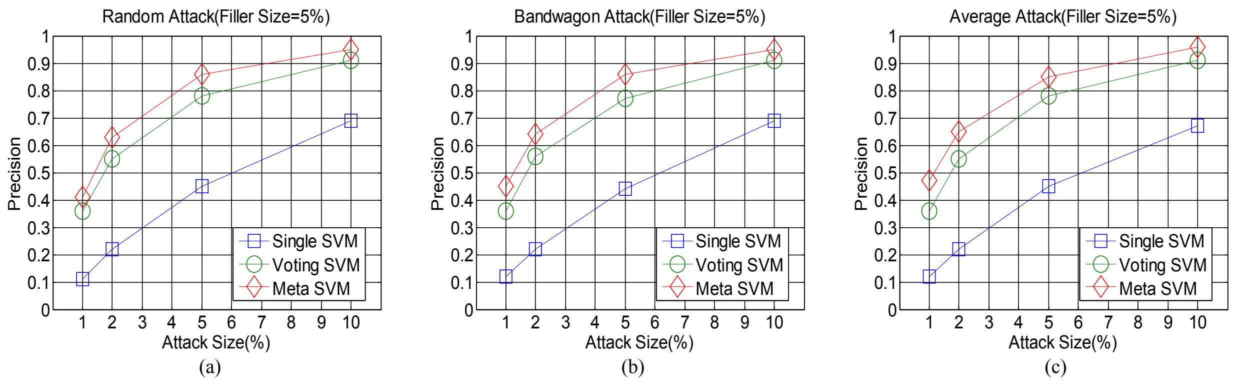

Figure 5. Precision on MovieLens 1M test sets for various attack models at filler size=5% across various attack sizes

As shown in Table Ⅲ, Table Ⅳ, Figure 4 and Figure 5, the recalls and precisions of Meta SVM on MovieLens 1M test sets, whose scale are larger than the scale of MovieLens 100K test sets, still keep at a high level.

In Table Ⅲ and Table Ⅳ, most recalls of Meta SVM are 0.99 and the minimum value has reached 0.95. These recalls are slightly lower than the recalls of Single SVM and Voting SVM. However, we believe that Meta SVM is more practical than the other two detection methods. Blew, we show the explanations for this viewpoint. In a recommender system, a group of attack profiles is particularly harmful against the recommendation results while the attack effects of several attack profiles is very limited [7][10]. In the worst case, the recall of Meta SVM only drops by 0.05 percentage points compared to the recall of Single SVM. Meta SVM can detect most of the attack profiles under the condition of holding high precisions while Single SVM and Voting SVM improve their recalls through increasing the misclassifications.

In Figure 4 and Figure 5, the precisions of Meta SVM increase with increasing of attack size, take Figure 4(a) for example, when the attack sizes are 1%, 2%, 5%, 10%, the precisions of Meta SVM are 0.4, 0.6, 0.81 and 0.92. These trends are also shown in Figure 2 and Figure 3. This is not hard to explain, with increasing of attack size, more attack profiles are detected while the number of misclassified genuine profiles keeps the same. These illustrate that Meta SVM has a good stability when detecting the attacks at large attack sizes.

The precision curves of Voting SVM are closer to the curves of Meta SVM in Figure 4 and Figure 5 than the curves in Figure 2 and Figure 3. This illustrates that the classification performance of Voting SVM is improved on the large-scale dataset. However, the precisions of Meta SVM are always higher than the precisions of Single SVM and Voting SVM. At 1% attack size, the average precision of Meta SVM increases by 3 percentage points and 0.2 percentage points compared to the average precision of Single SVM and Voting SVM respectively. At 10% attack size, while the average precisions of Single SVM and Voting SVM are 0.58 and 0.85 respectively, the average precision of Meta SVM reaches 0.94 higher than the average precision 0.91 in Figure 2 and Figure 3. These results illustrate that Meta SVM can effectively detect the attack profiles not only on a small-scale dataset but also on a large-scale dataset with a high precision.

V. Conclusions and Future Work

The detection of profile injection attacks is an important research area in the recommender system. We have made some beneficial explorations and attempts in this area. We propose an algorithm to create the diverse base-level training sets through flexible combinations of various attack types. Using these sets, we construct the diverse base-level classifiers to reduce the correlations of the misclassifications. Through relearning the results of the base-level classifiers, a meta-classifier is constructed. Based on base-level and meta-level classifiers, we propose a meta-learning-based detection algorithm which could effectively detect profile injection attacks. The experimental results on the different-scale datasets prove that the proposed approach can not only hold a high recall, but also effectively improve the precision.

In the future, we will study the more appropriate meta-learning strategies to improve the detection performance for profile injection attacks in collaborative recommender systems.

Acknowledgment

This work is partially supported by the Natural Science Foundation of Hebei Province, China (F2011203219) and the Special Fund for Fast Sharing of Science Paper in Net Era by CSTD of Ministry of Education, China (20101333110013).

References

[1] I. Bartolini, Z. Zhang, and D. Papadias, “Collaborative filtering with personalized skylines,” IEEE Transactions on Knowledge and Data Engineering, vol.23(2), pp.190-203, February 2011, doi:10.1109/TKDE.2010.86.

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, “Detecting profile injection attacks in collaborative recommender systems,” In Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE’06), June 2006, doi:10.1109/CEC-EEE.2006.34.

[3] S. K. Lam, and J. Riedl, “Shilling recommender systems for fun and profit,” In Proceedings of the 13th international conference on World Wide Web (WWW'04), pp.393-402, May 2004, doi:10.1145/988672.988726.

[4] R. Burke, B. Mobasher, and R. Bhaumik, “Limited knowledge shilling attacks in collaborative filtering systems,” In Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization (ITWP’05), August 2005, doi:10.1.1.135.2204.

[5] R. Burke, B. Mobasher, R. Bhaumik, and C. Williams, “Segment-Based injection attacks against collaborative filtering recommender systems,” In Proceedings of the International Conference on Data Mining (ICDM 2005), pp. 577-580, November 2005, doi:10.1109/ICDM.2005.127.

[6] M. O’mahony, N. Hurley, N. Kushmerick, and G. Silvestre, “Collaborative recommendation: a robustness analysis,” ACM Transactions on Internet Technology, vol.4(4), pp.344-377, November 2004, doi:10.1145/1031114.1031116.

[7] B. Mehta, and W. Nejdl, “Attack resistant collaborative filtering,” In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR’08), pp.75-82, July 2008, doi:10.1145/1390334.1390350.

[8] P.-A. Chirita, W. Nejdl, and C. Zamfir, “Preventing shilling attacks in online recommender systems,” In Proceedings of the 7th annual ACM international workshop on Web information and data management (WIDM ’05), pp.67-74, November 2005, doi:10.1145/1097047.1097061.

[9] X.-F. Su, H.-J. Zeng, and Z. Chen, “Finding group shilling in recommendation system,” Special interest tracks and posters of the 14th international conference on World Wide Web (WWW'05), pp.960-961, May 2005, doi:10.1145/1062745.1062818.

[10] B. Mehta, T. Hofmann, and P. Fankhauser, “Lies and propaganda: detecting spam users in collaborative filtering,” In Proceedings of the 12th international conference on Intelligent user interfaces (IUI’07), pp.14-21, January 2007, doi:10.1145/1216295.1216307.

[11] B. Mehta, “Unsupervised shilling detection for collaborative filtering,” In Proceedings of the 22nd national conference on Artificial intelligence (AAAI'07), pp.1402-1407, July 2007.

[12] B. Mehta, and W. Nejdl, “Unsupervised strategies for shilling detection and robust collaborative filtering,” User Modeling and User-Adapted Interaction, vol.19(1-2), pp.65-79, 2009, doi:10.1007/s11257-008-9050-4.

[13] K. Bryan, M. O’Mahony, and P. Cunningham, “Unsupervised retrieval of attack profiles in collaborative recommender systems,” In Proceedings of the 2008 ACM conference on Recommender systems (RecSys’08), pp.155-162, October 2008, doi:10.1145/1454008.1454034.

[14] N. Hurley, Z. Cheng, and M. Zhang, “Statistical attack detection,” In Proceedings of the third ACM conference on Recommender systems (RecSys’09), pp.149-156, October 2009, doi:10.1145/1639714.1639740.

[15] F. He, X. Wang, and B. Liu, “Attack detection by rough set theory in recommendation system,” IEEE International Conference on Granular Computing, pp.692-695, August 2010, doi:10.1109/GrC.2010.130.

[16] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, “Classification features for attack detection in collaborative recommender systems,” In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD’06), pp.542-547, August 2006, doi:10.1145/1150402.1150465.

[17] C. A.Williams, B. Mobasher, and R. Burke, “Defending recommender systems: detection of profile injection attacks,” Service Oriented Computing and Applications, vol.1(3), pp.157-170, 2007, doi:10.1007/s11761-007-0013-0.

[18] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Towards trustworthy recommender systems: an analysis of attack models and algorithm robustness,” ACM

Transactions on Internet Technology, vol.7(4), October 2007, doi:10.1145/1278366.1278372.

[19] Z. Fuzhi, J. Dongyan, C. Jinbo, “An improved PCA attack detection algorithm based on normal cloud model,” Journal of Computational Information Systems, vol.6(6), pp. 1959-1966, 2010.

[20] Z. Fuzhi; J. Dongyan, C. Jinbo, “A user profile injection attack detection algorithm based on normal cloud model and PCA,” ICIC Express Letters, vol.5(4A), pp. 925-930, 2011.

[21] X. Yuchen, L. Qiang; Z. Fuzhi, “User profile attack anomaly detection algorithm based on time series analysis,” Journal of Information and Computational Science, vol.7(11), pp. 2201-2206, 2010.

[22] R. Bhaumik, C. Williams, B. Mobasher, R. Burke, “Securing collaborative filtering against malicious attacks through anomaly detection,” In Proceedings of the 4th workshop on intelligent techniques for web personalization (ITWP'06), July 2006.

[23] Z. Lan, J. Gu, Z. Zheng, R. Thakur, and S. Coghlan, “A study of dynamic meta-learning for failure prediction in large-scale systems,” Journal of Parallel and Distributed Computing archive, vol.70(6), pp.630-643, June 2010, doi:10.1109/ICPP.2008.17.

[24] D. H. Wolpert, “Stacked generalization,” Neural Networks, vol.5(2), pp.241-259, 1992, doi:10.1.1.56.1533.

[25] P. Kordík, J. Koutník, J. Drchal, O. Kovářík, M. Čepek, and M. Šnorek, “Meta-learning approach to neural network optimization,” Neural Networks, vol.23(4), pp.568-582, May 2010, doi:10.1016/j.neunet.2010.02.003.

[26] M. Wasikowski, X.-W. Chen, “Combating the small sample class imbalance problem using feature selection,” IEEE Transactions on Knowledge and Data Engineering, vol.22(10), pp.1388-1400, October 2010, doi:10.1109/TKDE.2009.187.

[27] Y. Fu, H. Yan, J. Li, and R. Xiang, “Robust facial features localization on rotation arbitrary multi-view face in complex background,” Journal of Computers, vol.6(2), pp.337-342, February 2011, doi:10.4304/jcp.6.2.337-342.

[28] C.-F. Tsai, Y.-C. Linc, D. C.Yen, and Y.-M. Chen, “Predicting stock returns by classifier ensembles,” Applied Soft Computing, vol.11(2), pp.2452-2459, 2011, doi:10.1016/j.asoc.2010.10.001.

Fuzhi Zhang was born in 1964. Currently, he is a professor and PhD supervisor in School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. His research interests include intelligent information processing, network and information security, and service-oriented computing.

Quanqiang Zhou was born in 1985. Currently, he is a PhD student in School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. His research interests include intelligent information processing and personalization.