Recommender Systems Robust to Data Poisoning using Trim Learning

Seira Hidano and Shinsaku Kiyomoto
KDDI Research Inc., Saitama, Japan

Keywords: Recommender Systems, Matrix Factorization, Data Poisoning, Trim Learning.

Abstract: Recommender systems have been widely utilized in various e-commerce systems for improving user experience. However, since security threats, such as fake reviews and fake ratings, are becoming apparent, users are beginning to have their doubts about trust of such systems. The data poisoning attack is one of representative attacks for recommender systems. While acting as a legitimate user on the system, the adversary attempts to manipulate recommended items using fake ratings. Although several defense methods also have been proposed, most of them require prior knowledge on real and/or fake ratings. We thus propose recommender systems robust to data poisoning without any knowledge.

1 INTRODUCTION

Recommender systems have an essential role in improving user experience in recent e-commerce systems. Users can efficiently find out potentially preferable items by using the recommender system. However, recommended items are selected based on ratings from other users. Since in such an environment, anybody can easily provide malicious data, there is concern about security risks, such as fake reviews and fake ratings.

In this paper, we especially focus on data poisoning attacks for recommender systems. The data poisoning attack manipulates recommended items by injecting fake ratings to the system. Although several attacks have been proposed, defense techniques have not really reached maturity yet. Most of existing defense methods require prior knowledge on real and/or fake ratings. We thus propose recommender systems robust to data poisoning without any knowledge.

1.1 Related Work

The main objective of data poisoning attacks for recommender systems is to raise or lose the popularity of specific items. To this end, (Burke et al., 2005) discussed creating ratings of malicious users with limited knowledge, and introduced three attack models: *random attacks*, *average attacks*, and *bandwagon attacks*. (Williams et al., 2006) additionally provided an attack model called *obfuscated attacks*, which makes it more difficult to detect malicious users.

Defense methods are categorized into two models: supervised approach and non-supervised approach. In the supervised setting, detectors are trained with a set of rating data with a label of a legitimate or malicious user. Several features have been designed for this model (Chirita et al., 2005; Burke et al., 2006; Mobasher et al., 2007; Wu et al., 2011; Wu et al., 2012). Although this type of detectors achieve high accuracy, conditions are not practical. For instance, it is difficult to obtain rating data with a label in actual usecases. On the other hand, non-supervised methods generate a detector using a rating data set without labels. (Mehta, 2007) developed this type of first detector based on PCA, which we call the PCA-based method in this paper. However, it was shown that non-supervised methods are vulnerable to advanced attacks like the obfuscated attack (Li et al., 2016).

1.2 Our Contribution

We propose a new defense method without any knowledge, which is more effective than existing methods. Our main contribution is summarized as follows:

- We design a matrix factorization algorithm using trim learning. Matrix factorization is the most representative method for predicting unobserved ratings. Trim learning, proposed by (Jagielski et al., 2018), is a learning method robust to data poisoning for linear regression. By applying the concept of the trim learning to the matrix factorization, we develop a learning algorithm that precludes fake

Hidano, S. and Kiyomoto, S.
Recommender Systems Robust to Data Poisoning using Trim Learning.
DOI: 10.5220/0009180407210724
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 721-724
ISBN: 978-989-758-399-5; ISSN: 2184-4356

ratings with high accuracy.

- We demonstrate the effect of our proposed method with a real-world dataset. Assuming four types of data poisoning attacks, we compare the performance of our proposed method with another existing detection method. The results show that our proposed method improves robustness to data poisoning dramatically.

2 PRELIMINARIES

2.1 Recommender Systems

We focus on collaborating filtering based on matrix factorization that is widely used in real-world recommender systems (Koren et al., 2009). Let m and n be the numbers of users and items in the system, respectively. Let $\mathbf{M} \in \mathbb{R}^{m \times n}$ be a rating matrix including a large number of missing elements. We denote by Ω a set of indices whose ratings are observed. $\mathbf{M}_{i,j}$ indicates a rating of the ith user for the jth item. Matrix factorization is used to generate two latent factors $\mathbf{U} \in \mathbb{R}^{m \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times n}$, where $k \ll \min(m,n)$ is a positive integer. We call $\mathbf{U}$ and $\mathbf{V}$ a *user matrix* and an *item matrix*, respectively. These are given by solving the following optimization problem:

$$\min_{\mathbf{U},\mathbf{V}} \sum_{(i,j)\in\Omega} (\mathbf{M}_{i,j} - \mathbf{u}_i \mathbf{v}_j)^2 + \lambda(||\mathbf{U}||^2 + ||\mathbf{V}||^2), \quad (1)$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ indicate the ith row of $\mathbf{U}$ and the jth column of $\mathbf{V}$, respectively, and λ (≥ 0) is a regularization parameter. $||\cdot||$ is the Frobenius norm.

The recommender system predicts the values of missing elements in $\mathbf{M}$ by calculating $\widehat{\mathbf{M}} = \mathbf{UV}$. Recommended items are selected based on the predicted ratings.

Stochastic gradient decent (SGD) and *alternating minimization* are well-known as optimization methods to solve Eq. (1). Since the SGD method is faster and easier to implement for a sparse matrix like a rating matrix $\mathbf{M}$, we develop our learning algorithm based on it. The algorithm loops through all the ratings in a training set. We here define a prediction error for a given rating as

$$e_{i,j} := \mathbf{M}_{i,j} - \mathbf{u}_i \mathbf{v}_j. \quad (2)$$

At each iteration, both of the user matrix $\mathbf{U}$ and the item matrix $\mathbf{V}$ are modified as follows:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i + \eta \cdot (e_{i,j} \cdot \mathbf{u}_i - \lambda \cdot \mathbf{v}_j) \quad (3)$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \eta \cdot (e_{i,j} \cdot \mathbf{v}_i - \lambda \cdot \mathbf{u}_i), \quad (4)$$

where η (> 0) is a learning rate.

2.2 Data Poisoning Attacks

In the data poisoning attack for a recommender system, the adversary creates multiple user accounts, and injects fake ratings. Let $\mathbf{M}' \in \mathbb{R}^{m' \times n}$ be a fake rating matrix injected to the recommender system, where m' is the number of the malicious users. We define *attack size* as $\alpha := m'/m$.

We especially focus on an attack whereby the popularity of target items is elevated, which is called a *push attack*. Ratings from a malicious user for the push attack consist of three parts:

- *Target Items*: The items whose popularity is what the adversary would like to increase. The highest rate is assigned.
- *Filler Items*: The items are rated to make detection difficult. *Filler Size* β is measured as a ration of the number of the filler items to n.
- *Unrated Items*: The remaining items that are not rated occupy a majority of a data point.

Rating values for filler items are assigned based on either of the following four strategies (Burke et al., 2005; Burke et al., 2006):

- *Random Attacks*: The filler items are randomly rated around the overall average rating value.
- *Average Attacks*: The filler items are randomly rated around the average rating value of each item.
- *Bandwagon Attacks*: Some filler items are the most popular items and rated with the maximum rating value. The other filler items are randomly rated around the overall average rating value. In the experiments of this paper, half of filler items are assigned to the most popular items.
- *Obfuscated Attacks*: All the filler items are randomly selected among some highly popular items, and randomly rated around the overall average rating value.

2.3 Trim Learning

The trim learning, proposed by (Jagielski et al., 2018), is a learning method robust to data poisoning. Let $\mathcal{D}$ be a training set that consists of N legitimate samples and N' malicious samples. Let L be a loss function with model parameters θ. Jagielski et al. formally defined the optimization problem as follows:

$$\min_{\theta, I} L(\mathcal{D}^I, \theta)$$
$$\text{s.t. } I \subset [N+N'], |I| = N, \quad (5)$$

where $\mathcal{D}^I \subset \mathcal{D}$ is a set of data samples corresponding to indices I.

Jagielski et al. provided an optimization algorithm to solve Eq. (5) for linear regression. We apply this concept to matrix factorization, and develop a learning algorithm robust to data poisoning for a recommender system.

3 METHOD

We propose a learning method robust to data poisoning for a recommender system by combining matrix factorization and trim learning. This section defines an optimization problem for our method, and develops a robust matrix factorization algorithm.

3.1 Problem Setting

Since it is difficult for the adversary to imitate legitimate users, the behavior of malicious users is statistically different from that of legitimate users. As a result, training loss for malicious users will be larger, as compared to legitimate users. In addition, contaminated items such as target items and filler items make training loss larger for both legitimate users and malicious users. We thus design an algorithm to learn a model while trimming both of malicious users and contaminated items.

To this end, we define an optimization problem for our proposed method as follows:

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V},I_u,I_v} & L(\mathbf{M},\mathbf{U},\mathbf{V},I_u,I_v) \\ &= \sum_{(i,j)\in\Omega^{I_u\times I_v}} (\mathbf{M}_{i,j}-\mathbf{u}_i\mathbf{v}_j)^2+\lambda(||\mathbf{U}||^2+||\mathbf{V}||^2) \\ \text{s.t. } & I_u\subset[m],\ I_v\subset[n],\ |I_u|=m^*,\ |I_v|=n^*, \quad (6)\end{aligned}$$

where I_u and I_v are subsets of users and items, respectively. m^* and n^* are the numbers of users and items that are not trimmed, respectively. $\Omega^{I_u\times I_v}$ is a set of observed elements in $\mathbf{M}$ corresponding to only users I_u and items I_v.

3.2 Algorithm

We adopt alternative minimization for solving the optimization problem (6). Our algorithm first solves $\mathbf{U}$ and $\mathbf{V}$ while keeping I_u and I_v. Then I_u and I_v be solved with $\mathbf{U}$ and $\mathbf{V}$ fixed.

Let $\mathbf{U}^*$ and $\mathbf{V}^*$ be user and item matrices corresponding to I_u and I_v, respectively. Let $\mathbf{U}^c$ and $\mathbf{V}^c$ be user and item matrices corresponding to trimmed users $I_u^c=[m]\setminus I_u$ and trimmed items $I_v^c=[n]\setminus I_v$, respectively. In the optimization process of $\mathbf{U}$ and $\mathbf{V}$, $\mathbf{U}^*$ and $\mathbf{V}^*$ are first solved using the SGD method described in Section 2.1. After that, $\mathbf{U}^c$ (resp. $\mathbf{V}^c$) are solved with $\mathbf{V}^*$ (resp. $\mathbf{U}^*$) fixed. I_u and I_v can also be solved using alternative minimization. Algorithm 1 summarizes the algorithm of our proposed method.

Algorithm 1: Trim matrix factorization.

Input : $\mathbf{M}$, m, n, k, m^*, n^*, α, β, λ, η
Output: $\mathbf{U}^*$, $\mathbf{V}^*$, I_u, I_v

Initialize $I_u^{(0)}$, $I_v^{(0)}$.
$t\leftarrow 0$
repeat
  $\mathbf{U}^{*(t)},\mathbf{V}^{*(t)}\leftarrow \arg\min_{\mathbf{U}^*,\mathbf{V}^*} L(\mathbf{M},\mathbf{U}^*,\mathbf{V}^*,I_u^{(t)},I_v^{(t)})$
  $I_u^{c(t)}\leftarrow[m]\setminus I_u^{(t)}$, $I_v^{c(t)}\leftarrow[n]\setminus I_v^{(t)}$
  $\mathbf{U}^{c(t)}\leftarrow \arg\min_{\mathbf{U}^c} L(\mathbf{M},\mathbf{U}^c,\mathbf{V}^{*(t)},I_u^{c(t)},I_v^{(t)})$
  $\mathbf{V}^{c(t)}\leftarrow \arg\min_{\mathbf{V}^c} L(\mathbf{M},\mathbf{U}^{*(t)},\mathbf{V}^c,I_u^{(t)},I_v^{c(t)})$
  $\mathbf{U}^{(t)}\leftarrow \mathsf{concatenate}(\mathbf{U}^{*(t)},\mathbf{U}^{c(t)})$
  $\mathbf{V}^{(t)}\leftarrow \mathsf{concatenate}(\mathbf{V}^{*(t)},\mathbf{V}^{c(t)})$
  $I_u^{(t+1)}, I_v^{(t+1)}\leftarrow \arg\min_{I_u,I_v} L(\mathbf{M},\mathbf{U}^{(t)},\mathbf{V}^{(t)},I_u,I_v,)$
  $t\leftarrow t+1$
until $I_u^{(t)}=I_u^{(t-1)}\wedge I_v^{(t)}=I_v^{(t-1)}$
return $\mathbf{U}^{*(t)}$, $\mathbf{V}^{*(t)}$, $I_u^{(t)}$, $I_v^{(t)}$

4 EXPERIMENTS

4.1 Setup

We used the MovieLens 1M dataset (GroupLens Research, 2016) as a real-world dataset. We implemented four types of attacks described in Section 2.2. The attack size was set to $\alpha=0.05$, 0.1, 0.15 or 0.2. An item was randomly selected as a target item. The filler size was set to $\beta=0.05$, 0.1, 0.15 or 0.2, and filler items were also randomly selected for each value of β. We conducted each attack 10 times for each parameter set in order to stabilize results. Parameters of our proposed method were set to $k=32$, $m^*=m$, $n^*=(1-\beta)n-1$, $\lambda=0.01$, and $\eta=0.02$.

Here we define detection rate as a metric for our proposed method. The detection rate is the ratio of the number of trimmed malicious users to the total number of malicious users. We evaluated the effect of our proposed method using the average value of the detection rate. For comparison, we also implemented the PCA-based method (Mehta, 2007), and evaluated its performance with the same metric.

Table 1: Detection rates of our proposed method ($\beta = 0.1$).

Attack size	0.05	0.1	0.15	0.2
Random	0.887	0.941	0.963	0.974
Average	0.880	0.938	0.960	0.970
Bandwagon	0.793	0.859	0.897	0.918
Obfuscated	0.827	0.921	0.938	0.956

Table 2: Detection rates of PCA based method ($\beta = 0.1$).

Attack size	0.05	0.1	0.15	0.2
Random	0.0	0.010	0.029	0.121
Average	0.667	0.754	0.758	0.762
Bandwagon	0.733	0.770	0.802	0.780
Obfuscated	0.0	0.111	0.323	0.475

Table 3: Detection rates of our proposed method ($\alpha = 0.1$).

Filler size	0.05	0.1	0.15	0.2
Random	0.761	0.941	0.993	1.0
Average	0.757	0.938	0.980	1.0
Bandwagon	0.318	0.859	0.948	0.987
Obfuscated	0.333	0.921	0.970	1.0

Table 4: Detection rates of PCA based method ($\alpha = 0.1$).

Filler size	0.05	0.1	0.15	0.2
Random	0.0	0.010	0.367	0.711
Average	0.0	0.754	0.754	0.784
Bandwagon	0.06	0.770	0.770	0.770
Obfuscated	0.0	0.111	0.692	0.715

4.2 Evaluation Results

Tables 1 and 2 show detection rates of our proposed method and the PCA-based method, respectively, when the filler size $\beta = 0.1$. Tables 3 and 4 show the results for $\alpha = 0.1$. It can be shown that the detection rate of our proposed method is much higher than that of the PCA-based method for all the settings. Consequently, we can say that our proposed method improves robustness to data poisoning dramatically.

5 CONCLUSIONS

This paper proposed recommender systems robust to data poisoning. Our proposed method is a combination of matrix factorization and trim learning. The algorithm trains a model for recommendation while trimming malicious users and contaminated items. The experimental results showed that our proposed method improves robustness to data poisoning dramatically.

In the feature, we will conduct additional experiments with other real-world datasets as well as theoretical analysis of our proposed method. Furthermore, we will apply the concept of our proposed method to more complicated learning methods utilized in recommender systems.

REFERENCES

Burke, R., Mobasher, B., and Bhaumik, R. (2005). Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. (2006). Classification features for attack detection in collaborative recommender systems. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Chirita, P., Nejdl, W., and Zamfir, C. (2005). Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th ACM international workshop on Web information and data management (WIDM)*, pages 67–74.

GroupLens Research (2016). MovieLens 1M Dataset. http://grouplens.org/datasets/movielens/.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer and Information Science*, 42(8):30–37.

Li, W., Gao, M., Li, H., and Zeng, J. (2016). Shilling attack detection in recommender systems via selecting patterns analysis. *IEICE Transactions on Information and Systems*, E99-10(10).

Mehta, B. (2007). Unsupervised shilling detection for collaborative filtering. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*.

Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. (2007). Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology (TOIT)*, 7(23).

Williams, C., Mobasher, B., Burke, R. D., Sandvig, J. J., and Bhaumik, R. (2006). Detection of obfuscated attacks in collaborative recommender systems. In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*.

Wu, Z., Cao, J., Mao, B., and Wang, Y. (2011). Semi-SAD: Applying semi-supervised learning to shilling attack detection. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*.

Wu, Z., Wu, J., Cao, J., and Tao, D. (2012). HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.