Adversarial Attacks and Detection on reinforcement learning-based Interactive Recommender Systems

Anonymous Author(s)*

ABSTRACT

Adversarial attacks pose significant challenges for detecting adversarial attacks at an early stage. We propose attack-agnostic detection on reinforcement learning-based interactive recommendation systems. We first craft adversarial examples to show their diverse distributions and then augment recommendation systems by detecting potential attacks with a deep learning-based classifier based on the crafted data. Finally, we study the attack strength and frequency of adversarial examples and evaluate our model on standard datasets with multiple crafting methods. Our extensive experiments show that most adversarial attacks are effective, and both attack strength and attack frequency impact the attack performance. The strategically-timed attack achieves comparative attack performance with only 1/3 to 1/2 attack frequency. Besides, our black-box detector trained with one crafting method has the generalization ability over several crafting methods.

KEYWORDS

Adversarial Attack, Adversarial Examples Detection, Reinforcement Learning, Interactive Recommender System

ACM Reference Format:
Anonymous Author(s). 2020. Adversarial Attacks and Detection on reinforcement learning-based Interactive Recommender Systems. In *Proceedings of SIGIR '20: ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Interactive recommendation systems capture dynamic personalized user preferences by improving their strategies continuously [7, 12, 13]. They have attracted enormous attention and been applied in leading companies like Amazon, Netflix, and Youtube. The traditional methods to model user-system interactions include Multi-Armed Bandit (MAB) or Reinforcement Learning (RL). The former views action choices as a repeated single process, while the latter considers immediate and future rewards to model behaviors' long-term benefits. RL-based systems employ a Markov Decision Process (MDP) agent that estimates the value based on both actions and states, rather than merely on actions as done by MAB.

However, reinforcement learning-based models can be fooled by small disturbances on the input data [3, 11]. Small imperceptible noises, such as adversarial examples, may increase prediction error or reduce reward in supervised and RL tasks—the input noise can be transferred to attack different parameters even different models, including recurrent network and RL [2, 5]. Besides, the embedding vectors of users, items and relations are piped into RL-based recommendation models, making it challenging for humans to tell the true value or to dig out the real issues in the models. Attackers can easily leverage such characteristics to disrupt recommendation systems silently, making defending adversarial attacks a non-trivial task for RL-based recommendation systems.

In this work, we aim to develop a general detection model to detect attacks and increase the defence ability, which provides a practical strategy to overcome the dynamic 'arm-race' of attacks and to defend in the long run. We make the following contributions:

- We systematically investigate adversarial attacks and detection approaches with a focus on reinforcement learning-based recommendation systems and demonstrate the effectiveness of the designed adversarial examples and strategically-timed attack.
- We propose an encoder-classification detection model for attack-agnostic detection. The encoder captures the temporal relationship among sequence actions in reinforcement learning. We further use an attention-based classifier to highlight the critical time steps out of ample interactive space.
- We empirically show that even small perturbations can reduce the performance of most attack methods significantly. Our statistical validation shows that multiple attack methods generate similar actions of the attacked system, providing insights into improving the detection performance.

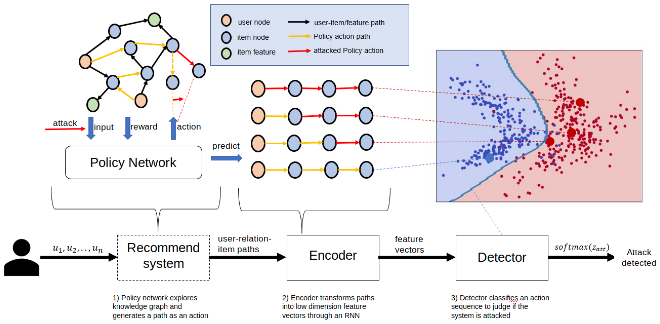

Figure 1: Our proposed Adversarial Attack and Detection Approach for RL-based Recommender Systems.

2 METHODOLOGY

2.1 RL-based Interactive Recommendation

Interactive recommendation systems suggest items to users and receives feedback. Given a user $u_j \in U = \{u_0, u_1, u_2, ..., u_n\}$, a set of items $I = \{i_0, i_1, i_2, ..., i_n\}$, and the user feedback history $i_{k_1}, i_{k_2}, ..., i_{k_{t-1}}$, the recommendation system suggests a new item i_{k_t}. This problem represents a Markov Decision Process as follows:

SIGIR '20, July 25–30, 2020, Xi'an, China
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/1122445.1122456

- State (s_t): a historical interaction between a user and the recommendation system computed by an embedding or encoder module.
- Action (a_t): an item or a set of items recommended by the RL agent.
- Reward (r_t): a variable related to a user's feedback to guide the reinforcement model towards true user preference.
- Policy ($\pi(a_t|s_t)$): a conditional probability distribution of items which the agent might recommend to a user u_i given the state of last time step s_{t-1}. The learning process aims to get an optimal policy.
- Value function ($Q(s_t, a_t)$): the agent's estimation of reward of current states s_t and recommended item a_t. We define the reward as the cosine similarity between user and item embedding vectors.

The reinforcement agent could follow REINFORCE with baseline or Actor-Critic algorithm that both consist of a value network and a policy network [14]. The attack model may generate adversarial examples using either the value network [5] or the policy network[10].

2.2 Attack Model

FGSM-based attack. We define an adversarial example as a little perturbation δ added onto the benign examples x, which can be a composition of embedding vectors of users, relations and items [14]. Unlike perturbations on images or texts, δ can be large due to the enormous manual work to check the embedding vectors. We define an adversarial example as

$$\begin{aligned} &\min_{\delta} R_T = \sum_{t=1}^{T} Q(s_t + \delta,\ a_t). \\ &a_t = \pi^*(a_t|s_t + \delta) \quad \text{subject to } S(s_t, s_t + \delta) \leq l \end{aligned} \tag{1}$$

Attack with smaller frequency. The strategically-timed attack [6] aims to decreases the attack frequency without sacrificing the performance of the un-targeted reinforcement attack. We formally present it below:

$$\delta_t = \delta_t * c_t \quad c_t \in \{0, 1\}, \quad \frac{\sum_{t=1}^{T} c_t}{T} < d \tag{2}$$

where c_t is a binary variable that controls when to attack; $d < T$ is the frequency of adversarial examples. There are two approaches to generate the binary sequence $c_{1:T}$ optimizing a hard integer programming problem and generating sequences via heuristic methods. Let p_0, p_1 be the two maximum probability of an policy π, we define c_t as follwos, which is different from [6]:

$$c_t = (p_0 - p_1) > threshold$$

In our experiments, we let the RL-based recommendation system to have a peak probability at the maximum action so as to test if the importance of the action to attackers using the above formula. In contrast, Jacobian-based Saliency Map Attack (JSMA) [9] and Deepfool [8] are based on the gradient of actions rather than the gradient of Q value. One key component of JSMA is saliency map computation, which decides which dimension of vectors (in Image classification is pixels) are modified. Deepfool pinpoints the attack dimension by comparison of affine distances between some class and temporal class.

2.3 Detection Model

The detection model is a supervised classifier, which detects adversarial examples based on the actions of the reinforcement agent in a general feature space. Suppose the action distributions of an agent are shifted by adversarial examples (Section 3 shows statistical evidence of the drift). Given an abnormal action sequence $a = \pi^*(a|s + \delta)$, the detection model aims to establisha separating hyperplane between adversarial examples and normal examples, thereby measuring the probability $p(y|a, \theta)$ or $p(y|\pi^*, s, \delta, \theta)$, where y is a binary variable indicating whether the input data are attacked.

To detect the adversarial examples presented in the last section, we employ an attention-based classifier. We first conduct a statistical analysis of the attacked actions in section 3. The detection model consists of two parts. The first is an encoder, to encode the action methods into a low dimensional feature vector. The second is a classifier to separate different data. We adopt this encoder-decoder model to make a bottleneck and filter out noisy information. The formulation of GRU is as follows:

$$\begin{aligned} z_t &= \sigma_g(W_z a_t + U_z h_{t-1}) \\ r_t &= \sigma_g(W_r a_t + U_r h_{t-1}) \\ \hat{h_t} &= tanh(W_h a_t + U_h \circ h_{t-1}) \\ h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h_t} \end{aligned} \tag{3}$$

We use an action sequence $a_{n1:T}$ to denote a series of user relation vectors or item embedding vectors and apply a recurrent model to encode the temporal relation into the feature vectors. We further adopt a single layer GRU network as our encoder and employ the attention-based dense net for detecting adversarial examples (formulated below).

$$\begin{aligned} \alpha_t &= Softmax(W_e e + b_e)_t \\ att, hid &= \sum_{t=1}^{T} \alpha_t h_t \\ p &= Softmax(W_{att} att + b_{att}) \end{aligned} \tag{4}$$

where e is the combined vector of action embedding and hidden states hid—we compute attention weights from embedding vectors and employ a liner unit to distribute probabilities to input time steps; h_t is the output of encoder. After processed through the attention layer, the vector is then piped into a linear unit with softmax to predict if the agent is polluted. The loss function is the cross entropy between the true label and corresponding probability,

$$J(Att(a_{1:T}), y) = -y \circ log(p)$$

3 EXPERIMENTS

In this section, we report our experiments to evaluate attack methods and our detection model.

3.1 Dataset and Experiment Setup

We conduct experiments following [1] and [14] over a real-world dataset, *Amazon dataset* [4]. This public dataset contains user reviews and metadata of the Amazon e-commerce platform from 1996 to 2014. We utilize three subsets, namely Beauty, Cellphones, and Clothing, as our experimental datasets. We directly use the dataset provided by [14] on Github to reproduce their experiments. Details about Amazon dataset analysis can be found in [14].

We conduct our attack and detection experiments based on [14]. We preprocess the dataset by filtering out feature words with higher TF-IDF scores than 0.1. Then, we use 70% data in each dataset as the training set (and the rest as the test set) and actions of reinforcement agent as the detection data. We define the actions of PGPR [14] as heterogeneous graph paths that start from users and have a length of 4. The three Amazon sub-datasets (Beauty, Cellphones, and Clothing) contain 22,363, 27,879, and 39387 users, respectively. To accelerate experiments, we use the first 10,000 users of each dataset to produce adversarial examples. Users in Beauty get, on average, 127.51 paths. The counterparts for Cellphones and Clothing are 121.92 and 122.71. We adopt the action file of l_∞ attack with an epsilon of 0.5 as the training set. As the number of paths is large, we utilize the first 100,000 paths for train and validation with split ratio 80/20. We randomly sampled 100,000 paths from each action file to form the test set.

After trained on training dataset, the subject models are then attacked by the adversarial methods. We slightly modify JSMA and Deepfool for our experiments—we create the saliency map by calculating the product of the target label and temporal label to achieve both effectiveness and higher efficiency (by 0.32 seconds per iteration) of JSMA; we also use sampling to decrease the computation load on a group of gradients for Deepfool. Besides, we set the hidden size of the GRU to 32 for the encoder, the drop rate of the attention-based classifier to 0.5, the maximum length of a user-item path to 4, the learning rate and weight decay of the optimization solver, Adam, to 5e-4 and 0.01, respectively.

3.2 Attack Experiments

This section reports our experiments on adversarial attacks. The first part shows the attack experiment results, followed by an analysis of the impact of attack frequency.

Adversarial attack results. We are interested in how vulnerable the agent is to perturbation in semantic embedding space. We consider an attack to be effective if a small perturbation leads to a notable performance reduction. We experimentally compare the performance of different attack methods (described in Section 2) in Table 1. We reuse the evaluation metrics of the original model, namely Normalized Discounted Cumulative Gain (NDCG), Recall, Hit Ratio (HR), and Precision for evaluation. All metrics are computed based on the top 10 sorted predictions for each user. Besides, all the metrics are presented in percentage without specific notion.

Table 1 shows the attack results share the same trend with the distribution discrepancy. Most attack methods significantly reduce the performance of the reinforcement system. FGSM l_1 achieves the best performance. It reveals that attacks on a single dimension can change the neural network's action drastically. Compared with l_1 and l_{inf} methods, FGSM l_2 is less effective, where the metrics just fluctuates around the original baseline (shown in Table 1). This is partly because the l_2 attack creates a small disturbance on original data. Specifically, JSMA chooses a small attack area but achieves comparable results as FGSM l_{inf}. Deepfool achieves the second least effective performance. Attacks on Clothing and Cellphones sub-datasets show similar effects.

Impact of attack frequency. We conduct two experiments on attack frequency, random attack, and strategically attack. In the random attack method, the adversarial examples are crafted with a frequency parameter, p_{freq}. In the strategically-timed attack, the adversarial examples are generated by the method shown in Section 2.2. The NDCG metric is presented in Figure 2; other metrics have a similar trend. It can be seen from 2 that the random attack performs worse than the strategically-timed attack. Generating strategically adversarial examples one third to half time steps achieves a significant reduction in all metrics.

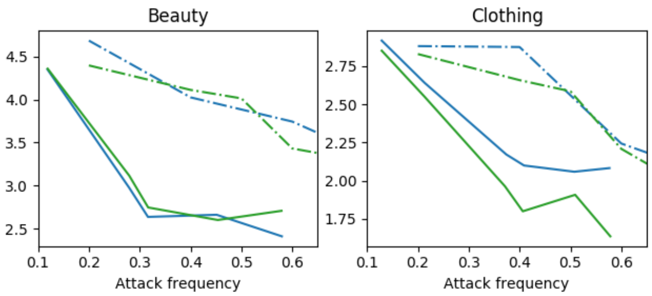

Figure 2: NDCG of attack frequency on Beauty and Clothing subsets. Dashdot lines represent random attacks, solid lines are strategically-timed attacks. Blue and green lines are FGSM l_{inf} and l_1 attacks respectively.

Analysis of adversarial examples. We use Maximum Mean Discrepancy as statistical measures to capture distribution distance. This divergence is defined as:

$$MMD(k, X_{org}, X_{adv}) = \sup_{k \in K} \left(\frac{1}{n} \sum_{i=1}^{n} k(x_{org,i}) - \frac{1}{m} \sum_{i=1}^{n} k(x_{adv,i}) \right)$$

where k is the kernel function, i.e., a radial basis function, X_{org}, X_{adv} are benign and adversarial examples.

MMD-org reveals the discrepancy between the original and adversarial datasets. While MMD-l_1 presents the discrepancy among different attack methods. The results (Table 1) show that the adversarial distribution is different from the original distribution. Also, the disturbed distributions are closed to each other regardless of the attack type. This insight makes it clear that we can use a classifier to separate benign data and adversarial data and it can detect several attacks at the same time, which might be transferred to other reinforcement learning attack detection tasks.

3.3 Detection Experiments

From a statistical perspective, the above analysis shows that one classifier can detect multiple types of attacks. We evaluate the

Table 1: Adversarial attack results, MMD between benign distribution and adversarial distribution on Amazon Beauty

Data	Parameters	NDCG	Recall	HR	Precision	MMD-org	MMD-l_1
Original	-	4.654	6.572	13.993	1.675	0.121	0.620
FGSM l_1	ϵ =0.1	2.695	3.714	6.599	0.693	0.604	0.010
FGSM l_2	ϵ =1.0	4.567	6.555	13.751	1.653	0.016	0.573
FGSM l_{inf}	ϵ =0.5	2.830	3.909	7.351	0.787	0.570	0.011
JSMA	-	2.984	3.844	8.254	0.931	0.412	0.034
Deepfool	-	3.280	4.352	9.548	1.050	0.177	0.458

detection performance of different models using Precision, Recall and F1 score.

We adopt an attention-based network for detection experiments. The detection model is trained on FGSM l_1 attack with ϵ at 0.1 for all datasets. The results (Table 2) show that our detection model achieves better performance on strong attacks. The detection precision and recall rise as the attack becomes stronger. l_∞ attack validates this trend, which shows that our model can detect weak attacks as well. The result of detection on l_2 attack can be reasoned with MMD analysis shown above, high precision and low recall show that most l_2 adversarial examples are close to benign data which confuses the detector. The l_1 attack with $\epsilon = 1.0$ validates that our detector performs well yet achieves worse performance on other tests of Cellphones dataset.

Table 2: Detection Result & Factor Analysis

Dataset	Attack	Precision	Recall	F1 Score
Beauty	l_1 0.1	0.919	0.890	0.904
	l_2 1.0	0.605	0.119	0.199
	l_{inf} 0.5	0.918	0.871	0.894
	JSMA	0.910	0.793	0.848
	Deepfool	0.915	0.840	0.876
Cellphones	l_1 0.1	0.801	0.781	0.791
	l_2 1.0	0.754	0.593	0.664
	l_{inf} 0.5	0.795	0.752	0.773
	l_1 1.0	0.810	0.825	0.817
Clothing	l_1 0.1	0.911	0.866	0.888
	l_2 1.0	0.541	0.099	0.168
	l_{inf} 0.5	0.912	0.879	0.895
Dataset	Frequency	Precision	Recall	F1 Score
Beauty	l_1 0.02	0.823	0.362	0.503
	l_1 0.08	0.918	0.872	0.894
	l_1 0.3	0.922	0.927	0.924
Dataset	Frequency	Precision	Recall	F1 Score
Beauty	l_1 0.579	0.921	0.912	0.917
	l_1 0.316	0.918	0.879	0.898
	l_1 0.118	0.837	0.401	0.543

Our results on factor analysis (Table 2) show that the detection model can detect attacks even under low attack frequencies. But the detection accuracy decreases as the attack frequency drops—the recall reduces significantly to 40.1% when 11.8% examples represent attacks.

4 CONCLUSION

Adversarial attacks on reinforcement learning-based recommendation system can degrade user experience. In this paper, we systematically study adversarial attacks and their factor impacts. We conduct statistical analysis to show classifiers, especially an attention-based detector, can well separate the detection data. Our extensive experiments show both our attack and detection models achieve satisfactory performance.

REFERENCES

[1] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI, 3312–3320.

[2] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 50–56.

[3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. (2014). arXiv:cs, stat/1412.6572 http://arxiv.org/abs/1412.6572

[4] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 507–517.

[5] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. (2017). http://arxiv.org/abs/1702.02284

[6] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. (2017). http://arxiv.org/abs/1703.06748

[7] Tariq Mahmood and Francesco Ricci. 2007. Learning and adaptivity in interactive recommender systems. In *Proceedings of the ninth international conference on Electronic commerce*. ACM, 75–84.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[9] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 372–387.

[10] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. 2017. Robust Deep Reinforcement Learning with Adversarial Attacks. (2017). http://arxiv.org/abs/1712.03632

[11] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing Properties of Neural Networks. (2013). arXiv:cs/1312.6199 http://arxiv.org/abs/1312.6199

[12] Nima Taghipour and Ahmad Kardan. 2008. A hybrid web recommender system based on q-learning. In *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 1164–1168.

[13] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. 2004. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research* 21 (2004), 393–428.

[14] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. *arXiv preprint arXiv:1906.05237* (2019).