A Belief Propagation Approach for Detecting Shilling Attacks in Collaborative Filtering

Jun Zou
Georgia Institute of Technology
Atlanta, GA 30332, USA
junzou@ece.gatech.edu

Faramarz Fekri
Georgia Institute of Technology
Atlanta, GA 30332, USA
fekri@ece.gatech.edu

ABSTRACT

Recommender systems have been widely used in e-commerce websites to suggest items that meet users' preferences. Collaborative filtering, which is the most popular recommendation algorithm, is vulnerable to shilling attacks, where a group of spam users collaborate to manipulate the recommendations. Several attack detection algorithms have been developed to detect spam users and remove them from the system. However, the existing algorithms focus mostly on rating patterns of users. In this paper, we develop a probabilistic inference framework that further exploits the target items for attack detection. In addition, the user features can also be conveniently incorporated in this framework. We utilize the Belief Propagation (BP) algorithm to perform inference efficiently. Experimental results verify that the proposed algorithm significantly improves detection performance as the number of target items increases.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering, Retrieval models.

Keywords

Collaborative Filtering, Shilling Attack, Belief Propagation

1. INTRODUCTION

Recommender systems are increasingly employed by e-commerce websites, e.g., Amazon.com and Netflix.com, to provide personalized recommendations to users. Collaborative Filtering (CF) is so far the most popular recommendation algorithm, which relies on historic ratings given by users on items to make recommendations. Unfortunately, it is vulnerable to the so called "shilling" attacks [6], in which a group of spam users collaborate to influence the recommendations for their benefits, e.g., to recommend their products more often.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2507875.

To protect the recommender systems against such attacks, one of the major approaches is to detect the spam users and remove them from the system. A number of detection algorithms have been proposed in the literature. Earlier work in [3] introduced several metrics for detecting the rating patterns of spam users. Then in [2], the authors improved the detection accuracy over [3] by including more model-specific features for classification. However, those feature-based algorithms suffer from low accuracy, as they only look at individual user rating patterns. In [7], the authors exploited the statistical properties of spam users, e.g., covariance, to perform detection via variable selection using Principle Component Analysis (PCA-VarSel). PCA-VarSel is very effective when spam users have low covariance, e.g., when they rate items randomly selected from all items, because genuine users have high covariance since they mostly rate only the popular items. However, it was shown in [4] that PCA-VarSel easily fails if spam users also selectively rate only those popular items.

All the existing works focused extensively on rating patterns of spam users, but they did not take into account the effect on the target items. In this work, we develop a probabilistic inference framework that further exploits the target items for attack detection. In particular, we propose a factorized probabilistic model, and apply the efficient Belief Propagation (BP) algorithm [5] for inference. Previously, [8] applied BP to build a privacy-preserving CF system.

2. SHILLING ATTACK DETECTION

2.1 Attack Models

The shilling attack includes two general classes: push attacks and nuke attacks. In push attacks, the spam users collaboratively promote the target items by rating them high, whereas in nuke attacks, the spam users demote the target items by rating them low. In addition, to effectively affect the recommendations made to the genuine users, each spam user also rates a set of filler items to increase similarity with those genuine users. Some well-known shilling attack models include Random attacks and Average attacks [6]. In Random attacks, the set of filler items are rated randomly on a distribution learnt from the ratings in the dataset, whereas in Average attacks, each filler item is rated around the average rating of this individual item, making the spam users more similar to the genuine users. Hence, the Average attack is more effective than the Random attack. Moreover, in both attack models, the set of target items are given the highest allowed rating for push attacks and the lowest rating for nuke attacks.

2.2 Probabilistic Modeling of Attack Detection

We assume there are a set of M users denoted by $\mathbb{U} = \{1, \ldots, M\}$ and a set of N items denoted by $\mathbb{I} = \{1, \ldots, N\}$ in the system. Let r_{ui} denote user u's rating on item i, and $\mathcal{R}$ denote the set of all observed ratings in the system. The set of users who have rated item i is denoted by U_i, and the set of items rated by user u is denoted by I_u. For each user u, we assign a binary variable m_u, where $m_u = 1$ if user u is a spam user and $m_u = 0$ otherwise. Similarly, for each item i, we assign a binary variable t_i, where $t_i = 1$ if item i is a target and $t_i = 0$ otherwise. Let $\mathcal{M} = \{m_u : 1 \leq u \leq M\}$ and $\mathcal{T} = \{t_i : 1 \leq i \leq N\}$. We denote by $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ the joint posterior probability distribution of $\mathcal{M}$ and $\mathcal{T}$ given the observed ratings in $\mathcal{R}$. To identify spam users, we need to find the marginal distributions of $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$. $P(m_u|\mathcal{R})$ can be directly computed from $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ by summing over all variables in $\mathcal{M}$ and $\mathcal{T}$ except m_u as follows

$$P(m_u|\mathcal{R}) = \sum_{\mathcal{M}\setminus m_u, \mathcal{T}} P(\mathcal{M}, \mathcal{T}|\mathcal{R}). \quad (1)$$

However, the computational complexity is $\mathcal{O}\left(2^{M+N}\right)$, which is exponential in the total number of users and items. In the following, we propose a proper factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$, and employ the efficient Belief Propagation (BP) algorithm that operates in a factor graph to infer $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$.

We begin by describing the local functions that $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ factorizes into. Since spam users collaboratively work together to influence the ratings on target items, the local dependency among spam users is induced by their ratings on target items. Let $\mathcal{M}_i = \{m_u : u \in U_i, r_{ui} = r_a\}$, where r_a denotes the maximum rating in cases of push attacks and the minimum rating in cases of nuke attacks. The reader may focus on push attacks, as the method and results hold for the nuke attacks as well. Hence, we introduce the local function $f_i(\mathcal{M}_i, t_i|\mathcal{R})$ to model the local probabilistic dependency among variables in $\{\mathcal{M}_i, t_i\}$. Given a configuration of $\mathcal{M}_i$, we can compute the rating bias caused by spam users as

$$\Delta r_i(\mathcal{M}_i) = \left| \frac{\sum_{u \in U_i} r_{ui}}{|U_i|} - \frac{\sum_{u \in U_i} r_{ui} - r_a \sum_{m \in \mathcal{M}_i} m}{|U_i| - \sum_{m \in \mathcal{M}_i} m} \right|, \quad (2)$$

where the first term in the absolute operator is the average rating on item i, including all ratings, and the second term is the unbiased average rating after removing ratings from spam users. Based on (2), we express $f_i(\mathcal{M}_i, t_i|\mathcal{R})$ as

$$f_i(\mathcal{M}_i, t_i|\mathcal{R}) = \frac{1}{1 + \exp\left((-1)^{1-t_i} \alpha_t \left(\Delta r_i(\mathcal{M}_i) - \delta_r\right)\right)}, \quad (3)$$

where $\alpha_t < 0$ and $\delta_r > 0$, and both are adjustable scalars. Given a configuration of $\mathcal{M}_i$, if the rating bias $\Delta r_i(\mathcal{M}_i)$ exceeds δ_r, this function assigns a larger value for $t_i = 1$ than for $t_i = 0$, and vice versa.

We also introduce a local factor function for each variable in $\mathcal{M}$ and $\mathcal{T}$, so as to incorporate the local information extracted from each individual user and item. Let $g_u(m_u|\mathcal{R})$ be the local function for $m_u \in \mathcal{M}$. $g_u(m_u|\mathcal{R})$ can be designed to take into account the rating patterns of individual users, such as those user features introduced in [2]. Here, for the purpose of illustration, we consider the use of a single feature ϕ_u of user u. A simple probabilistic discriminative classifier [1] based solely on feature ϕ_u can be given in the following form

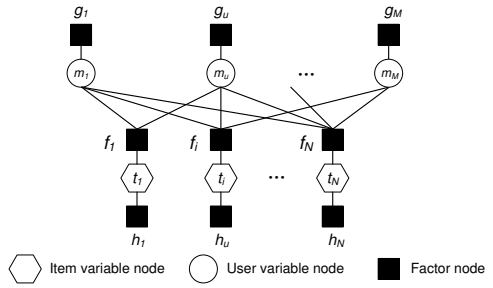


Figure 1: The factor graph for attack detection.

$$g_u(m_u|\mathcal{R}) = \frac{1}{1 + \exp\left((-1)^{1-m_u} \beta_1 (\phi_u - \tau_1)\right)}, \quad (4)$$

where β_1 and τ_1 are design parameters.

Likewise, we define a local factor function $h_i(t_i|\mathcal{R})$ for each $t_i \in \mathcal{T}$. We notice that the ratings of target items are more likely to have large variances than those of normal items, due to the eccentric ratings from spam users. Let φ_i denote the variance in ratings of item i. Similar to $g_u(m_u|\mathcal{R})$, we define $h_i(t_i|\mathcal{R})$ as follows

$$h_i(t_i|\mathcal{R}) = \frac{1}{1 + \exp\left((-1)^{1-t_i} \beta_2 (\varphi_i - \tau_2)\right)}, \quad (5)$$

where β_2 and τ_2 are design parameters.

Based on the local functions, we can express the factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ as

$$P(\mathcal{M}, \mathcal{T}|\mathcal{R}) = \frac{1}{Z} \prod_{i \in \mathbb{I}} f_i(\mathcal{M}_i, t_i|\mathcal{R}) \times \prod_{u \in \mathbb{U}} g_u(m_u|\mathcal{R}) \prod_{i \in \mathbb{I}} h_i(t_i|\mathcal{R}), \quad (6)$$

where Z is a normalization factor.

2.3 BP-based Inference in A Factor Graph

A factor graph is a bipartite graph that expresses the factorization structure of a function, where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function associated with the factor node [5]. We illustrate the factor graph that expresses the factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ by (6) in Fig. 1. Specifically, user variable m_u is represented by node m_u, and item variable t_i is represented by node t_i in Fig. 1. Each factor function in (6) is represented by a factor node. We connect each factor node g_i to item variable node t_i, and connect each factor node h_u to user variable node m_u. Also, we connect each factor node f_i to the user variable nodes in $\mathcal{M}_i$ and the item variable node t_i.

To infer the marginal distributions $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$, we apply BP in the factor graph to exploit the factorization for efficient inference. Since the constructed factor graph has loops, we cannot apply the standard BP algorithm for exact inference. We thus resort to "loopy" BP, which performs iterative message passing between variable nodes and factor nodes along the edges in the factor graph [5].

2.4 Complexity Reduction

While the BP algorithm can be much more efficient than direct computation using (1), the computational complexity

for generating messages at the factor node is exponential in the degree of the factor node. This computational burden can be quite significant at factor node f_i, where the complexity in terms of multiplications is $\mathcal{O}\left(|\mathcal{M}_i|2^{|\mathcal{M}_i|}\right)$ for generating a message. Since in recommender systems an item can be rated by over hundreds of users, it renders the BP algorithm almost practically infeasible. Hence, we propose a complexity reduction technique by reducing the degree of factor node f_i as follows.

We randomly divide the user variable nodes in $\mathcal{M}_i$ into $G_i = \lceil |\mathcal{M}_i|/D \rceil$ groups, where D is a small integer. Let $\mathcal{M}_i^{(k)}$ denote the set of user variable nodes in group k, and M_{ik} be the size of group k, where $M_{ik} = D$ for $1 \leq k < G_i$, and $M_{ik} = |\mathcal{M}_i| \bmod D$ for $k = G_i$. Assuming independence among groups, we can approximate (6) using

$$P(\mathcal{M}, \mathcal{T}|\mathcal{R}) = \frac{1}{Z} \prod_{i \in \mathbb{I}} \prod_{k=1}^{G_i} f_i^{(k)} \left(\mathcal{M}_i^{(k)}, t_i|\mathcal{R}\right) \times \prod_{u \in \mathbb{U}} g_u(m_u|\mathcal{R}) \prod_{i \in \mathbb{I}} h_i(t_i|\mathcal{R}), \quad (7)$$

where we derive $f_i^{(k)}\left(\mathcal{M}_i^{(k)}, t_i|\mathcal{R}\right)$ from (3) by replacing $\Delta r_i\left(\mathcal{M}_i\right)$ with

$$\Delta r_i\left(\mathcal{M}_i^{(k)}\right) = \left| \frac{\hat{R}_i w_{ik} + r_a M_{ik}}{|\hat{U}_i| w_{ik} + M_{ik}} - \frac{\hat{R}_i w_{ik} + r_a \sum_{m \in \mathcal{M}_i^{(k)}} m}{|\hat{U}_i| w_{ik} + \sum_{m \in \mathcal{M}_i^{(k)}} m} \right|,$$

where $\hat{U}_i = \{u : u \in U_i, r_{ui} \neq r_a\}$, $\hat{R}_i = \sum_{u \in \hat{U}_i} r_{ui}$, and $w_{ik} = M_{ik}/|\mathcal{M}_i|$. Note that we allocate $\hat{R}_i$, the sum of ratings from $\hat{U}_i$, to each group in proportion to the group size. Similarly as in Sec. 2.3, we construct a new factor graph for (7) and apply the BP algorithm. The computational complexity at factor node $f_i^{(k)}$ is $\mathcal{O}\left(D2^D\right)$. The overall complexity of the BP algorithm with complexity reduction is $\mathcal{O}\left(D2^D|\mathcal{R}_a|\right)$, where $\mathcal{R}_a$ is the subset of the observed ratings with value r_a.

3. EXPERIMENTAL EVALUATION

3.1 Experiment Setup

We evaluate the performance of the proposed BP-based attack detection algorithm using the 100K MovieLens dataset[1]. The dataset contains 100,000 ratings, all integers from 1 to 5, on 1682 items (movies) by 943 users. We treat the original users in the dataset as genuine users. To launch shilling attacks, a number of spam users are injected into the system. The ratio of spam users to genuine users is set as $\frac{1}{10}$. Each spam user randomly selects a set of filler items from the top 50% most popular items according to the number of ratings each item receives, similar to [4]. We refer to the ratio of filler items to all items as *filler size*. We consider the Average attack model described in Sec. 2.1, since it is more effective than the Random attack. The rating on each filler item follows a normal distribution with standard deviation σ, and its mean is set as the average rating received by the filler item. Finally, a set of items are selected as targets in the attack. For the push attack, each target item has an average rating between 1 and 3, whereas for the nuke attack, each target item has an average rating between 3 and 5. We assume all spam users have the same set of targets.

We evaluate the performance of the attack detection algorithms in terms of $Precision = |\mathcal{D} \cap \mathcal{S}|/|\mathcal{D}|$ and $Recall = |\mathcal{D} \cap \mathcal{S}|/|\mathcal{S}|$, where $\mathcal{D}$ and $\mathcal{S}$ denote the sets of detected spam users and true spam users, respectively. We compare the performance of various detection algorithms, including the proposed BP algorithm, the PCA-VarSel algorithm in [7], and the feature-based algorithm using (4).

To assess the detection performance, the PCA-VarSel algorithm described in [7] sorts the users in ascending order of their contribution to the principle components (PCs) obtained from PCA analysis, and selects the top-M_d listed users as spam users. For easy comparison and fairness, in the proposed BP algorithm, we likewise sort users in descending order of $P(m_u = 1|\mathcal{R})$, and also select the top-M_d users as spam users. Similarly, in the feature-based algorithm, we select the top-M_d users ranked in descending order of $g_u(m_u = 1|\mathcal{R})$. In our experiments, we set M_d as the number of true spam users injected into the system, so the *Precision* and *Recall* are equal.

In the proposed BP algorithm, we adopt the MeanVar feature introduced in [2] for ϕ_u in (4). The MeanVar ϕ_u of user u is computed as

$$\phi_u = \frac{\sum_{i \in I_u \setminus \bar{I}_u} (r_{ui} - \bar{r}_i)^2}{|I_u \setminus \bar{I}_u|},$$

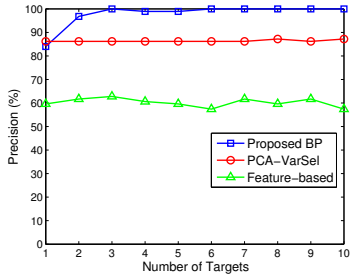
where $\bar{r}_i$ is the average rating of item i, and $\bar{I}_u = \{i : r_{ui} = r_a, i \in I_u\}$. The MeanVar feature is particularly effective for detecting spam users in Average attacks.

3.2 Results and Discussion

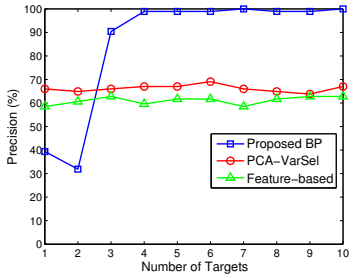
In Fig. 2, we present the results for detecting spam users in Push attacks, where we have set the parameters of the proposed BP algorithm as $\alpha_t = -3$ and $\delta_r = 0.35$ in (3), $\beta_1 = -1$ and $\tau_1 = 0.5$ in (4), $\beta_2 = 1$ and $\tau_2 = 1.5$ in (5), and the group size $D = 8$ for complexity reduction described in Sec. 2.4. The results show that the performance of the proposed BP algorithm improves significantly as the number of targets increases, reaching almost 100% precision when there are enough number of targets, whereas the other algorithms do not exhibit such improvement. This verifies that the proposed algorithm can effectively exploit the target items to detect spam users with high accuracy. Since the BP algorithm also incorporates the output $g_u(m_u|\mathcal{R})$ of the feature-based algorithm as illustrated in Fig. 1, we would like to examine the case when the performance of the feature-based algorithm is very poor. In Fig. 2c, the detection precision of the feature-based algorithm is only around 20% after we set $\sigma = 0.7$. Interestingly, we observe a threshold phenomenon for the BP algorithm, that is the BP algorithm has 0% precision when the number of targets is small, but when the number of targets exceeds a certain number, the BP algorithm provides almost 100% precision. This is because as the number of targets increases, the information gained from target items becomes dominant and corrects the results of the feature-based algorithm. Note that in practical implementations, we can incorporate into the BP algorithm more advanced feature-based algorithms for better performance. We have also performed experiments for Nuke attacks, and similar results can be observed, so we only present one set of those results in Fig. 3.

In Fig. 4, we investigate the impact of the filler size on the performance of the BP algorithm. We show the detec-

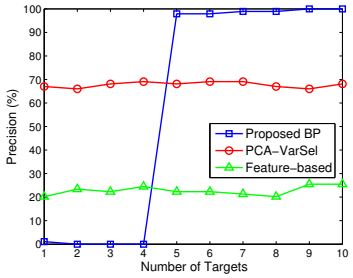
[1]Available at: http://www.grouplens.org/node/73.



(a) Filler size = 5% and $\sigma = 0.6$



(b) Filler size = 10% and $\sigma = 0.6$



(c) Filler size = 10% and $\sigma = 0.7$

Figure 2: Detection precision versus number of targets in *Push* attacks.

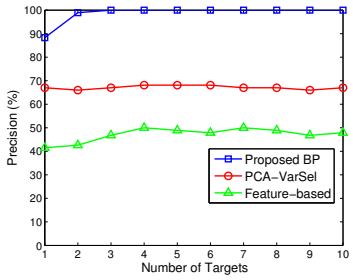
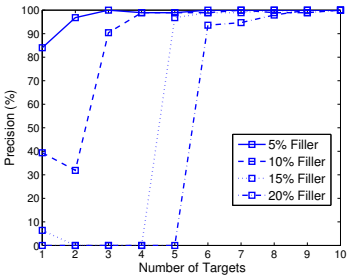
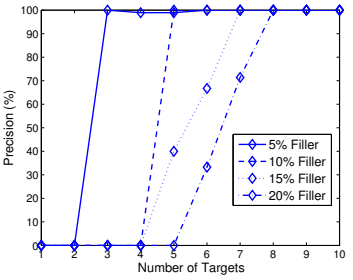


Figure 3: Detection precision versus number of targets in *Nuke* attacks. Filler size = 10% and $\sigma = 0.7$.



(a) Detection precision of spam users versus number of targets



(b) Detection precision of target items versus number of targets

Figure 4: Detection performance in *Push* attacks under varying filler sizes. $\sigma = 0.6$.

tion results for both spam users and target items in Fig. 4a and Fig. 4b, respectively. Here, to evaluate the detection performance for the target items, we infer $P(t_i|\mathcal{R})$ from (6) and select the top-N_t items ranked in descending order of $P(t_i = 1|\mathcal{R})$, where N_t is set as the number of true targets. Note that the existing algorithms only detect spam users. As the filler size increases, the BP algorithm suffers from performance loss for small numbers of targets. This is because the BP algorithm relies on information from target items, but the noise information from the filler items corrupts the useful information. Indeed, we can see in Fig. 4b that the detection precision for target items drops with increasing filler size. However, when there are enough target items to suppress the noise information, the BP algorithm still can achieve high detection precision. It is worth noting that in practice a higher cost is incurred for attackers to increase the filler size while decreasing the target number. Also, the BP algorithm can be enhanced with feature-based algorithms that deliver high detection accuracy for large filler sizes.

4. CONCLUSION

To protect the collaborative filtering systems against the shilling attacks, we proposed a BP-based algorithm for detecting spam users in a probabilistic inference framework. Different from the existing algorithms that rely solely on users' rating patterns, the proposed algorithm further exploits the target items. Considering the computational complexity for inference, we developed a factorized probabilistic model for attack detection and applied BP to perform inference efficiently. The proposed detection algorithm can also conveniently incorporate the existing algorithms for enhanced performance. We showed through experiments that the proposed BP algorithm significantly improves detection accuracy as the number of target items increases. We also observed that its performance degrades with increasing filler sizes. In the future work, we will develop more robust algorithms to deal with more sophisticated attack scenarios.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1115199.

5. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. In *Proc. KDD'06*, pages 542–547, 2006.

[3] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proc. WIDM'05*, pages 67–74, 2005.

[4] N. Hurley, Z. Cheng, and M. Zhang. Statistical attack detection. In *Proc. RecSys'09*, pages 149–156, 2009.

[5] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498–519, Feb. 2001.

[6] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *WWW'04*, pages 393–402, 2004.

[7] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: Detecting spam users in collaborative filtering. In *Proc. IUI'07*, pages 14–21, 2007.

[8] J. Zou, A. Einolghozati, and F. Fekri. Privacy-preserving item-based collaborative filtering using semi-distributed belief propagation. In *Proc. IEEE CNS'13*, 2013.