2013 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2013, SOUTHAMPTON, UK

GRAPH-BASED DETECTION OF SHILLING ATTACKS IN RECOMMENDER SYSTEMS

Zhuo Zhang, Sanjeev R. Kulkarni

Department of Electrical Engineering, Princeton University, Princeton, NJ 08540

ABSTRACT

Collaborative filtering has been widely used in recommender systems as a method to recommend items to users. However, by using knowledge of the recommendation algorithm, shilling attackers can generate fake profiles to increase or decrease the popularity of a targeted set of items. In this paper, we present a method to make recommender systems resistant to these attacks in the case that the attack profiles are highly correlated with each other. We formulate the problem as finding a maximum submatrix in the similarity matrix. We search for the maximum submatrix by transforming the problem into a graph and merging nodes by heuristic functions or finding the largest component. Experimental results show that the proposed approach can improve detection precision compared to state of art methods.

***Index Terms*— Collaborative Filtering, Recommender Systems, Robust, Graph, Heuristic, Largest Component**

1. INTRODUCTION

In recommender systems, there are lists of both users and items. Each user provides scores or linguistic terms to rate a subset of the possible items. With a large amount of information, users often find it hard to select an item from the large set of available items. Recommender systems are designed to recommend items based on relevant information for the specific user. The underlying principle is to find a group of users with similar tastes and then provide a prediction for the target user based on the preferences of similar users.

However, recommender systems are susceptible to so-called shilling attacks in which an attacker signs up as a number of "dummy" users and gives fake ratings in an attempt to increase (push) or decrease (nuke) the popularity of specific items by exploiting knowledge of the recommender system algorithm. An important problem in recommender system design is to find algorithms that are robust to attacks.

In this paper, we model shilling attacks as a group of users with highly correlated ratings. We formulate the detection problem as an optimization problem, to find an n-element subset of $\{1, ..., m\}$ specifying n rows and columns such that the corresponding $n \times n$ submatrix has maximal sum from the original $m \times m$ similarity matrix. To solve this NP-complete problem, we construct a graph based on the similarity matrix. The heuristic merging or largest component searching algorithm is further applied for the optimal solution. Experimental results show that our method successfully filters more than 90% of shilling attacks for different attack types.

Our paper makes the following contributions over prior work. 1) We do not make any assumption on the attack model except that attack users are highly correlated; 2) An optimization problem is formulated for the detection problem and two graph-based algorithms are applied; 3) Our algorithm does not need to specify the exact number of attack profiles but uses statistics to automatically determine it.

The rest of the paper is organized as follows. Section 2 introduces the background. Section 3 discusses details of our proposed algorithm. Experimental results and comparisons with existing methods are presented in Section 4.

2. BACKGROUND

2.1. Collaborative Filtering

In recommender systems, there is a set of users $U = \{u_1, ..., u_m\}$ and a set of items $I = \{i_1, ..., i_p\}$. For each user u, I_u denotes the corresponding subset of items that user u has rated. Let $I_{u,v}$ denote the subset of items that both user u and user v have rated. An $m \times p$ dimensional rating matrix R can be constructed as follows. Each element $r_{u,i}$ in R denotes user u's rating of item i. The rating data typically specifies only a small number of the elements of R. Let $\bar{r}_u$ denote the average rating of user u on all the items user u has rated. The goal is to predict an unknown rating $r_{u,i}$ that we believe a specific user u will provide for a target item i.

In recommender systems, the most commonly-used method is called collaborative filtering. It uses the Pearson Correlation function $s_{u,v}$ to measure the similarity between users u and v, which is defined as

$$s_{u,v} = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}} \quad (1)$$

This research was supported in part by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370, and by support from Deutsche Telekom T-Labs.

978-1-4799-1180-6/13/$31.00

Table 1. General Form of Attack Profiles

	I_S			I_F			I_N		
i_t	i_1^S	...	i_k^S	i_1^F	...	i_l^F	i_1^N	...	i_v^N
$\gamma(i_t)$	$\sigma(i_1^S)$	...	$\sigma(i_k^S)$	$\rho(i_1^F)$	...	$\rho(i_l^F)$	null	null	null

Then the final prediction is a weighted aggregation of some nearest neighbors' ratings as given by

$$r_{u,i} = \bar{r}_u + \frac{\sum_{v \in U} s_{u,v}(r_{v,i} - \bar{r}_v)}{\sum_{v \in U} |s_{u,v}|} \quad (2)$$

2.2. General Form of Attack Profiles

In order to push or nuke a target item, the fake profiles should first mimic the rating patterns of real users so that they can be the nearest neighbors of our target user. At the same time they should rate the specific item either much higher or lower than the average rating so that they can have a strong impact on the weighted sum. A general form of an attack profile is shown in Table 1, as four sets of items:
1) i_t: a singleton target item, of which the rating is $\gamma(i_t)$, usually the highest or lowest value in the system;
2) I_S: a set of selected items with particular characteristics determined by the attacker, with their ratings $\sigma(i_k^S)$ to mimic the rating pattern as real ones;
3) I_F: a set of filler items usually chosen randomly with random assigned ratings $\rho(i_l^F)$;
4) I_N: a set of unrated items.

2.3. Attack Models

In a typical attack, the target item i_t is usually set at either the highest score (for a push attack) or the lowest score (for a nuke attack). However, different choices of rating functions and selections of I_S and I_F lead to different attack models, some of which are described below. Here let $\bar{r}$ denote the average rating over all items and users, and let $\bar{r}_i$ denote the average rating of item i over all the users that have rated this item. Moreover, let $\bar{\sigma}$ denote the standard deviation of ratings over all items and users, and $\bar{\sigma}_i$ the standard deviation of ratings of item i over all the users who have rated this item. Let $N(r, \sigma^2)$ denote the Gaussian distribution with mean r and variance σ^2.
1) Random attack: $I_S = \phi$ and $\rho(i) \sim N(\bar{r}, \bar{\sigma}^2)$.
2) Average attack: $I_S = \phi$ and $\rho(i) \sim N(\bar{r}_i, \bar{\sigma}_i^2)$.
3) Bandwagon attack: I_S contains some number of popular items, $\sigma(i) = r_{\max}$ and $\rho(i) \sim N(\bar{r}_i, \bar{\sigma}_i^2)$.

2.4. Related Work

O'Mahony et al. summarize different types of attack strategies and empirically evaluate the robustness of memory-based collaborative filtering [1]. The work in [2–4] further extend the robustness analysis to model-based algorithms such as K-means, PLSA, SVD, PCA, and Matrix Factorization. Therefore three natural attributes are proposed for attack detection.

2.4.1. Generic Attributes

Attack profiles usually have low deviation from the mean value for most items, but high deviation from the mean for the attacked item. Therefore generic attributes such as Rating Deviation from Mean Agreement (RDMA) and Degree of Similarity with Top Neighbors (DegSim) [5] are used to classify fake profiles. Furthermore an unsupervised retrieval method (UnRAP) based on the matrix residues was proposed in [6].

2.4.2. Model-specific Attributes

Model-based methods assume that we have some prior knowledge about the attack model. Based on this model, ratings can be automatically divided into I_S and I_F. Finally several statistics such as Filler Mean Variance or Filler Mean Target Difference [7] can be computed from each subset to evaluate the authenticity of profiles.

2.4.3. Intra Attributes

Model-specific methods need information about the attack model, and usually need some training data to estimate parameters of the attack model. Otherwise I_S and I_F can not be easily separated. Unlike generic attributes and model-specific attributes which concentrate on characteristics within a single profile, intra attributes focus on statistics across profiles. As Mehta et al. mention in [4], spam users often work together and are highly correlated. Therefore a PCA-based method can be applied to remove the most correlated users. This approach orders all the user profiles based on the contribution of the principle component (or the top few principle components) and removes the most highly correlated items. However, [4,8] do not specify how to choose the number of principle components to be considered. [9, 10] assume high correlations between fake profiles as well, using clustering methods for shilling attack detection.

In the remainder of this paper, we will start from the same assumption that spam users are highly correlated, but we propose a different algorithm to find the most correlated group.

3. GRAPH-BASED FILTERING ALGORITHM

3.1. Problem Formulation

As we mentioned before, shilling attackers generally work as a group and have high correlation with each other. Suppose we have m user profiles of which n are fake. From the original rating matrix $R_{m \times p}$, we can derive the similarity matrix $S_{m \times m}$ based on Eqn (1). Our final goal is to find

the $n \times n$ submatrix with the maximum sum in the original $S_{m\times m}$ matrix, with the same columns and rows selected. We define $\vec{\delta} = (\delta_1, ..., \delta_m)$ where δ_i is an indicator function that represents whether column/row i is being selected. Therefore the problem is formulated as given below.

$$\vec{\delta} = \arg \max_{||\vec{\delta}||_1 = n} \frac{1}{|\vec{\delta}|^2} \vec{\delta} S \vec{\delta}^T \quad (3)$$

$$= \arg \max_{\sum_{i=1}^m \delta_i = n} \frac{1}{|\vec{\delta}|^2} \sum_{i=1}^m \sum_{j=1}^m s_{i,j}(\delta_i + \delta_j - \delta_i\delta_j) \quad (4)$$

where $\delta_i = 0$ or $1, \forall i = 1, ..., m$.

Searching for a maximal submatrix is typically referred to as a biclustering problem [11] and several algorithms have been proposed. Since (3) requires selecting the same columns and rows, it is different from traditional biclustering. However, a data matrix can be viewed as a weighted graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. Each vertex in V denotes a corresponding column/row and each edge between v_i and v_j has a weight $s_{i,j}$. Therefore we can get a suboptimal solution from the graph, a subset of points which have a high average weight within group. In the following subsections, we propose two algorithms, namely heuristic merging and largest component searching, to derive an approximate solution. After that, we use an iterative process to refine the attack group. There is always a tradeoff between the group size n and the average similarity within the group. Therefore, how to set up the stopping criterion is another challenge in this problem and we will address it later.

3.2. Heuristic Merging

A naive greedy algorithm selects one point out of the group with the highest average similarities with all points in the group, and adds it into the original group. When the group size is relatively large and most of them are fake profiles, this algorithm works very well. However, when the group size is relatively small, the initial points are hard to select. Therefore, we propose a generalized greedy, or heuristic merging algorithm, regarding each point initially as a separate cluster, merging them step by step by heuristic functions and finally leading to an optimal cluster with the right size. Before doing that, we first introduce some notation for convenience.

- $C_i^{(t)}$ is the node set of cluster i after t merging actions. $\bigcup_i C_i^{(t)}$=$\{1, ..., m\}$ and $C_i^{(t)} \bigcap C_j^{(t)} = \emptyset, \forall i \neq j$.
- $n_i^{(t)} = |C_i^{(t)}|$ is the number of nodes in cluster i after t merging actions.
- $d_{ij}^{(t)} = \frac{1}{n_i^{(t)} n_j^{(t)}} \sum_{x\in C_i^{(t)}, y\in C_j^{(t)}} s_{x,y}$ is the average distance between i and j after t merging actions, $i \neq j$.
- $f_i^{(t)} = \frac{1}{(n_i^{(t)})^2} \sum_{x\in C_i^{(t)}, y\in C_i^{(t)}} s_{x,y}$ is the average similarity within the group after t merging actions.

Initially each node belongs to an individual cluster, i.e. $C_i^{(0)} = \{i\}, \forall i = 1, ..., m$. Then at each time t we search for two clusters $C_i^{(t)}$, $C_j^{(t)}$ based on a heuristic function $f(C_i^{(t)}, C_j^{(t)})$ and merge them together. Our final goal is to find a cluster $C_{i_0}^{(t_0)}$ to maximize $f_{i_0}^{(t_0)}$ such that $n_{i_0}^{(t_0)} \geq n$.

In the problem above we have two objects to focus on, namely the size of the cluster $n_{i_0}^{(t_0)}$ and the average utility score $f_{i_0}^{(t_0)}$. Therefore we can choose our heuristic function h as either $n_{i_0}^{(t_0)}$ or $f_{i_0}^{(t_0)}$. But before the algorithms are introduced, let us first see the relationship after two clusters are combined.

Claim 1 Suppose in the $(t+1)$*th* merging action, $C_i^{(t)}$ and $C_j^{(t)}$ merge, i.e. $C_i^{(t+1)} = C_i^{(t)} \bigcup C_j^{(t)}$. Then

$$d_{ik}^{(t+1)} = \frac{n_i^{(t)} d_{ik}^{(t)} + n_j^{(t)} d_{jk}^{(t)}}{n_i^{(t)} + n_j^{(t)}}, \forall k \neq i.$$

$$f_i^{(t+1)} = \frac{(n_i^{(t)})^2 f_i^{(t)} + (n_j^{(t)})^2 f_j^{(t)} + 2 n_i^{(t)} n_j^{(t)} d_{ij}^{(t)}}{(n_i^{(t)} + n_j^{(t)})^2};$$

Proof.

$$\begin{aligned} d_{ik}^{(t+1)} &= \frac{\sum_{x\in C_i^{(t+1)}, y\in C_k^{(t+1)}} s_{x,y}}{n_i^{(t+1)} n_k^{(t+1)}} \\ &= \frac{\sum_{x\in C_i^{(t)} \bigcup C_j^{(t)}, y\in C_k^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)}) n_k^{(t)}} \\ &= \frac{\sum_{x\in C_i^{(t)}, y\in C_k^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)}) n_k^{(t)}} + \frac{\sum_{x\in C_j^{(t)}, y\in C_k^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)}) n_k^{(t)}} \\ &= \frac{n_i^{(t)} n_k^{(t)} d_{ik}^{(t)}}{(n_i^{(t)} + n_j^{(t)}) n_k^{(t)}} + \frac{n_j^{(t)} n_k^{(t)} d_{jk}^{(t)}}{(n_i^{(t)} + n_j^{(t)}) n_k^{(t)}} \\ &= \frac{n_i^{(t)} d_{ik}^{(t)} + n_j^{(t)} d_{jk}^{(t)}}{n_i^{(t)} + n_j^{(t)}}. \end{aligned}$$

$$\begin{aligned} f_i^{(t+1)} &= \frac{\sum_{x\in C_i^{(t+1)}, y\in C_i^{(t+1)}} s_{x,y}}{(n_i^{(t+1)})^2} \\ &= \frac{\sum_{x\in C_i^{(t)} \bigcup C_j^{(t)}, y\in C_i^{(t)} \bigcup C_j^{(t)}} s_{x,y}}{(n_i^{(t+1)})^2} \\ &= \frac{\sum_{x\in C_i^{(t)}, y\in C_i^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)})^2} + \frac{\sum_{x\in C_j^{(t)}, y\in C_j^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)})^2} \\ &\quad + 2\frac{\sum_{x\in C_i^{(t)}, y\in C_j^{(t)}} s_{x,y}}{(n_i^{(t)} + n_j^{(t)})^2} \\ &= \frac{(n_i^{(t)})^2 f_i^{(t)} + (n_j^{(t)})^2 f_j^{(t)} + 2 n_i^{(t)} n_j^{(t)} d_{ij}^{(t)}}{(n_i^{(t)} + n_j^{(t)})^2}. \end{aligned}$$

Algorithm 1 Heuristic Merging Algorithm

Input: $S_{m\times m}$, a symmetric correlation matrix of real numbers, and n, the maximum size of cluster.
Output: $C_{i_0}^{(t_0)}$, a cluster which has a size $n_{i_0}^{(t_0)} \geq n$ and local maximal average similarity $f_{i_0}^{(t_0)}$.
Initialization: $C_i^{(0)} = \{i\}, n_i^{(0)} = 1, d_{ij}^{(0)} = s_{ij}, f_i^{(0)} = 0$ $\forall i, j = 1, ..., m.\ t = 0$.
while $\max_i(n_i^{(t)}) \leq n$ **do**
  Find cluster i, j s.t. $h(C_i^{(t)}, C_j^{(t)})$ achieves maximum:
  $C_i^{(t+1)} \leftarrow C_i^{(t)} \bigcup C_j^{(t)}, n_i^{(t+1)} \leftarrow n_i^{(t)} + n_j^{(t)}$;
  $d_{ik}^{(t+1)} \leftarrow \frac{n_i^{(t)} d_{ik}^{(t)} + n_j^{(t)} d_{jk}^{(t)}}{n_i^{(t)} + n_j^{(t)}}, \forall k \neq i.$
  $f_i^{(t+1)} \leftarrow \frac{(n_i^{(t)})^2 f_i^{(t)} + (n_j^{(t)})^2 f_j^{(t)} + 2 n_i^{(t)} n_j^{(t)} d_{ij}^{(t)}}{(n_i^{(t)} + n_j^{(t)})^2}$;
  $t \leftarrow t + 1$;
end while
return $C_i^{(t)}$

Algorithm 2 Largest Component Searching Algorithm

Input: $S_{m\times m}$, a symmetric correlation matrix of real numbers, and n, the maximum size of cluster.
Output: C, a cluster with a size larger than n and a relatively high average similarity within the cluster $f(C)$.
Denote γ as the translation threshold, $\delta\gamma$ as the smallest distinguished threshold and $G_{m\times m}$ as the translated graph
Initial $\gamma \leftarrow 1$
repeat
  $\gamma \leftarrow \gamma - \delta\gamma$
  if $s_{u,v} > \gamma$ **then**
    Set u, v connected in G
  else
    Set u, v unconnected in G
  end if
  Find the largest component in G, denote as C
until $|C| \geq n$
return C

The heuristic merging algorithm is shown in Alg.1.

We can easily verify that when $h(C_i^{(t)}, C_j^{(t)}) = n_i^{(t+1)} = n_i^{(t)} + n_j^{(t)}$ (when $n_i^{(t+1)}$ is equal we will maximize $f_i^{(t+1)}$ to avoid trivial cases) the algorithm becomes the naive greedy algorithm, which adds the largest node into the original cluster at each step and converges within $n-1$ steps. When we set $h(C_i^{(t)}, C_j^{(t)}) = f_i^{(t+1)} = \frac{(n_i^{(t)})^2 f_i^{(t)} + (n_j^{(t)})^2 f_j^{(t)} + 2 n_i^{(t)} n_j^{(t)} d_{ij}^{(t)}}{(n_i^{(t)} + n_j^{(t)})^2}$, we allow nodes to merge freely regardless of the current cluster size. Finally, after at most $m-1$ steps, a cluster with more than n nodes will appear to be a solution. In the following, we will use $f_i^{(t+1)}$ as the merging heuristic function.

3.3. Searching for the Largest Component

The heuristic merging algorithm achieves a local optimal solution step by step. However, as we mentioned before, we must do it carefully for the first few steps in order to keep the local optimum not far away from the global optimum. Is there a way to find a group of highly correlated users in a single step, namely to find a cluster that maximizes the average similarity $f(C)$ for a cluster C? $f(C)$ is defined as

$$f(C) = \frac{1}{|C|^2} \sum_{x \in C, y \in C} s_{x,y}$$

A natural way would be to set a threshold to break some edges and find the largest component from the modified graph.

Here we set a threshold γ and convert $S_{m\times m}$ to a connected graph based on the following rule: if $s_{i,j} > \gamma$, we link the two vertices v_i and v_j; otherwise we break the link between two vertices. When γ is close to 1, all the vertices are separate from each other. As γ decreases, the original separate components connect with each other due to high correlation. In this case, the largest component in the graph, denoted as C, can be derived based on the classic algorithm introduced in [12] for an approximate solution of Eqn (3). Here, we can always choose a proper γ to make sure the size of C is around a prefixed number n. In the following step, we will further refine the set and automatically determine the final fake profile size n. The algorithm is shown in Alg.2.

3.4. Iterative Refinement

We expect that the solution derived from either heuristic merging or largest component searching will contain a certain level of noise since many genuine profiles could be highly similar with at least one of the fake profiles. Based on either algorithm, a real user's profile could be classified as an attack profile when one edge with large weight is connected to the attack group. However the attack profiles share high similarity with all the other attack profiles. Therefore we need a further refinement process for this algorithm.

Here a greedy algorithm is applied for refinement. We separate this process into two steps, deletion and addition. A pre-set number s steps of addition/deletion is specified. For the deletion step, we can calculate the average similarity in set C and remove the least correlated profile from C. We repeat this process s times to delete s profiles. And then we proceed to the addition step. The average similarity of all the profiles in $\bar{C}$ with profiles in C are calculated and the one with highest average similarity with C is added. The addition process is repeated s times as well. We repeatedly perform s deletions and s additions until the convergence is achieved. Note that usually we do not choose $s = 1$ but a larger number, say $s = 10$ in this case. This helps avoid local optima and helps push the final results toward the global optimal point. The algorithm is shown in Alg.3.

Algorithm 3 Iterative Refinement

Input: $S_{m\times m}$, a symmetric correlation matrix of real numbers, and C, a cluster with a high average similarity within the cluster $f(C)$.
Output: C, a cluster with a size larger than n and a relatively high average similarity within the cluster $f(C)$.
while $C \neq C_0$ **do**
  $C_0 = C$
  for $i = 1$ to s **do**
    $Cor_u \leftarrow \frac{\sum_{v\in C} s_{u,v}}{|C|}, \forall u \in \bar{C}$
    $C = C \cup \arg\max(Cor_u)_{u\in\bar{C}}$
  end for
  for $i = 1$ to s **do**
    $Cor_u \leftarrow \frac{\sum_{v\in C} s_{u,v}}{|C|}, \forall u \in C$
    $C = C/\arg\min(Cor_u)_{u\in C}$
  end for
end while
return C

4. EXPERIMENTS

4.1. Experimental Setup

In the experiments, we use the MovieLens dataset. It contains a total of 100,000 ratings from 943 users on 1,682 items. Each user rates at least 20 movies on a scale from 1 to 5. The density of the rating matrix is 6.3%. 80% of the ratings are randomly selected as the training set and the rest as testing.

We randomly pick 20 movies as the attack items for each test and artificially insert 50, 70 or 100 attack profiles with filler size 5%, making it the same density as the training dataset. Each movie is attacked individually and the average is reported in the following subsection. For a random attack, we generate ratings based on $N(\bar{r}, \bar{\sigma}^2)$, while for an average attack, we generate ratings based on $N(\bar{r}_i, \bar{\sigma}_i{}^2)$. For a bandwagon attack, we select movies {50, 56, 100, 127, 174, 181} as the selected set I_S and rate them as $r_{\max} = 5$. We select these movies because they are rated by more than 300 users and have an average rating larger than 4. We further generate two obfuscated attack models. One is noisy bandwagon attack, a special case of Average Over Popular Items (AOP) mentioned in [13], which randomly selects 3 out of the 6 popular movies mentioned above rated as $r_{\max}$ to avoid high correlations. The other one is mixed attack [9], which combines a 3% or 5% average attack together with a 3% or 5% noisy bandwagon attack.

For the target profiles with attacks, our algorithm varies the size of the highly correlated group to get a sequence, $G(n)$, denoting the maximal average similarity of a group with size n. Suppose $G_0(n)$ is the maximal average similarity sequence for profiles without attacks. The optimal number of fake profiles n^* is then obtained from Eqn (5) as

$$n^* = \arg\max_n \{G(n) - G_0(n)\} + n_0, \tag{5}$$

where n_0 is an empirical drift and we take $n_0 = 0.1n$ in the experiment.

We compare our Heuristic Merging (HM) and Largest Component Searching (LC) algorithm with the RDMA [5], DegSim [5] and UnRAP [6] algorithms. In the experiments, since we have no prior knowledge of the exact number of attack profiles, our algorithms derive n from Eqn (5). For RDMA, DegSim and UnRAP, we assume the exact number of attack profiles is known, which yields the same precision and recall in the results.

4.2. Evaluation Metrics

For the classification of attack profiles, we introduce two metrics, namely precision and recall to evaluate the performance of the detection algorithm.

$$\text{Precision} = \frac{TP}{TP+FP}, \text{ Recall} = \frac{TP}{TP+FN},$$

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

To further evaluate the effect of attacks, we define the prediction shift as follows.

$$\text{Prediction Shift} = \frac{\Sigma_{u,i}(\hat{r}'_{u,i} - \hat{r}_{u,i})}{N},$$

where N is the total number of known ratings, and $\hat{r}'_{u,i}$ and $\hat{r}_{u,i}$ are post- and pre-attack predictions respectively.

4.3. Experimental Results and Discussion

The experimental results comparing our proposed HM and LC with RMDA, DegSim and UnRAP are shown in Table 2. HM and LC perform very well in most cases except the 5% random attack case ($n \approx 50$). The reason is that when the attack size is small, there is no significant difference between genuine profiles and random attack profiles. Therefore neither algorithm can find the right size n^* due to the similar values of $G(n)$ and $G_0(n)$. Moreover, LC is more unstable than HM in the 7% and 10% random attack cases. This is because LC strongly depends on one largest weight to the group instead of an average, which may lead the results to be unstable. While for HM, since the average distance is updated all the time, it is always merging the optimal in the average sense.

UnRAP does the worst job in most cases because it assumes every rating is related to the column mean, row mean and overall mean. Random attacks and bandwagon attacks do not satisfy this assumption at all. For average attacks, due to the good fit of the model assumption, it does a slightly better job but still not that much. RMDA starts from a similar assumption to evaluate the genuineness of profiles, but is still not very effective due to the nature of actual ratings.

Table 2. Experimental Results

Attack Model		Random Attack			Average Attack			Bandwagon Attack			Average Over Popular Items			Mixed Attack	
Attack Size		5%	7%	10%	5%	7%	10%	5%	7%	10%	5%	7%	10%	3%+3%	5%+5%
Precision	HM	-	99.7%	99.1%	99.7%	99.9%	99.8%	92.4%	91.3%	90.1%	88.2%	87.7%	92.3%	91.9%	98.0%
	LC	-	26.8%	62.4%	83.2%	99.8%	99.0%	92.7%	91.7%	90.5%	90.3%	89.2%	88.3%	92.3%	95.2%
	DegSim	6.3%	7.3%	12.2%	20.2%	37.1%	59.3%	74.5%	77.1%	81.3%	64.2%	71.4%	71.5%	43.4%	66.4%
	RMDA	82.6%	83.0%	86.0%	74.2%	76.4%	77.2%	72.4%	78.7%	81.2%	71.3%	78.2%	81.6%	33.3%	56.7%
	UnRAP	1.0%	1.4%	1.4%	48.2%	47.3%	65.4%	8.2%	9.4%	26.1%	6.3%	12.9%	23.1%	23.3%	33.3%
Recall	HM	-	98.6%	98.7%	98.0%	98.8%	99.0%	100%	100%	99.7%	96.4%	92.9%	97.3%	97.2%	99.7%
	LC	-	46.3%	64.2%	81.0%	94.6%	99.0%	100%	100%	99.5%	96.2%	93.1%	92.1%	97.1%	96.8%
	DegSim	6.3%	7.3%	12.2%	20.2%	37.1%	59.3%	74.5%	77.1%	81.3%	64.2%	71.4%	71.5%	43.4%	66.4%
	RMDA	82.6%	83.0%	86.0%	74.2%	76.4%	77.2%	72.4%	78.7%	81.2%	71.3%	78.2%	81.6%	33.3%	56.7%
	UnRAP	1.0%	1.4%	1.4%	48.2%	47.3%	65.4%	8.2%	9.4%	26.1%	6.3%	12.9%	23.1%	23.3%	33.3%
Prediction Shift	HM	0.56	0.02	0.03	0.05	0.03	0.05	0.02	0.01	0.05	0.10	0.17	0.12	0.05	-0.04
	LC	0.56	0.57	0.32	0.13	-0.01	0.01	0.01	0.01	0.03	0.07	0.13	0.29	0.01	-0.03
	DegSim	0.46	0.58	0.74	0.86	0.93	0.90	0.52	0.65	0.72	0.56	0.63	0.65	0.72	0.78
	RMDA	0.19	0.24	0.24	0.38	0.42	0.57	0.51	0.63	0.69	0.52	0.65	0.68	0.93	0.96
	UnRAP	0.55	0.69	0.80	0.65	0.70	0.81	1.08	1.28	1.28	1.08	1.17	1.29	1.03	1.15

The DegSim method does not perform very well because in essence it starts from the assumption that fake profiles are highly correlated, but only focuses on the k nearest neighbors instead of overall information. HM and LC start from the same assumption but use the global optimal solution from graph-based algorithms. Note that HM and LC separate real and fake profiles almost perfectly but the estimate of n^* usually contains some error. Therefore sometimes either high precision or recall is achieved but not both. HM and LC lose power only when fake profiles are not highly correlated, or the attack size is not very large. But in either case, they cannot have a strong impact on the final recommendation and the prediction shift is usually small. Even in the two obfuscated attack models which weaken the model characteristics and high correlation, HM and LC still perform well while the performance of the other algorithms decreases significantly. Therefore, the algorithms we propose outperform state of art methods, especially when fake profiles are highly correlated.

5. REFERENCES

[1] M.P. O'Mahony, N.J. Hurley, and G.C.M. Silvestre, "Recommender systems: Attack types and strategies," in *Proceedings of the National Conference on Artificial Intelligence*, 2005, vol. 20, p. 334.

[2] B. Mobasher, R. Burke, and J. Sandvig, "Model-based collaborative filtering as a defense against profile injection attacks," in *Proceedings of the AAAI*, 2006, vol. 21, p. 1388.

[3] S. Zhang, Y. Ouyang, J. Ford, and F. Makedon, "Analysis of a low-dimensional linear model under recommendation attacks," in *Proceedings of the 29th ACM SIGIR conference*, 2006, pp. 517–524.

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

[5] P.A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of 7th workshop on Web information and data management*. ACM, 2005, pp. 67–74.

[6] Kenneth Bryan, Michael O'Mahony, and Pádraig Cunningham, "Unsupervised retrieval of attack profiles in collaborative recommender systems," in *Proceedings of the 2008 ACM RecSys*. ACM, 2008, pp. 155–162.

[7] C.A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: detection of profile injection attacks," *Service Oriented Computing and Applications*, vol. 1, no. 3, pp. 157–170, 2007.

[8] Q. Zhou and F. Zhang, "A hybrid unsupervised approach for detecting profile injection attacks in collaborative recommender systems," 2012.

[9] Runa Bhaumik, Bamshad Mobasher, and RD Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in *Proceedings of 7th IEEE ICML, Las Vegas, USA*, 2011, pp. 181–187.

[10] Jong-Seok Lee and Dan Zhu, "Shilling attack detectionła new approach for a trustworthy recommender system," *INFORMS Journal on Computing*, vol. 24, no. 1, pp. 117–131, 2012.

[11] Y. Cheng and G.M. Church, "Biclustering of expression data," in *Proceedings of the eighth international conference on intelligent systems for molecular biology*, 2000, vol. 8, pp. 93–103.

[12] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein, *Introduction to algorithms*, MIT press, 2001.

[13] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 149–156.