Data Poisoning Attacks in Contextual Bandits

Yuzhe Ma[1](✉), Kwang-Sung Jun[1], Lihong Li[2], and Xiaojin Zhu[1]

[1] University of Wisconsin-Madison, Madison, USA
ma234@wisc.edu, kjun@discovery.wisc.edu, jerryzhu@cs.wisc.edu
[2] Google Brain, Kirkland, WA, USA
lihong@google.com

Abstract. We study offline data poisoning attacks in contextual bandits, a class of reinforcement learning problems with important applications in online recommendation and adaptive medical treatment, among others. We provide a general attack framework based on convex optimization and show that by slightly manipulating rewards in the data, an attacker can force the bandit algorithm to pull a target arm for a target contextual vector. The target arm and target contextual vector are both chosen by the attacker. That is, the attacker can hijack the behavior of a contextual bandit. We also investigate the feasibility and the side effects of such attacks, and identify future directions for defense. Experiments on both synthetic and real-world data demonstrate the efficiency of the attack algorithm.

Keywords: Data poisoning · Contextual bandit · Adversarial attack

1 Introduction

As an important step toward trustworthy AI, adversarial learning studies robustness of machine learning systems against malicious attacks [7,10]. Training set poisoning is a type of attack where the adversary can manipulate the training data such that a machine learning algorithm trained on the poisoned data would produce a defective model. The defective model is often similar to a good model, but affords the adversary certain nefarious leverages [3,5,9,12,14,15,17]. Understanding training set poisoning is essential to developing defense mechanisms.

Recent studies on training set poisoning attack focused heavily on supervised learning. There has been little study on poisoning sequential decision making algorithms, even though they are widely employed in the real world. In this paper, we aim to fill in the gap by studying training set poisoning against contextual bandits. Contextual bandits are extensions of multi-armed bandits with side information and have seen wide applications in industry including news recommendation [13], online advertising [6], medical treatment allocation [11], and also promotion of users' well-being [8].

L. Bushnell et al. (Eds.): GameSec 2018, LNCS 11199, pp. 186–204, 2018.
https://doi.org/10.1007/978-3-030-01554-1_11

Let us take news recommendation as a running example for poisoning against contextual bandits. A news website has K articles (i.e., arms). It runs an adaptive article recommendation algorithm (the contextual bandit algorithm) to learn a policy in the backend. Every time a user (represented by a context vector) visits the website, the website displays an article that it thinks is most likely to interest the user based on the historical record of all users. Then the website receives a unit reward if the user clicks through the displayed article, and receives no reward otherwise. Usually the website keeps serving users throughout the day and updates its article selection policy periodically (say, during the nights or every few hours). This provides an opportunity for an attacker to perform *offline* data poisoning attacks, e.g. the attacker can sneak into the website backend at night before the policy is updated, and poison the rewards collected during the daytime. The website unknowingly updates its policy with the poisoned data. On the next day it behaves as the attacker wanted.

More generally, we study adversarial attacks in contextual bandit where the attacker poisons historical rewards in order to force the bandit to pull a target arm under a target context. One can view this attack as a form of offline reward shaping [16], but it is adversarial reward shaping. Our main contribution is an optimization-based attack framework for this attack setting. We also study the feasibility and side effect of the attack. We show on both synthetic and real-world data that the attack is effective. This exposes a security threat in AI systems that involve contextual bandits.

2 Review of Contextual Bandit

This section reviews contextual bandits, which will be the victim of the attack in this paper. A contextual bandit is an abstraction of many real-world decision making problems such as product recommendation and online advertising. Consider for example a news website which strives to recommend the most interesting news articles personalized for individual users. Every time a user visits the website, the website observes certain contextual information that describes the user such as age, gender, location, past news consumption patterns, etc. The website also has a pool of candidate news articles, one of which will be recommended and shown to the user. If the recommended article is interesting, the user may click on it; otherwise, the user may click on other items on the page or navigate to another page. The click probability here depends on both the user (via the context) and the recommended article. Such a dependency can be learned based on click logs and used for better recommendation for future users.

An important aspect of the problem is that the click feedback is observed only for the recommended article, not for others. In other words, the decision (choosing which article to show to a user) is irrevocable; it is impractical to force the user to revisit the webpage so as to recommend a different article. As a result, the feedback data being collected is necessarily biased towards the current recommendation algorithm being employed by the website, raising the need for balancing *exploration* and *exploitation* when choosing arms [13].

This is in stark contrast to a typical prediction task solved by supervised learning where predictions do not affect the data collection.

Formally, a contextual bandit has a set $\mathcal{X}$ of contexts and a set $\mathcal{A} = \{1, 2, \ldots, K\}$ of K arms. A contextual bandit algorithm proceeds in rounds $t = 1, 2, \ldots$. At round t, the algorithm observes a context vector $x_t \in \mathcal{R}^d$, chooses to pull an arm $a_t \in \mathcal{A}$, and observes a reward $r_t \in \mathcal{R}$. The goal of the algorithm is to maximize the total reward garnered over rounds. In the news recommendation example above, it is natural to define $r_t = 1$ if user clicks on the article and 0 otherwise, so that maximizing clicks is equivalent to maximizing the click-through rate, a critical business metric in online recommender systems.

In this work, we focus on the most popular and well-studied setting called linear bandits, where the expected reward is linear map of the context vector. Specifically, we assume each arm a is associated with an unknown vector $\theta_a \in \mathcal{R}^d$ with $\|\theta_a\|_2 \leq S$, so that for every t:

$$r_t = x_t^\top \theta_{a_t} + \eta_t \,, \tag{1}$$

where η_t is a σ-subGaussian noise. For simplicity, we assume η_t is unbounded and thus the reward can take any value in $\mathcal{R}$.

Most contextual bandit algorithms adopt the optimism-in-face-of-uncertainty (OFU) principle for efficient exploration. The OFU principle constructs an Upper Confidence Bound (UCB) for the mean reward of each arm based on historical data and then selects the arm with the highest UCB at each time step [1,4]. In round t, the historical data consists of the context, action, reward triples (x, a, r) from the previous $t - 1$ rounds. It is useful to split the historical data so that the feedback from the same arm is pooled together. Define $[K] = \{1, \ldots, K\}$. Let m_a be the number of times arm a was pulled up to time $t - 1$. This implies that $\sum_{a \in [K]} m_a = t - 1$. For each $a \in [K]$, let $X_a \in \mathcal{R}^{m_a \times d}$ be the design matrix for rounds, where arm a was pulled and each row of X_a is a previous context. Similarly, let $y_a \in \mathcal{R}^{m_a}$ be the corresponding reward (column) vector.

A UCB-style algorithm first forms a point estimate of θ_a by ridge regression

$$\hat{\theta}_a = (X_a^\top X_a + \lambda I)^{-1} X_a^\top y_a, \qquad \forall a \in [K], \tag{2}$$

where $\lambda > 0$ is a regularization parameter. At round t, the algorithm observes the context x_t and then selects the arm with the highest UCB:

$$a_t = \mathrm{argmax}_{a \in [K]} \left\{ x_t^\top \hat{\theta}_a + \alpha_a \|x_t\|_{V_a^{-1}} \right\}, \tag{3}$$

where $\|x_t\|_{V_a^{-1}} = \sqrt{x_t^\top V_a^{-1} x_t}$ is the Mahalanobis norm and $V_a = X_a^\top X_a + \lambda I$. Intuitively, for less frequently chosen a, the second term above tends to be large, thus encouraging exploration. The exploration parameter α_a is algorithm-specific. For example, in LinUCB [13] $\alpha_a = 1 + \sqrt{\frac{1}{2} \log \frac{2}{\delta}}$ and in OFUL [1] $\alpha_a = \sigma \sqrt{2 \log(\frac{\det(V_a)^{\frac{1}{2}} \det(\lambda I)^{-\frac{1}{2}}}{\delta})} + \lambda^{\frac{1}{2}} S$, where $\delta > 0$ is a confidence parameter. Here, we assume α_a may depend on input parameters like δ and observed data up to $t - 1$, but not x_t.

In Algorithm 1, we summarize the contextual bandit algorithm. While the bandit algorithm updates its $\hat{\theta}$ estimates in every round (step 3), in practice due to various considerations such updates often happen in mini-batches, e.g., several times an hour, or during the nights when fewer users visit the website [2,13]. Between these consecutive updates, the bandit algorithm follows a fixed policy obtained from the last update.

Algorithm 1. Contextual bandit algorithm

```
1: **Parameters**: confidence $\delta$, regularizer $\lambda$, UCB function $\alpha$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3:     Receive context $x_t$, estimate $\hat{\theta}_a, a \in [K]$ with (2).
4:     Pull arm $a_t = \text{argmax}_{a \in [K]} \left\{ x_t^\top \hat{\theta}_a + \alpha_a \|x_t\|_{V_a^{-1}} \right\}$.
5:     World generates reward $r_t = x_t^\top \theta_{a_t} + \eta_t$.
6:     Append $x_t$ and $r_t$ to $X_{a_t}$ and $y_{a_t}$, respectively.
7: **end for**
```

3 Attack Algorithm in Contextual Bandit

We now introduce an attacker with the following attack goal:

> **Attack Goal [$\mathbf{x}^* \rightarrow \mathbf{a}^*$]:** On a particular attack target context x^*, force the bandit algorithm to pull an attack target arm a^*.

For example, the attacker may want to manipulate the news service so that a particular article a^* is shown to users x^* from certain political bases. The attack is aimed at the current round t, or more generally the whole period when the arm-selection policy is fixed. Any suboptimal arm a^* can be the target arm. For concreteness, in our experiments the attacker always picks the worst arm a^* as the target arm. This is defined in the sense of the worst UCB, namely replacing argmax with argmin in (3), resulting in the target arm in (21).

We assume the attacker has full knowledge of the bandit algorithm and has access to all historical data. The attacker has the power to poison the historical reward vector[1] y_a, $\forall a \in [K]$. Specifically, the attacker can make arbitrary modifications $\Delta_a \in \mathcal{R}^{m_a}$, $\forall a \in [K]$ so that the reward vector for arm a becomes $y_a + \Delta_a$. After the poisoning attack, the ridge regression performed by the bandit algorithm yields a different solution:

$$\hat{\theta}_a = V_a^{-1} X_a^\top (y_a + \Delta_a). \tag{4}$$

Because such attacks happen on historical rewards in between bandit algorithm updates, we call it offline.

Now we can formally define the attack goal.

[1] In this paper we restrict the poisoning to modifying rewards for ease of exposition. More generally, the attacker can add, remove, or modify both the rewards and the context vectors. Our optimization-based attack framework can be generalized to such stronger attacks, though the optimization could become combinatorial.

Definition 1 (Weak attack). *A target context x^* is called* <u>*weakly attacked*</u> *into pulling target arm a^* if after attack the following inequalities are satisfied:*

$$x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} > x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \quad \forall a \neq a^*. \tag{5}$$

In other words, the algorithm is manipulated into choosing a^* for context x^*.

To avoid being detected, the attacker hopes to make the poisoning $\Delta_a, a \in [K]$ as small as possible. We measure the magnitude of the attack by the squared ℓ_2-norm $\sum_{a\in[K]}\|\Delta_a\|_2^2$.[2] We therefore formulate the attack as the following optimization problem:

$$\begin{aligned} \min_{\Delta_a : a\in[K]} \quad & \sum_{a\in[K]} \|\Delta_a\|_2^2 \\ \text{s.t.} \quad & x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} > x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \forall a \neq a^* \\ \text{where} \quad & \hat{\theta}_a = V_a^{-1}X_a^\top(y_a + \Delta_a), \quad \forall a. \end{aligned} \tag{6}$$

The weak attack above ensures that, given the target context x^*, the bandit algorithm is forced to pull arm a^* instead of any other arms. Unfortunately, the constraints do not result in a closed convex set. To formulate the attack as a convex optimization problem, we introduce a stronger notion of attack that implies weak attack:

Definition 2 (Strong attack). *A target context x^* is called* <u>*ϵ-strongly attacked*</u> *into pulling target arm a^*, for some $\epsilon > 0$, if after attack the following holds:*

$$x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} \geq \epsilon + x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \quad \forall a \neq a^*\,. \tag{7}$$

This is essentially a large margin condition which requires the UCB of a^* to be at least ϵ greater than the UCB of any other arm a. The margin parameter ϵ is chosen by the attacker. We achieve strong attack with the following optimization problem:

$$\begin{aligned} \min_{\Delta_a : a\in[K]} \quad & \sum_{a\in[K]} \|\Delta_a\|_2^2 \\ \text{s.t.} \quad & x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} \geq \epsilon + x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \quad \forall a \neq a^* \\ \text{where} \quad & \hat{\theta}_a = V_a^{-1}X_a^\top(y_a + \Delta_a), \forall a. \end{aligned} \tag{8}$$

The optimization problem above is a quadratic program with linear constraints in $\{\Delta_a\}_{a\in[K]}$. We summarize the attack in Algorithm 2. In the next section we discuss when the algorithm is feasible.

[2] The choice of norm is application dependent, see e.g., [15, Fig. 3]. Any norm works for the attack formulation.

Algorithm 2. Data Poisoning Attack in Contextual Bandit

```
**Input**: victim contextual bandit (Algorithm 1), target context $x^*$, target arm $a^*$,
  attack margin $\epsilon$, historical data $X_a, y_a, a \in [K]$.
Solve (8) for $\Delta_a, \forall a \in [K]$.
If a solution $\Delta_a$ is found, poison $y_a \leftarrow y_a + \Delta_a$; otherwise return infeasible.
```

4 Feasibility of Attack

While one can always write down the training set attack algorithm as optimization (8), there is no guarantee that such attack is feasible. In particular, the inequality constraints may result in an empty set. One may naturally ask: are there context vectors x^* that simply cannot be strongly attacked?[3] In this section we present a full characterization of the feasibility question for strong attack. As we will see, attack feasibility depends on the original training data. Understanding the answer helps us to gauge the difficulty of poisoning, and may aid the design of defenses.

The main result of this section is the following theorem that characterizes a sufficient and necessary condition for the strong attack to be feasible.

Theorem 1. *A context x cannot be strongly attacked into pulling a^* if and only if there exists $a \neq a^*$ such that the following two conditions are both satisfied:*

(i) $x \in \mathrm{Null}(X_{a^*}) \cap \mathrm{Null}(X_a)$, *and*
(ii) $\alpha_{a^*}||x||_{V_{a^*}^{-1}} < \epsilon + \alpha_a||x||_{V_a^{-1}}$.

Before presenting the proof, we first provide intuition. The key idea is that a context x cannot be strongly attacked if some non-target arm a is always better than a^* for x for any attack. This can happen because there are two terms in the arm selection criterion (3) while the attack can affect the first term only. It turns out that under the condition (i) the first term becomes zero. If there exists a non-target arm that has a larger second term than that of the target arm (the condition (ii)), then no attack can force the bandit algorithm to choose the target arm.

We present an empirical study on the feasibility of attack in Sect. 6.3.

Lemma 1. $x \in \mathrm{Null}(X_{a^*}) \Leftrightarrow x^\top V_{a^*}^{-1} X_{a^*}^\top = 0$*, where* $V_{a^*}^{-1} = X_{a^*}^\top X_{a^*} + \lambda I$.

Proof. First, we prove $x \in \mathrm{Null}(X_{a^*}) \Rightarrow x^\top V_{a^*}^{-1} X_{a^*}^\top = 0$. Note that

$$\begin{aligned} x \in \mathrm{Null}(X_{a^*}) &\Rightarrow X_{a^*}x = 0 \\ &\Rightarrow X_{a^*}^\top X_{a^*}x = 0 \\ &\Rightarrow (X_{a^*}^\top X_{a^*} + \lambda I)x = \lambda x \\ &\Rightarrow \frac{1}{\lambda}x = (X_{a^*}^\top X_{a^*} + \lambda I)^{-1}x = V_{a^*}^{-1}x. \end{aligned} \tag{9}$$

[3] Even if some context x^* cannot be strongly attacked, the attacker might be able to weakly attack it. Weak attack is sufficient for the attacker to force an arm pull of a^*. However, as $\epsilon \to 0$ strong attack approaches weak attack. Thus we only need to characterize strong attacks.

Therefore, we have

$$x^\top V_{a^*}^{-1} X_{a^*}^\top = \frac{1}{\lambda} x^\top X_{a^*}^\top = \frac{1}{\lambda}(X_{a^*}x)^\top = 0. \tag{10}$$

Now we show the other direction. Note that

$$\begin{aligned} x^\top V_{a^*}^{-1} X_{a^*}^\top = 0 &\Rightarrow x^\top V_{a^*}^{-1} X_{a^*}^\top X_{a^*} = 0 \\ &\Rightarrow x^\top V_{a^*}^{-1}(V_{a^*} - \lambda I) = 0 \\ &\Rightarrow x^\top = \lambda x^\top V_{a^*}^{-1} \\ &\Rightarrow (X_{a^*}^\top X_{a^*} + \lambda I)x = \lambda x \\ &\Rightarrow X_{a^*}^\top X_{a^*} x = 0 \\ &\Rightarrow x^\top X_{a^*}^\top X_{a^*} x = 0 \\ &\Rightarrow \|X_{a^*} x\|_2^2 = 0 \\ &\Rightarrow X_{a^*} x = 0\,, \end{aligned} \tag{11}$$

which implies $x \in \text{Null}(X_{a^*})$. □

Proof (Theorem 1). ($\Leftarrow$) According to Lemma 1, condition (i) implies

$$x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \varDelta_{a^*}) = x^\top V_a^{-1} X_a^\top (y_a + \varDelta_a) = 0. \tag{12}$$

Combined with (ii) we have for any $\varDelta_{a^*}$ and $\varDelta_a$,

$$\begin{aligned} x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \varDelta_{a^*}) + \alpha_{a^*} ||x||_{V_{a^*}^{-1}} \quad &= \quad \alpha_{a^*} ||x||_{V_{a^*}^{-1}} \\ < \quad \epsilon + \alpha_a ||x||_{V_a^{-1}} \quad &= \quad \epsilon + \alpha_a ||x||_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \varDelta_a)\,. \end{aligned} \tag{13}$$

Thus, x cannot be attacked.

($\Rightarrow$) This is equivalent to prove if $\forall a \neq a^*, \neg(i) \vee \neg(ii)$, then x can be attacked. To show x can be attacked, it suffices to find a solution for the optimization problem.

If $\neg(i)$, then $X_{a^*}x \neq 0$ or $X_a x \neq 0$. Assume $X_{a^*}x \neq 0$ (similar for the case $X_a x \neq 0$), then $x^\top V_{a^*}^{-1} X_{a^*}^\top \neq 0$. Let $p = X_{a^*} V_{a^*}^{-1} x$. For any $a \neq a^*$, arbitrarily fix some $\varDelta_a$, then define

$$q_a = \epsilon + \alpha_a ||x||_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \varDelta_a) - x^\top V_{a^*}^{-1} X_{a^*}^\top y_{a^*} - \alpha_{a^*} ||x||_{V_{a^*}^{-1}}. \tag{14}$$

Let $\varDelta_{a^*} = kp$, where $k = \max_{a \neq a^*} \frac{q_a}{\|p\|_2^2}$. Thus,

$$x^\top V_{a^*}^{-1} X_{a^*}^\top \varDelta_{a^*} = p^\top \varDelta_{a^*} = k\|p\|_2^2 \geq \frac{q_a}{\|p\|_2^2}\|p\|_2^2 = q_a, \quad \forall a \neq a^*. \tag{15}$$

Therefore, we have for all $a \neq a^*$ that

$$x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \varDelta_{a^*}) + \alpha_{a^*} ||x||_{V_{a^*}^{-1}} \geq \epsilon + \alpha_a ||x||_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \varDelta_a)\,, \tag{16}$$

which means x^* can be attacked.

If $\neg(ii)$, simply letting $\varDelta_{a^*} = -y_{a^*}$ and $\varDelta_a = -y_a$ suffices, concluding the proof. □

5 Side Effects of Attack

While the previous section characterized contexts x^* that cannot be strongly attacked, this section asks an opposite question: suppose the attacker was able to strongly attack some x^* by solving (8), what other contexts x are affected by the attack? For example, there might exist some context $x \neq x^*$ whose pre-attack chosen arm is $a(x) = 1$, but becomes $a'(x) = 2$. The side effects can be construed in two ways: on one hand the attack automatically influence more contexts than just x^*; on the other hand they make it harder for the attacker to conceal an attack. The latter may be utilized to facilitate detection by a defender. In this section, we study the side effect of attack and provide insights into future research directions on defense.

The side effect is quantified by the fraction of contexts in the context space such that the chosen arm is changed by the attacker. Specifically, let $\mathcal{X}$ be the context space and P be a probability measure over $\mathcal{X}$. Let $a(x)$ and $a'(x)$ be the pre-attack and post-attack chosen arm of a context x. Then the *side effect fraction* is defined as:

$$s = \int_{x \in \mathcal{X}} \mathbb{1}\left[a(x) \neq a'(x)\right] P(x) dx. \tag{17}$$

One can compute an *empirical side effect fraction* $\hat{s}$ as follows. First sample m contexts from P, and then let $\hat{s} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[a(x) \neq a'(x)\right]$. It is easy to show using Chernoff bound that $|s - \hat{s}|$ decays to 0 at the rate of $1/\sqrt{m}$.

We now give some properties of the side effect. Specifically, we first show if x is affected by the attack, cx is also affected by the attack for any $c > 0$.

Proposition 1. *If a context x satisfies $a(x) \neq a'(x)$, then $a(cx) \neq a'(cx)$ for any $c > 0$, where $a(x)$ and $a'(x)$ are the pre-attack and post-attack chosen arm of x. Moreover, $a'(cx) = a'(x)$, i.e., the post-attack chosen arms for cx and x are exactly the same.*

Proof. First, for any $a \neq a'(x)$, define

$$f_a(x) = x^\top \hat{\theta}_{a'(x)} + \alpha_{a'(x)} \|x\|_{V_{a'(x)}^{-1}} - x^\top \hat{\theta}_a - \alpha_a \|x\|_{V_a^{-1}}. \tag{18}$$

Note that $a'(x)$ is the best arm after attack, thus $f_a(x) > 0$, $\forall a \neq a'(x)$. Therefore, for any $c > 0$, we have

$$f_a(cx) = c f_a(x) > 0, \quad \forall a \neq a'(x), \tag{19}$$

which implies that $a'(cx) = a'(x)$. The same argument may be used to show $a(cx) = a(x)$. Therefore, $a'(cx) = a'(x) \neq a(x) = a(cx)$.

Proposition 1 shows that if a context x has a side effect, all contexts on the open ray $\{cx : c > 0\}$ also have the same side effect.

Proposition 2. *If a context x is strongly attacked, then cx is also strongly attacked for any $c \geq 1$.*

Proof. First, for any $a \neq a^*$, define

$$f_a(x) = x^\top \hat{\theta}_{a^*} + \alpha_{a^*} \|x\|_{V_{a^*}^{-1}} - x^\top \hat{\theta}_a - \alpha_a \|x\|_{V_a^{-1}} . \tag{20}$$

Since x is strongly attacked, we have $f_a(x) \geq \epsilon$, $\forall a \neq a^*$. Therefore $f_a(cx) = cf_a(x) \geq f_a(x) \geq \epsilon$, which shows that cx is also strongly attacked.

The above propositions are weak in that they do not directly quantify the side effect fraction s. They only tell us that when there is side effect, the affected contexts form a collection of rays. In the experiment section we empirically study the side effect fraction. Further theoretical understanding of the side effect is left as a future work.

6 Experiments

Our proposed attack algorithm works for any contextual bandit algorithm taking the form (3). Throughout the experiments, we choose to attack the OFUL algorithm that has a tight regret bound and can be efficiently implemented.

6.1 Attack Effectiveness and Effort: Toy Experiment

To study the effectiveness of the attack, we consider the following toy experiment. The bandit has $K = 5$ arms, and each arm has a payoff parameter $\theta_a \in \mathcal{R}^d$ where $d = 10$, distributed uniformly on the d-dimensional sphere, denoted $\mathcal{S}^d$. To generate θ_a, we first draw from a d-dimensional standard Gaussian distribution, $\tilde{\theta}_a \sim \mathcal{N}(\mathbf{0}, I_d)$ and then normalize: $\theta_a = \tilde{\theta_a} / \|\tilde{\theta}_a\|_2$.

Next, we construct the historical data as follows. We generate $n = 10^3$ historical context vectors $\{x_1, \ldots, x_n\}$ again uniformly on $\mathcal{S}^d$. For each historical context x, we pretend the world generates all K rewards $\{r_a : a \in \mathcal{A}\}$ from the K arms according to (1), where we set the noise level to $\sigma = 0.1$. We then choose an arm a randomly from a multinomial distribution: $a \sim \text{multi}(p_1, p_2, ..., p_K)$, where $p_{i'} = \frac{\exp(r_{i'})}{\sum_{i' \in \mathcal{A}} \exp(r_{i'})}$. This forms one data point (x, a, r_a), and we repeat it for all n points. We then group the historical data to form the appropriate matrices X_a, y_a for every $a \in \mathcal{A}$. Note that the historical data generated in this way is off-policy with respect to the bandit algorithm. The regularization and confidence parameters are $\lambda = 1$ and $\delta = 0.05$, respectively.

In each attack trial, we draw a single target context $x^* \in \mathcal{R}^d$ uniformly from $\mathcal{S}^d$. Without attack, the bandit would have chosen the arm with the highest UCB based on historical data (3). To illustrate the attack, we will do the opposite and set the attack target arm a^* as the one with the smallest UCB instead:

$$a^* = \text{argmin}_{a \in [K]} \left\{ {x^*}^\top \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}} \right\} , \tag{21}$$

where α_a is the UCB parameter of the OFUL algorithm [1]. We set the strong attack margin as $\epsilon = 0.001$. We then run the attack on x^* with Algorithm 2.

We run 100 attack trials. In each trial the arm parameters, historical data, and the target context x^* are regenerated. We make two main observations:

1. The attacker is effective. All ϵ-strongly attacks are successful.
2. The attacker's poisoning Δ is small. The total poisoning can be measured by $\|\Delta\|_2 = \sqrt{\sum_{a \in [K]} \|\Delta_a\|_2^2}$ in each attack trial. However, this quantity depends on the scale of the original pre-attack rewards y_a. It is more convenient to look at the poisoning effort ratio:

$$\frac{\|\Delta\|_2}{\|y\|_2} = \sqrt{\frac{\sum_{a \in [K]} \|\Delta_a\|_2^2}{\sum_{a \in [K]} \|y_a\|_2^2}}. \tag{22}$$

   Figure 1 shows the histogram for the poisoning effort ratio of the 100 attack trials. The ratio tends to be small, with a median of 0.26, which demonstrates that the attacker needs to only manipulate about 26% of the rewards.

These two observations indicate that poisoning attack in contextual bandit is easy to carry out.

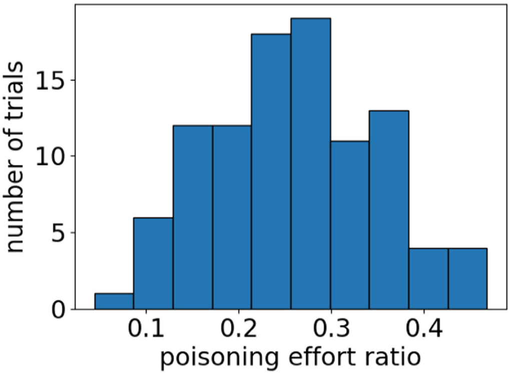


Fig. 1. Histogram of poisoning effort ratio in the toy experiment

We now analyze a single, representative attack trial to gain deeper insight into the attack strategy. In this trial, the UCBs of the 5 arms without attack are

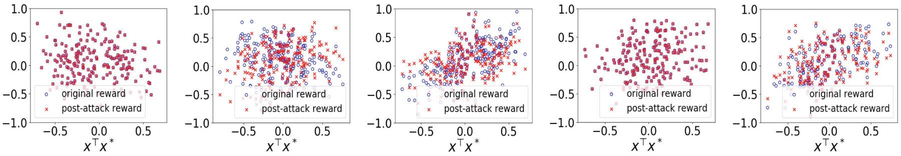
$$\text{pre-attack: } (0.204, 0.097, 0.959, 0.507, 0.818).$$

That is, arm 3 would have been chosen. As mentioned earlier, $a^* = 2$ is chosen to be the target arm as it has the smallest pre-attack UCB. After attack, the UCBs of all arms become:

$$\text{post-attack: } (0.204, 0.605, 0.604, 0.507, 0.604).$$

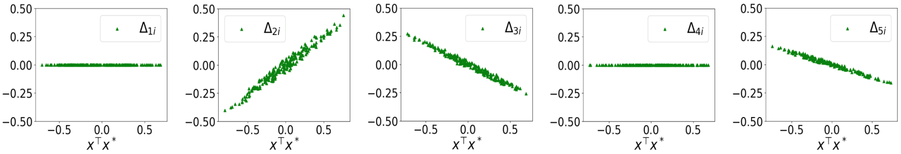
The attacker successfully forced the bandit to choose arm 2. It did so by poisoning the historical data to make arm 2 look better and arms 3 and 5 look worse. It left arms 1 and 4 unchanged.

Figure 2 shows the attack where each panel is the historical rewards where that arm was chosen. We show the original rewards (y_{ai}, blue circle) and post-attack rewards ($y_{ai} + \varDelta_{ai}$, red cross) for all historical points i where arm a was chosen. Intuitively, to decrease the UCB of arm a the attacker should reduce the reward if the historical context x is "similar" to x^*, and boost the reward otherwise. To see this, we sort the historical points by the inner product $x^\top x^*$ in ascending order. As shown in Fig. 2(c) and (e), the attacker gave the illusion that these arms are not good for x^* by reducing the rewards when $x^\top x^*$ is large. The attacker also increased the rewards when $x^\top x^*$ is very negative, which reinforces the illusion. In contrast, the attacker did the opposite on the target arm as shown in Fig. 2(b).



(a) arm 1 (b) arm 2 (c) arm 3 (d) arm 4 (e) arm 5

Fig. 2. Original reward y_{ai} and post-attack reward $y_{ai} + \varDelta_{ai}$ for each arm. (Color figure online)



(a) arm 1 (b) arm 2 (c) arm 3 (d) arm 4 (e) arm 5

Fig. 3. The reward poisoning $\varDelta_{ai}$ for each arm.

6.2 Attack on Real Data: Yahoo! News Recommendation

To further demonstrate the effectiveness of the attack algorithm in real applications, we now test it on the Yahoo! Front Page Today Module User Click Log Dataset (R6A).[4] The dataset contains a fraction of user click log for news articles displayed in the Featured Tab of the Today Module on Yahoo! Front Page (http://www.yahoo.com) during the first ten days in May 2009. Specifically, it contains about 46 million user visits, where each user is represented as a 6-dimensional contextual vector. When a user arrives, the Yahoo! Webscope program selects an article (an arm) from a candidate article pool and displays it

[4] URL: https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.

to the user. The system receives reward 1 if the user clicks on the article and 0 otherwise. Contextual information about users can be found in prior work [13].

To apply the attack algorithm, we require that the set of arms remain unchanged. However, the Yahoo! candidate article pool (i.e., the set of arms) varies as new articles are added and old ones are removed over time. Nonetheless, there are long periods of time where the set of arms is fixed. We restrict ourselves to such a stable time period for our experiment (specifically the period from 7:25 to 10:35 on May 1, 2009) in the Yahoo! data, which contains 243,667 user visits. During this period the bandit has $K = 20$ fixed arms. We further split the time period such that the first $n = 8000$ user visits are used as the historical training data to be poisoned, and the remaining $m = 163,667$ data points as the test data. The bandit learning algorithm uses regularization $\lambda = 1$. The confidence parameter is $\delta = 0.05$. The subGaussian parameter is set to $\sigma = \frac{1}{4}$ for binary rewards.

We simulate attacks on three target user context vectors: The most frequent user context vector $x^* = \bar{x}$, a middle user context vector $x^* =\rlap{$-$}x$, and the least frequent user context vector $x^* = \underline{x}$ in the test data. These three user context vectors appeared 5508, 106, and 1 times, respectively, in the test data. Note that there are potentially many distinct real-world users that are mapped to the same user contextual vector, therefore the "user" in our experiment does not necessarily mean a real-world individual that appeared thousands of times.

We again choose as the target arm a^* the worst arm on the target user as defined by (21). To determine the target arm, we first simulate the bandit algorithm on the original (pre-attack) training data, and then pick the arm with the smallest UCB for that user. For the three target users we consider, the target arms are 8, 3, and 8 respectively. The attacker uses attack margin $\epsilon = 0.001$.

Different from the toy example where the reward can be any value in $\mathcal{R}$, the reward in the Yahoo! dataset must be binary, corresponding to a click-or-not outcome of the recommendation. Therefore, the attacker must enforce $y_{ai} + \Delta_{ai} \in \{0, 1\}$. However, this results in a combinatorial problem. To preserve convexity, we instead relax the attacked reward into a box constraint: $y_{ai} + \Delta_{ai} \in [0, 1]$. We add these new constraints to (8) and solve the following optimization:

$$\begin{aligned}
\min_{\Delta \in \mathcal{R}^n} \quad & \sum_{a \in [K]} \|\Delta_a\|_2^2 \\
\text{s.t.} \quad & {x^*}^\top \hat{\theta}_{a^*} + \alpha_{a^*} \|x^*\|_{V_{a^*}^{-1}} \geq \epsilon + {x^*}^\top \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}}, \quad \forall a \neq a^*, \\
& y_{ai} + \Delta_{ai} \in [0, 1], \quad \forall i \in [m_a], \quad \forall a, \\
\text{where} \quad & \hat{\theta}_a = V_a^{-1} X_a^\top (y_a + \Delta_a), \quad \forall a.
\end{aligned} \tag{23}$$

After the real-valued Δ_{ai} is computed, the attacker performs rounding to turn $y_{ai} + \Delta_{ai}$ into 0 or 1. Specifically, the attacker thresholds $y_{ai} + \Delta_{ai}$ with a constant $c \in [0, 1]$, so that if $y_{ai} + \Delta_{ai} > c$, then let the post-attack reward be 1, otherwise let the post-attack reward be 0. Note that the poisoned rewards now correspond to "reward flipping" from 0 to 1 or vice versa by the attacker. In our experiment, we let the attacker try out 10^4 thresholds c equally distributed in

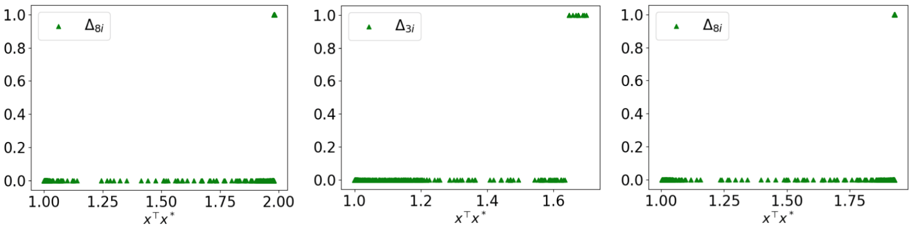
$[0, 1]$. The attacker examines different thresholds for two concerns. First, there is no guarantee that the thresholded solution still triggers the target arm pull, thus the attacker needs to check if the selected arm for x^* is a^*. If not, the corresponding threshold c is inadmissible. Second, among those thresholds that indeed trigger the target arm pull, the attacker selects the one that minimizes the number of flipped rewards, which corresponds to the smallest poisoning effort in the binary reward case.

In Table 1, we summarize the experimental results for attacking the three target users. Note that the attack is successful on all three target users. The best thresholds c for $\bar{x}$, $\sout{x}$ and $\underline{x}$ are 0.0449, 0.1911, and 0.0439, respectively. The number of flipped rewards is small compared to $n = 8000$, which demonstrates that the attacker only needs to spend little cost in order to force the bandit to pull the target arm. Note that the poisoning effect ratio is relatively large. This is because most of the pre-attack rewards are 0, in which case the denominator in (22) is small.

Table 1. Results of experiments on Yahoo! data

	$\bar{x}$	$\sout{x}$	$\underline{x}$
Strong attack successful?	True	True	True
Number [percentage] of flipped rewards	82 [1.0%]	9 [0.1%]	19 [0.2%]
Poisoning effort ratio	0.572	0.189	0.275

In Fig. 4, we show the reward poisoning $\varDelta$ on the historical data against the three target users, respectively. In all three cases, only a few rewards of the target arm are flipped from 0 to 1 by the attacker while those of the other arms remain unchanged. Therefore, we only show the reward poisoning on historical data restricted to the target arm (namely on y_{a^*}). The 82 and 19 flipped rewards overlap in Fig. 4(a) and (c). Note that the contexts of those flipped rewards are highly correlated with x^*.



(a) Most frequent user $x^* = \bar{x}$ (b) Medium frequent user $x^* = \sout{x}$ (c) Least frequent user $x^* = \underline{x}$

Fig. 4. The reward poisoning $\varDelta_{ai}$ on three target users.

6.3 Study on Feasibility

The attack feasibility depends on the historical contexts X, the bandit algorithm-specific UCB parameter α, the attack margin ϵ, the target arm a^*, and the target context x^*. To visualize the infeasible region of strong attack on context, we consider the following toy example.

The bandit has $K = 4$ arms. The attacker's target arm is $a^* = 4$, and the target context x^* lies in $\mathcal{R}^3$. The historical context vectors are

$$X_1 = [1,\ 0,\ 0],\quad X_2 = [0,\ -1,\ 1],\quad X_3 = [0,\ 2,\ 0],\quad X_4 = [2,\ 0,\ 0]. \qquad (24)$$

The problem parameters are $\sigma = S = \lambda = \epsilon = 1$ and $\delta = 0.05$. According to Theorem 1, any infeasible target context x^* satisfies $X_4 x^* = 0$. Thus such x^* must lie in the subspace spanned by the y-axis and z-axis. This allows us to show infeasible regions as 2D plots. In Fig. 5(a), we show the infeasible regions. We distinguish the infeasible region due to each non-target arm by a different color. For example, the infeasible region due to arm 1 consists of all contexts on which the target arm a^* can never be ϵ-better than arm 1 regardless of the attack. Note that the infeasible region due to arm 2 is a line segment of finite length, while that due to arm 3 is the whole $y = 0$ line. The shape of the infeasible region due to each non-target arm varies because the historical data differs and therefore the conditions in Theorem 1 characterizes different shapes. Note that the origin $x = 0$ satisfies the conditions in Theorem 1 and therefore is always infeasible.

One important observation is that, if the bandit algorithm is trained on more historical data, more context vectors x^* can potentially be strongly attacked. Formally, as indicated by Theorem 1 as the null space of historical context matrices $X_a, a \in [K]$ shrinks, the infeasible region shrinks as well. To demonstrate this, in Fig. 5(b) we add a context [0, 0, 0.5] to X_1 such that the historical contexts are:

$$X_1 = \begin{bmatrix} 1, & 0, & 0 \\ 0, & 0, & 0.5 \end{bmatrix},\quad X_2 = [0,\ -1,\ 1],\quad X_3 = [0,\ 2,\ 0],\quad X_4 = [2,\ 0,\ 0]. \qquad (25)$$

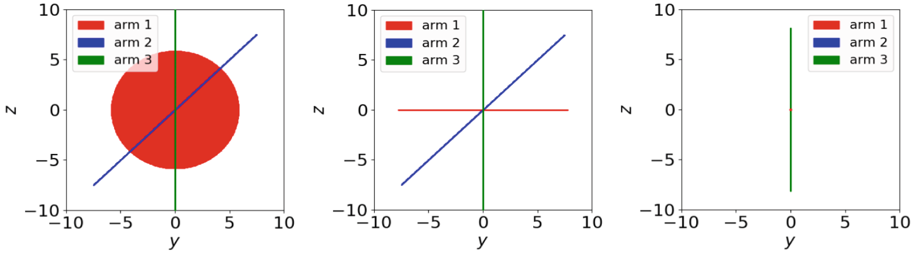
Now that $\text{Null}(X_1)$ is reduced, the infeasibility region due to arm 1 shrinks from the circle in Fig. 5(a) to a horizontal line segment in Fig. 5(b). However the infeasible region may not shrink to a subset of itself, as indicated by the line segment having wider length along y axis than the original circle, thus the shrink happens in the sense of being restricted to a lower-dimensional subspace.

Next we add a historical context $[0, 1, 0]$ to X_4:

$$X_1 = \begin{bmatrix} 1, & 0, & 0 \\ 0, & 0, & 0.5 \end{bmatrix},\quad X_2 = [0,\ -1,\ 1],\quad X_3 = [0,\ 2,\ 0],\quad X_4 = \begin{bmatrix} 2, & 0, & 0 \\ 0, & 1, & 0 \end{bmatrix}.$$

Then the infeasibility region due to arm 1 and arm 2 both shrink to the origin while arm 3 becomes a line segment, as shown in Fig. 5(c).

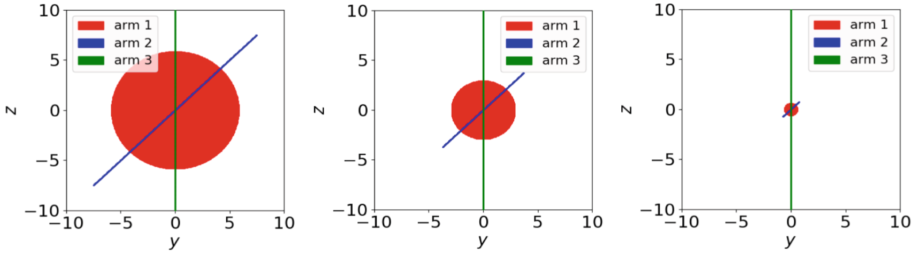
In practice, historical data is often abundant so that $\forall a \neq a^*$, $X_{a^*} \cup X_a$ spans the whole $\mathcal{R}^d$ space, and the only infeasible point is the origin. That is, the attacker can choose to attack essentially any context vector.



(a) original data (b) Context added to X_1 (c) Context added to X_4

Fig. 5. Infeasible region due to each non-target arm.

Another observation is that the infeasible region shrinks as the attack margin ϵ decreases, as shown in Fig. 6. The historical data for each arm is the same as (24). The reason is that a smaller ϵ makes the constraints in (8) easier to satisfy and therefore more contexts are feasible. As $\epsilon \to 0$ the infeasible region converges to those contexts that cannot be weakly attacked, which in this example is the line $y = 0$ in Fig. 6(c). Note that the contexts that cannot be weakly attacked are those that make (6) infeasible. Therefore, we see that without abundant historical data, there will be some contexts that can never be strongly attacked even when $\epsilon \to 0$. Also note that the origin $x^* = 0$ can never be strongly attacked by definition.



(a) $\epsilon = 1$ (b) $\epsilon = 0.5$ (c) $\epsilon = 0.1$

Fig. 6. Infeasible region shrinks as attack margin ϵ decreases.

6.4 Study on Side Effects

We first give an intuitive illustration of the side effect in 2D space. The bandit has $K = 3$ arms, where the arm parameters are θ_a. We generate $n = 1000$ historical data same as before with noise $\sigma = 0.1$. The target context x^* is uniformly sampled from $\mathcal{X}$. The bandit algorithm uses regularization weight $\lambda = 1$ and confidence parameter $\delta = 0.05$. Without attack, the UCB for the three arms are

$$\text{pre-attack: } (-0.419,\ 0.192,\ 1.013). \tag{26}$$

Therefore without attack arm 3 would have been chosen. By our design choice, the target arm is $a^* = 1$. The attacker uses margin $\epsilon = 0.001$. After attack the UCBs of all arms become:

$$\text{post-attack: } (0.290,\ 0.192,\ 0.289). \tag{27}$$

As shown in Fig. 7, the attacker forces the post-attack parameter of the best arm $\hat{\theta}_3$ to deviate from x^* while making $\hat{\theta}_1$ closer to x^*. Note that the attacker could also change the norm of the parameter. Note that arm 2 is not attacked, thus θ_2 and $\hat{\theta}_2$ overlap. The side effect is denoted by the brown arcs on the circle, where the arms chosen for those contexts are changed by the attacker. The side effect fraction for this example is $\hat{s} = 0.315$.

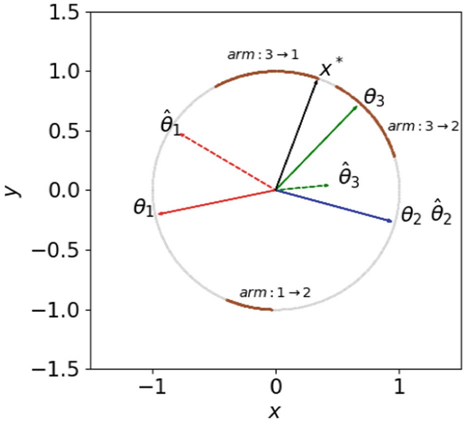
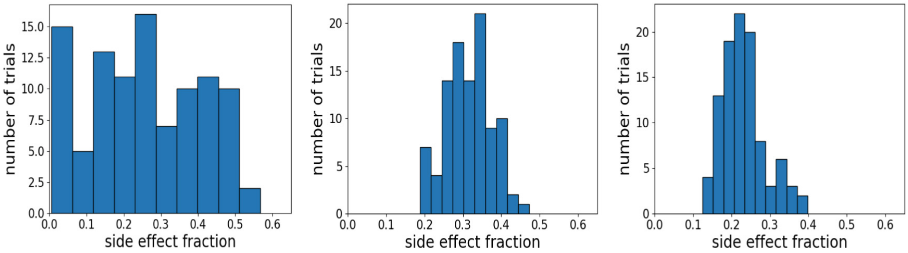


Fig. 7. Side effect shown in 2D context space.

Now we design a toy experiment to study how the side effect depends on the number of arms and the problem dimension. The context space $\mathcal{X}$ is the d-dimensional sphere $\mathcal{S}^d$ and P is uniform on the sphere. The bandit has K arms, where the arm parameters are sampled from P. Same as before, we generate $n = 2000$ historical data with noise $\sigma = 0.1$. The bandit algorithm uses regularization weight $\lambda = 1$. The target context x^* is sampled from P. The attacker's margin is $\epsilon = 0.001$ and the target arm a^* is the worst arm on the target context x^*. We sample $m = 10^3$ contexts from P to evaluate $\hat{s}$.

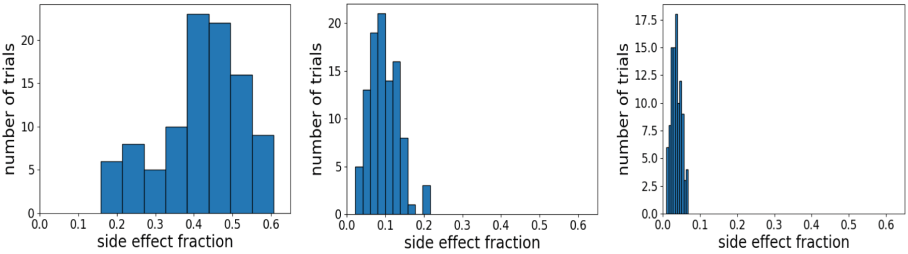
In Fig. 8, we fix $d = 2$ and show a histogram of $\hat{s}$ as the number of arm varies. Note that the attack affects about 30% users. The median $\hat{s}$ for the three panels are 0.249, 0.317, and 0.224 respectively, which shows that the side effect does not grow with the number of arms.

In Fig. 9, we fix $K = 5$ and show the side effect as the dimension d varies. The median $\hat{s}$ for the three panels are 0.435, 0.090, and 0.035, respectively, which implies that in higher dimensional space, the side effect tends to be smaller.



(a) $K = 2$ (b) $K = 20$ (c) $K = 200$

Fig. 8. Side effect fraction as arm number K increases.



(a) $d = 2$ (b) $d = 20$ (c) $d = 200$

Fig. 9. Side effect fraction as dimension d increases.

As the dimension d increases, the attack has less side effect. This exposes the hazard that in real-world applications where the problem dimension is high, the attack will be hard to detect from side effects.

We also study the side effect for the real data experiment. There we use the $m = 163,667$ test users to evaluate the side effect. The side effect fraction for the three users are 0.5391, 0.0750, and 0.5040, respectively. Note that the most frequent user and the least frequent user have a large side effect, which makes the attack easy to detect. In contrast, the side effect of the medium frequent user is extremely small. This implies that the attack can induce different level of side effect for different target users.

7 Conclusions and Future Work

We studied offline data poisoning attack of contextual bandits. We proposed an optimization-based attack framework against contextual bandit algorithms. By manipulating the historical rewards, the attack can successfully force the bandit algorithm to pull a pre-specified arm for some target context. Experiments on both synthetic and real-world data demonstrate the effectiveness of the attack. This exposes a security concern in AI systems that involve contextual bandits.

There are several future directions that can be explored. For example, our current attack only targets a single context x^*. Future work can characterize how to target a set of contexts simultaneously, i.e., force the bandit algorithm to pull the target arm for all contexts in some target set. In the simplest case where the set contains finitely many contexts, one can just replicate the constraint in (8) for each context in the set. The situation is more complicated if the target set is infinite or just too large. Another interesting question is how to develop defense mechanisms to protect the bandit from being attacked. As indicated in this paper, the defender can rely on the side effect to sense the existence of attacks. Conversely, it is also an open question how the attacker might attempt to minimize its side effect during the attack, so that the chances of being detected are minimized. Finally, in this paper we restrict the ability of the attacker to manipulating only the historical rewards. However, there are other types of attacks such as poisoning the historical contexts, adding additional data points, removing existing data points, or combinations of the above. The problem could become non-convex or even combinatorial depending on the type of the attack; some of these settings have been studied under the name "machine teaching" [18,19]. Future work needs to identify how to extend our current attack framework to more general settings.

Acknowledgment. This work is supported in part by NSF 1545481, 1704117, 1623605, 1561512, and the MADLab AF Center of Excellence FA9550-18-1-0166.

References

1. Abbasi-Yadkori, Y., Pál, D., Szepesvári, C.: Improved algorithms for linear stochastic bandits. In: Advances in Neural Information Processing Systems (NIPS), pp. 2312–2320 (2011)
2. Agarwal, A., et al.: Making contextual decisions with low technical debt (2016). coRR abs/1606.03966
3. Alfeld, S., Zhu, X., Barford, P.: Data poisoning attacks against autoregressive models. In: The 30th AAAI Conference on Artificial Intelligence (2016)
4. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2–3), 235–256 (2002)
5. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML), pp. 1467–1474 (2012)
6. Chapelle, O., Manavoglu, E., Rosales, R.: Simple and scalable response prediction for display advertising. ACM Trans. Intell. Syst. Technol. **5**(4), 61:1–61:34 (2014)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
8. Greenewald, K., Tewari, A., Murphy, S.A., Klasnja, P.V.: Action centered contextual bandits. In: Advances in Neural Information Processing Systems 30 (NIPS), pp. 5979–5987 (2017)
9. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: poisoning attacks and countermeasures for regression learning. arXiv preprint arXiv:1804.00308 (2018)
10. Joseph, A.D., Nelson, B., Rubinstein, B.I.P., Tygar, J.: Adversarial Machine Learning. Cambridge University Press, Cambridge (2018)

11. Kuleshov, V., Precup, D.: Algorithms for multi-armed bandit problems (2014). coRR abs/1402.6028
12. Li, B., Wang, Y., Singh, A., Vorobeychik, Y.: Data poisoning attacks on factorization-based collaborative filtering. In: Advances in Neural Information Processing Systems, pp. 1885–1893 (2016)
13. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web (WWW), pp. 661–670 (2010)
14. Mei, S., Zhu, X.: The security of latent Dirichlet allocation. In: The 18th International Conference on Artificial Intelligence and Statistics (AISTATS) (2015)
15. Mei, S., Zhu, X.: Using machine teaching to identify optimal training-set attacks on machine learners. In: The 29th AAAI Conference on Artificial Intelligence (2015)
16. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: theory and application to reward shaping. In: Proceedings of the 16th International Conference on Machine Learning (ICML), pp. 278–287 (1999)
17. Zhao, M., An, B., Yu, Y., Liu, S., Pan, S.J.: Data poisoning attacks on multi-task relationship learning. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, pp. 2628–2635 (2018)
18. Zhu, X.: Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In: The 29th AAAI Conference on Artificial Intelligence (AAAI "Blue Sky" Senior Member Presentation Track) (2015)
19. Zhu, X., Singla, A., Zilles, S., Rafferty, A.N.: An overview of machine teaching. arXiv e-prints, January 2018. https://arxiv.org/abs/1801.05927