Adversarial Attacks in Sequential Decision Making and Control

by

Yuzhe Ma

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2021

Date of final oral examination: 09/01/2021

The dissertation is approved by the following members of the Final Oral Committee:

Earlence Fernandes, Assistant Professor, Computer Sciences
Josiah Hanna, Assistant Professor, Computer Sciences
Kangwook Lee, Assistant Professor, Electrical and Computer Engineering
Yixuan (Sharon) Li, Assistant Professor, Computer Sciences
Xiaojin (Jerry) Zhu, Professor, Computer Sciences

I would like to dedicate this thesis to my beloved parents, Jin Ma and Jinfeng Xu.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Over the past decades, adversarial machine learning (AML) has become a prevalent topic in a wide range of domains. Of particular interest in AML is the question of how to adversarially manipulate machine learning algorithms in order to compromise the model performance. Studying attacks is not only beneficial to understanding the vulnerability of machine learners, but more importantly, provides guidance and insights into designing effective defense strategies. Existing attacks against machine learning can be broadly categorized into two classes depending on when and where the attack happens. In a typical data poisoning attack (a.k.a. training-time attack) setting, the attacker tampers the training data during training time to downgrade the utility of the learned model. On the other hand, in adversarial examples (a.k.a test-time attack), the attacker manipulates features of a target example during test time such that a pre-trained model makes a wrong prediction for that example.

Both data poisoning attacks and adversarial examples have been substantially studied in the traditional supervised learning setting (Biggio and Roli, 2018; Huang et al., 2011; Chakraborty et al., 2018). Due to the *iid* assumption of data points, the attack of supervised learners is a relatively easier task. For example, it is shown in (Mei and Zhu, 2015b) that poisoning a convex learner can be formulated as a bi-level optimization. One can transform the bi-level optimization into a single-level optimization using KKT conditions, and then computationally efficient solutions can be found. Similarly, in adversarial examples, since the test data points are *iid*, the attacker can perform one-shot attack on each individual test point to mislead the model into predicting a target label. A variety of efficient gradient-based attacks such as fast gradient sign method (FGSM) (Goodfellow et al., 2014) are capable of finding adversarial examples imperceptible to humans in the image space.

However, in many real-world scenarios, such as online recommendation (Li et al., 2010a), robotics control (Smart and Kaelbling, 2002; Peters et al., 2003; Gu et al., 2017), and medical treatment allocation (Yom-Tov et al., 2017; Yu et al., 2019), the learner needs to actively interact with an unknown environment to obtain

data in the form of a sequence/trajectory, and learns the optimal action-selection policy in an online manner. In such sequential decision process, the generated data are no longer *iid* due to two reasons. Firstly, future data will depend on the current state of the learning agent, and the sampled data must respect the state transition dynamics. Furthermore, the data generation will depend on how the agent interacts with the environment, i.e., the current action-selection policy of the agent. Given that the agent's policy keeps changing during the learning procedure, the data generation distribution also varies over time. Compared to supervised learning, the attack on sequential learners is a much harder problem because the attacker needs to take into account the temporal nature of the problem and the dependency between data points. For instance, modifying the current state of the agent may have long-term effect on the agent's future states. As a result, the attacker needs to carefully plan the modifications over the entire attack horizon in order to induce a desired effect in a targeted future time. In this thesis, we focus on the following research question: **How to design attack algorithms that can effectively compromise the performance of sequential decision making learners?** Towards answering the above question, this thesis conducts a systematic study on typical sequential decision making learners, including multi-armed bandit, reinforcement learning, industrial control systems, and also multi-agent game-theoretic learners.

To motivate the general idea behind our attack, we now explain how an attacker can be interpreted as a controller. Consider training a reinforcement learning agent. Assume at time t, the agent is at state s_t and takes action a_t. After that the environment generates reward r_t and transits the agent to the next state s_{t+1}. The agent consumes sequential data (s_t, a_t, r_t, s_{t+1}) and updates the policy π_t accordingly. One can view the agent as a dynamic system with the policy π_t being its meta state. The sequential data (s_t, a_t, r_t, s_{t+1}) can be viewed as control signals that guide the policy update, or meta state evolvement, of the agent. Without attack, the meta state π_t evolves towards the optimal policy. However, an attacker can enter the system and tamper the environmental data. As a result, the attacker takes over the environment and controls the meta state evolvement of the learning agent. Therefore, intrinsically the attacker can be viewed as an adversarial controller,

whose meta action is to perturb the sequential data. For example, in reward-manipulation attack, the meta action of the attacker is to perturb r_t to $r_t + \delta_t$. One can consider applying standard control theory to solve the optimal attack (Chen and Zhu, 2019; Ma et al., 2020). However, there are situations where control theory is not directly applicable (unknown or too complex state transition dynamics) or not preferred. In these cases, we can either design ad-hoc attack algorithms targeting specific learners or resort to the more general reinforcement learning framework.

Potential Attack Surfaces in Practice

One real-world example of attacks against sequential decision making systems is the Microsoft chatbot Tay — a conversational AI system with learning ability. After Tay was released on Twitter, some malicious users adversarially manipulated the chatbot into posting inflammatory and offensive tweets. In this example, the attacker (malicious twitter users) composed adversarial input data (e.g., racist remarks) and caused the system to corrupt. When design such attacks against real-world systems, the attacker needs to be aware of potential caveats or vulnerability of the victim system. In the following, we discuss a few possible attack surfaces for sequential decision making systems.

The first type of attack is the reward-manipulation attack. In many sequential decision making systems, the reward signals are generated from human feedbacks. For example, in online recommendation, the reward can be represented by user clicks/ratings, webpage duration time, customer reviews, etc. This reward feedback channel reveals potential security concerns, where adversarial users can provide fake reward feedback or some insider attacker can change existing user feedbacks. In both cases, our learning system receives corrupted input data, and the performance of the learned policy will be compromised. An example is movie rating systems — a group of malicious users can provide fake movie ratings (e.g., very low ratings) to manipulate the system into making a wrong decision (e.g., not recommend) on a target movie.

The other type of attack is the action-manipulation attack, in which the attacker

changes the actions selected by the decision-making system. More concretely, there are two separate cases. In the first case, the attacker can directly overwrite the original system action (as studied in (Liu and Lai, 2020)). However, in practice, such attack can be difficult to implement. One possibility is in online recommender systems, the data is usually saved in log files, then an insider attacker can secretly modify the action information in the log file. The other possibility is that the system already learned a policy $\pi(\cdot)$, a function that maps a state s to an action a. During the deployment phase, assume at time t the agent has state s_t. Then without attack, the policy chooses action $a_t = \pi(s_t)$. However, an attacker can change the state perceived by the agent to s'_t, so that the policy chooses a different action $a'_t = \pi(a'_t)$. Such attack can happen on game-play agents, where the agent state is usually represented as an image. Then an attacker can indirectly modify the action by perturbing the pixels in the input image. This corresponds to the adversarial example attack in the context of sequential decision making.

One can also combine reward manipulation and action manipulation together to form a hybrid attack. The counterpart of such attack in the supervised learning setting is called the backdoor attack (Chen et al., 2017a), which has not been fully studied within the sequential decision making scenario.

In multi-agent sequential decision making systems, there are two potential security threats — external attack and adversarial internal agent.

1. An external attacker directly changes the payoff function to induce a desired target behavior over the agents. In chapter 7, we study the external attack. We point out that while we approach the problem from an attack perspective, the algorithms and the theoretical results developed in chapter 7 also apply to benign entities who hope to help the sequential decision making systems in positive ways. For example, the government may hope to encourage citizens to volunteer for social work. To achieve that, the government gives incentives to volunteers.

2. In the multi-agent scenario, the environment that each agent is faced with will be affected by how other agents play. Therefore, an adversarial internal agent

can perform indirect attacks on the opponent (Gleave et al., 2019) or even the entire multi-agent system (Figura et al., 2021) by using an adversarial policy to shape the environment. The internal attack can be more implicit than the external attack, as it does not directly manipulate the data of the other agents, thus can be harder to detect.

Apart from the attack surface, there are other practical considerations. The most prominent one is how to evade detections. Intuitively, a stealthy attack should not induce remarkable changes to the original data flow. In particular, that could mean two different things. First, the magnitude of perturbation on any individual data point cannot be too large. Otherwise, an outlier detector can easily notice the abnormal data points, and thus the attacker. On the other hand, the attacker may not want to perform too frequent interventions on the system. This is because frequent interventions from outside the system can alert the agents. Both large-magnitude and high-frequency attack will result in heavy attack burden (or attack effort), which is not desirable to the attacker. Depending on the applications, the attacker may desire either small-magnitude attack, low-frequency attack, or both.

Thesis Outline

In Chapter 2, we study adversarial attacks against online stochastic bandits. In particular, we design ad-hoc attack algorithms that mislead bandit players into always selecting some target action (arm) desired by the attacker. As a result, the player suffers linear regret under attack. Furthermore, we show that the attacker only needs to incur sublinear attack cost to achieve this goal, therefore the attack is efficient.

In Chapter 3, we still focus on the bandit setting. However, we instead investigate the offline learning scenario, where the bandit player uses a fixed behavior policy to collect batch dataset, and then updates the policy offline. Such offline learning mode is widely adopted in real-world large-scale systems, as the policy update procedure can be very time-consuming thus frequent updates are not practical. In this situation, we study an attacker who performs one-shot perturbation on the

entire dataset, and cause the bandit to take a sub-optimal action. The problem becomes essentially equivalent to poisoning a supervised learner.

In Chapter 4 and 5, we move on to study attacks against reinforcement learners (RL), which are generalizations of bandit learners. Again, we consider both online and batch reinforcement learners. In Chapter 4, we target online RL algorithms, and demonstrate one can formulate the attack itself as another reinforcement learning problem. This approach takes the control view of the attacker. However, because the state transition of the victim RL agent is too complex, we cannot directly apply control theory. A generalization of control to complex (or unknown) state transition situations is the reinforcement learning framework. Therefore, it is quite natural to apply RL to solve the attack. In Chapter 5, we show that batch reinforcement learning is vulnerable to poisoning attacks. We provide theoretical upper bound on how much perturbation is needed to successfully induce the agent to learn a target sub-optimal policy. Besides that, we also looked into the linear quadratic regulator (LQR), a classic method used in state-feedback control systems. We show that an LQR that performs system identification using batch data is very sensitive to small error in the dataset. Consequently, an attack can injects tiny small error into the training data to corrupt the performance of LQR.

In Chapter 6, we examine a more applied example of control system – the Forward Collision Warning (FCW) system. This system can be found in most cars today, and it helps detect objects in front of the car during road driving, and produces warning lights when there is imminent danger of collision. At the core of this system is a module called Kalman Filter, which takes in measurements of distance and velocity of an object, and outputs a smooth estimate of the object state. We adopt the control view of the attacker, and formulate the attack problem as a Model Predictive Control (MPC). Our study shows that an attacker can sequentially perturb the measurements and cause the Kalman Filter to generate wrong estimates of the object state. As a result, the FCW produces wrong warning lights and distracted human drivers may suffer from car collision.

The previous results are all focused on a single decision making agent. Chapter 7, however, takes a preliminary step towards understanding attacks in multi-agent

decision making. As the simplest example, we consider multi-player matrix games, where several agents play the same matrix game multiple times. Every time, each player chooses an action and the reward/cost is determined by the action profile of all players. Standard results in game theory suggests if all players apply no-regret algorithms, then the empirical policy converges to some coarse correlated equilibrium. We assume the players are all no-regret learning agents, and demonstrate that an attacker can manipulate the reward/cost of each agent to induce a desired target action profile.

Contribution Statement. This chapter is joint work with Kwang-Sung Jun, Lihong Li and Xiaojin Zhu. The author Yuzhe Ma contributed to part of the theoretical analysis, and completed all the experiments. The paper version of this chapter appeared in NeurIPS18.

2.1 Adversarial Attacks on Stochastic Bandits

Designing trustworthy machine learning systems requires understanding how they may be attacked. There has been a surge of interest on adversarial attacks against supervised learning (Goodfellow et al., 2014; Joseph et al., 2018). In contrast, little is known on adversarial attacks against stochastic multi-armed bandits (MABs), a form of online learning with limited feedback. This is potentially hazardous since stochastic MABs are widely used in the industry to recommend news articles (Li et al., 2010b), display advertisements (Chapelle et al., 2014), improve search results (Kveton et al., 2015), allocate medical treatment (Kuleshov and Precup, 2014), and promote users' well-being (Greenewald et al., 2017), among many others. Indeed, as we show, an adversarial attacker can modify the reward signal to manipulate the MAB for nefarious goals.

Our main contribution is an analysis on reward-manipulation attacks. We distinguish three agents in this setting: "the world," "Bob" the bandit algorithm, and "Alice" the attacker. As in standard stochastic bandit problems, the world consists of K arms with sub-Gaussian rewards centered at $\mu_1, \ldots, \mu_K$. Note that we do *not* assume $\{\mu_i\}$ are sorted. Neither Bob nor Alice knows $\{\mu_i\}$. Bob pulls selected arms in rounds and attempts to minimize his regret. When Bob pulls arm $I_t \in [K]$ in round t, the world generates a random reward r_t^0 drawn from a sub-Gaussian distribution with expectation μ_{I_t}. However, Alice sits in-between the world and Bob and manipulates the reward into $r_t = r_t^0 - \alpha_t$. We call $\alpha_t \in \mathbb{R}$ the attack. If Alice decides not to attack in this round, she simply lets $\alpha_t = 0$. Bob then receives r_t,

without knowing the presence of Alice. Without loss of generality, assume arm K is a suboptimal "attack target" arm: $\mu_K < \max_{i=1\ldots K} \mu_i$. Alice's goal is to manipulate Bob into pulling arm K very often while making small attacks. Specifically, we show Alice can force Bob to pull the target arm $T - o(T)$ number of times with a cumulative attack cost of $\sum_{t=1}^{T} |\alpha_t| = O(\log(T))$.

The assumption that Alice does not know $\{\mu_i\}$ is significant because otherwise Alice can perform the attack trivially. To see this, with the knowledge of $\{\mu_i\}$ Alice would be able to compute the truncated reward gap $\Delta_i^\epsilon = \max\{\mu_i - \mu_K + \epsilon, 0\} \geqslant 0$ for all non-target arms $i \neq K$ for some small parameter $\epsilon > 0$. Alice can perform the following *oracle attack*: in any round where a non-target arm $I_t \neq K$ is pulled, attack with $\alpha_t = \Delta_{I_t}^\epsilon$. This oracle attack transforms the original bandit problem into one where all non-target arms have expected reward less than μ_K. It is well-known that if Bob runs a sublinear-regret algorithm (e.g., UCB (Auer et al., 2002a; Bubeck and Cesa-Bianchi, 2012a)), almost all arm pulls will concentrate on the now-best target arm K in the transformed bandit problem. Furthermore, Alice's cumulative attack cost will be sublinear in time, because the total number of non-target arm pulls is sublinear in the transformed problem. In practice, however, it is almost never the case that Alice knows $\mu_1, \ldots, \mu_K$ and hence the Δ_i^ϵ's. Thus the oracle attack is impractical. Our focus in this chapter is to design an attack that nearly matches the oracle attack, but for Alice who does not know $\{\mu_i\}$. We do so for two popular bandit algorithms, ϵ-greedy (Auer et al., 2002b) and UCB (Bubeck and Cesa-Bianchi, 2012a).

What damage can Alice do in practice? She can largely control the arms pulled by Bob. She can also control which arm appears to Bob as the best arm at the end. As an example, consider the news-delivering contextual bandit problem (Li et al., 2010b). The arms are available news articles, and Bob selects which arm to pull (i.e., which article to show to a user at the news site). In normal operation, Bob shows news articles to users to maximize the click-through rate. However, Alice can attack Bob to change his behavior. For instance, Alice can manipulate the rewards so that users from a particular political base are always shown particular news articles that can reinforce or convert their opinion. Conversely, Alice can coerce the bandit

to not show an important article to certain users. As another example, Alice may interfere with clinical trials (Kuleshov and Precup, 2014) to funnel most patients toward certain treatment, or make researchers draw wrong conclusions on whether treatment is better than control. Therefore, adversarial attacks on MAB deserve our attention. Insights gained from our study can be used to build defense in the future.

Finally, we note that our setting is motivated by modern industry-scale applications of contextual bandits, where arm selection, reward signal collection, and policy updates are done in a distributed way (Agarwal et al., 2016; Li et al., 2010b). Attacks can happen when the reward signal is joined with the selected arm, or when the arm-reward data is sent to another module for Bob to update his policy. In either case, Alice has access to both I_t and r_t^0 for the present and previous rounds.

2.2 Preliminaries

Before presenting our main attack algorithms, in this section we first discuss a simple heuristic attack algorithm which serves to illustrate the intrinsic difficulty of attacks. Throughout, we assume Bob runs a bandit algorithm with sublinear pseudo-regret $\mathbb{E}\sum_{t=1}^{T}(\max_{j=1}^{K}\mu_j - \mu_{I_t})$. As Alice does not know $\{\mu_i\}$ she must rely on the empirical rewards up to round $t-1$ to decide the appropriate attack α_t. The attack is online since α_t is computed on-the-fly as I_t and r_t^0 are revealed. The attacking protocol is summarized in Alg. 1.

Algorithm 1 Alice's attack against a bandit algorithm

1: **Input**: Bob's bandit algorithm, target arm K
2: **for** $t = 1, 2, \ldots$ **do**
3: Bob chooses arm I_t to pull.
4: World generates pre-attack reward r_t^0.
5: Alice observes I_t and r_t^0, and then decides the attack α_t.
6: Alice gives $r_t = r_t^0 - \alpha_t$ to Bob.
7: **end for**

We assume all arm rewards are σ^2-sub-Gaussian where σ^2 is known to both Alice and Bob. Let $N_i(t)$ be the number of pulls of arm i up to round t. We say the attack is *successful* after T rounds if the number of target-arm pulls is $N_K(T) = T - o(T)$ in expectation or with high probability, while minimizing the cumulative attack cost $\sum_{t=1}^{T} |\alpha_t|$.

For convenience we define the following quantities:

- $\tau_i(t) := \{s : s \leqslant t, I_s = i\}$, the set of rounds up to t where arm i is chosen,
- $\hat{\mu}_i^0(t) := N_i(t)^{-1} \sum_{s \in \tau_i(t)} r_s^0$, the pre-attack average reward of arm i up to round t, and
- $\hat{\mu}_i(t) := N_i(t)^{-1} \sum_{s \in \tau_i(t)} r_s$, the corresponding post-attack average reward.

The oracle attack, revisited While the oracle attack was impractical, it gives us a baseline for comparison. The oracle attack drags down the reward of all non-target arms,[1] and can be written as

$$\alpha_t = \mathbb{1}\{I_t \neq K\} \cdot \Delta^{\epsilon}_{I_t} .$$

Proposition 2.1 shows that the oracle attack succeeds and requires only a logarithmic attack cost. While more general statements for sublinear-regret algorithms can be made, we focus on logarithmic-regret bandit algorithms for simplicity. Throughout, omitted proofs can be found in our supplementary material.

Proposition 2.1. *Assume that Bob's bandit algorithm achieves an* $O(\log T)$ *regret bound. Then, Alice's oracle attack with* $\epsilon > 0$ *succeeds; i.e.,* $\mathbb{E}N_K(T) = T - o(T)$. *Furthermore, the expected total attack cost is* $O\left(\sum_{i=1}^{K-1} \Delta_i^{\epsilon} \log T\right)$.[2]

The heuristic constant attack A slight variant of the oracle attack is to attack all the non-target arms with a single constant amount $A > 0$, regardless of the

[1] The opposite strategy is to push up the target arm: $\alpha_t = \mathbb{1}\{I_t = K\} \cdot (\mu_K - \max_j \mu_j - \epsilon)$ to make arm K the best arm in post-attack rewards. However, a successful attack means that Alice pulls the target arm $T - o(T)$ times; the attack cost is necessarily linear in T, which is inefficient. Simulations that support "drag down" instead of "push up" are presented in Appendix A.4.

[2] For near-optimal algorithms like UCB (Auer et al., 2002a), one can find the optimal choice of ϵ. See our supplementary material for detail.

actual μ_i's:

$$\alpha_t = \mathbb{1}\{I_t \neq K\} \cdot A.$$

Let $\Delta_i := \Delta_i^0$. Unfortunately, this heuristic constant attack depends critically on the value of A compared to the unknown maximum gap $\max_i \Delta_i$. Proposition 2.2 states the condition under which the attack succeeds:

Proposition 2.2. *Assume that Bob's bandit algorithm achieves an* $O(\log T)$ *regret bound. Then, Alice's heuristic constant attack with* A *succeeds if and only if* $A > \max_i \Delta_i$. *If the attack succeeds, then the expected attack cost is* $O(A \log T)$.

Conversely, if $A < \max_i \Delta_i$ the attack fails. This is because in the transformed bandit problem, there exists an arm that has a higher expected reward than arm K, and Bob will mostly pull that arm. Therefore, the heuristic constant attack has to know an unknown quantity to guarantee a successful attack. Moreover, the attack is non-adaptive to the problem difficulty since some Δ_i's can be much smaller than A, in which case Alice pays an unnecessarily large attack cost.

We therefore ask the following question:

Does there exist an attacker Alice that guarantees a successful attack with cost adaptive to the problem difficulty?

The answer is yes. We present attack strategies against two popular bandit algorithms of Bob: ϵ-greedy and UCB. We show that Alice can indeed succeed in her attacks and incur cost as small as that of the oracle with an additive term due to the sub-Gaussian noise level σ.

2.3 Alice's Attack on ϵ-Greedy Bob

The ϵ-greedy strategy initially pulls each arm once in the first K rounds. For convenience, we assume that the target arm is pulled first: $I_1 = K$. Our results in this section can be adapted to any order of initialization with more complicated notation.

Bob's ϵ-greedy strategy has the following arm-selection rule for $t > K$ (Auer et al., 2002b):

$$I_t = \begin{cases} \text{draw uniform}[K], & \text{w.p. } \epsilon_t \quad \text{(exploration)} \\ \arg\max_i \hat{\mu}_i(t-1), & \text{otherwise (exploitation)} \end{cases}.$$

The strategy uses an exploration scheme $\{\epsilon_t\}$ over t. Alice's attack algorithm is not aware of $\{\epsilon_t\}$ though her cumulative attack cost $\sum |\alpha_t|$ will implicitly depend on it. Later in Corollary 2.2 we show that, for the typical decaying scheme $\epsilon_t \propto 1/t$, the cumulative attack cost is mild: $O(\log(t))$.

Alice wants to make Bob *always* pull the target arm during *exploitation* rounds. Since Alice has no influence on which arm is pulled during exploration, this attack goal is the strongest she can achieve. Here, Alg. 1 is specialized to ensure the following condition:

$$\hat{\mu}_{I_t}(t) \leqslant \hat{\mu}_K(t) - 2\beta(N_K(t)), \tag{1}$$

where we define $\beta(N)$ as

$$\beta(N) := \sqrt{\frac{2\sigma^2}{N} \log \frac{\pi^2 K N^2}{3\delta}}. \tag{2}$$

From this condition, we derive the actual attack α_t. Since

$$\hat{\mu}_{I_t}(t) = \frac{\hat{\mu}_{I_t}(t-1) N_{I_t}(t-1) + r_t^0 - \alpha_t}{N_{I_t}(t)}, \tag{3}$$

we set the attack in Alg. 1 as

$$\alpha_t = \left[\hat{\mu}_{I_t}(t-1) N_{I_t}(t-1) + r_t^0 - (\hat{\mu}_K(t) - 2\beta(N_K(t)))\, N_{I_t}(t)\right]_+, \tag{4}$$

where $[z]_+ = \max(0, z)$. Note α is always non-negative, thus the cumulative attack cost can be written without absolute value: $\sum_{t=1}^{T} \alpha_t$.

With this α_t, we claim that (i) Alice forces Bob to pull the target arm in all exploitation rounds as shown in Lemma 2.4, and (ii) the cumulative attack cost is

logarithmic in t for standard ϵ-greedy learner exploration scheme $\epsilon_t = O(1/t)$ as shown in Corollary 2.2. Our main result is the following general upper bound on the cumulative attack cost.

Theorem 2.1. *Let $\delta \leqslant 1/2$. With probability at least $1-2\delta$, for any T satisfying $\sum_{t=1}^{T} \epsilon_t \geqslant \frac{K}{e-2} \log(K/\delta)$,[3] Alice forces Bob running ϵ-greedy to choose the target arm in at least $\widetilde{N}_K(T)$ rounds, using a cumulative attack cost at most*

$$\sum_{t=1}^{T} |\alpha_t| < \left(\sum_{i=1}^{K} \Delta_i \right) \widetilde{N}(T) + (K-1) \cdot \left(\widetilde{N}(T)\beta(\widetilde{N}(T)) + 3\widetilde{N}(T)\beta(\widetilde{N}_K(T)) \right)$$

where

$$\begin{aligned} \widetilde{N}(T) &= \left(\frac{\sum_{t=1}^{T} \epsilon_t}{K} \right) + \sqrt{3 \log\left(\frac{K}{\delta}\right) \left(\frac{\sum_{t=1}^{T} \epsilon_t}{K} \right)}, \\ \widetilde{N}_K(T) &= T - \left(\sum_{t=1}^{T} \epsilon_t \right) - \sqrt{3 \log\left(\frac{K}{\delta}\right) \left(\sum_{t=1}^{T} \epsilon_t \right)}. \end{aligned}$$

Before proving the theorem, we first look at its consequence. If Bob's ϵ_t decay scheme is $\epsilon_t = \min\{1, cK/t\}$ for some $c > 0$ as recommended in (Auer et al., 2002b), Alice's cumulative attack cost is $O(\sum_{i=1}^{K} \Delta_i \log T)$ for large enough T, as the following corollary shows:

Corollary 2.2. *Inherit the assumptions in Theorem 2.1. Fix K and δ. If $\epsilon_t = cK/t$ for some constant $c > 0$, then*

$$\sum_{t=1}^{T} |\alpha_t| = \widehat{O}\left(\left(\sum_{i=1}^{K} \Delta_i \right) \log T + \sigma K \sqrt{\log T} \right), \tag{5}$$

where $\widehat{O}$ ignores $\log\log$ factors.

[3] One can drop this condition by considering slightly larger $\widetilde{N}(t)$ and smaller $\widetilde{N}_K(t)$. However, we keep the condition as it simplifies $\widetilde{N}(t)$ and $\widetilde{N}_K(t)$. We refer to the proof of Lemma 2.6 for detail.

Note that the two important constants are $\sum_i \Delta_i$ and σ. While a large σ can increase the cost significantly, the term with $\sum_i \Delta_i$ dominates the cost for large enough T. Specifically, $\sum_i \Delta_i$ is multiplied by $\log T$ that is of higher order than $\sqrt{\log T}$. We empirically verify the scaling of cost with T in Section 2.5.

To prove Theorem 2.1, we first show that β in (2) is a high-probability bound on the pre-attack empirical mean of all arms on all rounds. Define the event

$$E := \{\forall i, \forall t > K : |\hat{\mu}_i^0(t) - \mu_i| < \beta(N_i(t))\}. \tag{6}$$

Lemma 2.3. *For $\delta \in (0,1)$, $\mathbb{P}(E) > 1 - \delta$.*

The following lemma proves the first half of our claim.

Lemma 2.4. *For $\delta \leqslant 1/2$ and under event E, attacks (4) force Bob to always pull the target arm K in exploitation rounds.*

We now show that on average each attack on a non-target arm i is not much bigger than Δ_i.

Lemma 2.5. *For $\delta \leqslant 1/2$ and under event E, we have for all arm $i < K$ and all t that*

$$\sum_{s \in \tau_i(t)} |\alpha_s| < (\Delta_i + \beta(N_i(t)) + 3\beta(N_K(t)))\, N_i(t)\,.$$

Finally, we upper bound the number of non-target arm i pulls $N_i(T)$ for $i < K$. Recall the arm i pulls are only the result of exploration rounds. In round t the exploration probability is ϵ_t; if Bob explores, he chooses an arm uniformly at random. We also lower bound the target arm pulls $N_K(T)$.

Lemma 2.6. *Let $\delta < 1/2$. Suppose T satisfy $\sum_{t=1}^{T} \epsilon_t \geqslant \frac{K}{e-2} \log(K/\delta)$. With probability at least $1 - \delta$, for all non-target arms $i < K$,*

$$N_i(T) < \sum_{t=1}^{T} \frac{\epsilon_t}{K} + \sqrt{3 \sum_{s=1}^{T} \frac{\epsilon_t}{K} \log \frac{K}{\delta}}\,.$$

and for the target arm K,

$$N_K(T) > T - \sum_{t=1}^{T} \epsilon_t - \sqrt{3 \sum_{s=1}^{T} \epsilon_t \log \frac{K}{\delta}}.$$

We are now ready to prove Theorem 2.1.

Proof. The theorem follows immediately from a union bound over Lemma 2.5 and Lemma 2.6 below. We add up the attack costs over $K-1$ non-target arms. Then, we note that $N\beta(N)$ is increasing in N so $N_i(T)\beta(N_i(T)) \leqslant \widetilde{N}(T)\beta(\widetilde{N}(T))$. Finally, by Lemma A.2 in our supplementary material $\beta(N)$ is decreasing in N, so $\beta(N_K(T)) \leqslant \beta(\widetilde{N}_K(T))$. ∎

2.4 Alice's Attack on UCB Bob

Recall that we assume rewards are σ^2-sub-Gaussian. Bob's UCB algorithm in its basic form often assumes rewards are bounded in $[0,1]$; we need to modify the algorithm to handle the more general sub-Gaussian rewards. By choosing $\alpha = 4.5$ and $\psi : \lambda \mapsto \frac{\sigma^2\lambda^2}{2}$ in the (α,ψ)-UCB algorithm of (Bubeck and Cesa-Bianchi, 2012a, Section 2.2), we obtain the following arm-selection rule:

$$I_t = \begin{cases} t, & \text{if } t \leqslant K \\ \arg\max_i \left\{ \hat{\mu}_i(t-1) + 3\sigma\sqrt{\frac{\log t}{N_i(t-1)}} \right\}, & \text{otherwise.} \end{cases}$$

For the first K rounds where Bob plays each of the K arms once in an arbitrary order, Alice does not attack: $\alpha_t = 0$ for $t \leqslant K$. After that, attack happens only when $I_t \neq K$. Specifically, consider any round $t > K$ where Bob pulls arm $i \neq K$. It follows from the UCB algorithm that

$$\hat{\mu}_i(t-1) + 3\sigma\sqrt{\frac{\log t}{N_i(t-1)}} \geqslant \hat{\mu}_K(t-1) + 3\sigma\sqrt{\frac{\log t}{N_K(t-1)}}.$$

Alice attacks as follows. She computes an attack α_t with the smallest absolute value, such that

$$\hat{\mu}_i(t) \leqslant \hat{\mu}_K(t-1) - 2\beta(N_K(t-1)) - \Delta_0\,,$$

where $\Delta_0 \geqslant 0$ is a parameter of Alice. Since the post-attack empirical mean can be computed recursively by the following

$$\hat{\mu}_i(t) = \frac{N_i(t-1)\hat{\mu}_i(t-1) + r_t^0 - \alpha_t}{N_i(t-1)+1}\,,$$

where r_t^0 is the pre-attack reward; this enables us to write down in closed form Alice's attack:

$$\alpha_t = \left[N_i(t)\hat{\mu}_i^0(t) - \sum_{s\in\tau_i(t-1)} \alpha_s - N_i(t)\cdot(\hat{\mu}_K(t-1) - 2\beta(N_K(t-1)) - \Delta_0)\right]_+. \quad (7)$$

For convenience, define $\alpha_t = 0$ if $I_t = K$. We now present the main theorem on Alice's cumulative attack cost against Bob who runs UCB.

Theorem 2.3. *Suppose* $T \geqslant 2K$ *and* $\delta \leqslant 1/2$. *Then, with probability at least* $1-\delta$, *Alice forces Bob to choose the target arm in at least*

$$T - (K-1)\left(2 + \frac{9\sigma^2}{\Delta_0^2}\log T\right),$$

rounds, using a cumulative attack cost at most

$$\sum_{t=1}^{T}\alpha_t \leqslant \left(2 + \frac{9\sigma^2}{\Delta_0^2}\log T\right)\sum_{i<K}(\Delta_i + \Delta_0) + \sigma(K-1)\sqrt{32(2+\frac{9\sigma^2}{\Delta_0^2}\log T)\log\frac{\pi^2 K(2+\frac{9\sigma^2}{\Delta_0^2}\log T)^2}{3\delta}}\,.$$

While the bounds in the theorem are somewhat complicated, the next corollary is more interpretable and follows from a straightforward calculation. Specifically, we have the following by straightforward calculation:

Corollary 2.4. *Inherit the assumptions in Theorem 2.3 and fix* δ. *Then, the total number*

of non-target arm pulls is

$$O\left(K + \frac{K\sigma^2}{\Delta_0^2}\log T\right),$$

and the cumulative attack cost is

$$\widehat{O}\left(\left(1 + \frac{\sigma^2}{\Delta_0^2}\log T\right)\sum_{i<K}(\Delta_i + \Delta_0) + \sigma K \cdot \left(1 + \frac{\sigma}{\Delta_0}\sqrt{\log T}\right)\sqrt{\log\left(1 + \frac{K\sigma}{\Delta_0}\right)}\right),$$

where $\widehat{O}$ *ignores* $\log\log(T)$ *factors.*

We observe that a larger Δ_0 decreases non-target arm pulls (i.e. a more effective attack). The effect diminishes when $\Delta_0 > \sigma\sqrt{\log T}$ since $\frac{K\sigma^2}{\Delta_0^2}\log T < K$. Thus there is no need for Alice to choose a larger Δ_0. By choosing $\Delta_0 = \Theta(\sigma)$, the cost is $\widehat{O}(\sum_{i<K}\Delta_i \log T + \sigma K \log T)$. This is slightly worse than the cost of attacking ϵ-greedy where σ is multiplied by $\sqrt{\log T}$ rather than $\log T$. However, we find that a stronger attack is possible when the time horizon T is fixed and known to Alice ahead of time (i.e., the fixed budget setting). One can show that this choice $\Delta_0 = \Theta\left(\sigma\sqrt{\log T}\right)$ minimizes the cumulative attack cost, which is $\widehat{O}\left(K\sigma\sqrt{\log T}\right)$. This is a very strong attack since the dominating term w.r.t. T does not depend on $\sum_{i<K}\Delta_i$; in fact the cost associated with $\sum_{i<K}\Delta_i$ does not grow with T at all. This means that under the fixed budget setting algorithm-specific attacks can be better than the oracle attack that is algorithm-independent. Whether the same is true in the anytime setting (i.e., T is unknown ahead of time) is left as an open problem.

For the proof of Theorem 2.3 we use the following two lemmas.

Lemma 2.7. *Assume event* E *holds and* $\delta \leqslant 1/2$*. Then, for any* $i < K$ *and any* $t \geqslant 2K$*, we have*

$$N_i(t) \leqslant \min\{N_K(t), 2 + \frac{9\sigma^2}{\Delta_0^2}\log t\}. \tag{8}$$

Lemma 2.8. *Assume event* E *holds and* $\delta \leqslant 1/2$*. Then, at any round* $t \geqslant 2K$*, the cumulative attack cost to any fixed arm* $i < K$ *can be bounded as:*

$$\sum_{s\in\tau_i(t)}\alpha_s \leqslant N_i(t)\Big(\Delta_i + \Delta_0 + 4\beta(N_i(t))\Big).$$

Proof of Theorem 2.3. Suppose event E holds. The bounds are direct consequences of Lemmas 2.8 and 2.7 below, by summing the corresponding upper bounds over all non-target arms i. Specifically, the number of target arm pulls is $T - \sum_{i<K} N_i(T)$, and the cumulative attack cost is $\sum_{t=1}^{T} \alpha_t = \sum_{i<K} \sum_{t\in\tau_i(T)} \alpha_t$. Since event E is true with probability at least $1-\delta$ (Lemma 2.3), the bounds also hold with probability at least $1-\delta$. ∎

2.5 Simulations

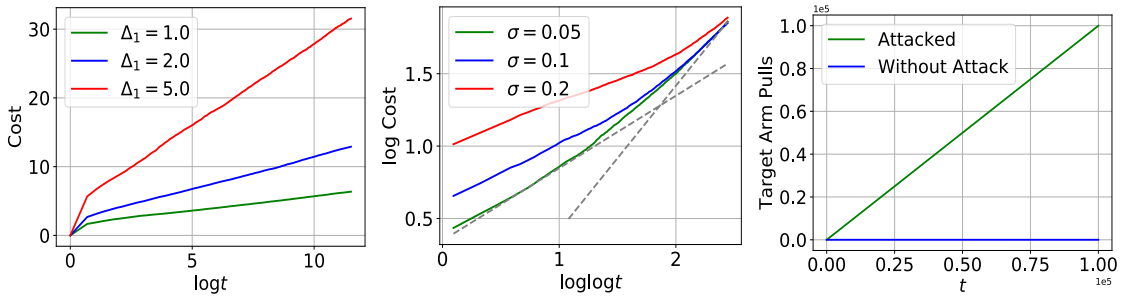
In this section, we run simulations on attacking ϵ-greedy and UCB algorithms to illustrate our theoretical findings.

Attacking ϵ-greedy The bandit has two arms. The reward distributions of arms 1 and 2 are $\mathcal{N}(\Delta_1, \sigma^2)$ and $\mathcal{N}(0, \sigma^2)$, respectively, with $\Delta_1 > 0$. Alice's target arm is arm 2. We let $\delta = 0.025$. Bob's exploration probability decays as $\epsilon_t = \frac{1}{t}$. We run Alice and Bob for $T = 10^5$ rounds; this forms one trial. We repeat 1000 trials.

In Figure 1a, we fix $\sigma = 0.1$ and show Alice's cumulative attack cost $\sum_{s=1}^{t} |\alpha_s|$ for different Δ_1 values. Each curve is the average over 1000 trials. These curves demonstrate that Alice's attack cost is proportional to $\log t$ as predicted by Corollary 2.2. As the reward gap Δ_1 becomes larger, more attack is needed to reduce the reward of arm 1, and the slope increases.

Furthermore, note that $\sum_{t=1}^{T} |\alpha_t| = \widehat{O}\left(\Delta_1 \log T + \sigma\sqrt{\log T}\right)$. Ignoring $\log\log T$ terms, we have $\sum_{t=1}^{T} |\alpha_t| \leqslant C(\Delta_1 \log T + \sigma\sqrt{\log T})$ for some constant $C > 0$ and large enough T. Therefore, $\log\left(\sum_{t=1}^{T} |\alpha_t|\right) \leqslant \max\{\log\log T + \log\Delta_1, \frac{1}{2}\log\log T + \log\sigma\} + \log C$. We thus expect the log-cost curve as a function of $\log\log T$ to behave like the maximum of two lines, one with slope $1/2$ and the other with slope 1. Indeed, we observe such a curve in Figure 1b where we fix $\Delta_1 = 1$ and vary σ. All the slopes eventually approach 1, though larger σ's take a longer time. This implies that the effect of σ diminishes for large enough T, which was predicted by Corollary 2.2.

In Figure 1c, we compare the number of target arm (the suboptimal arm 2) pulls with and without attack. This experiment is with $\Delta_1 = 0.1$ and $\sigma = 0.1$. Alice's



(a) Attack cost $\sum_{s=1}^{t} |\alpha_s|$ as Δ_1 varies

(b) Attack cost as σ varies; dotted lines depict slope $1/2$ and 1 for comparison.

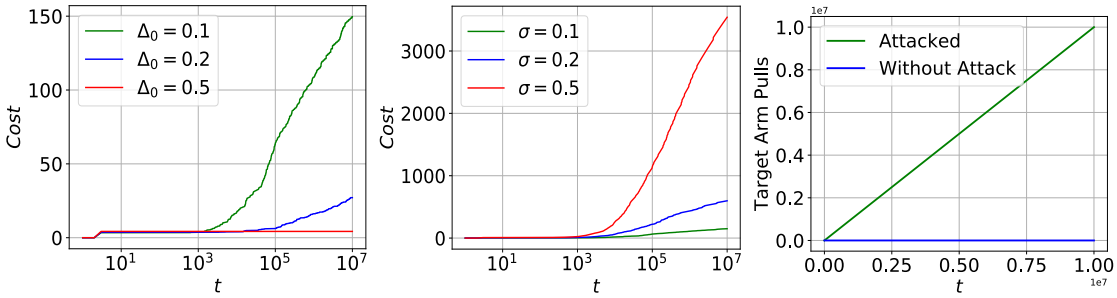
(c) Target arm pulls $N_K(t)$

Figure 1: Attack on ϵ-greedy bandit.

attack dramatically forces Bob to pull the target arm. In 10000 rounds, Bob is forced to pull the target arm 9994 rounds with the attack, compared to only 6 rounds if Alice was not present.

Attacking UCB The bandit has two arms. The reward distributions are the same as the ϵ-greedy experiment. We let $\delta = 0.05$. To study how σ and Δ_0 affects the cumulative attack cost, we perform two groups of experiments. In the first group, we fix $\sigma = 0.1$ and vary Alice's free parameter Δ_0 while in the second group, we fix $\Delta_0 = 0.1$ and vary σ. We perform 100 trials with $T = 10^7$ rounds.

Figure 2a shows Alice's cumulative attack cost as Δ_0 varies. As Δ_0 increases, the cumulative attack cost decreases. In Figure 2b, we show the cost as σ varies. Note that for large enough t, the cost grows almost linearly with $\log t$, which is implied by Corollary 2.4. In both figures, there is a large attack near the beginning, after which the cost grows slowly. This is because the initial attacks drag down the empirical average of non-target arms by a large amount, such that the target arm appears to have the best UCB for many subsequent rounds. Figure 2c again shows that Alice's attack forces Bob to pull the target arm: with attack Bob is forced to pull the target arm $10^7 - 2$ times, compared to only 156 times without attack.



(a) Attack cost $\sum_{s=1}^{t} \alpha_s$ as Δ_0 varies (b) Attack cost as σ varies (c) Target arm pulls $N_K(t)$

Figure 2: Attack on UCB learner.

3 ADVERSARIAL ATTACKS IN CONTEXTUAL BANDITS

Contribution Statement. This chapter is joint work with Kwang-Sung Jun, Lihong Li and Xiaojin Zhu. The author Yuzhe Ma is the leading author and completed most of the work, including the theoretical analysis and the experiments. The paper version of this chapter appeared in GameSec18.

3.1 Adversarial Attacks in Contextual Bandits

As an important step toward trustworthy AI, adversarial learning studies robustness of machine learning systems against malicious attacks (Goodfellow et al., 2014; Joseph et al., 2018). 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 (Alfeld et al., 2016; Biggio et al., 2012; Jagielski et al., 2018; Li et al., 2016a; Mei and Zhu, 2015a,b; Zhao et al., 2018a). 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 chapter, 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 (Li et al., 2010b), online advertising (Chapelle et al., 2014), medical treatment allocation (Kuleshov and Precup, 2014), and also promotion of users' well-being (Greenewald et al., 2017).

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 (Ng et al., 1999b), 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.

3.2 Review of Contextual Bandit

This section reviews contextual bandits, which will be the victim of the attack in this chapter. 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 (Li et al., 2010b). 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 $x_t \in \mathbb{R}^d$, chooses to pull an arm $a_t \in \mathcal{A}$, and observes a reward $r_t \in \mathbb{R}$. 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 $\theta_a \in \mathbb{R}^d$ with $\|\theta_a\|_2 \leqslant S$, so that for every t:

$$r_t = x_t^\top \theta_{a_t} + \eta_t \,, \tag{9}$$

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 $\mathbb{R}$.

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 (Auer et al., 2002a; Abbasi-Yadkori et al., 2011). 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 $X_a \in \mathbb{R}^{m_a \times d}$ be the design matrix for rounds, where arm a was pulled and each row of X_a is a previous context. Similarly, let $y_a \in \mathbb{R}^{m_a}$ be the corresponding reward (column) vector.

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

$$\hat{\theta}_a = (X_a^\top X_a + \lambda I)^{-1} X_a^\top y_a, \quad \forall a \in [K], \tag{10}$$

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:

$$a_t = \arg\max_{a \in [K]} \left\{ x_t^\top \hat{\theta}_a + \alpha_a \|x_t\|_{V_a^{-1}} \right\}, \tag{11}$$

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 (Li et al., 2010b) $\alpha_a = 1 + \sqrt{\frac{1}{2} \log \frac{2}{\delta}}$ and in OFUL (Abbasi-Yadkori et al., 2011) $\alpha_a = \sigma \sqrt{2 \log(\frac{\det(V_a)^{\frac{1}{2}} \det(\lambda I)^{-\frac{1}{2}}}{\delta})} + \lambda^{\frac{1}{2}} S$, 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 2, 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 (Li et al., 2010b;

Agarwal et al., 2016). Between these consecutive updates, the bandit algorithm follows a fixed policy obtained from the last update.

Algorithm 2 Contextual bandit algorithm

1: **Parameters**: confidence δ, regularizer λ, UCB function α.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Receive context x_t, estimate $\hat{\theta}_a, a \in [K]$ with (10).
4: Pull arm $a_t = \arg\max_{a \in [K]} \{x_t^\top \hat{\theta}_a + \alpha_a \|x_t\|_{V_a^{-1}}\}$.
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**

3.3 Attack Algorithm in Contextual Bandit

We now introduce an attacker with the following attack goal:

> **Attack goal** $[\mathbf{x}^* \to \mathbf{a}^*]$: 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 arg max with arg min in (11), resulting in the target arm in (29).

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[4] $y_a, \forall a \in [K]$. Specifically, the attacker can make arbitrary modifications

[4]In this chapter 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.

$\Delta_a \in \mathbb{R}^{m_a}, \forall a \in [K]$ 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:

$$\hat{\theta}_a = V_a^{-1} X_a^\top (y_a + \Delta_a). \tag{12}$$

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

Now we can formally define the attack goal.

Definition 3.1 (Weak attack)**.** *A target context x^* is called* weakly attacked *into pulling target arm a^* if after attack the following inequalities are satisfied:*

$$x^{*\top} \hat{\theta}_{a^*} + \alpha_{a^*} \|x^*\|_{V_{a^*}^{-1}} > x^{*\top} \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}}, \ \ \forall a \neq a^*. \tag{13}$$

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 $\sum_{a \in [K]} \|\Delta_a\|_2^2$.[5] We therefore formulate the attack as the following optimization problem:

$$\begin{aligned} \min_{\Delta_a : a \in [K]} \quad & \sum_{a \in [K]} \|\Delta_a\|_2^2 \\ \text{s.t.} \quad & x^{*\top} \hat{\theta}_{a^*} + \alpha_{a^*} \|x^*\|_{V_{a^*}^{-1}} > x^{*\top} \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}}, \forall a \neq a^* \\ \text{where} \quad & \hat{\theta}_a = V_a^{-1} X_a^\top (y_a + \Delta_a), \ \ \forall a. \end{aligned} \tag{14}$$

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:

[5] The choice of norm is application dependent, see e.g., (Mei and Zhu, 2015b, Figure 3). Any norm works for the attack formulation.

Definition 3.2 (Strong attack). *A target context x^* is called* ϵ-strongly attacked *into pulling target arm a^*, for some $\epsilon > 0$, if after attack the following holds:*

$$x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} \geqslant \epsilon + x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \ \ \forall a \neq a^*. \tag{15}$$

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:

$$\begin{aligned} \min_{\Delta_a : a\in[K]} \quad & \sum_{a\in[K]} \|\Delta_a\|_2^2 \\ \text{s.t.} \quad & x^{*\top}\hat{\theta}_{a^*} + \alpha_{a^*}\|x^*\|_{V_{a^*}^{-1}} \geqslant \epsilon + x^{*\top}\hat{\theta}_a + \alpha_a\|x^*\|_{V_a^{-1}}, \ \ \forall a \neq a^* \\ \text{where} \quad & \hat{\theta}_a = V_a^{-1}X_a^\top(y_a + \Delta_a), \forall a. \end{aligned} \tag{16}$$

The optimization problem above is a quadratic program with linear constraints in $\{\Delta_a\}_{a\in[K]}$. We summarize the attack in Algorithm 3. In the next section we discuss when the algorithm is feasible.

Algorithm 3 Data Poisoning Attack in Contextual Bandit

1: **Input**: victim contextual bandit (Algorithm 2), target context x^*, target arm a^*, attack margin ϵ, historical data $X_a, y_a, a \in [K]$.
2: Solve (16) for $\Delta_a, \forall a \in [K]$.
3: If a solution Δ_a is found, poison $y_a \leftarrow y_a + \Delta_a$; otherwise return `infeasible`.

3.4 Feasibility of Attack

While one can always write down the training set attack algorithm as optimization (16), 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?[6] 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 3.3. *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) $x \in \text{Null}(X_{a^*}) \cap \text{Null}(X_a)$, *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 (11) 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 Section 3.6.

Lemma 3.4. $x \in \text{Null}(X_{a^*}) \Leftrightarrow x^\top V_{a^*}^{-1} X_{a^*}^\top = 0$, *where* $V_{a^*} = X_{a^*}^\top X_{a^*} + \lambda I$.

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

$$\begin{aligned} x \in \text{Null}(X_{a^*}) &\Rightarrow X_{a^*} x = 0 \\ &\Rightarrow X_{a^*}^\top X_{a^*} x = 0 \\ &\Rightarrow (X_{a^*}^\top X_{a^*} + \lambda I) x = \lambda x \\ &\Rightarrow \frac{1}{\lambda} x = (X_{a^*}^\top X_{a^*} + \lambda I)^{-1} x = V_{a^*}^{-1} x. \end{aligned} \tag{17}$$

[6] 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

$$x^\top V_{a^*}^{-1} X_{a^*}^\top = \frac{1}{\lambda} x^\top X_{a^*}^\top = \frac{1}{\lambda}(X_{a^*} x)^\top = 0. \tag{18}$$

Now we show the other direction. Note that

$$\begin{aligned} x^\top V_{a^*}^{-1} X_{a^*}^\top = 0 &\Rightarrow x^\top V_{a^*}^{-1} X_{a^*}^\top X_{a^*} = 0 \\ &\Rightarrow x^\top V_{a^*}^{-1}(V_{a^*} - \lambda I) = 0 \\ &\Rightarrow x^\top = \lambda x^\top V_{a^*}^{-1} \\ &\Rightarrow (X_{a^*}^\top X_{a^*} + \lambda I)x = \lambda x \\ &\Rightarrow X_{a^*}^\top X_{a^*} x = 0 \\ &\Rightarrow x^\top X_{a^*}^\top X_{a^*} x = 0 \\ &\Rightarrow \|X_{a^*} x\|_2^2 = 0 \\ &\Rightarrow X_{a^*} x = 0\,, \end{aligned} \tag{19}$$

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

Theorem 3.3. ($\Leftarrow$) According to lemma 3.4, condition (i) implies

$$x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \Delta_{a^*}) = x^\top V_a^{-1} X_a^\top (y_a + \Delta_a) = 0. \tag{20}$$

Combined with (ii) we have for any Δ_{a^*} and Δ_a,

$$\begin{aligned} x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \Delta_{a^*}) + \alpha_{a^*}\|x\|_{V_{a^*}^{-1}} &= \alpha_{a^*}\|x\|_{V_{a^*}^{-1}} \\ < \epsilon + \alpha_a \|x\|_{V_a^{-1}} &= \epsilon + \alpha_a \|x\|_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \Delta_a)\,. \end{aligned} \tag{21}$$

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 Δ_a, then define

$$q_a = \epsilon + \alpha_a \|x\|_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \Delta_a) - x^\top V_{a^*}^{-1} X_{a^*}^\top y_{a^*} - \alpha_{a^*} \|x\|_{V_{a^*}^{-1}}. \quad (22)$$

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

$$x^\top V_{a^*}^{-1} X_{a^*}^\top \Delta_{a^*} = p^\top \Delta_{a^*} = k\|p\|_2^2 \geqslant \frac{q_a}{\|p\|_2^2} \|p\|_2^2 = q_a, \ \ \forall a \neq a^*. \quad (23)$$

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

$$x^\top V_{a^*}^{-1} X_{a^*}^\top (y_{a^*} + \Delta_{a^*}) + \alpha_{a^*} \|x\|_{V_{a^*}^{-1}} \geqslant \epsilon + \alpha_a \|x\|_{V_a^{-1}} + x^\top V_a^{-1} X_a^\top (y_a + \Delta_a), \quad (24)$$

which means x^* can be attacked.

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

■

3.5 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 (16), 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:

$$s = \int_{x \in \mathcal{X}} \mathbb{1}\left[a(x) \neq a'(x)\right] P(x) dx\,. \tag{25}$$

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 3.5. *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

$$f_a(x) = x^\top \hat{\theta}_{a'(x)} + \alpha_{a'(x)} \|x\|_{V^{-1}_{a'(x)}} - x^\top \hat{\theta}_a - \alpha_a \|x\|_{V_a^{-1}}\,. \tag{26}$$

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

$$f_a(cx) = c f_a(x) > 0, \;\; \forall a \neq a'(x)\,, \tag{27}$$

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 3.5 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 3.6. *If a context x is strongly attacked, then cx is also strongly attacked for any $c \geqslant 1$.*

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

$$f_a(x) = x^\top \hat{\theta}_{a^*} + \alpha_{a^*} \|x\|_{V_{a^*}^{-1}} - x^\top \hat{\theta}_a - \alpha_a \|x\|_{V_a^{-1}}. \tag{28}$$

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

3.6 Experiments

Our proposed attack algorithm works for any contextual bandit algorithm taking the form (11). Throughout the experiments, we choose to attack the OFUL algorithm that has a tight regret bound and can be efficiently implemented.

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 $\theta_a \in \mathbb{R}^d$ 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 (9), 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 $x^* \in \mathbb{R}^d$ uniformly from $\mathcal{S}^d$. Without attack, the bandit would have chosen the arm with the highest UCB based on historical data (11). To illustrate the attack, we will do the opposite and set the attack target arm a^* as the one with the smallest UCB instead:

$$a^* = \arg\min_{a \in [K]} \left\{ x^{*\top} \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}} \right\}, \tag{29}$$

where α_a is the UCB parameter of the OFUL algorithm (Abbasi-Yadkori et al., 2011). We set the strong attack margin as $\epsilon = 0.001$. We then run the attack on x^* with Algorithm 3.

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*:

$$\frac{\|\Delta\|_2}{\|y\|_2} = \sqrt{\frac{\sum_{a \in [K]} \|\Delta_a\|_2^2}{\sum_{a \in [K]} \|y_a\|_2^2}}. \tag{30}$$

   Figure 3 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.

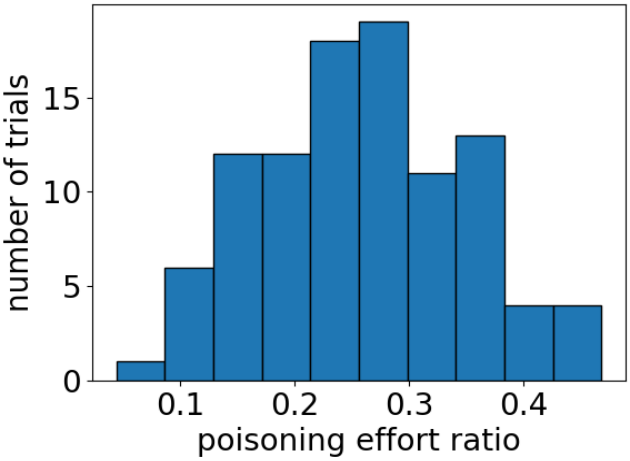


Figure 3: Histogram of poisoning effort ratio in the toy experiment

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

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

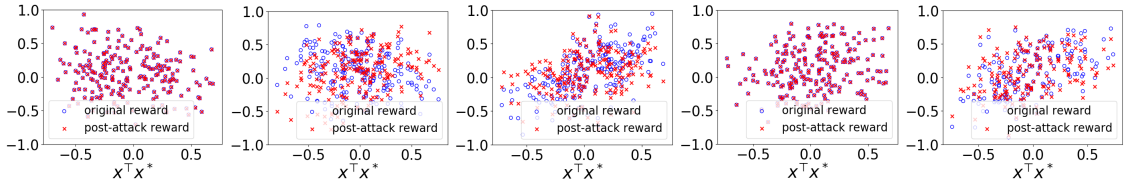
That is, arm 3 would have been chosen. As mentioned earlier, $a^* = 2$ is chosen to be the target arm as it has the smallest pre-attack UCB. After attack, the UCBs of all arms become:

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

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.

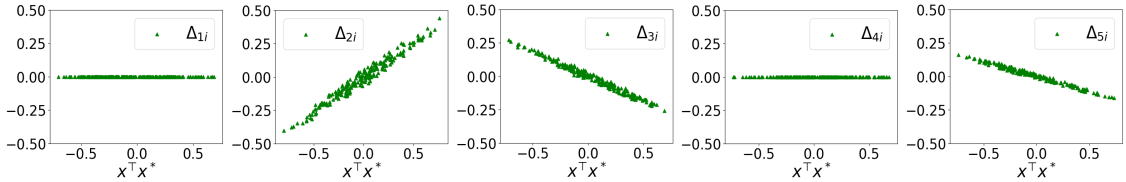
Figure 4 shows the attack where each panel is the historical rewards where that arm was chosen. We show the original rewards (y_{ai}, blue circle) and post-attack rewards ($y_{ai} + \Delta_{ai}$, red cross) for all historical points i where arm a was chosen. Intuitively, to decrease the UCB of arm a 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 Figure 4b and d, 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 Figure 4a.



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

Figure 4: Original reward y_{ai} and post-attack reward $y_{ai} + \Delta_{ai}$ for each arm.



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

Figure 5: The reward poisoning Δ_{ai} for each arm.

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).[7] 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 to the user. The

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

system receives reward 1 if the user clicks on the article and 0 otherwise. Contextual information about users can be found in prior work (Li et al., 2010b).

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 $x^* = \tilde{x}$, 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 (29). 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 $\mathbb{R}$, 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 (16) and solve the following optimization:

$$\begin{aligned} \min_{\Delta \in \mathbb{R}^n} \quad & \sum_{a \in [K]} \|\Delta_a\|_2^2 \\ \text{s.t.} \quad & x^{*\top} \hat{\theta}_{a^*} + \alpha_{a^*} \|x^*\|_{V_{a^*}^{-1}} \geqslant \epsilon + x^{*\top} \hat{\theta}_a + \alpha_a \|x^*\|_{V_a^{-1}}, \ \ \forall a \neq a^*, \\ & y_{ai} + \Delta_{ai} \in [0,1], \ \ \forall i \in [m_a], \ \ \forall a, \\ \text{where} \quad & \hat{\theta}_a = V_a^{-1} X_a^\top (y_a + \Delta_a), \ \ \forall a. \end{aligned} \tag{31}$$

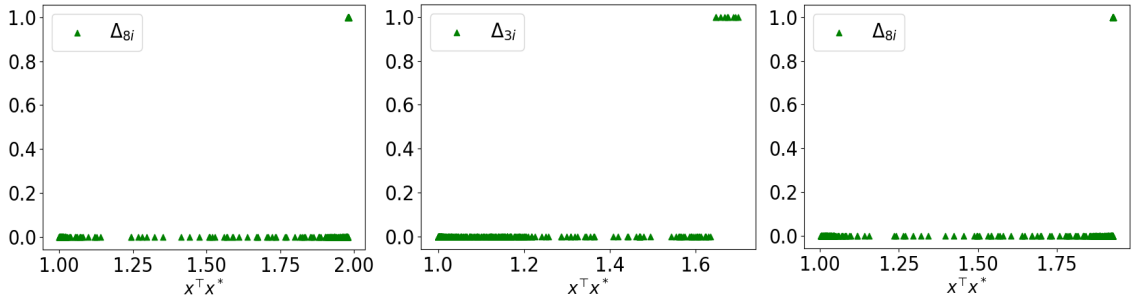
After the real-valued Δ_{ai} is computed, the attacker performs rounding to turn $y_{ai} + \Delta_{ai}$ into 0 or 1. Specifically, the attacker thresholds $y_{ai} + \Delta_{ai}$ with a constant $c \in [0,1]$, so that if $y_{ai} + \Delta_{ai} > c$, then let the post-attack reward be 1, otherwise let the 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 $\bar{x}$, $\varkappa$ 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 (30) is small.

In Figure 6, we show the reward poisoning Δ 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

	$\bar{x}$	$\not{x}$	$\underline{x}$
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

Table 1: Results of experiments on Yahoo! data



(a) Most frequent user $x^* = \bar{x}$ (b) Mid-frequent user $x^* = \not{x}$ (c) Least frequent user $x^* = \underline{x}$

Figure 6: The reward poisoning Δ_{ai} on three target users.

data restricted to the target arm (namely on y_{a^*}). The 82 and 19 flipped rewards overlap in Fig. 6 a and Fig. 6 c. Note that the contexts of those flipped rewards are highly correlated with x^*.

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 $\mathbb{R}^3$. The historical context vectors are

$$X_1 = [1,\ 0,\ 0],\ \ X_2 = [0,\ -1,\ 1],\ \ X_3 = [0,\ 2,\ 0],\ \ X_4 = [2,\ 0,\ 0]. \tag{32}$$

The problem parameters are $\sigma = S = \lambda = \epsilon = 1$ and $\delta = 0.05$. According to

Theorem 3.3, 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 Figure 7a, 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 3.3 characterizes different shapes. Note that the origin $x = 0$ satisfies the conditions in Theorem 3.3 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 3.3 as the null space of historical context matrices $X_a, a \in [K]$ shrinks, the infeasible region shrinks as well. To demonstrate this, in Figure 7b we add a context $[0, 0, 0.5]$ to X_1 such that the historical contexts are:

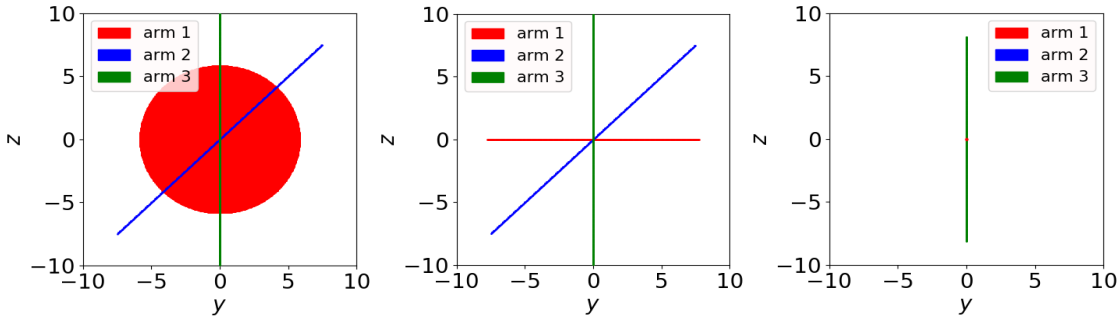
$$X_1 = \begin{bmatrix} 1, & 0, & 0 \\ 0, & 0, & 0.5 \end{bmatrix}, \quad X_2 = [0, \ -1, \ 1], \quad X_3 = [0, \ 2, \ 0], \quad X_4 = [2, \ 0, \ 0]\,. \tag{33}$$

Now that $\text{Null}(X_1)$ is reduced, the infeasibility region due to arm 1 shrinks from the circle in Figure 7a to a horizontal line segment in Figure 7b. 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 Figure 7c.



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

Figure 7: Infeasible region due to each non-target arm.

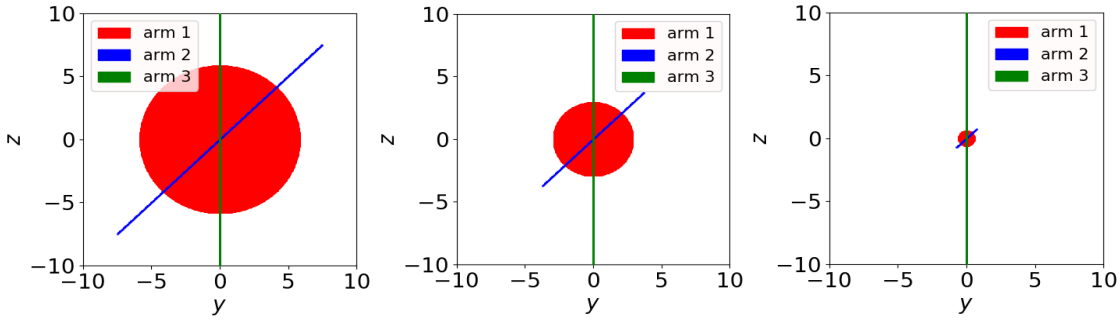
In practice, historical data is often abundant so that $\forall a \neq a^*$, $X_{a^*} \cup X_a$ spans the whole $\mathcal{R}^d$ space, and the only infeasible point is the origin. That is, the attacker can choose to attack essentially any context vector.

Another observation is that the infeasible region shrinks as the attack margin ϵ decreases, as shown in Figure 8. The historical data for each arm is the same as (32). The reason is that a smaller ϵ makes the constraints in (16) 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 Figure 8c. Note that the contexts that cannot be weakly attacked are those that make (14) 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.

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

$$\text{pre-attack: } (-0.419,\ 0.192,\ 1.013). \tag{34}$$



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

Figure 8: Infeasible region shrinks as attack margin ϵ decreases.

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:

$$\text{post-attack: } (0.290,\ 0.192,\ 0.289). \tag{35}$$

As shown in Figure 9, 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 $\hat{s} = 0.315$.

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 Figure 10, 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

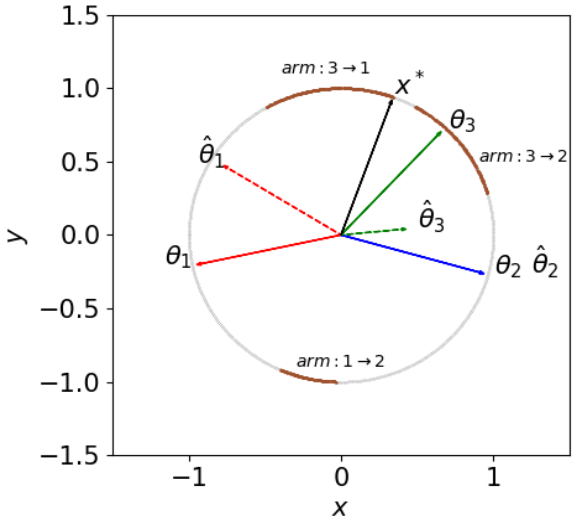
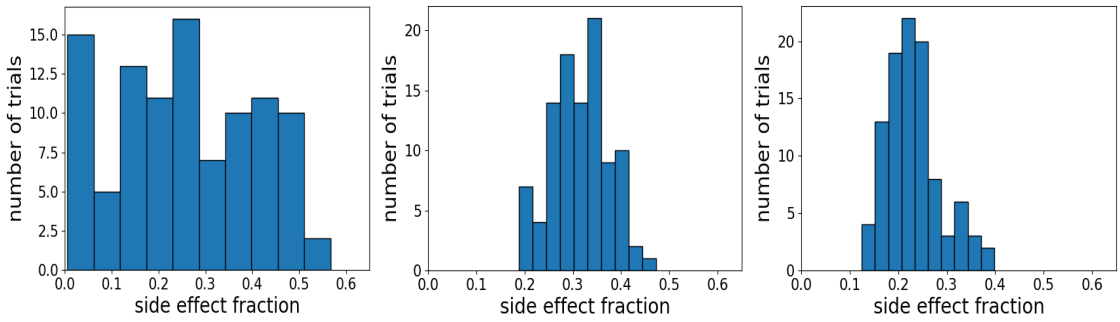


Figure 9: Side effect shown in 2D context space.

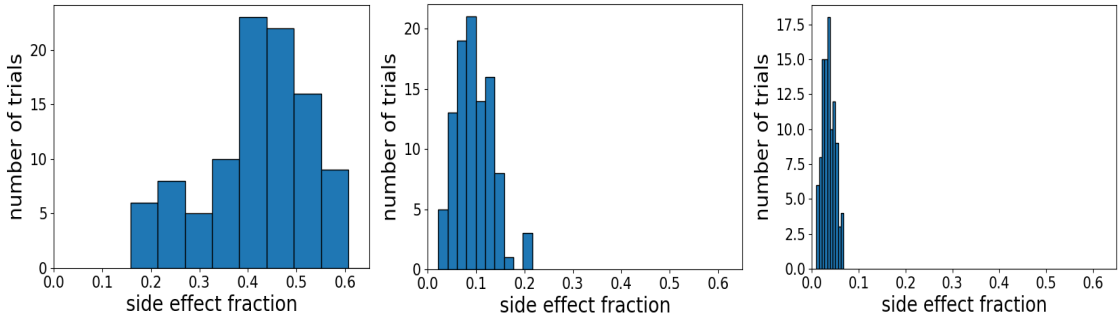
panels are 0.249, 0.317, and 0.224 respectively, which shows that the side effect does not grow with the number of arms.



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

Figure 10: side effect fraction as arm number K increases.

In Figure 11, 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) $d = 2$ (b) $d = 20$ (c) $d = 200$

Figure 11: 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.

3.7 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 (16) 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 chapter, 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 chapter 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" (Zhu, 2015; Zhu et al., 2018). Future work needs to identify how to extend our current attack framework to more general settings.

4 ADAPTIVE REWARD-POISONING ATTACKS AGAINST REINFORCEMENT LEARNING

Contribution Statement. This chapter is joint work with Xuezhou Zhang, Adish Singla and Xiaojin Zhu. The author Yuzhe Ma contributed to part of the theoretical analysis. The paper version of this chapter appeared in ICML20.

4.1 Introduction

In many reinforcement learning (RL) applications the agent extracts reward signals from user feedback. For example, in recommendation systems the rewards are often represented by user clicks, purchases or dwell time (Zhao et al., 2018b; Chen et al., 2019); in conversational AI, the rewards can be user sentiment or conversation length (Dhingra et al., 2016; Li et al., 2016b). In such scenarios, an adversary can manipulate user feedback to influence the RL agent in nefarious ways. Figure 12 describes a hypothetical scenario of how conversational AI can be attacked. One real-world example is that of the chatbot Tay, which was quickly corrupted by a group of Twitter users who deliberately taught it misogynistic and racist remarks shortly after its release (Neff and Nagy, 2016). Such attacks reveal significant security threats in the application of reinforcement learning.

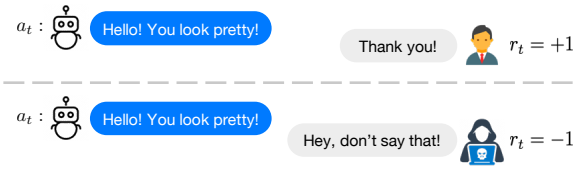


Figure 12: Example: an RL-based conversational AI is learning from real-time conversations with human users. the chatbot says "Hello! You look pretty!" and expects to learn from user feedback (sentiment). A benign user will respond with gratitude, which is decoded as a positive reward signal. An adversarial user, however, may express anger in his reply, which is decoded as a negative reward signal.

In this chapter, we formally study the problem of *training-time attack on RL via reward poisoning*. As in standard RL, the RL agent updates its policy π_t by performing action a_t at state s_t in each round t. The environment Markov Decision Process (MDP) generates reward r_t and transits the agent to s_{t+1}. However, the attacker can change the reward r_t to $r_t + \delta_t$, with the goal of driving the RL agent toward a target policy $\pi_t \to \pi^\dagger$.

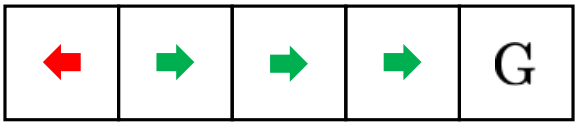


Figure 13: A chain MDP with attacker's target policy $\pi^\dagger$

Figure 13 shows a running example that we use throughout the chapter. The episodic MDP is a linear chain with five states, with left or right actions and no movement if it hits the boundary. Each move has a -0.1 negative reward, and G is the absorbing goal state with reward 1. Without attack, the optimal policy π^* would be to always move right. The attacker's goal, however, is to force the agent to learn the nefarious target policy $\pi^\dagger$ represented by the arrows in Figure 13. Specifically, the attacker wants the agent to move left and hit its head against the wall whenever the agent is at the left-most state.

Our main contributions are:

1. We characterize conditions under which such attacks are guaranteed to fail (thus RL is safe), and vice versa;

2. In the case where an attack is feasible, we provide upper bounds on the attack cost in the process of achieving $\pi^\dagger$;

3. We show that effective attacks can be found empirically using deep RL techniques.

4.2 Related Work

Test-time attacks against RL Prior work on adversarial attacks against reinforcement learning focused primarily on *test-time*, where the RL policy π is pre-trained and fixed, and the attacker manipulates the perceived state s_t to $s_t^\dagger$ in order to induce undesired action (Huang et al., 2017; Lin et al., 2017; Kos and Song, 2017; Behzadan and Munir, 2017). For example, in video games the attacker can make small pixel perturbation to a frame (Goodfellow et al., 2014)) to induce an action $\pi(s_t^\dagger) \neq \pi(s_t)$. Although test-time attacks can severely impact the performance of a deployed and fixed policy π, they do not modify π itself. For ever-learning agents, however, the attack surface includes π. This motivates us to study training-time attack on RL policy.

Reward Poisoning: Reward poisoning has been studied in bandits (Jun et al., 2018; Peltola et al., 2019; Altschuler et al., 2019; Liu and Shroff, 2019; Ma et al., 2018), where the authors show that adversarially perturbed reward can mislead standard bandit algorithms to pull a suboptimal arm or suffer large regret.

Reward poisoning has also been studied in *batch RL* (Zhang and Parkes, 2008; Zhang et al., 2009; Ma et al., 2019) where rewards are stored in a pre-collected batch data set by some behavior policy, and the attacker modifies the batch data. Because all data are available to the attacker at once, the batch attack problem is relatively easier. This chapter instead focuses on the *online* RL attack setting where reward poisoning must be done on the fly.

(Huang and Zhu, 2019) studies a restricted version of reward poisoning, in which the perturbation only depend on the current state and action: $\delta_t = \phi(s_t, a_t)$. While such restriction guarantees the convergence of Q-learning under the perturbed reward and makes the analysis easier, we show both theoretically and empirically that such restriction severely harms attack efficiency. Our results subsumes their results by considering more powerful attacks that can depend on the RL victim's Q-table Q_t. Theoretically, our analysis does not require the RL agent's underlying Q_t to converge while still providing robustness certificates; see section 4.4.

Reward Shaping: While this chapter is phrased from the adversarial angle, the framework and techniques are also applicable to the *teaching* setting, where a *teacher* aims to guide the agent to learn the *optimal policy* as soon as possible, by designing the reward signal. Traditionally, reward shaping and more specifically potential-based reward shaping (Ng et al., 1999a) has been shown able to speed up learning while preserving the optimal policy. (Devlin and Kudenko, 2012) extend potential-based reward shaping to be time-varying while remains policy-preserving. More recently, intrinsic motivations(Schmidhuber, 1991; Oudeyer and Kaplan, 2009; Barto, 2013; Bellemare et al., 2016) was introduced as a new form of reward shaping with the goal of encouraging exploration and thus speed up learning. Our work contributes by mathematically defining the teaching via reward shaping task as an optimal control problem, and provide computational tools that solve for problem-dependent high-performing reward shaping strategies.

4.3 The Threat Model

In the reward-poisoning attack problem, we consider three entities: the environment MDP, the RL agent, and the attacker. Their interaction is formally described by Alg 4.

The environment MDP is $\mathcal{M} = (S, A, R, P, \mu_0)$ where S is the state space, A is the action space, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $P : S \times A \times S \to \mathbb{R}$ is the transition probability, and $\mu_0 : S \to \mathbb{R}$ is the initial state distribution. We assume S, A are finite, and that a uniformly random policy can visit each (s, a) pair infinitely often.

We focus on an RL agent that performs standard Q-learning defined by a tuple $\mathcal{A} = (Q_0, \epsilon, \gamma, \{\alpha_t\})$, where Q_0 is the initial Q table, ϵ is the random exploration probability, γ is the discounting factor, $\{\alpha_t\}$ is the learning rate scheduling as a function of t. This assumption can be generalized: in the additional experiments provided in appendix B.8, we show how the same framework can be applied to attack general RL agents, such as DQN. Denote Q^* as the optimal Q table that

satisfies the Bellman's equation:

$$Q^*(s,a) = \mathbf{E}_{P(s'|s,a)}\left[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a')\right] \tag{36}$$

and denote the corresponding optimal policy as $\pi^*(s) = \arg\max_a Q^*(s,a)$. For notational simplicity, we assume π^* is unique, though it is easy to generalize to multiple optimal policies.

Algorithm 4 Reward Poisoning against Q-learning

PARAMETERS: Agent parameters $\mathcal{A} = (Q_0, \epsilon, \gamma, \{\alpha_t\})$, MDP parameters $\mathcal{M} = (S, A, R, P, \mu_0)$.

1: **for** $t = 0, 1, \ldots$ **do**
2: agent at state s_t, has Q-table Q_t.
3: agent acts according to ϵ-greedy behavior policy

$$a_t \leftarrow \begin{cases} \arg\max_a Q_t(s_t,a), & \text{w.p. } 1-\epsilon \\ \text{uniform from } A, & \text{w.p. } \epsilon. \end{cases} \tag{37}$$

4: environment transits $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, produces reward $r_t = R(s_t, a_t, s_{t+1})$.

5: attacker poisons the reward to $r_t + \delta_t$.
6: agent receives $(s_{t+1}, r_t + \delta_t)$, performs Q-learning update:

$$\begin{aligned} &Q_{t+1}(s_t,a_t) \leftarrow (1-\alpha_t)Q_t(s_t,a_t) + \\ &\alpha_t\left(r_t + \delta_t + \gamma \max_{a' \in A} Q_t(s_{t+1},a')\right) \end{aligned} \tag{38}$$

7: environment resets if episode ends: $s_{t+1} \sim \mu_0$.
8: **end for**

The Threat Model The attacker sits between the environment and the RL agent. In this chapter we focus on white-box attacks: the attacker has knowledge of the environment MDP and the RL agent's Q-learning algorithm, except for their future

randomness. Specifically, at time t the attacker observes the learner Q-table Q_t, state s_t, action a_t, the environment transition s_{t+1} and reward r_t. The attacker can choose to add a perturbation $\delta_t \in \mathbb{R}$ to the current environmental reward r_t. The RL agent receives poisoned reward $r_t + \delta_t$. We assume the attack is inf-norm bounded: $|\delta_t| \leqslant \Delta, \forall t$.

There can be many possible attack goals against an RL agent: forcing the RL agent to perform certain actions; reaching or avoiding certain states; or maximizing its regret. In this chapter, we focus on a specific attack goal: **policy manipulation**. Concretely, the goal of policy manipulation is to force a target policy $\pi^\dagger$ on the RL agent for as many rounds as possible.

Definition 4.1. *Target (partial) policy $\pi^\dagger : S \mapsto 2^A$: For each $s \in S$, $\pi^\dagger(s) \subseteq A$ specifies the set of actions desired by the attacker.*

The partial policy $\pi^\dagger$ allows the attacker to desire multiple target actions on one state. In particular, if $\pi^\dagger(s) = A$ then s is a state that the attacker "does not care." Denote $S^\dagger = \{s \in S : \pi^\dagger(s) \neq A\}$ the set of **target states** on which the attacker does have a preference. In many applications, the attacker only cares about the agent's behavior on a small set of states, namely $|S^\dagger| \ll |S|$.

For RL agents utilizing a Q-table, a target policy $\pi^\dagger$ induces a set of Q-tables:

Definition 4.2. *Target Q-table set*

$$\mathcal{Q}^\dagger := \{Q : \max_{a \in \pi^\dagger(s)} Q(s,a) > \max_{a \notin \pi^\dagger(s)} Q(s,a), \forall s \in S^\dagger\}$$

If the target policy $\pi^\dagger$ always specifies a singleton action or does not care on all states, then $\mathcal{Q}^\dagger$ is a convex set. But in general when $1 < |\pi^\dagger(s)| < |A|$ on any s, $\mathcal{Q}^\dagger$ will be a union of convex sets and is itself non-convex.

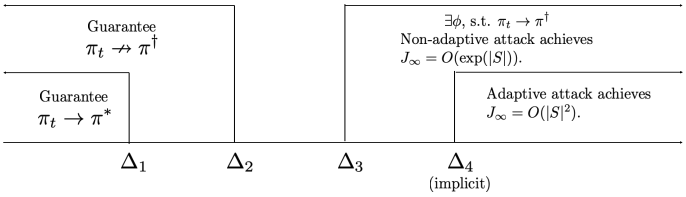


Figure 14: A summary diagram of the theoretical results.

4.4 Theoretical Guarantees

Now, we are ready to formally define the *optimal attack* problem. At time t, the attacker observes an *attack state* (N.B. distinct from MDP state s_t):

$$\xi_t := (s_t, a_t, s_{t+1}, r_t, Q_t) \in \Xi \tag{39}$$

which jointly characterizes the MDP and the RL agent. The attacker's goal is to find an *attack policy* $\phi : \Xi \to [-\Delta, \Delta]$, where for $\xi_t \in \Xi$ the *attack action* is $\delta_t := \phi(\xi_t)$, that minimizes the number of rounds on which the agent's Q_t disagrees with the attack target $\mathcal{Q}^\dagger$:

$$\min_{\phi} \quad \mathbb{E}_\phi \sum_{t=0}^{\infty} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger], \tag{40}$$

where the expectation accounts for randomness in Alg 4. We denote $J_\infty(\phi) = \mathbb{E}_\phi \sum_{t=0}^{\infty} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]$ the total attack cost, and $J_T(\phi) = \mathbb{E}_\phi \sum_{t=0}^{T} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]$ the finite-horizon cost. We say the attack is *feasible* if (40) is finite.

Next, we characterize attack feasibility in terms of poison magnitude constraint Δ, as summarized in Figure 14. Proofs to all the theorems can be found in the appendix.

Attack Infeasibility

Intuitively, smaller Δ makes it harder for the attacker to achieve the attack goal. We show that there is a threshold Δ_1 such that for any $\Delta < \Delta_1$ the RL agent is eventually safe, in that $\pi_t \to \pi^*$ the correct MDP policy. This implies that (40) is infinite and

the attack is infeasible. There is a potentially larger Δ_2 such that for any $\Delta < \Delta_2$ the attack is also infeasible, though π_t may not converge to π^*.

While the above statements are on π_t, our analysis is via the RL agent's underlying Q_t. Note that under attack the rewards $r_t + \delta_t$ are no longer stochastic, and we cannot utilize the usual Q-learning convergence guarantee. Nonetheless, we show that Q_t is bounded in a polytope in the Q-space.

Theorem 4.3 (Boundedness of Q-learning)**.** *Assume that $\delta_t < \Delta$ for all t, and the stepsize α_t's satisfy that $\alpha_t \leqslant 1$ for all t, $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. Let Q^* be defined as* (36)*. Then, for any attack sequence $\{\delta_t\}$, there exists $N \in \mathbb{N}$ such that, with probability 1, for all $t \geqslant N$, we have*

$$Q^*(s,a) - \frac{\Delta}{1-\gamma} \leqslant Q_t(s,a) \leqslant Q^*(s,a) + \frac{\Delta}{1-\gamma}. \tag{41}$$

Remark 1: The bounds in Theorem 4.3 are in fact tight. The lower and upper bound can be achieved by setting $\delta_t = -\Delta$ or $+\Delta$ respectively.

We immediately have the following two infeasibility certificates.

Corollary 4.4 (Strong Infeasibility Certificate)**.** *Define*

$$\Delta_1 = (1-\gamma) \min_s \left[Q^*(s, \pi^*(s)) - \max_{a \neq \pi^*(s)} Q^*(s,a) \right] /2.$$

If $\Delta < \Delta_1$, there exist $N \in \mathbb{N}$ such that, with probability 1, for all $t > N$, $\pi_t = \pi^*$. In other words, eventually the RL agent learns the optimal MDP policy π^* despite the attacks.

Corollary 4.5 (Weak Infeasibility Certificate)**.** *Given attack target policy $\pi^\dagger$, define*

$$\Delta_2 = (1-\gamma) \max_s \left[Q^*(s, \pi^*(s)) - \max_{a \in \pi^\dagger(s)} Q^*(s,a) \right] /2.$$

If $\Delta < \Delta_2$, there exist $N \in \mathbb{N}$ such that, with probability 1, for all $t > N$, $\pi_t(s) \notin \pi^\dagger(s)$ for some $s \in S^\dagger$. In other words, eventually the attacker is unable to enforce $\pi^\dagger$ (though π_t may not settle on π^* either).

Intuitively, an MDP is difficult to attack if its margin,

$$\min_s \left[Q^*(s, \pi^*(s)) - \max_{a \neq \pi^*(s)} Q^*(s, a) \right]$$

is large. This suggests a defense: for RL to be robust against poisoning, the environmental reward signal should be designed such that the optimal actions and suboptimal actions have large performance gaps.

Attack Feasibility

We now show there is a threshold Δ_3 such that for all $\Delta > \Delta_3$ the attacker can enforce $\pi^\dagger$ for all but finite number of rounds.

Theorem 4.6. *Given a target policy $\pi^\dagger$, define*

$$\Delta_3 = \frac{1+\gamma}{2} \max_{s \in S^\dagger} [\max_{a \notin \pi^\dagger(s)} Q^*(s, a) - \max_{a \in \pi^\dagger(s)} Q^*(s, a)]_+ \tag{42}$$

where $[x]_+ \coloneqq \max(x, 0)$. Assume the same conditions on α_t as in Theorem 4.3. If $\Delta > \Delta_3$, there is a feasible attack policy $\phi^{sas}_{\Delta_3}$. Furthermore, $J_\infty(\phi^{sas}_{\Delta_3}) \leqslant O(L^5)$, where L is the covering number.

Theorem 4.6 is proved by constructing an attack policy $\phi^{sas}_{\Delta_3}(s_t, a_t)$, detailed in Alg. 5. Note that this attack policy does not depend on Q_t. We call this type of attack *non-adaptive attack*. Under such construction, one can show that Q-learning converges to the target policy $\pi^\dagger$. Recall the covering number L is the upper bound on the minimum sequence length starting from any (s, a) pair and follow the MDP until all (state, action) pairs appear in the sequence (Even-Dar and Mansour, 2003). It is well-known that ϵ-greedy exploration has a covering time $L \leqslant O(e^{|S|})$ (Kearns and Singh, 2002). Prior work has constructed examples on which this bound is tight (Jin et al., 2018). We show in appendix B.3 that on our toy example ϵ-greedy indeed has a covering time $O(e^{|S|})$. Therefore, the objective value of (40) for non-

Algorithm 5 The Non-Adaptive Attack $\phi^{sas}_{\Delta_3}$

PARAMETERS: target policy $\pi^\dagger$, agent parameters $\mathcal{A} = (Q_0, \epsilon, \gamma, \{\alpha_t\})$, MDP parameters $\mathcal{M} = (S, A, R, P, \mu_0)$, maximum magnitude of poisoning Δ.
def Init($\pi^\dagger, \mathcal{A}, \mathcal{M}$):

1: Construct a Q-table Q', where $Q'(s, a)$ is defined as

$$\begin{cases} Q^*(s,a) + \dfrac{\Delta}{(1+\gamma)}, & \text{if } s \in S^\dagger, a \in \pi^\dagger(s) \\ Q^*(s,a) - \dfrac{\Delta}{(1+\gamma)}, & \text{if } s \in S^\dagger, a \notin \pi^\dagger(s) \\ Q^*(s,a), & \text{if } s \notin S^\dagger \end{cases}$$

2: Calculate a new reward function

$$R'(s,a) = Q'(s,a) - \gamma \mathbf{E}_{P(s'|s,a)}\left[\max_{a'} Q'(s',a')\right].$$

3: Define the attack policy $\phi^{sas}_{\Delta_3}$ as:

$$\phi^{sas}_{\Delta_3}(s,a) = R'(s,a) - \mathbf{E}_{P(s'|s,a)}\left[R(s,a,s)\right], \forall s, a.$$

def Attack(ξ_t):

1: Return $\phi^{sas}_{\Delta_3}(s_t, a_t)$

adaptive attack is upper-bounded by $O(e^{|S|})$. In other words, the non-adaptive attack is slow.

Fast Adaptive Attack (FAA)

We now show that there is a fast adaptive attack ϕ^{ξ}_{FAA} which depends on Q_t and achieves J_∞ polynomial in $|S|$. The price to pay is a larger attack constraint Δ_4, and the requirement that the attack target states are sparse: $k = |S^\dagger| \leqslant O(\log |S|)$. The FAA attack policy ϕ^{ξ}_{FAA} is defined in Alg. 6.

Conceptually, the FAA algorithm ranks the target states in descending order by their distance to the starting states, and focusing on attacking one target state at a time. Of central importance is the temporary target policy ν_i, which is designed to navigate the agent to the currently focused target state $s^\dagger_{(i)}$, while not altering the already achieved target actions on target states of earlier rank. This allows FAA to achieve a form of program invariance: after FAA achieves the target policy in a target state $s^\dagger_{(i)}$, the target policy on target state (i) will be preserved indefinitely. We provide a more detailed walk-through of Alg. 6 with examples in appendix B.5.

Definition 4.7. *Define the shortest ϵ-**distance** from s to s' as*

$$\begin{aligned} &d_\epsilon(s, s') = \min_{\pi \in \Pi} \mathbf{E}_{\pi_\epsilon}[T] \\ &s.t.\ s_0 = s, s_T = s', s_t \neq s', \forall t < T \end{aligned} \tag{43}$$

*where π_ϵ denotes the epsilon-greedy policy based on π. Since we are in an MDP, there exists a common (partial) policy $\pi_{s'}$ that achieves $d_\epsilon(s, s')$ for all source state $s \in S$. Denote $\pi_{s'}$ as the **navigation policy** to s'.*

Definition 4.8. *The ϵ-**diameter** of an MDP is defined as the longest shortest ϵ-distance between pairs of states in S:*

$$D_\epsilon = \max_{s, s' \in S} d_\epsilon(s, s') \tag{44}$$

Theorem 4.9. *Assume that the learner is running ϵ-greedy Q-learning algorithm on an episodic MDP with ϵ-diameter D_ϵ and maximum episode length H, and the attacker aims at k distinct target states, i.e. $|S^\dagger| = k$. Then, ϕ^ξ_{FAA} is feasible, and we have*

$$J_\infty(\phi^\xi_{FAA}) \leqslant k\frac{|S||A|H}{1-\epsilon} + \frac{|A|}{1-\epsilon}\left[\frac{|A|}{\epsilon}\right]^k D_\epsilon, \tag{45}$$

given that Δ is large enough that the $\mathtt{Clip}_\Delta()$ function in Alg. 6 never takes effect.

How large is D_ϵ? For MDPs with underlying structure as undirected graphs, such as the grid worlds, it is shown that the expected hitting time of a uniform

Algorithm 6 The Fast Adaptive Attack (FAA)

PARAMETERS: target policy $\pi^\dagger$, margin η, agent parameters $\mathcal{A} = (Q_0, \epsilon, \gamma, \{\alpha_t\})$, MDP parameters $\mathcal{M} = (S, A, R, P, \mu_0)$.
def Init($\pi^\dagger, \mathcal{A}, \mathcal{M}, \eta$):

1: Given (s_t, a_t, Q_t), define the hypothetical Q-update function without attack as $Q'_{t+1}(s_t, a_t) = (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t\,(r_t + \gamma(1 - \mathrm{EOE})\max_{a' \in A} Q_t(s_{t+1}, a'))$.

2: Given (s_t, a_t, Q_t), denote the greedy attack function at s_t w.r.t. a target action set A_{s_t} as $g(A_{s_t})$, defined as

$$\begin{cases} \frac{1}{\alpha_t}[\max_{a \notin A_{s_t}} Q_t(s_t, a) - \\ \qquad Q'_{t+1}(s_t, a_t) + \eta]_+ & \text{if } a_t \in A_{s_t} \\ \frac{1}{\alpha_t}[\max_{a \in A_{s_t}} Q_t(s_t, a) - \\ \qquad Q'_{t+1}(s_t, a_t) + \eta]_- & \text{if } a_t \notin A_{s_t}. \end{cases}$$

3: Define $\mathrm{Clip}_\Delta(\delta) = \min(\max(\delta, -\Delta), \Delta)$.
4: Rank the target states in descending order as $\{s^\dagger_{(1)}, ..., s^\dagger_{(k)}\}$, according to their shortest ϵ-distance to the initial state $\mathbf{E}_{s \sim \mu_0}\left[d^\epsilon(s, s_{(i)})\right]$.
5: **for** $i = 1, ..., k$ **do**
6: Define the temporary target policy ν_i as

$$\nu_i(s) = \begin{cases} \pi_{s^\dagger_{(i)}}(s) & \text{if } s \notin \{s^\dagger_{(j)} : j \leqslant i\} \\ \pi^\dagger(s) & \text{if } s \in \{s^\dagger_{(j)} : j \leqslant i\}. \end{cases}$$

7: **end for**

def Attack(ξ_t):

1: **for** $i = 1, ..., k$ **do**
2: **if** $\arg\max_a Q_t(s^\dagger_{(i)}, a) \notin \pi^\dagger(s^\dagger_{(i)})$ **then**
3: Return $\delta_t \leftarrow \mathrm{Clip}_\Delta(g(\{\nu_i(s_t)\}))$.
4: **end if**
5: **end for**
6: Return $\delta_t \leftarrow \mathrm{Clip}_\Delta(g(\{\pi^\dagger(s_t)\}))$.

random walk is bounded by $O(|S|^2)$(Lawler, 1986). Note that the random hitting time tightly upper bounds the optimal hitting time, a.k.a. the ϵ-diameter D_ϵ, and they match when $\epsilon = 1$. This immediately gives us the following result:

Corollary 4.10. *If in addition to the assumptions of Theorem 4.9, the maximal episode length* $H = O(|S|)$, *then* $J_\infty(\phi^\xi_{FAA}) \leqslant O(e^k|S|^2|A|)$ *in Grid World environments. When the number of target states is small, i.e.* $k \leqslant O(\log|S|)$, $J_\infty(\phi^\xi_{FAA}) \leqslant O(\text{poly}(|S|))$.

Remark 2: Theorem 4.9 and Corollary 4.10 can be thought of as defining an implicit Δ_4, such that for any $\Delta > \Delta_4$, the clip function in Alg. 6 never take effect, and ϕ^ξ_{FAA} achieves polynomial cost.

Illustrating Attack (In)feasibility Δ Thresholds

The theoretical results developed so far can be summarized as a diagram in Figure 14. We use the chain MDP in Figure 13 to illustrate the four thresholds $\Delta_1, \Delta_2, \Delta_3, \Delta_4$ developed in this section. On this MDP and with this attack target policy $\pi^\dagger$, we found that $\Delta_1 = \Delta_2 = 0.0069$. The two matches because this $\pi^\dagger$ is the easiest to achieve in terms of having the smallest upperbound Δ_2. Attackers whose poison magnitude $|\delta_t| < \Delta_2$ will not be able to enforce the target policy $\pi^\dagger$ in the long run.

We found that $\Delta_3 = 0.132$. We know that $\phi^{sas}_{\Delta_3}$ should be feasible if $\Delta > \Delta_3$. To illustrate this, we ran $\phi^{sas}_{\Delta_3}$ with $\Delta = 0.2 > \Delta_3$ for 1000 trials and obtained estimated $J_{10^5}(\phi^{sas}_{\Delta_3}) = 9430$. The fact that $J_{10^5}(\phi^{sas}_{\Delta_3}) \ll T = 10^5$ is empirical evidence that $\phi^{sas}_{\Delta_3}$ is feasible. We found that $\Delta_4 = 1$ by simulation. The adaptive attack ϕ^ξ_{FAA} constructed in Theorem 4.9 should be feasible with $\Delta = \Delta_4 = 1$. We run ϕ^ξ_{FAA} for 1000 trials and observed $J_{10^5}(\phi^\xi_{FAA}) = 30.4 \ll T$, again verifying the theorem. Also observe that $J_{10^5}(\phi^\xi_{FAA})$ is much smaller than $J_{10^5}(\phi^{sas}_{\Delta_3})$, verifying the foundamental difference in attack efficiency between the two attack policies as shown in Theorem 4.6 and Corollary 4.10.

While FAA is able to force the target policy in polynomial time, it's not necessarily the optimal attack strategy. Next, we demonstrate how to solve for the optimal attack problem in practice, and empirically show that with the techniques from

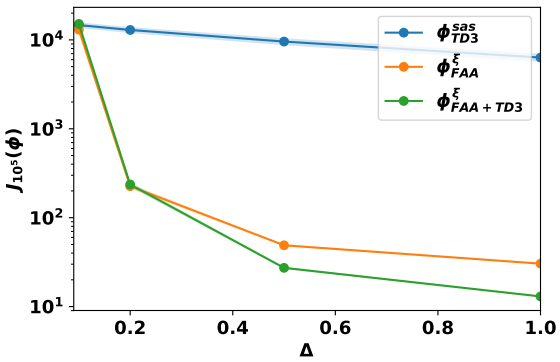


Figure 15: Attack cost $J_{10^5}(\phi)$ on different Δ's. Each curve shows mean ± 1 standard error over 1000 independent test runs.

Deep Reinforcement Learning (DRL), we can find efficient attack policies in a variety of environments.

4.5 Attack RL with RL

The attack policies $\phi^{sas}_{\Delta_3}$ and ϕ^{ξ}_{FAA} were manually constructed for theoretical analysis. Empirically, though, they do not have to be the most effective attacks under the relevant Δ constraint.

In this section, we present our key computational insight: the attacker can find an effective attack policy by relaxing the attack problem (40) so that the relaxed problem can be effectively solved with RL. Concretely, consider the higher-level attack MDP $\mathcal{N} = (\Xi, \Delta, \rho, \tau)$ and the associated optimal control problem:

- The attacker observes the attack state $\xi_t \in \Xi$.
- The attack action space is $\{\delta_t \in \mathbb{R} : |\delta_t| \leqslant \Delta\}$.
- The original attack loss function $\mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]$ is a 0-1 loss that is hard to optimize. We replace it with a continuous surrogate loss function ρ that measures how

close the current agent Q-table Q_t is to the target Q-table set:

$$\rho(\xi_t) = \sum_{s \in S^\dagger} \left[\max_{a \notin \pi^\dagger(s)} Q_t(s,a) - \max_{a \in \pi^\dagger(s)} Q_t(s,a) + \eta \right]_+ \tag{46}$$

where $\eta > 0$ is a margin parameter to encourage that $\pi^\dagger(s)$ is strictly preferred over $A \backslash \pi^\dagger(s)$ with no ties.

- The attack state transition probability is defined by $\tau(\xi_{t+1} \mid \xi_t, \delta_t)$. Specifically, the new attack state $\xi_{t+1} = (s_{t+1}, a_{t+1}, s_{t+2}, r_{t+1}, Q_{t+1})$ is generated as follows:
  - s_{t+1} is copied from ξ_t if not the end of episode, else $s_{t+1} \sim \mu_0$.
  - a_{t+1} is the RL agent's exploration action drawn according to (37), note it involves Q_{t+1}.
  - s_{t+2} is the RL agent's new state drawn according to the MDP transition probability $P(\cdot \mid s_{t+1}, a_{t+1})$.
  - r_{t+1} is the new (not yet poison) reward according to MDP $R(s_{t+1}, a_{t+1}, s_{t+2})$.
  - The attack δ_t happens. The RL agent updates Q_{t+1} according to (38).

With the higher-level attack MDP $\mathcal{N}$, we relax the optimal attack problem (40) into

$$\phi^* = \arg\min_{\phi} \mathbb{E}_\phi \sum_{t=0}^{\infty} \rho(\xi_t) \tag{47}$$

One can now solve (47) using Deep RL algorithms. In this chapter, we choose Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), a state-of-the-art algorithm for continuous action space. We use the same set of hyperparameters for TD3 across all experiments, described in appendix B.6.

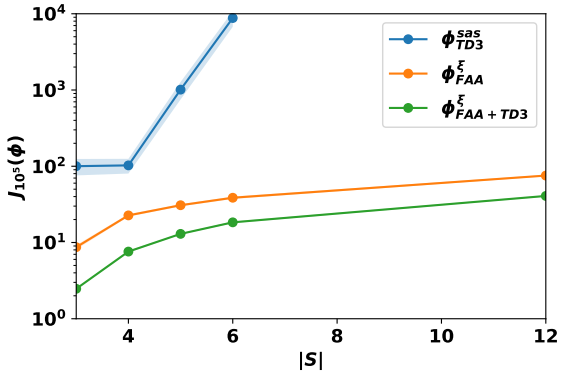


Figure 16: Attack performances on the chain MDPs of different lengths. Each curve shows mean ± 1 standard error over 1000 independent test runs.

4.6 Experiments

In this section, We make empirical comparisons between a number of attack policies ϕ: We use the naming convention where the superscript denotes non-adaptive or adaptive policy: ϕ^{sas} depends on (s_t, a_t, s_{t+1}) but not Q_t. Such policies have been extensively used in the reward shaping literature and prior work (Ma et al., 2019; Huang and Zhu, 2019) on reward poisoning; ϕ^{ξ} depends on the whole attack state ξ_t. We use the subscript to denote how the policy is constructed. Therefore, ϕ^{ξ}_{TD3} is the attack policy found by solving (47) with TD3; $\phi^{\xi}_{FAA+TD3}$ is the attack policy found by TD3 initialized from FAA (Algorithm 6), where TD3 learns to provide an additional δ'_t on top of the δ_t generated by ϕ^{ξ}_{FAA}, and the agent receives $r_t + \delta_t + \delta'_t$ as reward; ϕ^{sas}_{TD3} is the attack policy found using TD3 with the restriction that the attack policy only takes (s_t, a_t, s_{t+1}) as input.

In all of our experiments, we assume a standard Q-learning RL agent with parameters: $Q_0 = 0^{S\times A}$, $\epsilon = 0.1, \gamma = 0.9, \alpha_t = 0.9, \forall t$. The plots show ± 1 standard error around each curve (some are difficult to see). We will often evaluate an attack policy ϕ using a Monte Carlo estimate of the 0-1 attack cost $J_T(\phi)$ for $T = 10^5$, which approximates the objective $J_\infty(\phi)$ in (40).

Efficiency of Attacks across different Δ's

Recall that $\Delta > \Delta_3$, $\Delta > \Delta_4$ are sufficient conditions for manually-designed attack policies $\phi^{sas}_{\Delta_3}$ and ϕ^{ξ}_{FAA} to be feasible, but they are not necessary conditions. In this experiment, we empirically investigate the feasibilities and efficiency of non-adaptive and adaptive attacks across different Δ values.

We perform the experiments on the chain MDP in Figure 13. Recall that on this example, $\Delta_3 = 0.132$ and $\Delta_4 = 1$ (implicit). We evaluate across 4 different Δ values, $[0.1, 0.2, 0.5, 1]$, covering the range from Δ_3 to Δ_4. The result is shown in Figure 15.

We are able to make several interesting observations:
(1) All attacks are feasible (y-axis $\ll T$), even when Δ falls under the thresholds Δ_3 and Δ_4 for corresponding methods. This suggests that the feasibility thresholds are not tight.
(2) For non-adaptive attacks, as Δ increases the best-found attack policies ϕ^{sas}_{TD3} achieve small improvement, but generally incur a large attack cost.
(3) Adaptive attacks are very efficient when Δ is large. At $\Delta = 1$, the best adaptive attack $\phi^{\xi}_{FAA+TD3}$ achieves a cost of merely 13 (takes 13 steps to always force $\pi^\dagger$ on the RL agent). However, as Δ decreases the performance quickly degrades. At $\Delta = 0.1$ adaptive attacks are only as good as non-adaptive attacks. This shows an interesting transition region in Δ that our theoretical analysis does not cover.

Adaptive Attacks are Faster

In this experiment, we empirically verify that, while both are feasible, adaptive attacks indeed have an attack cost $O(\text{Poly}|S|)$ while non-adaptive attacks have $O(e^{|S|})$. The 0-1 costs $1[\pi_t \neq \pi^\dagger]$ are in general incurred at the beginning of each $t = 0 \dots T$ run. In other words, adaptive attacks achieve $\pi^\dagger$ faster than non-adaptive attacks. We use several chain MDPs similar to Figure 13 but with increasing number of states $|S| = 3, 4, 5, 6, 12$. We provide a large enough $\Delta = 2 \gg \Delta_4$ to ensure the feasibility of all attack policies. The result is shown in Figure 16. The best-found non-adaptive attack ϕ^{sas}_{TD3} is approximately straight on the log-scale plot, suggesting attack cost J growing exponentially with MDP size $|S|$. In contrast, the two adaptive

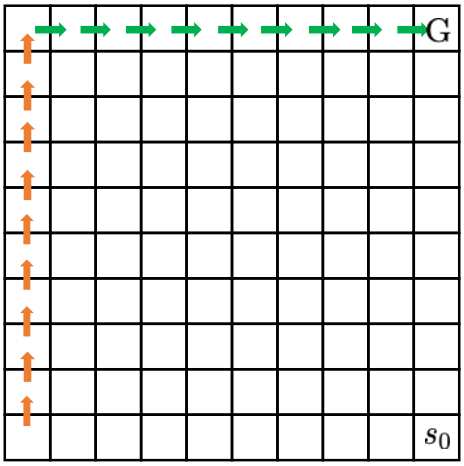


Figure 17: The 10×10 Grid World. s_0 is the starting state and G the terminal goal. Each move has a -0.1 negative reward, and a $+1$ reward for arriving at the goal. We consider two partial target policies: $\pi_1^\dagger$ marked by the green arrows, and $\pi_2^\dagger$ by *both* the green and the orange arrows.

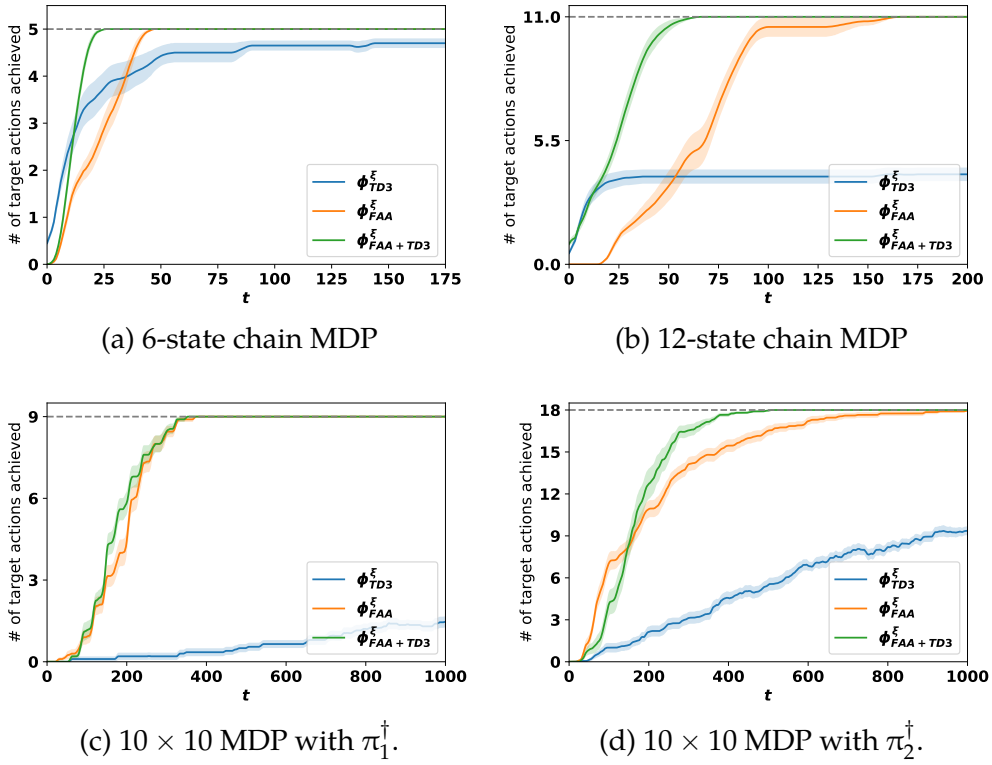
attack polices ϕ^ξ_{FAA} and $\phi^\xi_{\mathrm{FAA+TD3}}$ actually achieves attack cost linear in $|S|$. This is not easy to see from this log-scaled plot; We reproduce Figure 16 without the log scale in the appendix B.7, where the linear rate can be clearly verified. This suggests that the upperbound developed in Theorem 4.9 and Corollary 4.10 can be potentially improved.

Ablation Study

In this experiment, we compare three adaptive attack policies: ϕ^ξ_{TD3} the policy found by out-of-the-box TD3, ϕ^ξ_{FAA} the manually designed FAA policy, and $\phi^\xi_{\mathrm{FAA+TD3}}$ the policy found by using FAA as initialization for TD3.

We use three MDPs: a 6-state chain MDP, a 12-state chain MDP, and a 10×10 grid world MDP.. The 10×10 MDP has two separate target policies $\pi_1^\dagger$ and $\pi_2^\dagger$, see Figure 17.

For evaluation, we compute the number of target actions achieved $|\{s \in S^\dagger : \pi_t(s) \in \pi^\dagger(s)\}|$ as a function of t. This allows us to look more closely into the progress made by an attack. The results are shown in Figure 18.



(a) 6-state chain MDP

(b) 12-state chain MDP

(c) 10×10 MDP with $\pi_1^\dagger$.

(d) 10×10 MDP with $\pi_2^\dagger$.

Figure 18: Experiment results for the ablation study. Each curve shows mean ± 1 standard error over 20 independent test runs. The gray dashed lines indicate the total number of target actions.

First, observe that across all 4 experiments, attack policy ϕ^{ξ}_{TD3} found by out-of-the-box TD3 never succeeded in achieving all target actions. This indicates that TD3 alone cannot produce an effective attack. We hypothesize that this is due to a lack of effective exploration scheme: when the target states are sparse ($|S^\dagger| \ll |S|$) it can be hard for TD3 equipped with Gaussian exploration noise to locate all target states. As a result, the attack policy found by vanilla TD3 is only able to achieve the target actions on a subset of frequently visited target states.

Hand-crafted ϕ^{ξ}_{FAA} is effective in achieving the target policies, as is guaranteed by our theory. Nevertheless, we found that $\phi^{\xi}_{\text{FAA+TD3}}$ always improves upon ϕ^{ξ}_{TD3}. Recall that we use FAA as the initialization and then run TD3. This indicates that

TD3 can be highly effective with a good initialization, which effectively serves as the initial exploration policy that allows TD3 to locate all the target states.

Of special interest are the two experiments on the 10×10 Grid World with different target policies. Conceptually, the advantage of the adaptive attack is that the attacker can perform explicit navigation to lure the agent into the target states. An efficient navigation policy that leads the agent to all target states will make the attack very efficient. Observe that in Figure 17, both target polices form a chain, so that if the agent starts at *the beginning of the chain*, the target actions naturally lead the agent to the subsequent target states, achieving efficient navigation.

Recall that the FAA algorithm prioritizes the target states farthest to the starting state. In the 10×10 Grid World, the farthest state is the top-left grid. For target states $S_1^\dagger$, the top-left grid turns out to be the beginning of the *target chain*. As a result, ϕ^ξ_{FAA} is already very efficient, and $\phi^\xi_{\mathrm{FAA+TD3}}$ couldn't achieve much improvement, as shown in 18c. On the other hand, for target states $S_2^\dagger$, the top-left grid is in the middle of the target chain, which makes ϕ^ξ_{FAA} not as efficient. In this case, $\phi^\xi_{\mathrm{FAA+TD3}}$ makes a significant improvement, successfully forcing the target policy in about 500 steps, whereas it takes ϕ^ξ_{FAA} as many as 1000 steps, about twice as long as $\phi^\xi_{\mathrm{FAA+TD3}}$.

4.7 Conclusion

In this chapter, we studied the problem of reward-poisoning attacks on reinforcement-learning agents. Theoretically, we provide robustness certificates that guarantee the truthfulness of the learned policy when the attacker's constraint is stringent. When the constraint is loose, we show that by being adaptive to the agent's internal state, the attacker can force the target policy in polynomial time, whereas a naive non-adaptive attack takes exponential time. Empirically, we formulate that the reward poisoning problem as an optimal control problem on a higher-level attack MDP, and developed computational tools based on DRL that is able to find efficient attack policies across a variety of environments.

5 POLICY POISONING IN BATCH REINFORCEMENT LEARNING AND CONTROL

Contribution Statement. This chapter is joint work with Xuezhou Zhang, Wen Sun and Xiaojin Zhu. The author Yuzhe Ma is the leading author and completed most of the work, including the theoretical analysis and the experiments. The paper version of this chapter appeared in NeurIPS19.

5.1 Introduction

With the increasing adoption of machine learning, it is critical to study security threats to learning algorithms and design effective defense mechanisms against those threats. There has been significant work on adversarial attacks (Biggio and Roli, 2018; Huang et al., 2011). We focus on the subarea of data poisoning attacks where the adversary manipulates the training data so that the learner learns a wrong model. Prior work on data poisoning targeted victims in supervised learning (Mei and Zhu, 2015b; Koh et al., 2018; Wang and Chaudhuri, 2018; Zhang and Zhu, 2019) and multi-armed bandits (Jun et al., 2018; Ma et al., 2018; Liu and Shroff, 2019). We take a step further and study data poisoning attacks on reinforcement learning (RL). Given RL's prominent applications in robotics, games and so on, an intentionally and adversarially planted bad policy could be devastating.

While there has been some related work in test-time attack on RL, reward shaping, and teaching inverse reinforcement learning (IRL), little is understood on how to training-set poison a reinforcement learner. We take the first step and focus on *batch* reinforcement learner and controller as the victims. These victims learn their policy from a batch training set. We assume that the attacker can modify the rewards in the training set, which we show is sufficient for policy poisoning. The attacker's goal is to force the victim to learn a particular target policy (hence the name policy poisoning), while minimizing the reward modifications. Our main contribution is to characterize batch policy poisoning with a unified optimization

framework, and to study two instances against tabular certainty-equivalence (TCE) victim and linear quadratic regulator (LQR) victim, respectively.

5.2 Related Work

Of particular interest is the work on *test-time attacks* against RL (Huang et al., 2017). Unlike policy poisoning, there the RL agent carries out an already-learned and fixed policy π to e.g. play the Pong Game. The attacker perturbs pixels in a game board image, which is part of the state s. This essentially changes the RL agent's perceived state into some s'. The RL agent then chooses the action $a' := \pi(s')$ (e.g. move down) which may differ from $a := \pi(s)$ (e.g. move up). The attacker's goal is to force some specific a' on the RL agent. Note π itself stays the same through the attack. In contrast, ours is a data-poisoning attack which happens at training time and aims to change π.

Data-poisoning attacks were previously limited to supervised learning victims, either in batch mode (Biggio et al., 2012; Xiao et al., 2015; Li et al., 2016a; Mei and Zhu, 2015b) or online mode (Wang and Chaudhuri, 2018; Zhang and Zhu, 2019). Recently data-poisoning attacks have been extended to multi-armed bandit victims (Jun et al., 2018; Ma et al., 2018; Liu and Shroff, 2019), but not yet to RL victims.

There are two related but distinct concepts in RL research. One concept is reward shaping (Ng et al., 1999a; Asmuth et al., 2008; Devlin and Kudenko, 2012; Wiewiora, 2003) which also modifies rewards to affect an RL agent. However, the goal of reward shaping is fundamentally different from ours. Reward shaping aims to speed up convergence to the *same* optimal policy as without shaping. Note the differences in both the target (same vs. different policies) and the optimality measure (speed to converge vs. magnitude of reward change).

The other concept is teaching IRL (Cakmak and Lopes, 2012; Brown and Niekum, 2019; Kamalaruban et al., 2019). Teaching and attacking are mathematically equivalent. However, the main difference to our work is the victim. They require an IRL agent, which is a specialized algorithm that estimates a reward function from

demonstrations of (state, action) trajectories alone (i.e. no reward given). In contrast, our attacks target more prevalent RL agents and are thus potentially more applicable. Due to the difference in the input to IRL vs. RL victims, our attack framework is completely different.

5.3 Preliminaries

A Markov Decision Process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition kernel where $\Delta_{\mathcal{S}}$ denotes the space of probability distributions on $\mathcal{S}$, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is a discounting factor. We define a policy $\pi : \mathcal{S} \to \mathcal{A}$ as a function that maps a state to an action. We denote the Q function of a policy π as $Q^{\pi}(s, a) = \mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^{\tau} R(s_{\tau}, a_{\tau}) \mid s_0 = s, a_0 = a, \pi]$, where the expectation is over the randomness in both transitions and rewards. The Q function that corresponds to the optimal policy can be characterized by the following Bellman optimality equation:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \max_{a' \in \mathcal{A}} Q^*(s', a'), \tag{48}$$

and the optimal policy is defined as $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} Q^*(s, a)$.

We focus on RL victims who perform batch reinforcement learning. A training item is a tuple $(s, a, r, s') \in \mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S}$, where s is the current state, a is the action taken, r is the received reward, and s' is the next state. A training set is a batch of T training items denoted by $D = (s_t, a_t, r_t, s'_t)_{t=0:T-1}$. Given training set D, a model-based learner performs learning in two steps:

Step 1. The learner estimates an MDP $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}, \gamma)$ from D. In particular, we assume the learner uses maximum likelihood estimate for the transition kernel $\hat{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta_{\mathcal{S}}$

$$\hat{P} \in \arg\max_{P} \sum_{t=0}^{T-1} \log P(s'_t|s_t, a_t), \tag{49}$$

and least-squares estimate for the reward function $\hat{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$

$$\hat{R} = \arg\min_{R} \sum_{t=0}^{T-1} (r_t - R(s_t, a_t))^2. \tag{50}$$

Note that we do not require (49) to have a unique maximizer $\hat{P}$. When multiple maximizers exist, we assume the learner arbitrarily picks one of them as the estimate. We assume the minimizer $\hat{R}$ is always unique. We will discuss the conditions to guarantee the uniqueness of $\hat{R}$ for two learners later.

Step 2. The learner finds the optimal policy $\hat{\pi}$ that maximizes the expected discounted cumulative reward on the estimated environment $\hat{M}$, i.e.,

$$\hat{\pi} \in \arg\max_{\pi:\mathcal{S}\mapsto\mathcal{A}} \mathbb{E}_{\hat{P}} \sum_{\tau=0}^{\infty} \gamma^\tau \hat{R}(s_\tau, \pi(s_\tau)), \tag{51}$$

where s_0 is a specified or random initial state. Note that there could be multiple optimal policies, thus we use $\in$ in (51). Later we will specialize (51) to two specific victim learners: the tabular certainty equivalence learner (TCE) and the certainty-equivalent linear quadratic regulator (LQR).

5.4 Policy Poisoning

We study policy poisoning attacks on model-based batch RL learners. Our threat model is as follows:

Knowledge of the attacker. The attacker has access to the original training set $D^0 = (s_t, a_t, r_t^0, s_t')_{t=0:T-1}$. The attacker knows the model-based RL learner's algorithm. Importantly, the attacker knows how the learner estimates the environment, i.e., (49) and (50). In the case (49) has multiple maximizers, we assume the attacker knows exactly the $\hat{P}$ that the learner picks.

Available actions of the attacker. The attacker is allowed to arbitrarily modify the rewards $\mathbf{r}^0 = (r_0^0, ..., r_{T-1}^0)$ in D^0 into $\mathbf{r} = (r_0, ..., r_{T-1})$. As we show later, changing r's but not s, a, s' is sufficient for policy poisoning.

Attacker's goals. The attacker has a pre-specified target policy $\pi^\dagger$. The attack goals are to (1) force the learner to learn $\pi^\dagger$, (2) minimize attack cost $\|\mathbf{r} - \mathbf{r}^0\|_\alpha$ under an α-norm chosen by the attacker.

Given the threat model, we can formulate policy poisoning as a bi-level optimization problem[8]:

$$\min_{\mathbf{r}, \hat{\mathsf{R}}} \quad \|\mathbf{r} - \mathbf{r}^0\|_\alpha \tag{52}$$

$$\text{s.t.} \quad \hat{\mathsf{R}} = \arg\min_{\mathsf{R}} \sum_{t=0}^{T-1} (r_t - \mathsf{R}(s_t, a_t))^2 \tag{53}$$

$$\{\pi^\dagger\} = \arg\max_{\pi: \mathcal{S} \mapsto \mathcal{A}} \mathbb{E}_{\hat{P}} \sum_{\tau=0}^{\infty} \gamma^\tau \hat{\mathsf{R}}(s_\tau, \pi(s_\tau)). \tag{54}$$

The $\hat{P}$ in (54) does not involve $\mathbf{r}$ and is precomputed from D^0. The singleton set $\{\pi^\dagger\}$ on the LHS of (54) ensures that the target policy is learned uniquely, i.e., there are no other optimal policies tied with $\pi^\dagger$. Next, we instantiate this attack formulation to two representative model-based RL victims.

Poisoning a Tabular Certainty Equivalence (TCE) Victim

In tabular certainty equivalence (TCE), the environment is a Markov Decision Process (MDP) with finite state and action space. Given original data $D^0 = (s_t, a_t, r_t^0, s_t')_{0:T-1}$, let $T_{s,a} = \{t \mid s_t = s, a_t = a\}$, the time indexes of all training items for which action a is taken at state s. We assume $T_{s,a} \geqslant 1, \forall s, a$, i.e., each state-action pair appears at least once in D^0. This condition is needed to ensure that the learner's estimate $\hat{P}$ and $\hat{\mathsf{R}}$ exist. Remember that we require (50) to have a unique solution. For the TCE learner, $\hat{\mathsf{R}}$ is unique as long as it exists. Therefore, $T_{s,a} \geqslant 1, \forall s, a$ is sufficient to guarantee a unique solution to (50). Let the poisoned data be $D = (s_t, a_t, r_t, s_t')_{0:T-1}$.

[8] As we will show, the constraint (54) could lead to an open feasible set (e.g., in (57)) for the attack optimization (52)-(54), on which the minimum of the objective function (52) may not be well-defined. In the case (54) induces an open set, we will consider instead a closed subset of it, and optimize over the subset. How to construct the closed subset will be made clear for concrete learners later.

Instantiating model estimation (49), (50) for TCE, we have

$$\hat{P}(s' \mid s, a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} \mathbb{1}\left[s'_t = s'\right], \tag{55}$$

where $\mathbb{1}[]$ is the indicator function, and

$$\hat{R}(s, a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t. \tag{56}$$

The TCE learner uses $\hat{P}, \hat{R}$ to form an estimated MDP $\hat{M}$, then solves for the optimal policy $\hat{\pi}$ with respect to $\hat{M}$ using the Bellman equation (48). The attack goal (54) can be naively characterized by

$$Q(s, \pi^\dagger(s)) > Q(s, a), \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s). \tag{57}$$

However, due to the strict inequality, (57) induces an open set in the Q space, on which the minimum of (52) may not be well-defined. Instead, we require a stronger attack goal which leads to a closed subset in the Q space. This is defined as the following ϵ-robust target Q polytope.

Definition 5.1. *(ϵ-robust target Q polytope) The set of ϵ-robust Q functions induced by a target policy $\pi^\dagger$ is the polytope*

$$\mathcal{Q}_\epsilon(\pi^\dagger) = \{Q : Q(s, \pi^\dagger(s)) \geqslant Q(s, a) + \epsilon, \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s)\} \tag{58}$$

for a fixed $\epsilon > 0$.

The margin parameter ϵ ensures that $\pi^\dagger$ is the unique optimal policy for any Q in the polytope. We now have a solvable attack problem, where the attacker wants

to force the victim's Q function into the ϵ-robust target Q polytope $\mathcal{Q}_\epsilon(\pi^\dagger)$:

$$\min_{\mathbf{r}\in\mathbb{R}^T,\hat{R},Q\in\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}} \quad \|\mathbf{r}-\mathbf{r}^0\|_\alpha \tag{59}$$

$$\text{s.t.} \quad \hat{R}(s,a) = \frac{1}{|T_{s,a}|}\sum_{t\in T_{s,a}} r_t \tag{60}$$

$$Q(s,a) = \hat{R}(s,a) + \gamma\sum_{s'}\hat{P}(s'|s,a)\,Q\left(s',\pi^\dagger(s')\right), \forall s, \forall a, \tag{61}$$

$$Q(s,\pi^\dagger(s)) \geqslant Q(s,a) + \epsilon, \forall s\in\mathcal{S}, \forall a\neq\pi^\dagger(s). \tag{62}$$

The constraint (61) enforces Bellman optimality on the value function Q, in which $\max_{a'\in\mathcal{A}} Q(s',a')$ is replaced by $Q\left(s',\pi^\dagger(s')\right)$, since the target policy is guaranteed to be optimal by (62). Note that problem (59)-(62) is a convex program with linear constraints given that $\alpha \geqslant 1$, thus could be solved to global optimality. However, we point out that (59)-(62) is a more stringent formulation than (52)-(54) due to the additional margin parameter ϵ we introduced. The feasible set of (59)-(62) is a subset of (52)-(54). Therefore, the optimal solution to (59)-(62) could in general be a sub-optimal solution to (52)-(54) with potentially larger objective value. We now study a few theoretical properties of policy poisoning on TCE. All proofs are in the appendix. First of all, the attack is always feasible.

Proposition 5.2. *The attack problem* (59)-(62) *is always feasible for any target policy* $\pi^\dagger$.

Proposition 5.2 states that for any target policy $\pi^\dagger$, there exists a perturbation on the rewards that teaches the learner that policy. Therefore, the attacker changing r's but not s, a, s' is already sufficient for policy poisoning.

We next bound the attack cost. Let the MDP estimated on the clean data be $\hat{M}^0 = (\mathcal{S},\mathcal{A},\hat{P},\hat{R}^0,\gamma)$. Let Q^0 be the Q function that satisfies the Bellman optimality equation on $\hat{M}^0$. Define $\Delta(\epsilon) = \max_{s\in\mathcal{S}}[\max_{a\neq\pi^\dagger(s)} Q^0(s,a) - Q^0(s,\pi^\dagger(s)) + \epsilon]_+$, where $[]_+$ takes the maximum over 0. Intuitively, $\Delta(\epsilon)$ measures how suboptimal the target policy $\pi^\dagger$ is compared to the clean optimal policy π^0 learned on $\hat{M}^0$, up to a margin parameter ϵ.

Theorem 5.3. *Assume* $\alpha \geqslant 1$ *in* (59). *Let* $\mathbf{r}^*$, $\hat{R}^*$ *and* Q^* *be an optimal solution to* (59)-(62), *then*

$$\frac{1}{2}(1-\gamma)\Delta(\epsilon)\left(\min_{s,a}|T_{s,a}|\right)^{\frac{1}{\alpha}} \leqslant \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leqslant \frac{1}{2}(1+\gamma)\Delta(\epsilon)T^{\frac{1}{\alpha}}. \tag{63}$$

Corollary 5.4. *If* $\alpha = 1$, *then the optimal attack cost is* $O(\Delta(\epsilon)T)$. *If* $\alpha = 2$, *then the optimal attack cost is* $O(\Delta(\epsilon)\sqrt{T})$. *If* $\alpha = \infty$, *then the optimal attack cost is* $O(\Delta(\epsilon))$.

Note that both the upper and lower bounds on the attack cost are linear with respect to $\Delta(\epsilon)$, which can be estimated directly from the clean training set D^0. This allows the attacker to easily estimate its attack cost before actually solving the attack problem.

Poisoning a Linear Quadratic Regulator (LQR) Victim

As the second example, we study an LQR victim that performs system identification from a batch training set (Dean et al., 2017). Let the linear dynamical system be

$$s_{t+1} = As_t + Ba_t + w_t, \forall t \geqslant 0, \tag{64}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $s_t \in \mathbb{R}^n$ is the state, $a_t \in \mathbb{R}^m$ is the control signal, and $w_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ is a Gaussian noise. When the agent takes action a at state s, it suffers a quadratic loss of the general form

$$L(s,a) = \frac{1}{2}s^\top Qs + q^\top s + a^\top Ra + c \tag{65}$$

for some $Q \succeq 0$, $R \succ 0$, $q \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Here we have redefined the symbols Q and R in order to conform with the notation convention in LQR: now we use Q for the quadratic loss matrix associated with state, not the action-value function; we use R for the quadratic loss matrix associated with action, not the reward function. The previous reward function $R(s,a)$ in general MDP (section 5.3) is now equivalent to the negative loss $-L(s,a)$. This form of loss captures various LQR control problems. Note that the above linear dynamical system can be viewed as

an MDP with transition kernel $P(s' \mid s, a) = \mathcal{N}(As + Ba, \sigma^2 I)$ and reward function $-L(s, a)$. The environment is thus characterized by matrices A, B (for transition kernel) and Q, R, q, c (for reward function), which are all unknown to the learner.

We assume the clean training data $D^0 = (s_t, a_t, r_t^0, s_{t+1})_{0:T-1}$ was generated by running the linear system for multiple episodes following some random policy (Dean et al., 2017). Let the poisoned data be $D = (s_t, a_t, r_t, s_{t+1})_{0:T-1}$. Instantiating model estimation (49), (50), the learner performs system identification on the poisoned data:

$$(\hat{A}, \hat{B}) \in \operatorname*{arg\,min}_{(A,B)} \frac{1}{2} \sum_{t=0}^{T-1} \|As_t + Ba_t - s_{t+1}\|_2^2 \tag{66}$$

$$(\hat{Q}, \hat{R}, \hat{q}, \hat{c}) = \operatorname*{arg\,min}_{(Q \succeq 0, R \succeq \epsilon I, q, c)} \frac{1}{2} \sum_{t=0}^{T-1} \left\| \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t \right\|_2^2. \tag{67}$$

Note that in (67), the learner uses a stronger constraint $R \succeq \epsilon I$ than the original constraint $R \succ 0$, which guarantees that the minimizer can be achieved. The conditions to further guarantee (67) having a unique solution depend on the property of certain matrices formed by the clean training set D^0, which we defer to appendix C.4.

The learner then computes the optimal control policy with respect to $\hat{A}$, $\hat{B}$, $\hat{Q}$, $\hat{R}$, $\hat{q}$ and $\hat{c}$. We assume the learner solves a discounted version of LQR control

$$\max_{\pi: \mathcal{S} \mapsto \mathcal{A}} \quad -\mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau \left(\frac{1}{2} s_\tau^\top \hat{Q} s_\tau + \hat{q}^\top s_\tau + \pi(s_\tau)^\top \hat{R} \pi(s_\tau) + \hat{c} \right) \right] \tag{68}$$

$$\text{s.t.} \quad s_{\tau+1} = \hat{A} s_\tau + \hat{B} \pi(s_\tau) + w_\tau, \forall \tau \geqslant 0. \tag{69}$$

where the expectation is over w_τ. It is known that the control problem has a closed-form solution $\hat{a}_\tau = \hat{\pi}(s_\tau) = K s_\tau + k$, where

$$K = -\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B} \right)^{-1} \hat{B}^\top X \hat{A}, \quad k = -\gamma (\hat{R} + \gamma \hat{B}^\top X \hat{B})^{-1} \hat{B}^\top x. \tag{70}$$

Here $X \succeq 0$ is the unique solution of the Algebraic Riccati Equation,

$$X = \gamma \hat{A}^\top X \hat{A} - \gamma^2 \hat{A}^\top X \hat{B} \left(\hat{R} + \gamma \hat{B}^\top X \hat{B} \right)^{-1} \hat{B}^\top X \hat{A} + \hat{Q}, \tag{71}$$

and x is a vector that satisfies

$$x = \hat{q} + \gamma(\hat{A} + \hat{B}K)^\top x. \tag{72}$$

The attacker aims to force the victim into taking target action $\pi^\dagger(s), \forall s \in \mathbb{R}^n$. Note that in LQR, the attacker cannot arbitrarily choose $\pi^\dagger$, as the optimal control policy K and k enforce a linear structural constraint between $\pi^\dagger(s)$ and s. One can easily see that the target action must obey $\pi^\dagger(s) = K^\dagger s + k^\dagger$ for some $(K^\dagger, k^\dagger)$ in order to achieve successful attack. Therefore we must assume instead that the attacker has a target policy specified by a pair $(K^\dagger, k^\dagger)$. However, an arbitrarily linear policy may still not be feasible. A target policy $(K^\dagger, k^\dagger)$ is feasible if and only if it is produced by solving some Riccati equation, namely, it must lie in the following set:

$$\{(K, k) : \exists Q \succeq 0, R \succeq \epsilon I, q \in \mathbb{R}^n, c \in \mathbb{R}, \text{such that (70), (71), and (72) are satisfied}\}. \tag{73}$$

Therefore, to guarantee feasibility, we assume the attacker always picks the target policy $(K^\dagger, k^\dagger)$ by solving an LQR problem with some attacker-defined loss function. We can now pose the policy poisoning attack problem:

$$\min_{r, \hat{Q}, \hat{R}, \hat{q}, \hat{c}, X, x} \quad \|\mathbf{r} - \mathbf{r}^0\|_\alpha \tag{74}$$

$$\text{s.t.} \quad -\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top X \hat{A} = K^\dagger \tag{75}$$

$$-\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top x = k^\dagger \tag{76}$$

$$X = \gamma \hat{A}^\top X \hat{A} - \gamma^2 \hat{A}^\top X \hat{B} \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top X \hat{A} + \hat{Q} \tag{77}$$

$$x = \hat{q} + \gamma(\hat{A} + \hat{B}K^\dagger)^\top x \tag{78}$$

$$(\hat{Q}, \hat{R}, \hat{q}, \hat{c}) = \underset{(Q \succeq 0, R \succeq \epsilon I, q, c)}{\arg\min} \sum_{t=0}^{T-1} \left\| \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t \right\|_2^2 \tag{79}$$

$$X \succeq 0. \tag{80}$$

Note that the estimated transition matrices $\hat{A}$, $\hat{B}$ are not optimization variables because the attacker can only modify the rewards, which will not change the learner's estimate on $\hat{A}$ and $\hat{B}$. The attack optimization (74)-(80) is hard to solve

due to the constraint (79) itself being a semi-definite program (SDP). To overcome the difficulty, we pull all the positive semi-definite constraints out of the lower-level optimization. This leads to a more stringent surrogate attack optimization (see appendix C.3). Solving the surrogate attack problem, whose feasible region is a subset of the original problem, in general gives a suboptimal solution to (74)-(80). But it comes with one advantage: convexity.

5.5 Experiments

Throughout the experiments, we use CVXPY (Diamond and Boyd, 2016) to implement the optimization. All code can be found in:
https://github.com/myzwisc/PPRL_NeurIPS19.

Policy Poisoning Attack on TCE Victim

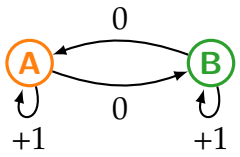
Experiment 1. We consider a simple MDP with two states A, B and two actions: *stay* in the same state or *move* to the other state, shown in figure 19a. The discounting factor is $\gamma = 0.9$. The MDP's Q values are shown in table 19b. Note that the optimal policy will always pick action *stay*. The clean training data D^0 reflects this underlying MDP, and consists of 4 tuples:

$$(A, \textit{stay}, 1, A) \quad (A, \textit{move}, 0, B) \quad (B, \textit{stay}, 1, B) \quad (B, \textit{move}, 0, A)$$

Let the attacker's target policy be $\pi^\dagger(s) =$*move*, for any state s. The attacker sets $\epsilon = 1$ and uses $\alpha = 2$, i.e. $\|\mathbf{r} - \mathbf{r}^0\|_2$ as the attack cost. Solving the policy poisoning attack optimization problem (59)-(62) produces the poisoned data:

$$(A, \textit{stay}, 0, A) \quad (A, \textit{move}, 1, B) \quad (B, \textit{stay}, 0, B) \quad (B, \textit{move}, 1, A)$$

with attack cost $\|\mathbf{r} - \mathbf{r}^0\|_2 = 2$. The resulting poisoned Q values are shown in table 19c. To verify this attack, we run TCE learner on both clean data and poisoned data. Specifically, we estimate the transition kernel and the reward function as



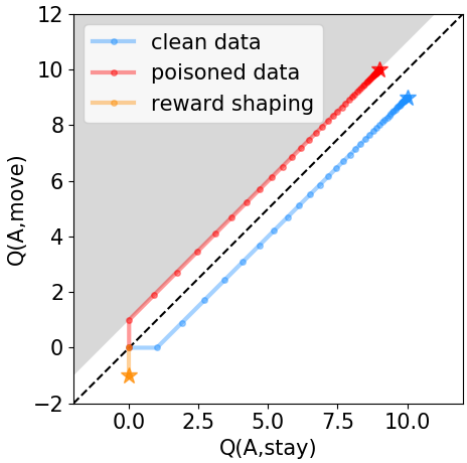
(a) A toy MDP with two states.

	stay	move
A	10	9
B	10	9

(b) Original Q values.

	stay	move
A	9	10
B	9	10

(c) Poisoned Q values.



(d) Trajectory for the Q values of state A during value iteration.

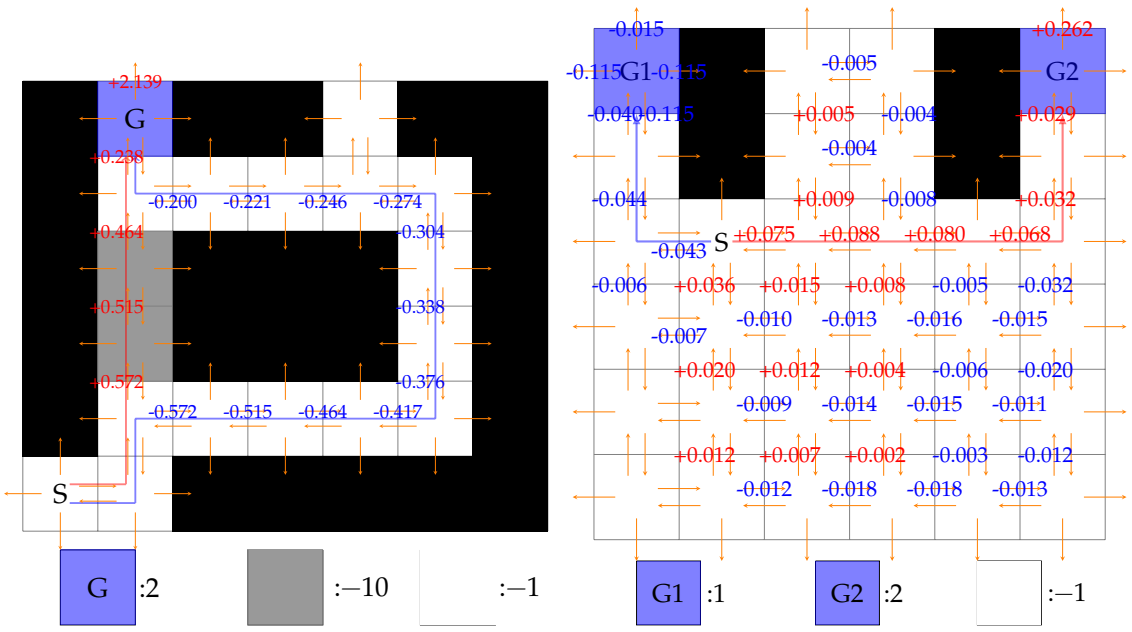
Figure 19: Poisoning TCE in a two-state MDP.

in (55) and (56) on each data set, and then run value iteration until the Q values converge. In Figure 19d, we show the trajectory of Q values for state A, where the x and y axes denote $Q(A, stay)$ and $Q(A, move)$ respectively. All trajectories start at $(0, 0)$. The dots on the trajectory correspond to each step of value iteration, while the star denotes the converged Q values. The diagonal dashed line is the (zero margin) policy boundary, while the gray region is the ϵ-robust target Q polytope with an offset $\epsilon = 1$ to the policy boundary. The trajectory of clean data converges to a point below the policy boundary, where the action $stay$ is optimal. With the poisoned data, the trajectory of Q values converge to a point exactly on the boundary of the ϵ-robust target Q polytope, where the action $move$ becomes optimal. This validates our attack.

We also compare our attack with reward shaping (Ng et al., 1999a). We let the potential function $\phi(s)$ be the optimal value function $V(s)$ for all s to shape the clean dataset. The dataset after shaping is

$$(A, stay, 0, A) \quad (A, move, -1, B) \quad (B, stay, 0, B) \quad (B, move, -1, A)$$

In Figure 19d, we show the trajectory of Q values after reward shaping. Note that same as on clean dataset, the trajectory after shaping converges to a point also below the policy boundary. This means reward shaping can not make the learner learn a different policy from the original optimal policy. Also note that after reward shaping, value iteration converges much faster (in only one iteration), which matches the benefits of reward shaping shown in (Ng et al., 1999a). More importantly, this illustrates the difference between our attack and reward shaping.



(a) Grid world with a single terminal state G.

(b) Grid world with two terminal states G_1 and G_2.

Figure 20: Poisoning TCE in grid-world tasks.

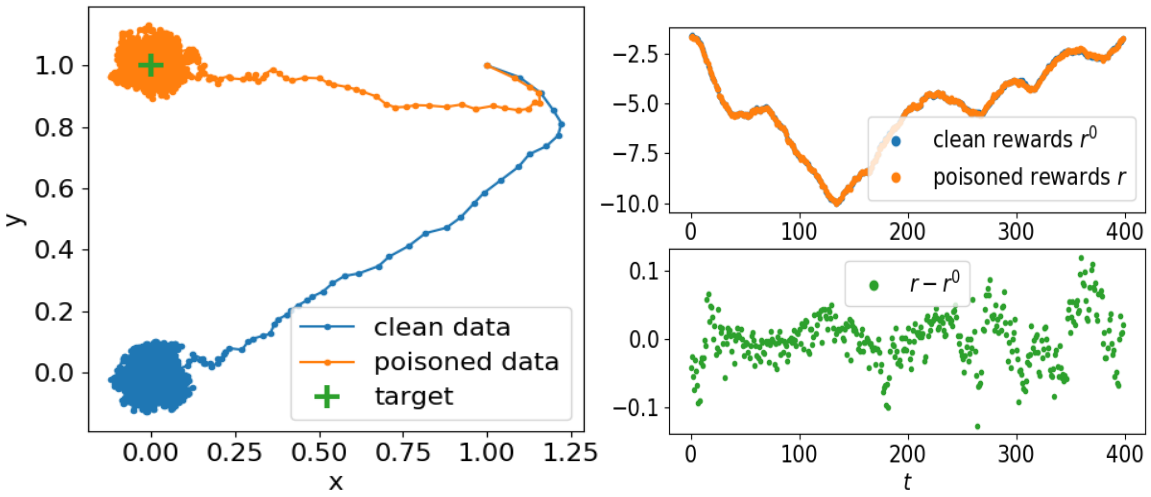
Experiment 2. As another example, we consider the grid world tasks in (Cakmak and Lopes, 2012). In particular, we focus on two tasks shown in figure 20a and 20b. In figure 20a, the agent starts from S and aims to arrive at the terminal cell G. The black regions are walls, thus the agent can only choose to go through the white or gray regions. The agent can take four actions in every state: go left, right, up or down, and stays if the action takes it into the wall. Reaching a gray, white, or the terminal state results in rewards -10, -1, 2, respectively. After the agent arrives at

the terminal state G, it will stay there forever and always receive reward 0 regardless of the following actions. The original optimal policy is to follow the blue trajectory. The attacker's goal is to force the agent to follow the red trajectory. Correspondingly, we set the target actions for those states on the red trajectory as along the trajectory. We set the target actions for the remaining states to be the same as the original optimal policy learned on clean data.

The clean training data contains a single item for every state-action pair. We run the attack with $\epsilon = 0.1$ and $\alpha = 2$. Our attack is successful: with the poisoned data, TCE generates a policy that produces the red trajectory in Figure 20a, which is the desired behavior. The attack cost is $\|\mathbf{r} - \mathbf{r}^0\|_2 \approx 2.64$, which is small compared to $\|\mathbf{r}^0\|_2 = 21.61$. In Figure 20a, we show the poisoning on rewards. Each state-action pair is denoted by an orange arrow. The value tagged to each arrow is the modification to that reward, where red value means the reward is increased and blue means decreased. An arrow without any tagged value means the corresponding reward is not changed by attack. Note that rewards along the red trajectory are increased, while those along the blue trajectory are reduced, resulting in the red trajectory being preferred by the agent. Furthermore, rewards closer to the starting state S suffer larger poisoning since they contribute more to the Q values. For the large attack +2.139 happening at terminal state, we provide an explanation in appendix C.5.

Experiment 3. In Figure 20b there are two terminal states G1 and G2 with reward 1 and 2, respectively. The agent starts from S. Although G2 is more profitable, the path is longer and each step has a -1 reward. Therefore, the original optimal policy is the blue trajectory to G1. The attacker's target policy is to force the agent along the red trajectory to G2. We set the target actions for states as in experiment 2. The clean training data contains a single item for every state-action pair. We run our attack with $\epsilon = 0.1$ and $\alpha = 2$. Again, after the attack, TCE on the poisoned dataset produces the red trajectory in figure 20b, with attack cost $\|\mathbf{r} - \mathbf{r}^0\|_2 \approx 0.38$, compared to $\|\mathbf{r}^0\|_2 = 11.09$. The reward poisoning follows a similar pattern to experiment 2.

Policy Poisoning Attack on LQR Victim



(a) Clean and poisoned vehicle trajectory.

(b) Clean and poisoned rewards.

Figure 21: Poisoning a vehicle running LQR in 4D state space.

Experiment 4. We now demonstrate our attack on LQR. We consider a linear dynamical system that approximately models a vehicle. The state of the vehicle consists of its 2D position and 2D velocity: $s_t = (x_t, y_t, v_t^x, v_t^y) \in \mathbb{R}^4$. The control signal at time t is the force $a_t \in \mathbb{R}^2$ which will be applied on the vehicle for h seconds. We assume there is a friction parameter η such that the friction force is $-\eta v_t$. Let m be the mass of the vehicle. Given small enough h, the transition matrices can be approximated by (64) where

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 - h\eta/m & 0 \\ 0 & 0 & 0 & 1 - h\eta/m \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m & 0 \\ 0 & h/m \end{bmatrix}. \tag{81}$$

In this example, we let $h = 0.1$, $m = 1$, $\eta = 0.5$, and $w_t \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 0.01$. The vehicle starts from initial position $(1, 1)$ with velocity $(1, -0.5)$, i.e., $s_0 = (1, 1, 1, -0.5)$. The true loss function is $L(s, a) = \frac{1}{2} s^\top Q s + a^\top R a$ with $Q = I$ and

$R = 0.1I$ (i.e., $Q = I, R = 0.1I, q = 0, c = 0$ in (65)). Throughout the experiment, we let $\gamma = 0.9$ for solving the optimal control policy in (68). With the true dynamics and loss function, the computed optimal control policy is

$$K^* = \begin{bmatrix} -1.32 & 0 & -2.39 & 0 \\ 0 & -1.32 & 0 & -2.39 \end{bmatrix}, k^* = \begin{bmatrix} 0 & 0 \end{bmatrix}, \tag{82}$$

which will drive the vehicle to the origin.

The batch LQR learner estimates the dynamics and the loss function from a batch training data. To produce the training data, we let the vehicle start from state s_0 and simulate its trajectory with a random control policy. Specifically, in each time step, we uniformly sample a control signal a_t in a unit sphere. The vehicle then takes action a_t to transit from current state s_t to the next state s_{t+1}, and receives a reward $r_t = -L(s_t, a_t)$. This gives us one training item (s_t, a_t, r_t, s_{t+1}). We simulate a total of 400 time steps to obtain a batch data that contains 400 items, on which the learner estimates the dynamics and the loss function. With the learner's estimate, the computed clean optimal policy is

$$\hat{K}^0 = \begin{bmatrix} -1.31 & 1.00e{-2} & -2.41 & 2.03e{-3} \\ -1.97e{-2} & -1.35 & -1.14e{-2} & -2.42 \end{bmatrix}, \hat{k}^0 = \begin{bmatrix} -4.88e{-5} & 4.95e{-6} \end{bmatrix}. \tag{83}$$

The clean optimal policy differs slightly from the true optimal policy due to the inaccuracy of the learner's estimate. The attacker has a target policy $(K^\dagger, k^\dagger)$ that can drive the vehicle close to its target destination $(x^\dagger, y^\dagger) = (0, 1)$ with terminal velocity $(0, 0)$, which can be represented as a target state $s^\dagger = (0, 1, 0, 0)$. To ensure feasibility, we assume that the attacker starts with the loss function $\frac{1}{2}(s - s^\dagger)^\top Q(s - s^\dagger) + a^\top R a$ where $Q = I, R = 0.1I$. Due to the offset this corresponds to setting $Q = I, R = 0.1I, q = -s^\dagger, c = \frac{1}{2}{s^\dagger}^\top Q s^\dagger = 0.5$ in (65). The attacker then solves the Riccati equation with its own loss function and the learner's estimates $\hat{A}$ and $\hat{B}$ to arrive at

the target policy

$$K^\dagger = \begin{bmatrix} -1.31 & 9.99e{-3} & -2.41 & 2.02e{-3} \\ -1.97e{-2} & -1.35 & -1.14e{-2} & -2.42 \end{bmatrix}, k^\dagger = \begin{bmatrix} -0.01 & 1.35 \end{bmatrix}. \quad (84)$$

We run our attack (74)-(80) with $\alpha = 2$ and $\epsilon = 0.01$ in (79). Figure 21 shows the result of our attack. In Figure 21a, we plot the trajectory of the vehicle with policy learned on clean data and poisoned data respectively. Our attack successfully forces LQR into a policy that drives the vehicle close to the target destination. The wiggle on the trajectory is due to the noise w_t of the dynamical system. On the poisoned data, the LQR victim learns the policy

$$\hat{K} = \begin{bmatrix} -1.31 & 9.99e{-3} & -2.41 & 2.02e{-3} \\ -1.97e{-2} & -1.35 & -1.14e{-2} & -2.42 \end{bmatrix}, \hat{k} = \begin{bmatrix} -0.01 & 1.35 \end{bmatrix}, \quad (85)$$

which matches exactly the target policy $K^\dagger, k^\dagger$. In Figure 21b, we show the poisoning on rewards. Our attack leads to very small modification on each reward, thus the attack is efficient. The total attack cost over all 400 items is only $\|\mathbf{r} - \mathbf{r}^0\|_2 = 0.73$, which is tiny small compared to $\|\mathbf{r}^0\|_2 = 112.94$. The results here demonstrate that our attack can dramatically change the behavior of LQR by only slightly modifying the rewards in the dataset.

Finally, for both attacks on TCE and LQR, we note that by setting the attack cost norm $\alpha = 1$ in (52), the attacker is able to obtain a *sparse* attack, meaning that only a small fraction of the batch data needs to be poisoned. Such sparse attacks have profound implications in adversarial machine learning as they can be easier to carry out and harder to detect. We show detailed results in appendix C.5.

5.6 Conclusion

We presented a policy poisoning framework against batch reinforcement learning and control. We showed the attack problem can be formulated as convex optimization. We provided theoretical analysis on attack feasibility and cost. Experiments

show the attack can force the learner into an attacker-chosen target policy while incurring only a small attack cost.

6 SEQUENTIAL ATTACKS ON KALMAN FILTER-BASED FORWARD COLLISION WARNING SYSTEMS

Contribution Statement. This chapter is joint work with Jon Sharp, Ruizhe Wang, Earlence Fernandes and Xiaojin Zhu. The author Yuzhe Ma is the leading author and completed most of the work, including the theoretical analysis and part of the experiments. The simulator used in this chapter was built by Jon and Ruizhe. The paper version of this chapter appeared in AAAI21.

6.1 Introduction

Advanced Driver Assistance Systems (ADAS) are hybrid human-machine systems that are widely deployed on production passenger vehicles (National Highway Traffic Safety Administration, 2020). They use sensing, traditional signal processing and machine learning to detect and raise alerts about unsafe road situations and rely on the human driver to take corrective actions. Popular ADAS examples include Forward Collision Warning (FCW), Adaptive Cruise Control and Autonomous Emergency Braking (AEB).

Although ADAS hybrid systems are designed to increase road safety when drivers are distracted, attackers can negate their benefits by strategically tampering with their behavior. For example, an attacker could convince an FCW or AEB system that there is no imminent collision until it is too late for a human driver to avoid the crash.

We study the robustness of ADAS to attacks. The core of ADAS typically involves tracking the states (e.g., distance and velocity) of road objects using Kalman filter (KF). Downstream logic uses this tracking output to detect unsafe situations before they happen. We focus our efforts on Forward Collision Warning (FCW), a popular ADAS deployed on production vehicles today. FCW uses KF state predictions to detect whether the ego vehicle (vehicle employing the ADAS system) is about to collide with the most important object in front of it and will alert the human driver

in a timely manner. Thus, our concrete attack goal is to trick the KF that FCW uses and make it output incorrect state predictions that would induce false or delayed alerts depending on the specific physical situation.

Recent work has examined the robustness of road object state tracking for autonomous vehicles (Jia et al., 2020). Their attacks create an instantaneous manipulation to the Kalman filter inputs without considering its sequential nature, the downstream logic that depends on filter output, or the physical dynamics of involved vehicles. This leads to temporarily hijacked Kalman filter state predictions that are incapable of ensuring that downstream logic is reliably tricked into producing false alerts. By contrast, we adopt an online planning view of attacking KFs that accounts for: (1) their sequential nature where current predictions depend on past measurements; and (2) the downstream logic that uses KF output to produce warnings. Our attack technique also considers a simplified model of human reaction to manipulated FCW warning lights.

We propose a novel Model Predictive Control (MPC)-based attack that can sequentially manipulate measurement inputs to a KF with the goal of stealthily hijacking its behavior. Our attacks force FCW alerts that mask the true nature of the physical situation involving the vehicles until it is too late for a distracted human driver to take corrective actions.

We evaluate our attack framework by creating a high-fidelity driving simulation using CARLA (Dosovitskiy et al., 2017), a popular tool for autonomous vehicle research and development. We create test scenarios based on real-world driving data (National Highway Traffic Safety Administration, 2011; European New Car Assessment Programme, 2018) and demonstrate the practicality of the attack in causing crashes involving the victim vehicle. Anonymized CARLA simulation videos of our attacks are available at `https://sites.google.com/view/attack-kalman-filter`.

6.2 Background

Forward Collision Warning provides audio-visual alerts to warn human drivers of imminent collisions. Fig. 22 shows the pipeline of a prototypical FCW hybrid

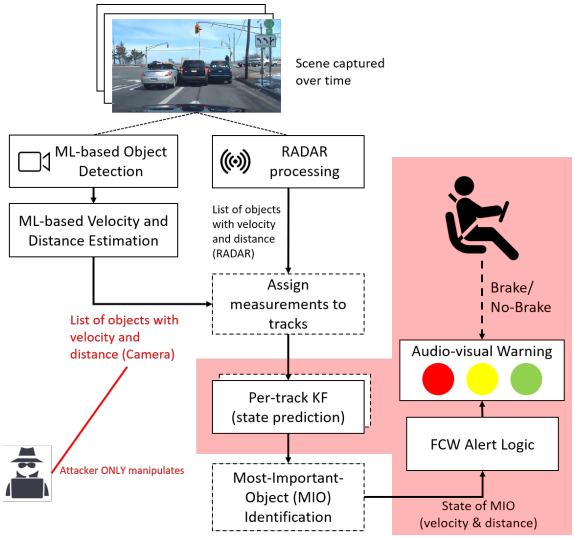


Figure 22: Overview of Forward Collision Warning (FCW) hybrid human-machine system. We take a first step to understanding the robustness of this system to attackers who can compromise sensor measurements. Therefore, we filter the problem to its essence (shaded parts) — the Kalman filter that tracks the most important object (MIO) and the downstream logic that decides how to warn the driver.

system (MATLAB, 2020b): (1) It uses camera and RADAR sensors to perceive the environment; (2) It processes sensor data using a combination of traditional signal processing and machine learning algorithms to derive object velocities and distances; (3) A Kalman filter tracks the Most Important Object (MIO) state and makes predictions about its future states; (4) FCW logic uses Kalman filter predictions to determine whether a collision is about to occur and creates audio-visual warnings; (5) A human driver reacts to FCW alerts. These alerts can be either: green – indicating no danger, yellow – indicating potential danger of forward collision, and red – indicating imminent danger where braking action must be taken.

We focus on attacking the core steps of FCW (shaded parts of Fig. 22). Thus, we assume there is a single MIO in front of the ego vehicle and a single Kalman filter actively tracking its state. The steps of measurement assignment and MIO identification will not be considered in this chapter.

We have two attack goals that will comprehensively demonstrate the vulnerability of FCW hybrid systems — the attacker should trick FCW into showing no red alerts when there is an imminent collision with the most important object (MIO), and vice versa — the attacker should trick FCW into showing red alerts when there is no collision, inducing a human to react with braking that can potentially lead to a rear-end crash with a trailing vehicle.

Kalman Filtering

At the core of FCW is the Kalman Filter, which estimates the state of the MIO based on sensor measurements. In this chapter, the state of the MIO is represented as $x_t = (d_t^1, v_t^1, a_t^1, d_t^2, v_t^2, a_t^2)$, where d_t^1, v_t^1, a_t^1 are the distance, velocity and acceleration of the MIO along the driving direction, and d_t^2, v_t^2, a_t^2 for the lateral direction (perpendicular to driving direction). Then KF models the evolution of x_t as

$$x_{t+1} = Ax_t + \omega_t, t \geqslant 1, \tag{86}$$

where A is the state-transition matrix and $\omega_t \sim N(0, \Omega)$ is Gaussian noise. The underlying state x_t is unknown, but one can obtain measurements y_t of the state as

$$y_t = Cx_t + \psi_t, t \geqslant 1, \tag{87}$$

where C is the measurement matrix and $\psi_t \sim N(0, \Psi)$ is the measurement noise. In our setting, $y_t \in \mathbb{R}^8$ contains vision and radar measurements of the MIO distance and velocity along two directions, i.e., $y_t = (d_t^{1,v}, v_t^{1,v}, d_t^{2,v}, v_t^{2,v}, d_t^{1,r}, v_t^{1,r}, d_t^{2,r}, v_t^{2,r})$, where we use superscripts v, r for vision and radar, and numbers 1, 2 for driving and lateral direction, respectively. Given the state dynamics (86) and measurement model (87), KF provides a recursive formula to estimate the state based on sequential measurements obtained over time. Concretely, KF starts from some initial state and covariance prediction $\hat{x}_1$ and $\hat{\Sigma}_1$. Then for any $t \geqslant 2$, KF first applies (88) to correct the predictions based on measurements y_t. The corrected state and covariance

matrix are denoted by $\bar{x}_t$ and $\bar{\Sigma}_t$.

$$\begin{aligned} \bar{x}_t &= (I - H_{t-1}C)\hat{x}_{t-1} + H_{t-1}y_t, \\ \bar{\Sigma}_t &= (I - H_{t-1}C)\hat{\Sigma}_{t-1}. \end{aligned} \tag{88}$$

where $H_{t-1} = \hat{\Sigma}_{t-1}C^\top(C\hat{\Sigma}_{t-1}C^\top + \Psi)^{-1}$. Next, KF applies (89) to predict state and covariance for the next step.

$$\hat{x}_t = A\bar{x}_t, \quad \hat{\Sigma}_t = A\bar{\Sigma}_t A^\top + \Omega. \tag{89}$$

The correction and prediction steps are applied recursively as t grows. Note that the derivation of covariance matrix is independent of y_t, thus can be computed beforehand.

Warning Alert Logic and Human Model

In this chapter, we analyze an FCW warning alert logic that uses state prediction $\hat{x}_t$ to decide warning lights. Denote $\hat{x}_t = (\hat{d}_t^1, \hat{v}_t^1, \hat{a}_t^1, \hat{d}_t^2, \hat{v}_t^2, \hat{a}_t^2)$, then the warning light ℓ_t output by FCW at step t is one of the following three cases:

- Safe (Green): The MIO is moving away, or the distance to MIO remains constant, i.e., $\hat{v}_t^1 \geqslant 0$.
- Caution (Yellow): The MIO is moving closer, but still at a distance further than the minimum safe distance $d^*(\hat{v}_t^1)$, i.e., $\hat{v}_t^1 < 0$ and $\hat{d}_t^1 > d^*(\hat{v}_t^1)$. We define the safe distance as $d^*(\hat{v}_t^1) = -1.2\hat{v}_t^1 + (\hat{v}_t^1)^2/0.8g$, where g is 9.8 m/s^2.
- Warn (Red): The MIO is moving closer, and at a distance less than the minimum safe distance, i.e., $\hat{v}_t^1 < 0$ and $\hat{d}_t^1 \leqslant d^*(\hat{v}_t^1)$.

The FCW alert logic can be summarized as:

$$F(\hat{x}_t) = \begin{cases} \text{green} & \text{if } \hat{v}_t^1 \geqslant 0, \\ \text{yellow} & \text{if } \hat{v}_t^1 < 0, \hat{d}_t^1 > d^*(\hat{v}_t^1), \\ \text{red} & \text{if } \hat{v}_t^1 < 0, \hat{d}_t^1 \leqslant d^*(\hat{v}_t^1). \end{cases} \tag{90}$$

Given the FCW warning light, the human driver could be in one of the following two states – applying the brake pedal, or not applying/releasing the brake. We take into account human reaction time h^*; warning lights must sustain at least h^* steps before the human driver switches state. That is, the driver brakes after h^* steps since the first red light, and releases the brake after h^* steps since the first yellow/green light. In appendix D.5, we provide an algorithmic description of the human model.

6.3 Attack Problem Formulation

We assume that the attacker has full knowledge of the KF parameters (i.e., white-box attacker). The attacker can directly manipulate measurements (i.e., false data injection), but only pertaining to the vision component, and not the RADAR data. Our attack framework is agnostic of whether the attacker manipulates camera or RADAR, but we choose to only manipulate camera because of the increasing presence of deep learning techniques in ADAS and their general vulnerability to adversarial examples (Szegedy et al., 2013; Eykholt et al., 2018b; Athalye et al., 2017; Sharif et al., 2016). We envision that future work can integrate our results into adversarial example algorithms to create physical attacks.

We further restrict the attacker to only making physically plausible changes to the vision measurements. This is because an anomaly detection system might filter out physically implausible measurements (e.g., change of 10^4m/s over one second). Concretely, we require that the distance and velocity measurement after attack must lie in $[\underline{d}, \bar{d}]$ and $[\underline{v}, \bar{v}]$ respectively. We let $[\underline{d}, \bar{d}] = [0, 75]$ and $[\underline{v}, \bar{v}] = [-30, 30]$. Finally, we assume that at any time step, the attacker knows the true measurement only for

that time step, but does not know future measurements. To address this difficulty of an unknown future, we propose a model predictive control (MPC)-based attack framework that consists of an outer problem and an inner problem, where the inner problem is an instantiation of the outer problem with respect to attacker-envisioned future in every step of MPC. In the following, we first introduce the outer problem formulation.

Outer Attack Problem

Our attacker has a pre-specified target interval $\mathcal{T}^\dagger$, and aims at changing the warning lights output by FCW in $\mathcal{T}^\dagger$. As a result, the human driver sees different lights and takes unsafe actions. Specifically, for any time $t \in \mathcal{T}^\dagger$, the attacker hopes to cause the FCW to output a desired target light $\ell_t^\dagger$, as characterized by (97), in which $F(\cdot)$ is the FCW alert logic (90). To accomplish this, the attacker manipulates measurements in an attack interval $\mathcal{T}^a$. In this chapter, we assume $\mathcal{T}^\dagger \subset \mathcal{T}^a$. Furthermore, we consider only the scenario where $\mathcal{T}^\dagger$ and $\mathcal{T}^a$ have the same last step, since attacking after the target interval is not needed. Let δ_t be the manipulation at step t, and $\tilde{y}_t = y_t + \delta_t$ be measurement after attack. We refer to the i-th component of δ_t as δ_t^i. We next define the attack effort as the cumulative change over measurements $J = \sum_{t\in\mathcal{T}^a} \delta_t^\top R \delta_t$. where $R \succ 0$ is the effort matrix. The attacker hopes to minimize the attack effort.

Meanwhile, the attacker cannot arbitrarily manipulate measurements. We consider two constraints on the manipulation. First, MIO distance and velocity are limited by simple natural physics, as shown in (96). Moreover, similar to the norm ball used in adversarial examples, we impose another constraint that restricts the attacker's manipulation $\|\delta_t\|_\infty \leqslant \Delta$ (see (95)). We refer to $\mathcal{T}^s = \mathcal{T}^a \backslash \mathcal{T}^\dagger$, the difference between $\mathcal{T}^a$ and $\mathcal{T}^\dagger$, as the stealthy (or planning) interval. During $\mathcal{T}^s$, the attacker can induce manipulations before the target interval with advance planning, and by doing so, hopefully better achieve the desired effect in the target interval. However, for the sake of stealthiness, the planned manipulation should not change the original lights in $\mathcal{T}^s$. This can be characterized by the stealthiness constraint (98), where

ℓ_t is the original light.

Given the above, the attack can be formulated as an optimization problem:

$$\begin{aligned}
\min_{\delta_t} \quad & J = \sum_{t \in \mathcal{T}^a} \delta_t^\top R \delta_t, && (91)\\
\text{s.t.} \quad & \tilde{y}_t = y_t + \delta_t, \forall t \in \mathcal{T}^a, && (92)\\
& \tilde{x}_t = A(I - H_{t-1}C)\tilde{x}_{t-1} + AH_{t-1}\tilde{y}_t, && (93)\\
& \delta_t^i = 0, \forall i \in \mathcal{I}_{\text{radar}}, \forall t \in \mathcal{T}^a, && (94)\\
& \|\delta_t\| \leqslant \Delta, \forall t \in \mathcal{T}^a, && (95)\\
& \tilde{d}_t^{1,\nu} \in [\underline{d}, \bar{d}], \tilde{v}_t^{1,\nu} \in [\underline{v}, \bar{v}], \forall t \in \mathcal{T}^a, && (96)\\
& F(\tilde{x}_t) = \ell_t^\dagger, \forall t \in \mathcal{T}^\dagger, && (97)\\
& F(\tilde{x}_t) = \ell_t, \forall t \in \mathcal{T}^s. && (98)
\end{aligned}$$

The constraint (93) specifies the evolution of the state prediction under the attacked measurements $\tilde{y}_t$. (94) enforces no change on radar measurements, where $\mathcal{I}_{\text{radar}} = \{5, 6, 7, 8\}$ contains indexes of all radar components . The attack optimization is hard to solve due to three reasons:

(1). The problem could be non-convex.

(2). The problem could be be infeasible.

(3). The optimization is defined on measurements y_t that are not visible until after $\mathcal{T}^a$, while the attacker must design manipulations δ_t during $\mathcal{T}^a$ in an online manner.

We now explain how to address the above three issues.

The only potential sources of non-convexity in our attack are (97) and (98). We now explain how to derive a surrogate convex problem using $\ell_t^\dagger = \ell_t^o =$ red as an example. The other scenarios are similar, thus we leave the details to Appendix D.2.

The constraint $F(\tilde{x}_t) = \text{red}$ is equivalent to

$$\tilde{v}_t^{1,\nu} < 0, \tag{99}$$

$$\tilde{d}_t^{1,\nu} \leqslant -1.2\tilde{v}_t^{1,\nu} + \frac{1}{0.8g}(\tilde{v}_t^{1,\nu})^2., \tag{100}$$

The above constraints result in non-convex optimzation mainly because (100) is nonlinear. To formulate a convex problem, we now introduce surrogate constraints that are tighter than (99), (100) but guarantee convexity.

Proposition 6.1. *Let* $U(d) = 0.48g - \sqrt{(0.48g)^2 + 0.8gd}$. *Let* $\epsilon > 0$ *be any positive number. Then for any* $d_0 \geqslant 0$, *the surrogate constraints* (101), (102) *are tighter than* $F(\tilde{x}_t) = \text{red}$, *and induce convex attack optimization.*

$$\tilde{v}_t^{1,\nu} \leqslant -\epsilon, \tag{101}$$

$$\tilde{v}_t^{1,\nu} \leqslant U'(d_0)(\tilde{d}_t^{1,\nu} - d_0) + U(d_0) - \epsilon. \tag{102}$$

We provide a proof and guidance on how to select d_0 in Appendix D.2. With the surrogate constraints, the attack optimization becomes convex. However, the surrogate optimization might still be infeasible. To address the feasibility issue, we further introduce slack variables into (101), (102) to allow violation of stealthiness and target lights:

$$\tilde{v}_t^{1,\nu} \leqslant -\epsilon + \xi_t, \tag{103}$$

$$\tilde{v}_t^{1,\nu} \leqslant U'(d_0)(\tilde{d}_t^{1,\nu} - d_0) + U(d_0) - \epsilon + \zeta_t. \tag{104}$$

We include these slack variables in the objective function:

$$J = \underbrace{\sum_{t\in\mathcal{T}^a} \delta_t^\top R \delta_t}_{\text{total manipulation } J_1} + \lambda \underbrace{\sum_{t\in\mathcal{T}^s} (\xi_t^2 + \zeta_t^2)}_{\text{stealthiness violation } J_2} + \lambda \underbrace{\sum_{t\in\mathcal{T}^\dagger} (\xi_t^2 + \zeta_t^2)}_{\text{target violation } J_3}. \tag{105}$$

Then, the surrogate attack optimization is

$$\min_{\delta_t} \quad J = J_1 + \lambda J_2 + \lambda J_3, \tag{106}$$

$$\text{s.t.} \quad \text{(92)-(96), (103), (104).} \tag{107}$$

Proposition 6.2. *The attack optimization* (106)-(107) *with surrogate constraints and slack variables is convex and feasible.*

Inner Attack Problem: MPC-based Attack

In the outer surrogate attack (106)-(107), we need to assume the attacker knows the measurements y_t in the entire attack interval $\mathcal{T}^a$ beforehand. However, the attacker cannot know the future. Instead, he can only observe and manipulate the current measurement in an online manner. To address the unknown future issue, we adopt a control perspective and view the attacker as an adversarial controller of the KF, where the control action is the manipulation δ_t. We then apply MPC, an iterative control method that progressively solves (106)-(107). By using MPC, the attacker is able to adapt the manipulation to the instantiated measurements revealed over time while accounting for unknown future measurements.

Specifically, in each step t, the attacker has observed all past measurements $y_1, ...y_{t-1}$ and the current measurement y_t. Thus, the attacker can infer the clean state $\hat{x}_t$ in the case of no attacker intervention. Based on $\hat{x}_t$, the attacker can recursively predict future measurements by simulating the environmental dynamics without noise, i.e., $\forall \tau > t$:

$$x'_\tau = Ax'_{\tau-1}, \hat{y}_\tau = Cx'_\tau. \tag{108}$$

The recursion starts from $x'_t = \hat{x}_t$. The attacker then replaces the unknown measurements in the outer attack by its prediction $\hat{y}_\tau$ $(\tau > t)$ to derive the following

inner attack:

$$\min_{\delta_{\tau:\tau\geqslant t}} \quad J = \sum_{\tau\in\mathcal{T}^a} \delta_\tau^\top R \delta_\tau + \lambda \sum_{\tau\in\mathcal{T}^a} (\xi_\tau^2 + \zeta_\tau^2), \tag{109}$$

$$\text{s.t.} \quad \tilde{y}_\tau = \hat{y}_\tau + \delta_\tau, \forall \tau \geqslant t, \tag{110}$$

$$(93)\text{-}(96), (103), (104) \text{ (defined on } \tau \geqslant t). \tag{111}$$

The attacker solves the above inner attack in every step t. Assume the solution is $\delta_\tau(\tau \geqslant t)$. Then, the attacker only implements the manipulation on the current measurement, i.e., $\tilde{y}_t = y_t + \delta_t$, and discards the future manipulations. After that, the attacker enters step $t+1$ and applies MPC again to manipulate the next measurement. This procedure continues until the last step of the attack interval $\mathcal{T}^a$. We briefly illustrate the MPC-based attack in algorithm 7.

Algorithm 7 MPC-based attack.

Input: target interval $\mathcal{T}^\dagger$, target lights $\ell_t^\dagger, t \in \mathcal{T}^\dagger$, stealthy interval $\mathcal{T}^s$, original lights $\ell_t, t \in \mathcal{T}^s$.
Initialize $\hat{x}_1$ and $\hat{\Sigma}_1$. Let $\tilde{x}_1 = \hat{x}_1$, $\mathcal{T}^a = \mathcal{T}^s \cup \mathcal{T}^\dagger$.
for $t \leftarrow 2, ..., T$ **do**
  environment generates measurement y_t
  if $t \in \mathcal{T}^a$ **then**
    attacker infers clean state $\hat{x}_t$ without attack.
    attacker predicts future $\hat{y}_t$ with (108)
    attacker solves (109)-(111) to obtain δ_τ $(\tau \geqslant t)$
    attacker manipulates y_t to $\tilde{y}_t = y_t + \delta_t$
    $\tilde{x}_t$ evolves to $\tilde{x}_{t+1}$ according to $\tilde{y}_t$
  else
    $\tilde{x}_t$ evolves to $\tilde{x}_{t+1}$ according y_t.
  end if
end for

6.4 Experiments on CARLA Simulation

In this section, we empirically study the performance of the MPC-based attack. We first describe the simulation setup.

Simulation Setup

We use CARLA (Dosovitskiy et al., 2017), a high-fidelity vehicle simulation environment, to generate measurement data that we input to the Kalman filter-based FCW. CARLA supports configurable sensors and test tracks. We configure the simulated vehicle to contain a single forward-facing RGB camera (800x600 pixels), a forward-facing depth camera of the same resolution, and a single forward-facing RADAR (15° vertical detection range, 6000 points/sec, 85 m maximum detection distance). We took this configuration from a publicly-available FCW implementation (MATLAB, 2020b). The simulation runs at 20 frames/sec and thus, each sensor receives data at that rate. Furthermore, this configuration is commonly available on production vehicles today (Joseph A. Gregor, 2017), and thus, our simulation setup matches real-world FCW systems from a hardware perspective.

For each time step of the simulation, CARLA outputs a single RGB image, a depth map image, and variable number of RADAR points. We use YOLOv2 (Redmon et al., 2016) to produce vehicle bounding boxes, the Hungarian pairwise matching algorithm (Kuhn, 1955) to match boxes between frames, and the first derivative of paired depth map image readings to produce vehicle detections from vision with location and velocity components. Details of processing and formatting of CARLA output can be found in Appendix D.1. This process produces measurements that match ground truth velocity and distance closely.

Although there are infinitely many possible physical situations where an FCW alert could occur involving two vehicles, they reside in a small set of equivalence classes. The National Highway Traffic Safety Administration (NHTSA) has outlined a set of testing conditions for assessing the efficacy of FCW alerts (National Highway Traffic Safety Administration, 2011). It involves a two vehicles on a straight test

track at varying speeds. Based on these real-world testing guidelines, we develop the following two scenarios:

MIO-10: Collision between two moving vehicles

The ego and MIO travel on a straight road, with a negative relative velocity between the two vehicles. Specifically, the ego travels at 27 m/s (~60 mph) and the MIO at 17 m/s (~38 mph). These correspond to typical freeway speed differences of adjacent vehicles. In the absence of any other action, the ego will eventually collide with the MIO. In our simulations, we let this collision occur and record camera and RADAR measurements throughout. Since the relative velocity of the MIO to the ego is -10m/s, we refer to this dataset as MIO-10.

MIO+1: No collision

The ego and MIO travel on a straight road, with a positive relative velocity between the two vehicles. Specifically, the ego travels at 27 m/s (~60 mph) and the MIO at 28 m/s (~63 mph). A trailing vehicle moving at 27 m/s follows the ego 7 m behind. In the absence of any other action, the ego and trailing vehicle will not collide. We collect measurements until the MIO moves out of sensor range of the ego. We refer to this dataset as MIO+1.

The above scenarios correspond to basic situations where the ego vehicle has an unobstructed view of the MIO and represents a best-case for the FCW system. Attacks on these two settings are the hardest to achieve and comprehensively demonstrate the efficacy of our MPC-based attack.

Attack Setup

We perform preprocessing of CARLA measurements to remove outliers and interpolate missing data (see Appendix D.3). Each step of our KF corresponds to one frame of the CARLA simulated video sequence (i.e., 0.05 seconds). We assume that the KF initializes its distance and velocity prediction to the average of the first vision and RADAR measurements. The acceleration is initialized to 0 in both directions.

The covariance matrix is initialized to that used by Matlab FCW (MATLAB, 2020b). Throughout the experiments, we let the effort matrix $R = I$, the margin parameter $\epsilon = 10^{-3}$, and $\lambda = 10^{10}$. We assume the human reaction time is $h^* = 24$ steps (i.e., 1.2 seconds in our simulation).

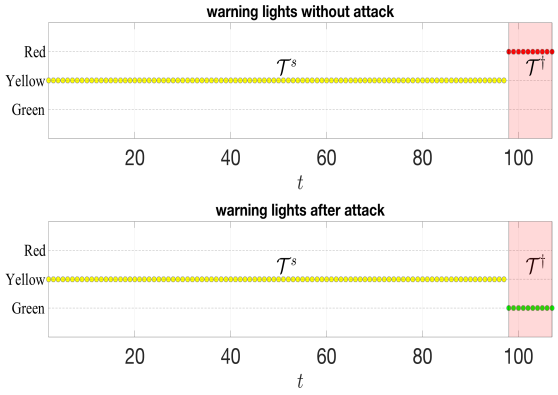
MIO-10 dataset

We first simulate FCW to obtain the original warning lights without attack. The first red light appears at step 98. Before this step, the lights are all yellow. Without attack, the human driver will notice the red warning at step 98. After 1.2 seconds of reaction time (24 steps), the driver will start braking at step 122. The ground-truth distance to the MIO at the first application of brakes is 14.57m. During braking, the distance between the ego vehicle and the MIO reduces by $10^2/0.8g \approx 12.76$m before stabilizing. Since this is less than the ground-truth distance of 14.58m before braking, the crash can be avoided. This validates the potential effectiveness of FCW.

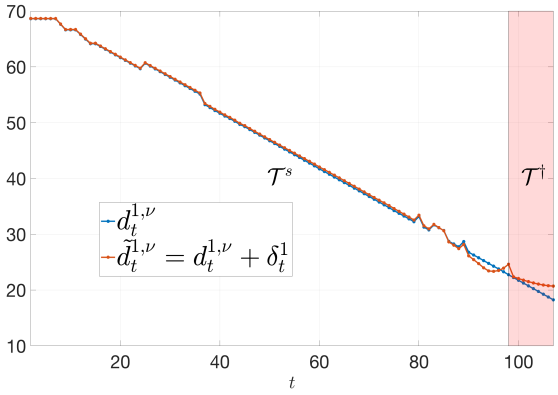
Our attacker aims to cause a crash. To accomplish this, the attacker suppresses the first 10 red warnings, so that the first red warning is delayed to step 108. As a result, the driver starts braking at step 132. The ground-truth distance to MIO at this step is 9.58m, which is below the minimum distance needed to avoid collision (12.76m). As such, a collision will occur. Therefore, we let the target interval be $\mathcal{T}^\dagger = [98, 107]$, and the target lights be $\ell_t^\dagger = \text{green}, \forall t \in \mathcal{T}^\dagger$.

MIO+1 dataset

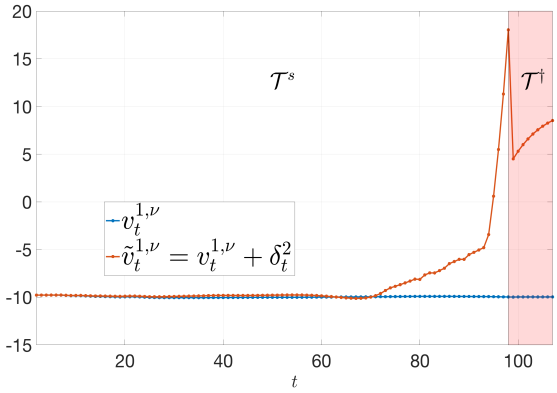
In this scenario, the original warning lights without attack are all green. There is a trailing vehicle 7 m behind the ego vehicle, driving at the same velocity as the ego vehicle. Our attacker aims at causing the FCW to output red lights, so that the ego vehicle suddenly brakes unnecessarily and causes a rear collision with the trailing vehicle. To this end, the attacker changes the green lights in the interval [100, 139] to red, in which case the ego vehicle driver starts braking at step 124, after 1.2 seconds of reaction time. If the warning returns to green at step 140, the driver will react after 1.2 seconds and stop braking at step 164. Therefore, the driver



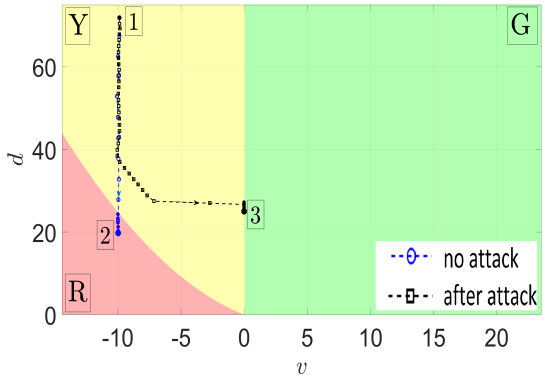
(a) The warning lights.



(b) The manipulation on distance.



(c) The manipulation on velocity.



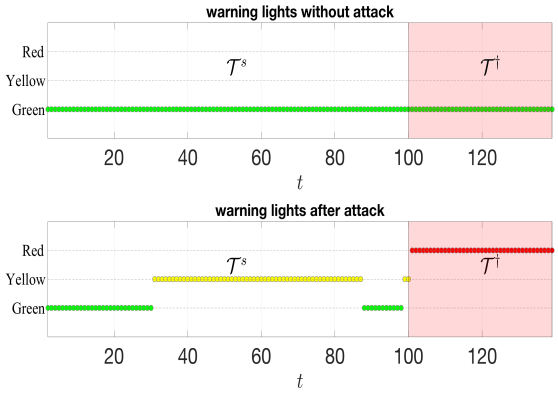
(d) The state trajectory.

Figure 23: Attacks on the MIO-10 dataset.

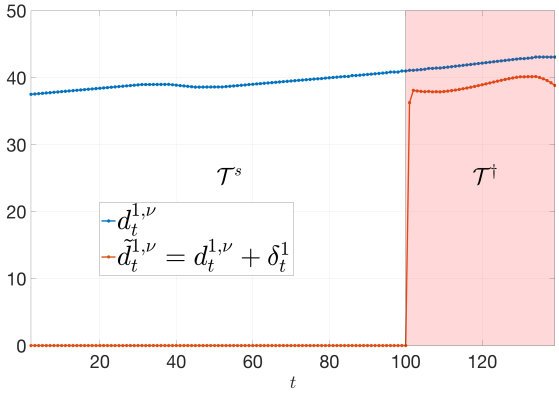
continuously brakes for at least $(164 - 124) \times 0.05 = 2$ seconds. Assuming the driver of the trailing vehicle is distracted, then during those 2 seconds, the distance between the trailing and the ego vehicle reduces by $0.2\text{g} \times 2^2 = 7.84\text{m} > 7\text{m}$, thus causing a rear-collision. Therefore, we let the target interval be $\mathcal{T}^\dagger = [100, 139]$ and the target lights be $\ell_t^\dagger = \text{red}, \forall t \in \mathcal{T}^\dagger$.

The MPC-based Attack Is Successful

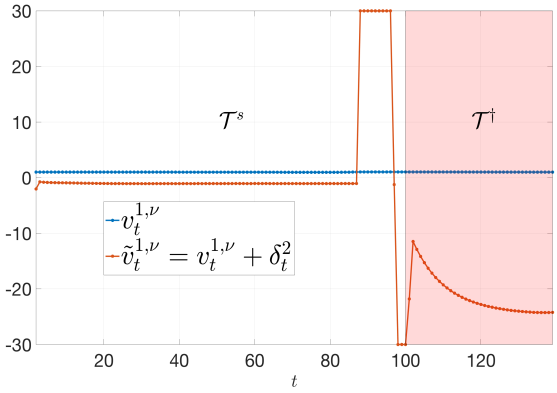
Our first result shows that the MPC-based attack can successfully cause the FCW to output the desired warning lights in the target interval $\mathcal{T}^\dagger$. In this experiment, we let



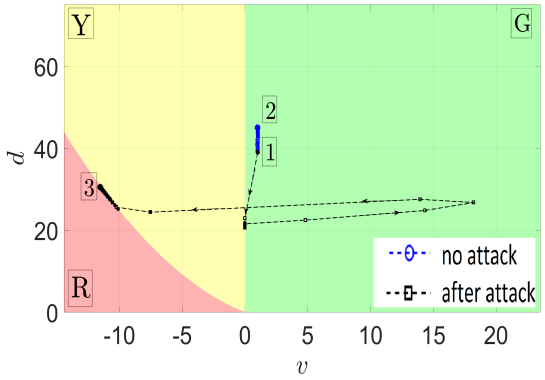
(a) The warning lights.



(b) The manipulation on distance.



(c) The manipulation on velocity.



(d) The state trajectory.

Figure 24: Attacks on the MIO+1 dataset.

$\Delta = \infty$ and the stealthy interval $\mathcal{T}^s$ start at step 2. In Fig. 23a and 24a, we show the warning lights in $\mathcal{T}^\dagger$ (shaded in red). For MIO-10, the attacker achieves the desired red lights in the entire $\mathcal{T}^\dagger$, while maintaining the original yellow lights in $\mathcal{T}^s$. For MIO+1, the attacker failed to achieve the red warning at step 100, but is successful in all later steps. We verified that the attack still leads to a collision. In fact, the attacker can tolerate at most two steps of failure in the beginning of $\mathcal{T}^\dagger$ while still ensuring that the collision occurs. There is an unintended side effect in $\mathcal{T}^s$ where green lights are changed to yellow. However, this side effect is minor since the driver will not brake when yellow lights are produced. In many production vehicles, green and yellow lights are not shown to the driver — only the red warnings are

shown.

In Fig. 23b, 23c, we note that for MIO-10, the manipulation is mostly on velocity, and there are early planned manipulations starting from step 70. A large increase in velocity happens at step 100 (the first step of $\mathcal{T}^\dagger$), which causes the KF's velocity estimation to be positive, resulting in a green light. After that, velocity measurements are further increased to maintain a positive velocity estimation. In Fig. 24b, 24c, we show manipulations on MIO+1. The overall trend is that the attacker reduces the perceived MIO distance and velocity. As a result, KF estimates the MIO to be close than the safe distance in $\mathcal{T}^\dagger$, thus red lights are produced. During interval [88,96], There is an exceptional increase of velocity. We provide a detailed explanation for that increase in Appendix D.4.

In Fig. 23d, 24d, we show the trajectory of KF state prediction projected onto the distance-velocity space during interval $\mathcal{T}^a$. We partition the 2D space into three regions, green (G), yellow (Y) and red (R). Each region contains the states that trigger the corresponding warning light. The trajectory without attack (blue) starts from location 1 and ends at 2. After attack, the trajectory (dark) is steered into the region of the desired warning light, ending at location 3. Note that during $\mathcal{T}^\dagger$, the state after attack lies on the boundary of the desired region. This is because our attack minimizes manipulation effort. Forcing a state deeper into the desired region would require more effort, increasing the attacker's cost.

Attack Is Easier with More Planning Space

Our second result shows that the attack is easier when the attacker has more time to plan, or equivalently, a longer stealthy interval $\mathcal{T}^s$. The stealthy interval is initially of full length, which starts from step 2 until the last step prior to $\mathcal{T}^\dagger$. Then, we gradually reduce the length by $1/4$ of the full length until the interval is empty. This corresponds to 5, 3.75, 2.5, 1.25 and 0 seconds of planning space before the target interval $\mathcal{T}^\dagger$. We denote the number of light violations in $\mathcal{T}^\dagger$ as $V^\dagger = \sum_{t\in\mathcal{T}^\dagger} \tilde{\ell}_t \neq \ell_t^\dagger$, and similarly V^s for $\mathcal{T}^s$. We let $\Delta = \infty$. In Table 2 and 3, we show $V^\dagger$, V^s together with J_1, J_2, J_3 and J as defined in (105) for MIO-10 and MIO+1 respectively. Note that

Table 2: $V^\dagger$, V^s, J_1, J_2, J_3 and J for the MIO-10 dataset.

	MPC-based attack						Greedy attack					
$\mathcal{T}^s$	$V^\dagger$	V^s	J_1	J_2	J_3	J	$V^\dagger$	V^s	J_1	J_2	J_3	J
0	1	0	7.1e3	0	7.4	7.4e10	1	0	4.6e3	98.4	7.4	1.1e12
1.25	0	0	4.4e3	0	0	4.3e3	0	23	1.3e5	3.3e3	0	3.3e13
2.5	0	0	4.4e3	0	0	4.4e3	0	47	2.0e5	5.4e3	0	5.4e13
3.75	0	0	4.4e3	0	0	4.4e3	0	71	2.5e5	7.6e3	0	7.5e13
5	0	0	4.4e3	0	0	4.4e3	0	96	2.9e5	9.2e3	0	9.2e13

Table 3: $V^\dagger$, V^s, J_1, J_2, J_3 and J for the MIO+1 dataset.

	MPC-based attack						Greedy attack					
$\mathcal{T}^s$	$V^\dagger$	V^s	J_1	J_2	J_3	J	$V^\dagger$	V^s	J_1	J_2	J_3	J
0	3	0	3.3e4	0	1.2e2	1.2e12	3	0	1.1e5	0	1.2e2	1.2e12
1.25	1	14	7.6e4	6.8	11.0	1.8e11	0	25	1.7e5	6.1e3	0	6.1e13
2.5	1	39	1.1e5	4.2	6.9	1.1e11	0	49	2.3e5	1.1e4	0	1.1e14
3.75	1	58	1.5e5	3.5	5.9	9.4e10	0	74	3.0e5	1.6e4	0	1.6e14
5	1	58	1.8e5	3.3	5.6	9.0e10	0	98	3.5e5	2.0e4	0	2.0e14

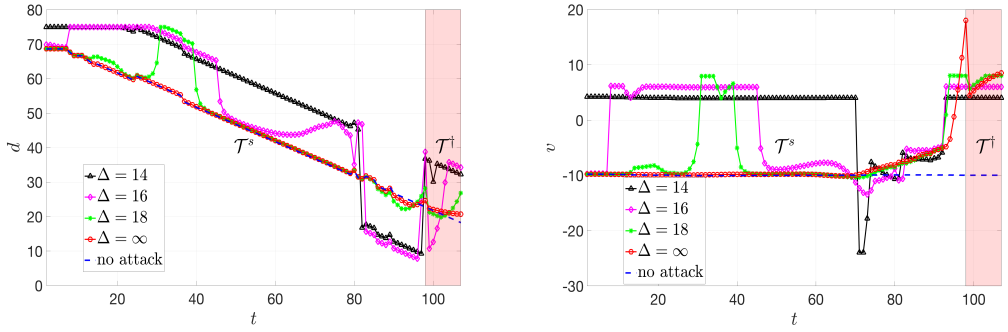
on both datasets, the violation $V^\dagger$ and the total objective J decrease as the length of $\mathcal{T}^s$ grows, showing that the attacker can better accomplish the attack goal given a longer interval of planning.

On MIO-10, when $\mathcal{T}^s$ is empty, the attack fails to achieve the desired warning in all target steps. However, given 1.25s of planning before $\mathcal{T}^\dagger$, the attacker forces the desired lights throughout $\mathcal{T}^\dagger$. Similarly, on MIO+1, when $\mathcal{T}^s$ is empty, the attack fails in the first three steps of $\mathcal{T}^\dagger$, and the collision will not happen. Given 1.25s of planning before $\mathcal{T}^\dagger$, the attack only fails in the first step of $\mathcal{T}^\dagger$, and the collision happens. This demonstrates that planning in $\mathcal{T}^s$ benefits the attack.

Attack Is Easier as Δ Increases

In this section, we show that the attack becomes easier as the upper bound on the manipulation Δ grows. In this experiment, we focus on the MIO-10 dataset and let $\mathcal{T}^\dagger$ start from step 2. In Fig 25, we show the manipulation on measurements for $\Delta = 14, 16, 18$ and ∞. The number of green lights achieved by the attacker in the target interval is 0, 4, 10 and 10 respectively. This shows the attack is easier for larger Δ. Note that for smaller Δ, the attacker's manipulation becomes flatter

due to the constraint $\|\delta_t\| \leqslant \Delta$. But, more interestingly, the attacker needs to start the attack earlier to compensate for the decreasing bound. We also note that the minimum Δ to achieve the desired green lights over the entire target interval (to integer precision) is 18.



(a) Manipulation on distance.

(b) Manipulation on velocity.

Figure 25: Manipulation on measurements with different upper bound Δ. As Δ grows, the attack becomes easier.

Comparison Against Greedy Attacker

In this section, we introduce a greedy baseline attacker. For MIO-10, since the attack goal is to achieve green lights in $\mathcal{T}^\dagger$, the greedy attacker always increases the distance and velocity to the maximum possible value, i.e.,

$$\tilde{d}_t^{1,\nu} = \min\{d_t^{1,\nu} + \Delta, \bar{d}\}, \tilde{v}_t^{1,\nu} = \min\{v_t^{1,\nu} + \Delta, \bar{v}\}, \forall t \in \mathcal{T}^a.$$

Similarly, for MIO+1, the attacker always decreases the distance and velocity to the minimum possible value.

In table 2 and 3, we compare the performance of greedy and our MPC-based attack. On both datasets, each attack strategy achieves a small number of violations $V^\dagger$ in $\mathcal{T}^\dagger$. However, the greedy attack suffers significantly more violations V^s in $\mathcal{T}^s$ than does MPC. Furthermore, these violations are more severe, reflected by the much larger J_2 of the greedy attack. As an example, on MIO+1, the greedy attack changes the original green lights in $\mathcal{T}^s$ to red, while our attack only changes green

to yellow. The greedy attack also results in larger total effort J_1 and objective value J. Therefore, we conclude that our attack outperforms the baseline greedy attack overall. In appendix D.6, we provide more detailed results of the greedy attack.

6.5 Related Work

Attacks on Object Tracking. Recent work has examined the vulnerability of multi-object tracking (MOT) (Jia et al., 2020). Although this work does consider the downstream logic that uses the outputs of ML-based computer vision, our work goes beyond in several ways. First, we consider a hybrid system that involves human and machine components. Second, we consider the more realistic case of sensor fusion involving RADAR and camera measurements that is deployed in production vehicles today. Prior work assumed a system that only uses a single camera sensor. Third, we examine a complete FCW pipeline that uses object tracking data to make predictions about collisions and issues warnings to drivers. Prior work only considered MOT without any further logic that is necessarily present in realistic systems. Finally, our attack algorithm accounts for the sequential nature of decision making in ADAS.

Vision Adversarial Examples. ML models are vulnerable to adversarial examples (Szegedy et al., 2013), with a bulk of research in the computer vision space (Goodfellow et al., 2014; Papernot et al., 2016; Carlini and Wagner, 2017; Shafahi et al., 2018; Chen et al., 2017a). Recent work has demonstrated physical attacks where objects in the real world can be manipulated in ways that cause the models to output wrong decisions (Brown et al., 2017; Athalye et al., 2017; Eykholt et al., 2018a; Sharif et al., 2016). For example, attackers can throw inconspicuous stickers on stop signs and cause the model to output a speed limit sign (Eykholt et al., 2018b). However, all of this work studies the ML model in isolation without taking into account the cyber-physical system that uses model decisions. By contrast, we contribute the first study that examines the security of FCW — a hybrid human-machine system that incorporates machine learning and human behavior. We introduce a control-based attack framework that can account for these aspects while remaining stealthy to

the human driver.

Control-based Attacks on KF. Prior work in control theory has studied false data injection attacks on Kalman filters (Bai et al., 2017; Kung et al., 2016; Zhang and Venkitasubramaniam, 2016; Chen et al., 2016; Yang et al., 2016; Chen et al., 2017b). Our work assumes a similar attack modality – the attacker can induce changes to measurements. However, prior work does not consider the downstream logic and human behavior that depends on KF output. By contrast, we contribute a planning-based attack framework that considers all of these aspects, and we show end-to-end attacks that can cause crashes in distracted driving scenarios.

6.6 Conclusion

We formulate the adversarial attack of Kalman Filter as an optimal control problem, and propose an MPC-based attack algorithm. We demonstrate our attack on FCW, an ADAS that adopts KF to produce warning lights. We show that our attack can manipulate the FCW to output incorrect warnings, which mislead human drivers to behave unsafely and cause crash. Our study incorporates a human behavior model, and is applicable to general machine-human hybrid systems.

7 ADVERSARIAL ATTACKS IN GAMES

Contribution Statement. This chapter is joint work with Young Wu and Xiaojin Zhu. The author Yuzhe Ma is the leading author and completed most of the work, including the theoretical analysis and the experiments. The paper version of this chapter is prepared for submission when the thesis is under construction.

7.1 Introduction

In recent years, there has been a surge of interest in adversarial attacks against sequential decision making learners, such as multi-armed bandit (Jun et al., 2018; Ma et al., 2018; Liu and Lai, 2020; Yang et al., 2021) and reinforcement learning (Ma et al., 2019; Zhang et al., 2020; Liu and Shroff, 2019; Garcelon et al., 2020; Rakhsha et al., 2020). Most prior works consider only a single learning agent that interacts with a fixed underlying environment. However, little is known about attacks in a multi-agent sequential decision making scenario, where agents interact with each other and the reward or state transition depends on the behavior of all the agents together. In reality, multi-agent learning systems are prevalent and have been widely used in different domains, including games (Silver et al., 2017; Vinyals et al., 2019), robotics control (Dudek et al., 1996; Vorotnikov et al., 2018), economics (Mannion et al., 2016; Kutschinski et al., 2003; Zheng et al., 2020), etc. Therefore, it is imperative to understand how these systems could be adversarially manipulated by attackers, which will give insight into designing more robust and defensive multi-agent learning systems.

In this chapter, we take a preliminary step towards understanding the vulnerability of multi-agent sequential decision making in the presence of an attacker. In particular, we formally study a special class of the multi-agent learning scenario – the repeated matrix game with finite horizon. In this game, there are several players who play the same matrix game repeatedly for T rounds. The goal of each player is to gain as much payoff as possible over time, or in other words, minimize the

regret compared to the best action in hindsight. Many real-world examples fall into the class of repeated matrix games, such as multi-round rock-paper-scissors. To investigate potential security issues in these games, we assume an attacker who has the ability to perturb the payoff of the game, and the attack goals are (1) to force the players to play at a pre-specified target action profile for $T - o(T)$ rounds; (2) to keep the total change to the payoffs at $o(T)$. A real-world example is that a nefarious economic practitioner may hope to enforce marketers to not trade with each other, leading to low economic welfare. We point out that while we study the problem from an attack angle, our results also apply for benign goals, e.g., guide the players to take an action profile that is beneficial to the society. For example, in the volunteer game (see section 7.5), a program organizer may hope to encourage the players to volunteer.

A critical concept in games is the Nash equilibrium, which characterizes a set of strategies such that no player can unilaterally deviate from its strategy to gain more payoff. In repeated matrix games, (Auer et al., 2002c) first established the connection between Nash equilibrium and no-regret learners. Specifically, for two-player zero-sum games, if both players apply no-regret algorithms, then the average policy converges to some Nash equilibrium. A more general result for multi-player games is that the empirical distribution of the policies converge to some coarse correlated equilibrium (Fu, 2018). In this regard, our attack is able to shape the equilibrium learned by the players toward a pre-specified target action profile.

Our contributions are summarized as below. (1). We show that for repeated matrix games, an attacker can force the players to play at a target action profile $T - o(T)$ rounds, while incurring only $o(T)$ total change to the payoff. (2).Our attack can shape the equilibrium learned by the players. Specifically, depending on the nature of the game, our attack can either change the Nash equilibrium or the coarse correlated equilibrium toward the target action profile. (3) We empirically evaluate the performance of our attack on two games — the rock-paper-scissors and the volunteer's dilemma, which confirms our theoretical analysis.

7.2 Problem Definition

In this section, we fix some notations. There are M players. The action set of player i is denoted by $\mathcal{A}_i$. In this chapter, we assume $\mathcal{A}_i$ is finite, and we use A_i to represent the size of $\mathcal{A}_i$. The game will repeat T times. The players maintain their own action selection policies $\pi_i^t \in \Delta^{\mathcal{A}_i}$ over time, where $\Delta^{\mathcal{A}_i}$ is the probability simplex over $\mathcal{A}_i$. In each round t, every player i samples an action a_i^t according to strategy π_i^t, which forms a joint action profile $a^t = (a_i^t, ..., a_M^t)$. We use $a_{-i}^t = (a_1^t, ..., a_{i-1}^t, a_{i+1}^t, ..., a_M^t)$ to mean the actions selected by all players except player i at round t. The environment then generates the loss vector $\ell^o(a^t) = (\ell_1^o(a^t), ..., \ell_M^o(a^t))$, where $\ell^o(\cdot)$ is the loss function of the original matrix game and $\ell_i^o(\cdot)$ specifies the loss of player i. We assume $\forall i, a, \ell_i^o(a) \in \mathcal{L}$, where $\mathcal{L}$ is the set of loss values of the game (which is often finite). For example, in rock-paper-scissors game, $\mathcal{L} = \{-1, 0, 1\}$, where -1 means the player wins, 1 means the player loses and 0 is a tie. After $\ell^o(a^t)$ is generated, each player i receives the loss $\ell_i^o(a^t)$ and updates their policy accordingly. Note that each player only observes their own loss but does not see the actions or losses of the other players.

Protocol 8 Attack in repeated matrix game

Knowledge of the attacker: $M, \mathcal{A}_i, ..., \mathcal{A}_M, a^\dagger, \ell^o$, and the regret rate α of the learners
Attack goal: enforce $N^T(a^\dagger) = \Omega(T)$.

1: **for** $t = 1, 2, \ldots, T$ **do**
2: The attacker prepares the loss function $\ell^t(\cdot) = (\ell_1^t(\cdot), ..., \ell_M^t(\cdot))$ for all action profiles, based on the game history $\ell^1, a^1, ..., \ell^{t-1}, a^{t-1}$.
3: The players choose actions $a^t = (a_1^t, ..., a_M^t)$, where $a_i^t \sim \pi_i^t, \forall i \in [M]$.
4: Each player i observes the poisoned loss $\ell_i^t(a^t)$ and updates strategy π_i^t.
5: The attacker incurs attack cost $C(\ell^o, \ell^t, a^t)$.
6: **end for**

Attack Protocol: See Protocol 8. We study an attacker who has the ability to perturb the loss. Specifically, in every round t, the attacker prepares a different loss function $\ell^t(\cdot)$ based on the history of game, i.e., $\ell^1, a^1, ..., \ell^{t-1}, a^{t-1}$. Note that the attacker has to prepare the loss function $\ell^t(\cdot)$ for all action profiles ("cells" in

the payoff matrix), and ℓ^t cannot depend on the current actions a^t: The attacker's prepared loss function ℓ^t has to be committed in the beginning of round t. Also note that we use superscript t to mean the prepared loss function $\ell^t(\cdot)$ can be time-variant. In certain cases, we will omit the superscript if $\ell^t(\cdot)$ is time-invariant. Moreover, it is desirable for the attacker to avoid detection by using the (usually discrete and finite) natural game loss values, i.e., $\ell_i^t(a) \in \mathcal{L}, \forall i, a$. However, we will initially relax this constraint by allowing intermediate loss values in the interval $convex(\mathcal{L})$, and come back to this issue in section 7.4. The players then choose an action profile a^t. As a result, the players observe the poisoned loss $\ell^t(a^t)$ instead of $\ell^o(a^t)$, and they update their policies π_i^t using $\ell^t(a^t)$. Meanwhile, the attacker incurs attack cost $C(\ell^o, \ell^t, a^t)$, whose structure will be discussed below.

Attack Goal: The attacker has two goals simultaneously:

- It has a desired target action profile $a^\dagger$ (which may not coincide with the natural solution concept of the game). The attacker wants to force the players to choose $a^\dagger$ as often as possible. Define $N^T(a) = \sum_{t=1}^T \mathbb{1}[a^t = a]$ to mean the number of rounds where the action profile selected by all players is a. Then this attack goal is to enforce $N^T(a^\dagger) = T - o(T)$.
- The attacker desires small perturbations to the loss function $\ell^o(\cdot)$. We define the attack cost by a non-negative attack cost function $C(\ell^o, \ell^t, a) \geqslant 0$.

The attack cost is subtle and depends on the application. We highlight two possible attack cost functions:

Definition 7.1. *(Attack preparation cost). The attack preparation cost is defined as*

$$C_{prep}(\ell^o, \ell^t) := \sum_a C(\ell^o, \ell^t, a). \tag{112}$$

$$C_{prep}^T = \sum_{t=1}^T C_{prep}(\ell^o, \ell^t). \tag{113}$$

Note that for each round t, the attack preparation cost ignores the identity of a^t but instead measures the total amount of "preparation" the attacker has to do over all potential action profiles.

Remark 7.2. *For any time-invariant loss function $\ell(\cdot)$ that satisfy $C(\ell^o, \ell, a) > 0$ for some a, the attack preparation cost is always linear, i.e., $C^T_{prep} = \Omega(T)$. The attack preparation cost is more unforgiving to the attacker.*

Definition 7.3. *(Attack execution cost). The attack execution cost is defined as*

$$C_{exec}(\ell^o(a^t), \ell^t(a^t)) := C(\ell^o, \ell^t, a^t). \tag{114}$$

$$C^T_{exec} = \sum_{t=1}^{T} C_{exec}(\ell^o(a^t), \ell^t(a^t)). \tag{115}$$

Compared to the attack preparation cost, the attack execution cost measures only the perturbation on the loss of the selected action profile a^t. From now on, we focus on analyzing the attack execution cost.

We also make the following technical assumption on the attack cost function.

Assumption 7.4. *The attack cost function C is η-Lipschitz with respect to the p-norm difference of $\ell(a)$ for some $p \geqslant 1$, i.e,*

$$\forall a, C(\ell^o, \ell^t, a) \leqslant \eta \|\ell^o(a) - \ell^t(a)\|_p. \tag{116}$$

The above assumption is satisfied for commonly used attack cost functions. As an example, suppose $C(\ell^o, \ell^t, a)$ is just the p-norm difference, i.e., $\|\ell^t(a) - \ell^o(a)\|_p$, then the assumption is trivially satisfied with $\eta = 1$. Note that we assumed $\mathcal{L}$ is bounded, thus we can define $L = \min_{x \in \mathcal{L}} x$ and $U = \max_{x \in L} x$. Then we have $\forall i, a, |\ell^t_i(a) - \ell^o_i(a)| \leqslant U - L$, thus $C(\ell^o, \ell^t, a) \leqslant \eta \|\ell^o(a) - \ell^t(a)\|_p \leqslant \eta M^{\frac{1}{p}}(U - L)$, which means C is always bounded. Also note that a direct result of Lipschitzness is that if $\ell^o(a) = \ell^t(a)$, then $C(\ell^o, \ell^t, a) = 0$.

7.3 Assumptions on the Learners: No-Regret Learning

The attacker assumes that the players want to achieve approximate coarse correlated equilibrium (CCE) or approximate Nash equilibrium (NE); and for that the players are each running a no-regret learning algorithm like EXP3P (Bubeck and Cesa-Bianchi, 2012b). It is well-known that for two-player ($M = 2$) zero-sum games, no-regret learners could learn some NE (Blum and Monsour, 2007). More general results suggest that for multi-player ($M \geqslant 2$) general-sum games, no-regret learners can learn some CCE (Fu, 2018). We first define the regret.

Definition 7.5. *(Regret). For any player i, the best-in-hindsight regret with respect to a sequence of loss functions $\ell_i^t(\cdot, a_{-i}^t), t \in [T]$, is defined as*

$$R_i^T = \sum_{t=1}^{T} \ell_i^t(a_i^t, a_{-i}^t) - \min_{a_i \in \mathcal{A}_i} \sum_{t=1}^{T} \ell_i^t(a_i, a_{-i}^t). \tag{117}$$

The expected regret is defined as $\mathbf{E}\left[R_i^T\right]$, where the expectation is with respect to the random selection of actions $a^t, t \in [T]$ over all players.

A few important remarks are in order.

Remark 7.6. *The loss functions $\ell_i^t(\cdot, a_{-i}^t), t \in [T]$ depend on the actions selected by the other players a_{-i}^t, while a_{-i}^t depends on $a^1, ..., a^{t-1}$ of all players in the first $t-1$ rounds. Therefore, $\ell_i^t(\cdot, a_{-i}^t)$ depends on $a_i^1, ..., a_i^{t-1}$. That means, from player i's perspective, it is faced with a non-oblivious (adaptive) adversary (Slivkins, 2019).*

Remark 7.7. *Note that $a_i^* := \arg\min_{a_i \in \mathcal{A}_i} \sum_{t=1}^{T} \ell_i^t(a_i, a_{-i}^t)$ in* (117) *would have meant a baseline in which player i always plays the best-in-hindsight action a_i^* throughout time. Such baseline play should have caused all other rational players to change their plays away from $a_{-i}^1, ..., a_{-i}^T$. However, we are disregarding this fact in defining* (117) *. For this reason,* (117) *is not fully counterfactual, and is called the best-in-hindsight regret in the*

literature (Bubeck and Cesa-Bianchi, 2012a). The same is true when we define expected regret and introduce randomness in player i*'s* a^t*.*

Our key assumption is that the learners achieve sublinear expected regret. This assumption is satisfied by standard bandit algorithms such as EXP3.P (Bubeck and Cesa-Bianchi, 2012b).

Assumption 7.8. *(No-regret Learner) We assume the players apply no-regret learning algorithm that achieves expected regret* $\mathbf{E}\left[R_i^T\right] = O(T^\alpha), \forall i$ *for some* $\alpha \in [0, 1)$*.*

The attacker assumes no-regret learning players because it is a standard way for players to achieve approximate solution concepts such as Nash equilibrium or more generally the coarse correlated equilibrium in repeated matrix games. In particular, there are two standard results. We briefly explain these two results as below without diving into more details.

Remark 7.9. *Let* $\pi^t = (\pi_1^t, ...\pi_M^t)$ *be the joint strategy of all players at round* t*. Then for two-player (*$M = 2$*) zero-sum games (i.e,* $\sum_i \ell_i^o(a) = 0, \forall a$*), no-regret learners guarantee* $\mathbf{E}\left[\frac{1}{T}\sum_t \pi^t\right]$ *converges to some Nash equilibrium.*

Remark 7.10. *Let* $\pi^t = (\pi_1^t, ...\pi_M^t)$ *be the joint strategy of all players at round* t*. Consider the following empirical distribution* D_T*: first draw* $t \sim U([1:T])$*, where* U *is the uniform distribution, and then each player* i *follows strategy* π_i^t*. For multi-player (*$M \geqslant 2$*) general-sum games, no-regret learners guarantee* $\mathbf{E}\left[D_T\right]$ *converges to some coarse correlated equilibrium.*

7.4 Attacking No-regret Learners in a Game

We mentioned earlier that the attacker desires for the poisoned loss values ℓ^t to lie in the natural game value set $\mathcal{L}$ for better stealth. In many games, the $\mathcal{L}$ is a finite discrete set. For example, in rock-paper-scissors, $\mathcal{L}$ contains only three values, indicating three outcomes of the game — win, lose or tie. The discreteness of $\mathcal{L}$ complicates our design of attack algorithms. In this section, we first relax the

discreteness constraint, and allow the loss after attack to take arbitrary continuous value in $\tilde{\mathcal{L}} = [L, U]$, where $L = \min_{x \in \mathcal{L}} x$ and $U = \max_{x \in L} x$. We assume $U > L$, which is satisfied when $\mathcal{L}$ contains at least two distinct values. We will revisit the discrete $\mathcal{L}$ issue in the next section.

With more loss values in $\tilde{\mathcal{L}}$ to choose from, we develop attack algorithms targeting no-regret learners. We consider two scenarios separately – the bounded-away target loss case and boundary target loss case. The former is a simpler scenario to attack, while the latter is more complicated. However, in both cases, efficient attack algorithms exist.

Bounded-away Target Loss

In the first scenario, we have the following assumption on the original loss function.

Assumption 7.11. *(Bounded-away Target Loss). We assume the original loss of the target action profile satisfies* $\exists \rho \in (0, \frac{1}{2}(U - L)], \forall i, \ell_i^o(a^\dagger) \in [L + \rho, U - \rho]$.

This assumption allows the attacker to keep $\ell^o(a^\dagger)$ unchanged (so it does not incur large attack execution cost because eventually $a^\dagger$ should be overwhelmingly played), while leaving room to modify other entries in ℓ^o such that $a^\dagger$ becomes strictly dominant. Our main result is that under assumption 7.11, an attacker can boost $N^T(a^\dagger) = T - O(T^\alpha)$ with $O(T^\alpha)$ attack execution cost. Specifically, our attacker prepares the following time-invariant loss function.

$$\forall i, a, \ell_i(a) = \begin{cases} \ell_i^o(a^\dagger) - (1 - \frac{d(a)}{M})\rho & \text{if } a_i = a_i^\dagger, \\ \ell_i^o(a^\dagger) + \frac{d(a)}{M}\rho & \text{if } a_i \neq a_i^\dagger, \end{cases} \tag{118}$$

where $d(a) = \sum_{j=1}^{M} \mathbb{1}\left[a_j = a_j^\dagger\right]$.

Lemma 7.12. *The attacker loss function* (118) *has the following properties.*

1. $\forall i, a, \ell_i(a) \in \tilde{\mathcal{L}}$, *thus* ℓ *is valid.*
2. *For every player* i, *the target action* $a_i^\dagger$ *strictly dominates any other action by* $(1 - \frac{1}{M})\rho$, *i.e.,* $\ell_i(a_i, a_{-i}) = \ell_i(a_i^\dagger, a_{-i}) + (1 - \frac{1}{M})\rho, \forall i, a_i \neq a_i^\dagger, a_{-i}$.

3. $\ell(a^\dagger) = \ell^o(a^\dagger)$.

4. *If the original loss ℓ^o is zero-sum, then ℓ is also zero-sum.*

Remark 7.13. *The property 3 in Lemma 7.12 is particularly important. Specifically, when the players take the desired target actions $a^\dagger$, the attacker maintains the loss unchanged. That means, the attack execution cost $C_{exec}(\ell^o(a^\dagger), \ell^t(a^\dagger)) = 0$. As we will prove, under attack* (118)*, the players are forced to select $a^\dagger$ in $T - o(T)$rounds. During those rounds, the attack execution cost is always 0, thus the total attack execution cost will be $o(T)$.*

Proof. First note that $\forall i$ and $\forall a$, we have

$$\ell_i(a) \in [\ell_i^o(a^\dagger) - \rho, \ell_i^o(a^\dagger) + \rho] \subseteq [L, U]. \tag{119}$$

Therefore, ℓ is a valid loss function.

$\forall a_{-i}$, let $a = (a_i, a_{-i})$ for some $a_i \neq a_i^\dagger$, and $b = (a_i^\dagger, a_{-i})$, then we have $d(b) = d(a) + 1$, thus

$$\ell_i(a) - \ell_i(b) = \ell_i^o(a^\dagger) + \frac{d(a)}{M}\rho - \ell_i^o(a^\dagger) + (1 - \frac{d(b)}{M})\rho = (1 - \frac{1}{M})\rho. \tag{120}$$

Therefore, the target action $a_i^\dagger$ strictly dominates any other action by $(1 - \frac{1}{M})\rho$.

When $a = a^\dagger$, we have $d(a) = M$, thus by our design, we have $\forall i$,

$$\ell_i(a^\dagger) = \ell_i^o(a^\dagger) - (1 - \frac{d(a)}{M})\rho = \ell_i^o(a^\dagger) - (1 - \frac{M}{M})\rho = \ell_i^o(a^\dagger). \tag{121}$$

Therefore, $\ell(a^\dagger) = \ell^o(a^\dagger)$.

Finally, we prove that if ℓ^o is zero-sum, then ℓ is also zero-sum. To see that, for any a, we sum over all players to obtain

$$\begin{aligned}\sum_{i=1}^{M} \ell_i(a) &= \sum_{i:a_i = a_i^\dagger} \left(\ell_i^o(a^\dagger) - (1 - \frac{d(a)}{M})\rho\right) + \sum_{i:a_i \neq a_i^\dagger} \left(\ell_i^o(a^\dagger) + \frac{d(a)}{M}\rho\right) \\ &= \sum_i \ell_i^o(a^\dagger) - d(a)(1 - \frac{d(a)}{M})\rho + (M - d(a))\frac{d(a)}{M}\rho = \sum_{i=1}^{M} \ell_i^o(a^\dagger) = 0,\end{aligned} \tag{122}$$

where the last equality is due to that the original game is zero-sum. ■

Given Lemma 7.12, we next state our first main result.

Theorem 7.14. *Under assumption 7.11, an attacker that uses loss function* (118) *to perform attack can cause* $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^\alpha)$ *while incurring expected attack execution cost* $\mathbf{E}\left[C^T_{exec}\right] = O(\eta M^{1+\frac{1}{p}} T^\alpha)$.

Proof. Since the attacker perturbs $\ell^o(\cdot)$ to $\ell(\cdot)$, the players are equivalently running no-regret algorithms under the cost ℓ. Note that according to Lemma 7.12, $a_i^\dagger$ is the optimal action for player i, and taking a non-target action results in $(1-\frac{1}{M})\rho$ regret regardless of a_{-i}, thus the expected regret of player i is

$$\mathbf{E}\left[R_i^T\right] = \mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a_i^t \neq a_i^\dagger\right](1-\frac{1}{M})\rho\right] = (1-\frac{1}{M})\rho\left(T - \mathbf{E}\left[N_i^T(a_i^\dagger)\right]\right) \tag{123}$$

Therefore we have,

$$\forall i, \mathbf{E}\left[N_i^T(a_i^\dagger)\right] = T - \frac{M}{(M-1)\rho}\mathbf{E}\left[R_i^T\right] = T - O(\mathbf{E}\left[R_i^T\right]) = T - O(T^\alpha). \tag{124}$$

Therefore, we have

$$\begin{aligned} &T - \mathbf{E}\left[N^T(a^\dagger)\right] = \mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a^t \neq a^\dagger\right]\right] = \mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a_j^t \neq a_j^\dagger \text{ for some } j\right]\right] \\ &\leqslant \mathbf{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{M} \mathbb{1}\left[a_j^t \neq a_j^\dagger\right]\right] = \sum_{j=1}^{M} \mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a_j^t \neq a_j^\dagger\right]\right] \\ &= \sum_{j=1}^{M}\left(T - \mathbf{E}\left[N_j(a_j^\dagger)\right]\right) = O(MT^\alpha). \end{aligned} \tag{125}$$

Thus $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^\alpha)$.

Next we bound the attack cost. Note that $\ell^o(a^\dagger) = \ell(a^\dagger)$, thus when $a^t = a^\dagger$, by our assumption of the attack cost function, we have

$$C_{exec}(\ell^o(a^t), \ell(a^t)) = C(\ell^o, \ell, a^\dagger) = 0 \tag{126}$$

On the other hand, when $a^t \neq a^\dagger$, due to the Lipschitzness of the attack cost function C, we have $C_{exec} \leqslant \eta M^{\frac{1}{p}}(U - L)$. Therefore, the expected attack execution cost is

$$\begin{aligned} \mathbf{E}\left[C_{exec}^T\right] &= \mathbf{E}\left[\sum_{t=1}^{T} C_{exec}(\ell^o(a^t), \ell(a^t))\right] \\ &\leqslant \eta M^{\frac{1}{p}}(U - L)\mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a^t \neq a^\dagger\right]\right] = O(\eta M^{1+\frac{1}{p}} T^\alpha). \end{aligned} \tag{127}$$

where we have reused the result already proved in (125). ■

We have two direct results from Theorem 7.14. First, a standard no-regret algorithm EXP3.P (Bubeck and Cesa-Bianchi, 2012b) achieves $\mathbf{E}\left[R_i^T\right] = O(T^{\frac{1}{2}})$. Therefore, by plugging $\alpha = \frac{1}{2}$ into Theorem 7.14, we have the following first corollary.

Corollary 7.15. *Assume the no-regret learning algorithm is EXP3.P. Then an attacker can cause* $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^{\frac{1}{2}})$ *while incurring expected attack execution cost* $\mathbf{E}\left[C_{exec}^T\right] = O(\eta M^{1+\frac{1}{p}} T^{\frac{1}{2}})$.

Our second corollary shows that if the original loss ℓ^o is zero-sum, then our attacker can mislead the players to believe that $a^\dagger$ is a Nash equilibrium.

Corollary 7.16. *Assume there are two players, i.e.,* $M = 2$*, and the original loss function* $\ell^o(\cdot)$ *is zero-sum. Then under attack* (118)*, the expected averaged policy* $\mathbf{E}\left[\bar{\pi}_i^T\right] = \mathbf{E}\left[\frac{1}{T}\sum_t \pi_i^t\right]$ *converges to a point mass distribution concentrated on* $a_i^\dagger$*, thus the players believe that the pure strategy* $a^\dagger$ *is a Nash equilibrium.*

Proof. For two-player zero-sum games, the players applying no-regret algorithm believe that $\mathbf{E}\left[\bar{\pi}^T\right]$ converges to some Nash equilibrium. Next we prove that $\mathbf{E}\left[\bar{\pi}_i^T\right]$

converges to a point mass distribution concentrated on $a_i^\dagger$. We use $\pi_i^t(a)$ to denote the probability of choosing action a at round t. Then we have

$$\begin{aligned}\mathbf{E}\left[\bar{\pi}_i^T(a_i^\dagger)\right] &= \frac{1}{T}\mathbf{E}\left[\sum_{t=1}^{T}\pi_i^t(a_i^\dagger)\right] = \frac{1}{T}\mathbf{E}\left[\sum_{t=1}^{T}\mathbf{E}\left[\mathbb{1}\left[a_i^t = a_i^\dagger\right]\right]\right] \\ &= \frac{1}{T}\mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a_i^t = a_i^\dagger\right]\right] = \frac{1}{T}\mathbf{E}\left[N_i^T(a_i^\dagger)\right] = \frac{T - O(T^\alpha)}{T} \to 1.\end{aligned} \tag{128}$$

Therefore, the players believe that $a_i^\dagger, i \in [M]$ form a Nash equilibrium. ∎

Boundary Target Loss

When the loss of the target action $\ell^o(a^\dagger)$ hits the boundary of $\tilde{\mathcal{L}}$, the previous time-invariant attack (129) no longer works. In this section, we show that for the boundary target loss scenario, the attacker can still induce $N^T(a^\dagger) = T - o(T)$ while incurring $o(T)$ attack execution cost. The strategy is to use a time-variant loss function. Specifically, let $\epsilon \in (0, 1-\alpha]$ and $\rho_t = t^{\alpha+\epsilon-1}, \forall t \geqslant 1$, then our attacker prepares the following time-variant loss functions.

$$\forall i, a, \ell_i^t(a) = \begin{cases} (1-\rho_t)\ell_i^o(a^\dagger) + \frac{1}{2}(U+L)\rho_t - \frac{1}{2}(U-L)\left(1 - \frac{d(a)}{M}\right)\rho_t & \text{if } a_i = a_i^\dagger, \\ (1-\rho_t)\ell_i^o(a^\dagger) + \frac{1}{2}(U+L)\rho_t + \frac{1}{2}(U-L)\frac{d(a)}{M}\rho_t & \text{if } a_i \neq a_i^\dagger, \end{cases} \tag{129}$$

where $d(a) = \sum_{i=1}^{M}\mathbb{1}\left[a_i = a_i^\dagger\right]$.

Lemma 7.17. *The attacker loss function* (129) *has the following properties.*

1. $\forall i, a, \ell_i^t(a) \in \tilde{\mathcal{L}}$, *thus the loss function is valid.*
2. *For every player* i, *the target action* $a_i^\dagger$ *strictly dominates any other action by* $\frac{1}{2}(U-L)(1-\frac{1}{M})\rho_t$, *i.e.,* $\ell_i(a_i, a_{-i}) \geqslant \ell_i(a_i^\dagger, a_{-i}) + \frac{1}{2}(U-L)(1-\frac{1}{M})\rho_t, \forall i, t, a_i \neq a_i^\dagger, a_{-i}$.
3. $\forall t, C(\ell^o(a^\dagger), \ell^t(a^\dagger)) \leqslant \frac{1}{2}(U-L)\eta M^{\frac{1}{p}}\rho_t$

Proof. Note that $\rho_t \in (0,1]$ and $1 - \frac{d(a)}{M} \leqslant 1$, thus $\forall i$ and $\forall a$, we have

$$\begin{aligned}&(1-\rho_t)\ell_i^o(a^\dagger) + \frac{U+L}{2}\rho_t - \frac{1}{2}(U-L)\left(1-\frac{d(a)}{M}\right)\rho_t \\ &\geqslant (1-\rho_t)L + \frac{U+L}{2}\rho_t - \frac{1}{2}(U-L)\rho_t = L\end{aligned} \tag{130}$$

Also note that $\frac{d(a)}{M} \leqslant 1$, thus

$$\begin{aligned}&(1-\rho_t)\ell_i^o(a^\dagger) + \frac{U+L}{2}\rho_t + \frac{1}{2}(U-L)\frac{d(a)}{M}\rho_t \\ &\leqslant (1-\rho_t)U + \frac{U+L}{2}\rho_t + \frac{1}{2}(U-L)\rho_t = U\end{aligned} \tag{131}$$

Therefore, $\forall i, a, \ell_i^t(a) \in [L, U]$.

Second, $\forall i$ and $\forall a_{-i}$, let $a = (a_i, a_{-i})$ for some $a_i \neq a_i^\dagger$, and let $b = (a_i^\dagger, a_{-i})$, then we have $d(b) = d(a) + 1$, thus one can obtain

$$\ell_i^t(a) - \ell_i^t(b) = \frac{1}{2}(U-L)\frac{d(a)}{M}\rho_t + \frac{1}{2}(U-L)\left(1-\frac{d(b)}{M}\right)\rho_t = \frac{1}{2}(U-L)(1-\frac{1}{M})\rho_t. \tag{132}$$

To see the third property, note that $\ell_i^t(a^\dagger) = (1-\rho_t)\ell_i^o(a^\dagger) + \frac{U+L}{2}\rho_t$, thus

$$\forall i, \ell_i^t(a^\dagger) - \ell^o(a^\dagger) = -\rho_t\left(\ell_i^o(a^\dagger) - \frac{U+L}{2}\right) \tag{133}$$

Note that $|\ell_i^o(a^\dagger) - \frac{U+L}{2}| \leqslant \frac{1}{2}(U-L)$, thus $\|\ell^o(a^\dagger) - \ell^t(a^\dagger)\|_p \leqslant \frac{1}{2}(U-L)M^{\frac{1}{p}}\rho_t$. By the Lipschitzness assumption on C, we have $C(\ell^o(a^\dagger), \ell^t(a^\dagger)) \leqslant \frac{1}{2}(U-L)\eta M^{\frac{1}{p}}\rho_t$.

■

Given Lemma 7.17, we provide our second main result.

Theorem 7.18. *$\forall \epsilon \in (0, 1-\alpha]$, an attacker that uses loss function* (129) *to perform attack can cause* $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^{1-\epsilon})$ *while incurring expected attack execution cost* $\mathbf{E}\left[C_{exec}^T\right] = O(M^{1+\frac{1}{p}}T^{1-\epsilon} + M^{\frac{1}{p}}T^{\alpha+\epsilon})$.

Remark 7.19. *By choosing a larger ϵ in Theorem 7.18, the attacker can increase* $\mathbf{E}\left[N^T(a^\dagger)\right]$. *However, the total attack execution cost can grow. The attack cost attains the minimum order* $O\left(M^{\frac{1}{p}}(1+M)T^{\frac{1+\alpha}{2}}\right)$ *when* $\epsilon = \frac{1-\alpha}{2}$. *The corresponding number of target action selection is* $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^{\frac{1+\alpha}{2}})$

Proof. Under attack, the players are equivalently running no-regret algorithms under loss $\ell^t(\cdot)$ instead of $\ell^o(\cdot)$. Note that according to the second property in Lemma 7.17, $a_i^\dagger$ is the optimal action for player i, and taking a non-target action results in $\frac{1}{2}(U-L)(1-\frac{1}{M})\rho_t$ regret regardless of a_{-i}, thus the expected regret of player i is

$$\begin{aligned}\mathbf{E}\left[R_i^T\right] &= \mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a_i^t \neq a_i^\dagger\right]\frac{1}{2}(U-L)(1-\frac{1}{M})\rho_t\right] \\ &= \frac{1}{2}(U-L)(1-\frac{1}{M})\mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a_i^t \neq a_i^\dagger\right]\rho_t\right]\end{aligned} \tag{134}$$

Now note that $\rho_t = t^{\alpha+\epsilon-1}$ is monotonically decreasing as t grows, thus we have

$$\sum_{t=1}^{T}\mathbb{1}\left[a_i^t \neq a_i^\dagger\right]\rho_t \geqslant \sum_{t=N_i(a_i^\dagger)+1}^{T} t^{\alpha+\epsilon-1} = \sum_{t=1}^{T} t^{\alpha+\epsilon-1} - \sum_{t=1}^{N_i(a_i^\dagger)} t^{\alpha+\epsilon-1} \tag{135}$$

Then note that

$$\sum_{t=1}^{T} t^{\alpha+\epsilon-1} \geqslant \int_{t=1}^{T} t^{\alpha+\epsilon-1} = \frac{1}{\alpha+\epsilon}T^{\alpha+\epsilon} \tag{136}$$

and

$$\sum_{t=1}^{N_i(a_i^\dagger)} t^{\alpha+\epsilon-1} \leqslant \int_{t=0}^{N_i(a_i^\dagger)} t^{\alpha+\epsilon-1} = \frac{1}{\alpha+\epsilon}\left(N_i^T(a_i^\dagger)\right)^{\alpha+\epsilon} \tag{137}$$

Therefore, we have

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{1}\left[a_i^t \neq a_i^\dagger\right] \rho_t &\geqslant \frac{1}{\alpha+\epsilon}\left(T^{\alpha+\epsilon} - \left(N_i^T(a_i^\dagger)\right)^{\alpha+\epsilon}\right) \\ &= \frac{1}{\alpha+\epsilon} T^{\alpha+\epsilon}\left(1 - (1 - \frac{T - N_i^T(a_i^\dagger)}{T})^{\alpha+\epsilon}\right) \\ &\geqslant \frac{1}{\alpha+\epsilon} T^{\alpha+\epsilon} \frac{T - N_i^T(a_i^\dagger)}{T}(\alpha+\epsilon) \\ &= T^{\alpha+\epsilon} - T^{\alpha+\epsilon-1} N_i^T(a_i^\dagger).\end{aligned} \tag{138}$$

Therefore, we have

$$\begin{aligned}\mathbf{E}\left[R_i^T\right] &= \frac{1}{2}(U-L)(1-\frac{1}{M})\mathbf{E}\left[\sum_{t=1}^{T} \mathbb{1}\left[a_i^t \neq a_i^\dagger\right] \rho_t\right] \\ &\geqslant \frac{1}{2}(U-L)(1-\frac{1}{M})\mathbf{E}\left[\left(T^{\alpha+\epsilon} - T^{\alpha+\epsilon-1} N_i^T(a_i^\dagger)\right)\right] \\ &= \frac{1}{2}(U-L)(1-\frac{1}{M})\left(T^{\alpha+\epsilon} - T^{\alpha+\epsilon-1}\mathbf{E}\left[N_i^T(a_i^\dagger)\right]\right)\end{aligned} \tag{139}$$

As a result, we have

$$\forall i, \mathbf{E}\left[N_i^T(a_i^\dagger)\right] \geqslant T - \frac{2M}{(M-1)(U-L)}\mathbf{E}\left[R_i^T\right] T^{1-\alpha-\epsilon} = T - O(T^{1-\epsilon}). \tag{140}$$

By a similar argument to (125), we have $\mathbf{E}\left[N^T(a^\dagger)\right] = T - O(MT^{1-\epsilon})$.

We now analyze the attack execution cost. Note that by the third property in Lemma 7.17, when $a^t = a^\dagger$, $C_{exec}(\ell^o(a^t), \ell^t(a^t)) = C(\ell^o, \ell^t, a^\dagger) \leqslant \frac{1}{2}(U - L)\eta M^{\frac{1}{p}}\rho_t$. On the other hand, when $a^t \neq a^\dagger$, we have $C_{exec}(\ell^o(a^t), \ell^t(a^t)) \leqslant$

$(U-L)\eta M^{\frac{1}{p}}$. Therefore, the expected attack execution cost is

$$\begin{aligned}\mathbf{E}\left[C_{exec}^{T}\right] &\leqslant (U-L)\eta M^{\frac{1}{p}}\mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a^{t}\neq a^{\dagger}\right]\right]+\frac{1}{2}(U-L)\eta M^{\frac{1}{p}}\mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a^{t}=a^{\dagger}\right]\rho_{t}\right]\\ &= (U-L)\eta M^{\frac{1}{p}}(T-\mathbf{E}\left[N^{T}(a^{\dagger})\right])+\frac{1}{2}(U-L)\eta M^{\frac{1}{p}}\sum_{t=1}^{T}\rho_{t},\end{aligned} \tag{141}$$

where $T-\mathbf{E}\left[N^{T}(a^{\dagger})\right]=O(MT^{1-\epsilon})$ as already proved. Also note that

$$\mathbf{E}\left[\sum_{t=1}^{T}\mathbb{1}\left[a^{t}=a^{\dagger}\right]\rho_{t}\right]\leqslant\sum_{t=1}^{T}\rho_{t}=\sum_{t=1}^{T}t^{\alpha+\epsilon-1}\leqslant\int_{t=0}^{T}t^{\alpha+\epsilon-1}=\frac{1}{\alpha+\epsilon}T^{\alpha+\epsilon}. \tag{142}$$

Therefore, we have

$$\begin{aligned}\mathbf{E}\left[C_{exec}^{T}\right] &\leqslant (U-L)\eta M^{\frac{1}{p}}O(MT^{1-\epsilon})+\frac{\eta(U-L)}{2(\alpha+\epsilon)}M^{\frac{1}{p}}T^{\alpha+\epsilon}\\ &= O(M^{1+\frac{1}{p}}T^{1-\epsilon}+M^{\frac{1}{p}}T^{\alpha+\epsilon}).\end{aligned} \tag{143}$$

■

Corollary 7.20. *Assume the no-regret learning algorithm is EXP3.P. Then by picking $\epsilon=\frac{1}{4}$ in Theorem 7.18, an attacker can cause $\mathbf{E}\left[N^{T}(a^{\dagger})\right]=T-O(MT^{\frac{3}{4}})$ while incurring $\mathbf{E}\left[C_{exec}^{T}\right]=O\left(M^{\frac{1}{p}}(1+M)T^{\frac{3}{4}}\right)$ attack cost.*

Attack Subject to Discrete Loss $\mathcal{L}$

In previous sections, we assume the loss can take arbitrary continuous value in the relaxed loss range $\tilde{\mathcal{L}}=[L,U]$. However, there are many real-world situations where continuous loss does not have a natural interpretation. For example, in the rock-paper-scissors game, the loss is interpreted as win, lose or tie, thus $\mathcal{L}$ can only take value in $\{-1,0,1\}$. For such games, we provide a probabilistic attack adapted from (129). We empirically show that in doing so, the attack is still efficient. Specifically, the attacker prepares the following stochastic perturbation on the loss.

$$\forall i, a, \hat{\ell}_i^t(a) = \begin{cases} U & \text{with probability } \frac{\ell_i^t(a)-L}{U-L} \\ L & \text{with probability } \frac{U-\ell_i^t(a)}{U-L}, \end{cases} \tag{144}$$

where $\ell_i^t(a)$ is defined as in (129). Note that since $L \in \mathcal{L}$ and $U \in \mathcal{L}$, the stochastic loss function (144) always produces loss that lies in $\mathcal{L}$; in fact, $\hat{\ell}_i^t(a)$ always lies on the boundary of $\mathcal{L}$.

7.5 Experiments

In this section, we perform empirical evaluations of our attack. Throughout the experiments, we use EXP3.P (Bubeck and Cesa-Bianchi, 2012b) as the no-regret learner. We choose the attack cost function as $C(\ell^o(a), \ell^t(a)) = |\ell^o(a) - \ell^t(a)|$. We investigate two examples of games— the Rock-Paper-Scissors (RPS) game and the Volunteer Dilemma (VD).

The Rock-Paper-Scissors (RPS) Game

In the rock paper scissors game, there are two players, and each player has three actions — rock (R), paper (P), and scissors (S). The loss function takes value in $\mathcal{L} = \{-1, 0, 1\}$. If a player loses, he suffers 1 loss, and the other player gains reward 1 (i.e., suffers -1 loss). If there is a tie, then both players suffer 0 loss. We show the original loss function ℓ^o in table 4, where the entries are the losses for the row and the column player respectively. For now, we relax the discrete set $\mathcal{L}$ to $\tilde{\mathcal{L}} = [-1, 1]$, and allow the loss function after attack to take values in $\tilde{\mathcal{L}}$.

In our first experiment (RPS1), the attacker desires the players to form a tie with Rock-Rock as often as possible, i.e. the target action profile is $a^\dagger = (R, R)$. Note that $\forall i, \ell_i^o(a^\dagger) = 0$ while $L = -1$ and $U = 1$, thus this is the bounded-away target loss attack scenario where the attacker can choose $\rho = 1$. The attacker can use the time-invariant loss function (118) to perform the attack. We consider four different time horizons: $T = 10^4, 10^5, 10^6$, and 10^7. For each T, we run our attack and record the total number of rounds where $a^t \neq a^\dagger$, i.e., $T - N^T(a^\dagger)$, and the total

	R	P	S
R	$0,0$	$1,-1$	$-1,1$
P	$-1,1$	$0,0$	$1,-1$
S	$1,-1$	$-1,1$	$0,0$

Table 4: The original loss function ℓ^o of the rock-paper-scissors game.

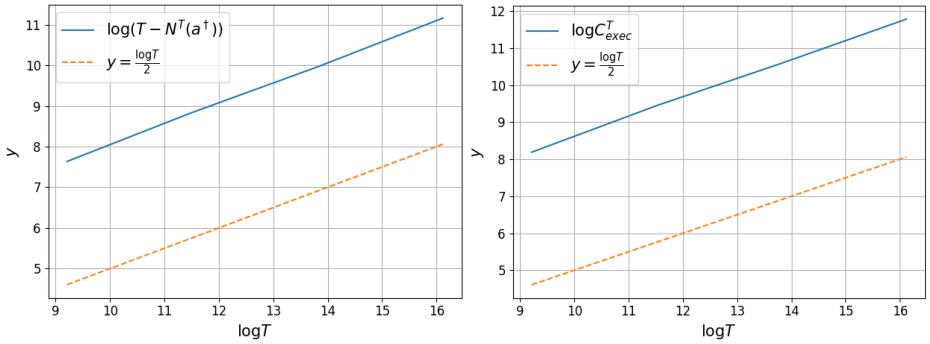
	R	P	S
R	$0,0$	$-0.5,0.5$	$-0.5,0.5$
P	$0.5,-0.5$	$0,0$	$0,0$
S	$0.5,-0.5$	$0,0$	$0,0$

Table 5: RPS1: The poisoned loss function ℓ for target $a^\dagger = (R, R)$ under time-invariant attack (118) with $M = 2, \rho = 1$.

attack execution cost C^T_{exec}. [9]. Note that according to our analysis, $\log(T - N^T(a^\dagger))$ scales as $\frac{1}{2}\log T$. In Figure 26a, we show $\log(T - N^T(a^\dagger))$ as a function of $\log T$, and we plot the line with the anticipated slope $\frac{1}{2}$ for comparison. We observe that the slopes match exactly, which is consistent with our theoretical results. In Figure 26b, we show $\log C^T_{exec}$ as a function of $\log T$. Again, the slope matches the theoretical value $\frac{1}{2}$. For $T = 10^7$, the attack forces $N^T(a^\dagger) = 9.93 \times 10^6$, which is 99.3% of the total rounds. The total attack execution cost is $C^T_e = 1.31 \times 10^5$. On average, each round incurs 0.013 loss perturbation. In table 5, we show the loss function after attack. Note that the loss of the target action profile (R, R) remains the same after attack. Furthermore, (R, R) strictly dominates the other actions by 0.5. We remind the reader that the players do not see the whole table 5 at once before the game; instead, they only experience their own payoff (the ith element) in the selected entries $\ell_i(a^t)$ corresponding to the action profiles a^t played out by their no-regret algorithms over time.

In our second experiment (RPS2), the setting remains the same, but the attacker target action profile becomes $a^\dagger = (R, P)$. That is, the attacker hopes to make the row player play Rock while the column player play Paper as often as possible. Note that now the loss of the target action $\ell^o(a^\dagger) = (1, -1)$ hits the boundary of $\tilde{\mathcal{L}}$, thus the attacker has to apply the time-variant attack. We simulated five different attack ρ_t

[9] Our analysis is about the expected value of $T - N^T(a^\dagger)$ and C^T_e, thus ideally one should run multiple trials for each T and average the statistics to obtain an approximation to $\mathbf{E}\left[T - N^T(a^\dagger)\right]$ and $\mathbf{E}\left[C^T_e\right]$. However, in our experiment, we observe that the random number $T - N^T(a^\dagger)$ and C^T_e both have small variance. Therefore, we only run one trial and report the random numbers instead.

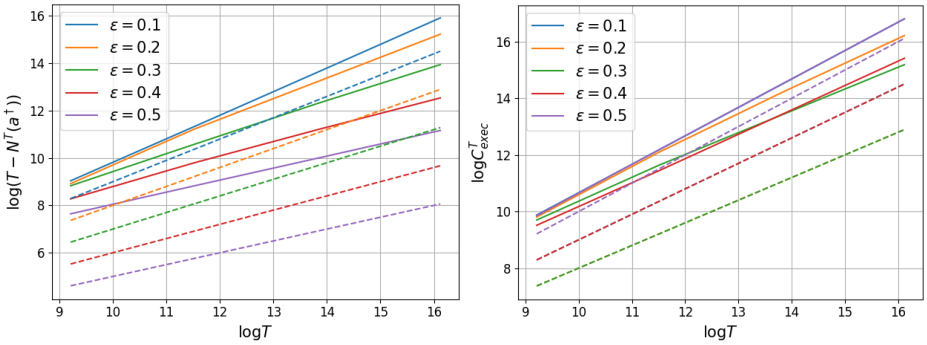


(a) Number of rounds where $a^t \neq a^\dagger$ (b) The attack execution cost.

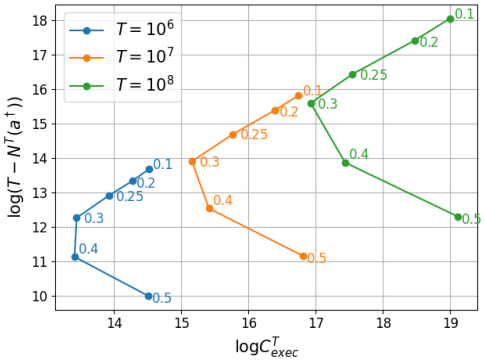
Figure 26: RPS1: $a^\dagger = (R, R)$ time-invariant attacks on RPS.

sequences in (129), corresponding to $\epsilon = 0.1, 0.2, 0.3, 0.4$, and 0.5. In Figure 27a, we show $\log(T - N^T(a^\dagger))$ as a function of $\log T$ for different ϵ with solid lines. According to Theorem 7.18, $\log(T - N^T(a^\dagger))$ scales as $(1-\epsilon)\log T$. To verify this result, we plot $y = (1-\epsilon)x$ for different ϵ with dashed lines in the same color as the corresponding solid line. We see that the slopes are indeed consistent with $1-\epsilon$. In Figure 27b, we show the attack execution cost. The lines for $\epsilon = 0.1$ and $\epsilon = 0.5$ overlap. According to Theorem 7.18, $\log C^T_{exec}$ scales as $\log(T^{\alpha+\epsilon} + T^{1-\epsilon})$, which is dominated by $\max(\alpha+\epsilon, 1-\epsilon)\log T$. To verify this result, we plot $y = \max(\alpha+\epsilon, 1-\epsilon)x$ for different ϵ with dashed lines in the same color as the corresponding solid line. Note that the lines of $\epsilon = 0.2$ and 0.3 overlap, and the lines of $\epsilon = 0.1$ and 0.4 overlap. We see that the slopes are roughly consistent. Also note that for $\epsilon \leqslant 0.3$, the cost reduces as ϵ grows, while for $\epsilon > 0.3$, the cost increases as ϵ grows. This coincides with Theorem 7.18, specifically, the tradeoff between the two terms in the upper bound $O(T^{1-\epsilon} + T^{\alpha+\epsilon})$. However, Theorem 7.18 suggests that the minimum cost is achieved at $\epsilon = \frac{1-\alpha}{2} = 0.25$, while Figure 27b implies that the cost is minimum at some $\epsilon \in (0.3, 0.4)$. We believe the inconsistency is due to not large enough horizon T. For $T = 10^7$, our attack with $\epsilon = 0.3$ enforces $N^T(a^\dagger) = 8.87 \times 10^6$, which is 88.7% percent of the total rounds. The attack execution cost is $C^T_e = 4.00 \times 10^6$. For $\epsilon = 0.4$, the attack enforces $N^T(a^\dagger) = 9.72 \times 10^6$ with execution cost $C^T_e = 4.95 \times 10^6$. In table 6, we show a few loss functions at different round t during the attack. Note that

after attack, the target actions (R, P) strictly dominate the other actions. Besides that, table 6 shows that the dominance gap diminishes as t grows. This is especially due to the third property in Lemma 7.17, where ρ_t decreases monotonically. In Figure 27c, we further investigate the inconsistency that the attack execution cost achieves the minimum at some $\epsilon \in (0.3, 0.4)$ rather than $\epsilon^* = 0.25$. We let $T = 10^6, 10^7, 10^8$, and $\epsilon = 0.1, 0.2, 0.25, 0.3, 0.4, 0.5$. For each T, we plot $\log N^T(a^\dagger)$ against $\log C^T_{exec}$ and we marked out the corresponding ϵ values on the curve. Note that for different T, the pattern remains the same — as ϵ grows, $\log N^T(a^\dagger)$ increases monotonically, while $\log C^T_{exec}$ first reduces and then increases. We also note that as T becomes larger, the ϵ with the minimum attack cost is closer to the desired $\epsilon^* = 0.25$.



(a) Number of rounds where $a^t \neq a^\dagger$ (b) The attack execution cost.



(c) $\log(T - N^T(a^\dagger))$ against $\log C^T_{exec}$.

Figure 27: RPS2: $a^\dagger = (R, P)$ time-variant attacks on RPS.

	R	P	S
R	$-0.5, 0.5$	$0, 0$	$-0.5, 0.5$
P	$0, 0$	$0.5, -0.5$	$0, 0$
S	$0, 0$	$0.5, -0.5$	$0, 0$

(a) ℓ^1.

	R	P	S
R	$0.4, -0.4$	$0.6, -0.6$	$0.4, -0.4$
P	$0.6, -0.6$	$0.8, -0.8$	$0.6, -0.6$
S	$0.6, -0.6$	$0.8, -0.8$	$0.6, -0.6$

(b) ℓ^{10}.

	R	P	S
R	$0.91, -0.91$	$0.94, -0.94$	$0.91, -0.91$
P	$0.94, -0.94$	$0.97, -0.97$	$0.94, -0.94$
S	$0.94, -0.94$	$0.97, -0.97$	$0.94, -0.94$

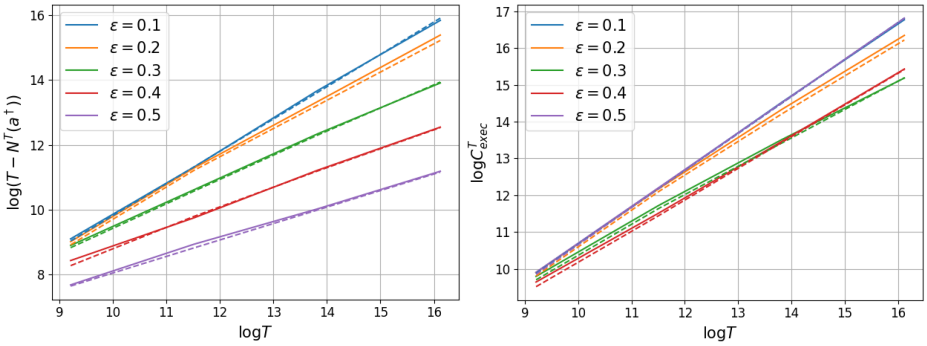
(c) ℓ^{1000}.

Table 6: RPS2: The attack loss functions ℓ^t for selected t (with $\epsilon = 0.3$). Note the target entry $a^\dagger = (R, P)$ converges toward (1,-1).

In the third experiment (RPS3), we compare the performance of the stochastic attack (144) against the original non-stochastic version (129). Again, $a^\dagger = (R, P)$. Recall the purpose of stochastic attacks is to use natural game loss values in $\mathcal{L}$ to make the attack less detectable. Thus we hope to show that the stochastic attacks do not lose too much potency compared to the non-stochastic attacks (which had to use unnatural loss values as in RPS1 and RPS2). In Figure 28, we show the number of non-target action selections and the total attack execution cost for the stochastic attack. We plot the results of the original non-stochastic version in dashed lines for comparison. Note that the two attacks have almost identical performance in terms of $N^T(a^\dagger)$, but the stochastic attack may incur slightly larger attack execution cost, e.g., for $\epsilon = 0.2$. Overall, though, stochastic attacks did not lose much and may be preferred by attackers.

The Volunteer Dilemma (VD)

Our second example is the volunteer's dilemma (Wikipedia contributors, 2021). There are $M \geqslant 2$ players, and each player has two actions — volunteer or defect. When there exists at least one volunteer, those players who do not volunteer gain benefit. In our case, we assume the benefit is 1 as in (Wikipedia contributors, 2021), which can be interpreted as -1 loss. The volunteers, however, receive no payoff. On the other hand, if no player volunteers, then every player suffers a large penalization. We assume the penalization (or loss) is 10 as in (Wikipedia contributors, 2021). We show the loss function for an individual player i in table 7.



(a) Number of rounds where $a^t \neq a^\dagger$ (b) The attack execution cost.

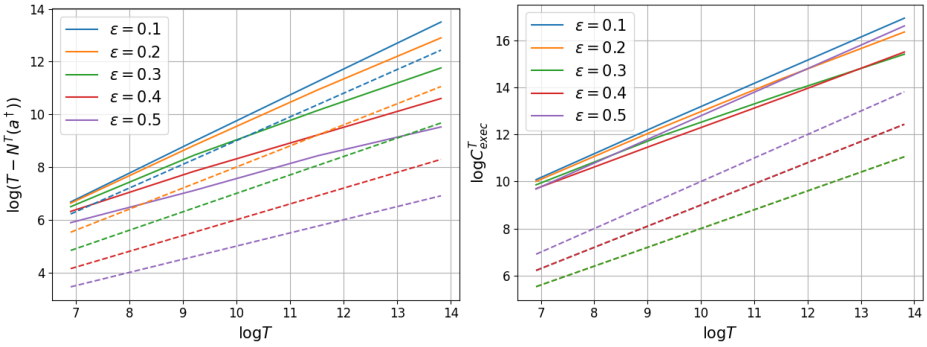
Figure 28: RPS3: Time-variant stochastic attack for $a^\dagger = (R, P)$ with natural loss values in $\mathcal{L}$. The dashed lines are the corresponding non-stochastic attacks with unnatural loss values in RPS2.

		Other players	
		exists some volunteer	no volunteer exists
Player i	volunteer	0	0
	defect	-1	10

Table 7: The loss function ℓ^o_i for individual player i in the volunteer dilemma.

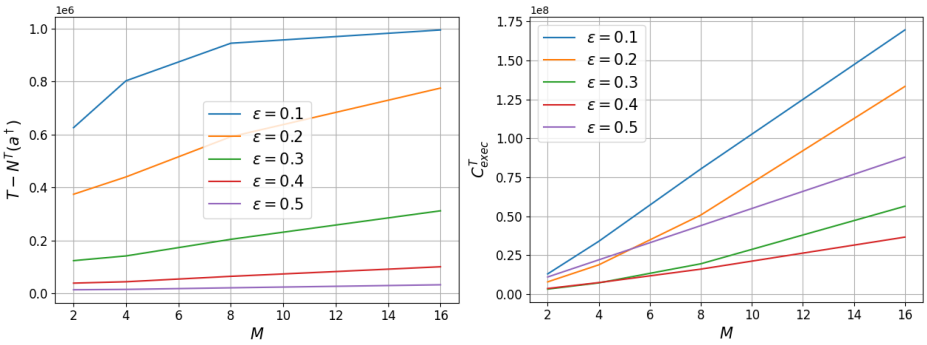
The attacker aims at coaxing no volunteers, i.e., the target action $a^\dagger_i$ is defect for any player i. Note that $\forall i, \ell^o_i(a^\dagger) = 10$, which achieves the upper bound of $\mathcal{L}$. Therefore, the attacker needs to apply the time-variant attack (129). We experimented with $M = 3$, $T = 10^3, 10^4, 10^5, 10^6$ and $\epsilon = 0.1, 0.2, 0.3, 0.4, 0.5$ in this experiment. In Figure 29, we show the number of rounds where $a^t \neq a^\dagger$ and the total execution cost respectively. Note that the slope for $\log(T - N^T(a^\dagger))$ matches $1 - \epsilon$, and the slope of $\log C^T_{exec}$ roughly matches $\max(\alpha + \epsilon, 1 - \epsilon)$.

We now study how the number of players M influences the attack performance. For the VD game, we fix $T = 10^6$, and rerun the experiments for $M = 2, 4, 8, 16$ players with different ϵ. Figure 30 shows $T - N^T(a^\dagger)$ and the attack execution cost C^T_{exec}. As M grows, $N^T(a^\dagger)$ decreases while the execution cost C^T_{exec} increases. This result is consistent with Theorem 7.18.



(a) Number of rounds where $a^t \neq a^\dagger$ (b) The attack execution cost.

Figure 29: Time-variant attack on VD ($M = 3$).



(a) Number of rounds where $a^t \neq a^\dagger$ (b) The attack execution cost.

Figure 30: As the number of player M grows, $N^T(a^\dagger)$ decreases and C^T_{exec} grows.

7.6 Conclusion

In this chapter, we designed attacks against no-regret game players. We show that an attacker can force all players to select a target action profile in $T - o(T)$ rounds, while incurring only $o(T)$ attack execution cost. There are some future research questions: (1) How to design attacks in more complicated multi-agent learning scenarios where the players adopt real game-theoretic behaviors, such as stochastic Markov games. (2) How to design defense mechanisms against our attack.

8 CONCLUSIONS AND FUTURE WORK

In this thesis, we provided a systematic study on adversarial attacks against several classic sequential decision making and control systems, and we demonstrated both theoretically and empirically that these systems are susceptible to reward-manipulation attacks — a type of security threat that directly perturbs the reward feedback channel. We distill the following key principle in designing efficient attacks that apply to most sequential decision making systems.

Principle of Attack: assume the goal of the attacker is to enforce a target action $a^\dagger$ on the victim agent, then the attacker should use the following two principles to design attack.

1. Do "not" (or just slightly) perturb the reward whenever the agent selects the target action $a^\dagger$.

2. When the agent selects an action $a \neq a^\dagger$, manipulate the reward to ensure that after attack, $a^\dagger$ appears to be better than a.

The rationality behind the above the attack design is the following. By the second principle, after attack, the target action $a^\dagger$ becomes the optimal action to the agent, thus a reasonable agent should take $a^\dagger$ very often, e.g., in $T - o(T)$ rounds. In other words, the agent fails to take $a^\dagger$ in only $o(T)$ rounds. Then by the first principle, the attacker incurs significant reward manipulation only when the agent fails to take the target action, which happens in $o(T)$ rounds as we have argued. Therefore, the cumulative reward perturbation (i.e, attack cost) is $o(T)$.

In the following, we envision some research problems to study in the future. One interesting question is to study attacks in the multi-agent sequential decision making scenario. There are some new considerations in this case. First, the agents in the multi-agent scenario usually have game-theoretic reasoning ability. That means, each agent will reason about other agents' response before making a decision. To characterize the state of an agent, we may need to include the whole sequential-decision making history. As a result, the policy set becomes much richer

than the single-agent scenario. Therefore, the attacker is faced with a much more complicated learning system. Second, the attacker might only be able to manipulate a small group of agents (e.g., ϵ fraction of agents). A defender can exploit this fact to design effective defense mechanisms. This is a similar consideration to the ϵ-fraction data corruption in the robust statistics literature, although the corruption we consider here is at the agent level rather than the data level. Finally, it might be harder for the attacker to evade detection in the multi-agent setting. For example, in collaborative multi-agent learning, the agents can communicate with each other and diagnose the system together.

The other interesting question is how to design effective defense mechanisms to mitigate the effect of attack. In the single-agent setting, the primary method is to use robust statistics for policy update. This method has been substantially studied in multi-armed bandits (Guan et al., 2020; Niss and Tewari, 2019) and reinforcement learning (Zhang et al., 2021b,a). Some other defense methods include (Banihashem et al., 2021; Lu et al., 2021; Zhong et al., 2021; Ding et al., 2021). However, how to design defense in the multi-agent setting remains an under-explored research problem. If the attacker can only manipulate ϵ fraction of agents, one potential defense approach is to adapt existing methods in robust statistics.

In this thesis, we proved theoretical upper bound on the attack cost (or effort) for several learners such as stochastic bandit, batch reinforcement learning and no-regret game players. It remains unclear whether our bound is also necessary. In (Zuo, 2020), the authors provided attack cost lower bounds for concrete bandit algorithms, including ϵ-greedy and UCB. However, there is no general lower bound for broader class of sequential decision making algorithms, which is an interesting theoretical question to study.

Finally, it is important to demonstrate the attack and defense algorithms on (simulated) real-world sequential decision making systems, including online recommender system, movie rating system, dialogue generation system, autonomous driving system, gaming system, among others. It is also imperative to construct standard datasets and benchmarks for researchers to evaluate and compare the performance of different attacks and defenses. During attack implementation, prac-

titioners should pay special attention to the following aspects — how frequent the attack is, can the attack evade detections, how much perturbation the attack introduces to the system, and how to exploit these observations to guide the design of practical defense mechanisms.

A APPENDIX FOR ADVERSARIAL ATTACKS ON STOCHASTIC BANDITS

A.1 Details on the oracle and constant attack

Logarithmic regret and the suboptimal arm pull counts. For simplicity, denote by i^* the *unique* best arm; that is, $i^* = \arg\max_{i=1,\dots,K} \mu_i$. We show that a logarithmic regret bound implies that the arm pull count of arm $i \neq i^*$ is at most logarithmic in T.

Lemma A.1. *Assume that a bandit algorithm enjoys a regret bound of* $O(\log(T))$. *Then,* $\mathbb{E}N_i(T) = O(\log(T))$, $\forall i \neq i^*$.

Proof. The logarithmic regret bound implies that for a large enough T there exists $C > 0$ such that $\sum_{i=1}^{K} \mathbb{E}N_i(T)(\mu_{i^*} - \mu_i) \leqslant C \log T$. Therefore, for any $i \neq i^*$, we have $\mathbb{E}N_i(T)(\mu_{i^*} - \mu_i) \leqslant C \log T$, which implies that

$$\mathbb{E}N_i(T) \leqslant \frac{C}{\mu_{i^*} - \mu_i} \log T = O(\log T)\,.$$

■

Proof of Proposition 2.1 By Lemma A.1, a logarithmic regret bound implies that the bandit algorithm satisfies $\mathbb{E}N_i(T) = O(\log(T))$. That is, for a large enough T, $\mathbb{E}N_i(T) \leqslant C_i \log(T)$ for some $C_i > 0$. Based on the view that the oracle attack effectively shifts the means $\mu_1, \cdots, \mu_K$, the best arm is now the K-th arm. Then, $\mathbb{E}N_K(T) = T - \sum_{i \neq K} \mathbb{E}N_i(T) \geqslant T - \sum_{i \neq K} C_i \log T = T - o(T)$, which proves the first statement.

For the second statement, we notice that $\mathbb{E}N_i(T) = C_i \log T$ for any $i \neq K$ and that we do not attack the K-th arm. Therefore,

$$\mathbb{E}\left[\sum_{t=1}^{T} |\alpha_t|\right] = \sum_{i=1}^{K-1} \mathbb{E}N_i(T) \cdot \Delta_i^{\epsilon} \leqslant \sum_{i=1}^{K-1} C_i \Delta_i^{\epsilon} \log T = O\left(\sum_{i=1}^{K-1} \Delta_i^{\epsilon} \log T\right)\,.$$

Proof of Proposition 2.2 By Lemma A.1, a logarithmic regret bound implies that the bandit algorithm satisfies $\mathbb{E}N_i(T) = O(\log(T))$. Note that the constant attack effectively shifts the means of all the arms by A' except for the K-th arm. Since $A' > \max_i \Delta_i$, the best arm is now the K-th arm. Then, $\mathbb{E}N_K(T) = T - \sum_{i=1}^{K-1} \mathbb{E}N_i(T) \geqslant T - \sum_{i=1}^{K-1} C_i \log T = T - o(T)$, which proves the first statement.

For the second statement, we notice that $\mathbb{E}N_i(T) = C_i \log T$ for any $i \neq K$, and we do not attack the K-th arm. Therefore,

$$\mathbb{E}\left[\sum_{t=1}^{T} |\alpha_t|\right] = \sum_{i=1}^{K-1} \mathbb{E}N_i(T) \cdot A' \leqslant A' \sum_{i=1}^{K-1} C_i \log T = O(A' \cdot \log T)\,.$$

The best ϵ for Alice's oracle attack Consider the case where Bob employs a near-optimal bandit algorithm such as UCB (Auer et al., 2002a), which enjoys $\mathbb{E}N_i(T) = \Theta(1 + \Delta_i^{-2} \log T)$. When the time horizon T is known ahead of time, one can compute the best ϵ ahead of time. Hereafter, we omit unimportant constants for simplicity. Since Alice employs the oracle attack, Bob pulls each arm $C + \epsilon^{-2} \log(T)$ times for some $C > 0$ in expectation. Assuming that the target arm is K, the attack cost is

$$\sum_{i=1}^{K-1} \Delta_i^{\epsilon} \cdot (C + \epsilon^{-2} \log(T)) = C \sum_{i=1}^{K-1} \Delta_i + (K-1)C \cdot \epsilon + \sum_{i=1}^{K-1} \left(\frac{\Delta_i}{\epsilon^2} + \frac{1}{\epsilon}\right) \log T$$

To balance the two terms, one can see that ϵ has to grow with T and the term Δ_i/ϵ^2 is soon dominated by $1/\epsilon$. Thus, for large enough T the optimal choice of ϵ is $\sqrt{C \log(T)}$, which leads to the attack cost of $O(K\sqrt{C \log T})$.

A.2 Details on attacking the ϵ-greedy strategy

Lemma A.2. *For $\delta \leqslant 1/2$, the $\beta(N)$ defined in (2) is monotonically decreasing in N.*

Proof. It suffices to show that $f(x) = \frac{2\sigma^2}{x} \log \frac{\pi^2 K x^2}{3\delta}$ is decreasing for $x \geqslant 1$. Note that

$\delta \leqslant 1/2 \leqslant \frac{K}{3}(\frac{\pi}{e})^2$, thus for $x \geqslant 1$ we have

$$\begin{aligned} f'(x) &= -\frac{2\sigma^2}{x^2}\log\frac{\pi^2 K x^2}{3\delta} + \frac{2\sigma^2}{x}\frac{3\delta}{\pi^2 K x^2}\frac{2\pi^2 K x}{3\delta} \\ &= \frac{2\sigma^2}{x^2}(2 - \log\frac{\pi^2 K x^2}{3\delta}) \leqslant \frac{2\sigma^2}{x^2}(2 - \log\frac{\pi^2 K}{3\delta}) \\ &\leqslant \frac{2\sigma^2}{x^2}(2 - \log e^2) = 0. \end{aligned}$$

■

Proof of Corollary 2.2 When T is larger than the following threshold:

$$\frac{K+1}{K}(\sum_{t=1}^{T}\epsilon_t) + \sqrt{12\log(K/\delta)(\frac{K+1}{K}\sum_{t=1}^{T}\epsilon_t)},$$

we have $\widetilde{N}_K(T) \geqslant \widetilde{N}(T)$. Because $\beta(N)$ is decreasing in N,

$$\widetilde{N}(T)\beta(\widetilde{N}(T)) + 3\widetilde{N}(T)\beta(\widetilde{N}_K(T)) \leqslant 4\widetilde{N}(T)\beta(\widetilde{N}(T)). \tag{145}$$

Due to the the exploration scheme of the strategy,

$$\sum_{t=1}^{T}\epsilon_t = cK\sum_{t=1}^{T}1/t \leqslant cK(\log(T)+1).$$

Thus by the definition of $\tilde{N}(T)$,

$$\tilde{N}(T) \leqslant c(\log T + 1) + \sqrt{3\log\left(\frac{K}{\delta}\right)c(\log T + 1)}.$$

For sufficiently large T, there exists a constant c_2 depending on c, K, δ to further upper bound the RHS as follows:

$$c(\log T + 1) + \sqrt{3 \log\left(\frac{K}{\delta}\right) c(\log T + 1)} \leqslant c_2 \log T := \check{N}(T). \tag{146}$$

Since $N\beta(N)$ is increasing in N, combining (145) and (146) we have for sufficiently large T,

$$\widetilde{N}(T)\beta(\widetilde{N}(T)) + 3\widetilde{N}(T)\beta(\widetilde{N}_K(T)) \leqslant 4\check{N}(T)\beta(\check{N}(T)).$$

Plugging this upper bound into Theorem 2.1,

$$\begin{aligned}\sum_{t=1}^{T} \alpha_t &< \left(\sum_{i=1}^{K} \Delta_i\right) \check{N}(T) + 4(K-1)\check{N}(T)\beta(\check{N}(T)) \\ &= c_2 \left(\sum_{i=1}^{K} \Delta_i\right) \log T + \sqrt{32c_2}(K-1)\sigma \cdot \sqrt{\log T \left(2 \log\log T + \log \frac{\pi^2 K c_2^2}{3\delta}\right)}\end{aligned} \tag{147}$$

Proof of Lemma 2.3 Let $\{X_j\}_{j=1}^{\infty}$ be a sequence of *i.i.d.* σ^2-sub-Gaussian random variables with mean μ. Let $\hat{\mu}_N^0 = \frac{1}{N}\sum_{j=1}^{N} X_j$. By Hoeffding's inequality

$$\mathbb{P}(|\hat{\mu}_N^0 - \mu| \geqslant \eta) \leqslant 2\exp\left(-\frac{N\eta^2}{2\sigma^2}\right).$$

Define $\delta_{iN} := \frac{6\delta}{\pi^2 N^2 K}$. Apply union bound over arms i and pull counts $N \in \mathbb{N}$,

$$\mathbb{P}\left(\exists i, N : |\hat{\mu}_{i,N}^0 - \mu_i| \geqslant \beta(N)\right) \leqslant \sum_{i=1}^{K}\sum_{N=1}^{\infty} \delta_{iN} = \delta.$$

Proof of Lemma 2.4 We show by induction that at the end of any round $t \geqslant K$ Algorithm 1 maintains the invariance

$$\hat{\mu}_K(t) > \hat{\mu}_i(t), \quad \forall i < K, \tag{148}$$

which forces the learner to pull arm K if $t+1$ is an exploitation round.

Base case: By definition the learner pulls arm K first, then all the other arms once. During round $t = 2 \dots K$ the attack algorithm ensures $\hat{\mu}_i(t) \leqslant \hat{\mu}_K(t) - 2\beta(1) < \hat{\mu}_K(t)$ for arms $i < K$, trivially satisfying (148).

Induction: Suppose (148) is true for rounds up to $t-1$. Consider two cases for round t:

If round t is an exploration round and $I_t \neq K$ is pulled, then only $\hat{\mu}_{I_t}(t)$ changes; the other arms copy their empirical mean from round $t-1$. The attack algorithm ensures $\hat{\mu}_K(t) \geqslant \hat{\mu}_{I_t}(t) + 2\beta(N_K(t)) > \hat{\mu}_{I_t}(t)$. Thus (148) is satisfied at t.

Otherwise either t is exploration and K is pulled; or t is exploitation – in which case K is pulled because by inductive assumption (148) is satisfied at the end of $t-1$. Regardless, this arm K pull is not attacked by Algorithm 1 and its empirical mean is updated by the pre-attack reward. We show this update does not affect the dominance of $\hat{\mu}_K(t)$. Consider any non-target arm $i < K$. Denote the last time $\hat{\mu}_i$ was changed by t'. Note $t' < t$ and $N_K(t') < N_K(t)$. At round t', Algorithm 1 ensured that $\hat{\mu}_i(t') \leqslant \hat{\mu}_K(t') - 2\beta(N_K(t'))$. We have:

$$
\begin{aligned}
\hat{\mu}_K(t) &= \hat{\mu}^0_K(t) && \text{(arm } K \text{ never attacked)} \\
&> \mu^0_K - \beta(N_K(t)) && \text{((6) lower bound)} \\
&> \mu^0_K - \beta(N_K(t')) && \text{(Lemma A.2)} \\
&> \hat{\mu}_K(t') - 2\beta(N_K(t')) && \text{((6) upper bound)} \\
&\geqslant \hat{\mu}_i(t') && \text{(Algorithm 1)} \\
&= \hat{\mu}_i(t)\,.
\end{aligned}
$$

Thus (148) is also satisfied at round t.

Proof of Lemma 2.5 Without loss of generality assume in round t arm i is pulled and the attacker needed to attack the reward (i.e. $I_t = i$ and $\alpha_t > 0$). By defini-

tion (4),

$$\begin{aligned}\alpha_t &= \hat{\mu}_i(t-1)N_i(t-1) + r_t^0 - (\hat{\mu}_K(t) - 2\beta(N_K(t)))\,N_i(t)\\ &= \sum_{s\in\tau_i(t-1)} (r_s^0 - \alpha_s) + r_t^0 - (\hat{\mu}_K(t) - 2\beta(N_K(t)))\,N_i(t)\\ &= \sum_{s\in\tau_i(t)} r_s^0 - \sum_{s\in\tau_i(t-1)} \alpha_s - (\hat{\mu}_K(t) - 2\beta(N_K(t)))\,N_i(t).\end{aligned}$$

Therefore, the cumulative attack on arm i is

$$\begin{aligned}\sum_{s\in\tau_i(t)} \alpha_s &= \sum_{s\in\tau_i(t)} r_s^0 - (\hat{\mu}_K(t) - 2\beta(N_K(t)))\,N_i(t)\\ &= \left(\hat{\mu}_i^0(t) - \hat{\mu}_K(t) + 2\beta(N_K(t))\right) N_i(t).\end{aligned}$$

One can think of the term in front of $N_i(t)$ as the amortized attack cost against arm i. By Lemma 2.3,

$$\begin{aligned}\hat{\mu}_i^0(t) &< \mu_i + \beta(N_i(t))\\ \hat{\mu}_K(t) = \hat{\mu}_K^0(t) &> \mu_K - \beta(N_K(t))\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{s:I_s=i}^{t} \alpha_s &< (\mu_i - \mu_K + \beta(N_i(t)) + 3\beta(N_K(t)))\,N_i(t)\\ &\leqslant (\Delta_i + \beta(N_i(t)) + 3\beta(N_K(t)))\,N_i(t).\end{aligned}$$

The last inequality follows from the gap definition $\Delta_i := [\mu_i - \mu_K]_+$.

Proof of Lemma 2.6 Fix a non-target arm $i < K$. Let X_t be the Bernoulli random variable for round T being arm i pulled. Then,

$$\begin{aligned} N_i(T) &= \sum_{t=1}^{T} X_t \\ \mathbb{E}[X_t] &= \frac{\epsilon_t}{K} \\ \mathbb{V}[X_t] &= \frac{\epsilon_t}{K}(1-\frac{\epsilon_t}{K}) < \frac{\epsilon_t}{K}\,. \end{aligned}$$

Since X_t's are independent random variables, we may apply Lemma 9 of (Agarwal et al., 2014), so that for any $\lambda \in [0,1]$, with probability at least $1-\delta/K$,

$$\begin{aligned} \sum_{t=1}^{T}(X_t - \frac{\epsilon_t}{K}) &\leqslant (e-2)\lambda \sum_{t=1}^{T} \mathbb{V}[X_t] + \frac{1}{\lambda}\log\frac{K}{\delta} \\ &< (e-2)\lambda \sum_{t=1}^{T} \mathbb{E}[X_t] + \frac{1}{\lambda}\log\frac{K}{\delta}\,. \end{aligned}$$

Choose $\lambda = \sqrt{\frac{\log(K/\delta)}{(e-2)\sum_{t=1}^{T}\mathbb{E}[X_t]}}$, and we get that

$$\begin{aligned} \sum_{t=1}^{T} X_t &< \sum_{t=1}^{T}\frac{\epsilon_t}{K} + 2\sqrt{(e-2)\sum_{t=1}^{T}\frac{\epsilon_t}{K}\log\frac{K}{\delta}} \\ &< \sum_{t=1}^{T}\frac{\epsilon_t}{K} + \sqrt{3\sum_{t=1}^{T}\frac{\epsilon_t}{K}\log\frac{K}{\delta}} := \tilde{N}(T)\,. \end{aligned}$$

The same reasoning can be applied to all non-target arm $i < K$. [10]

The case with the target arm is similar, with the only change that $\mathbb{E}[X_t] > 1-\epsilon_t$

[10] Note the upper bound above is valid for T such that $\sum_{t=1}^{T}\epsilon_t \geqslant \frac{K}{e-2}\log(K/\delta)$ only as otherwise λ is greater than 1. One can get rid of such a condition by a slightly looser bound. Specifically, using $\lambda = 1$ gives us a bound that holds true for all T. We then take the max of the two bounds, which can be simplified as $\sum_{t=1}^{T} X_t < (e-1)\sum_{t=1}^{T}\frac{\epsilon_t}{K} + \sqrt{3\sum_{t=1}^{T}\frac{\epsilon_t}{K}\log\frac{K}{\delta}} + \log\frac{K}{\delta}$. The condition on T in Theorem 2.1 can be removed using this bound. However, by keeping the mild assumption on T we keep the exposition simple.

and $\mathbb{V}[X_t] < \epsilon_t$, leading to the lower bound:

$$N_K(T) > T - \sum_{t=1}^{T} \epsilon_t - \sqrt{3 \sum_{t=1}^{T} \epsilon_t \log \frac{K}{\delta}} =: \tilde{N}_K(T)\,.$$

Finally, a union bound is applied to all K arms to complete the proof.

A.3 Details on attacking the UCB strategy

Proof of Lemma 2.7 Fix some $t \geqslant 2K$. If $N_i(t) \leqslant 2$ for all $i < K$, then $N_K(t) \geqslant 2$, which implies $N_i(t) \leqslant \min\{N_K(t), 2\}$. Thus, (8) holds trivially and we are done.

Now fix any $i < K$ such that $N_i(t) > 2$. As the desired upper bound is non-decreasing in t, we only need to prove the result for t where $I_t = i$. Let t' be the previous time where arm i was pulled. Note that t' satisfies $K < t' < t$ as $N_i(t) > 2$, so the attacker has started attacking at round t'. This implies that $N_i(t'-1) + 1 = N_i(t') = N_i(t-1) = N_i(t) - 1$.

On one hand, it is clear that after attack $\alpha_{t'}$ was added at round t', the following holds:

$$\hat{\mu}_i(t') \leqslant \hat{\mu}_K(t') - 2\beta(N_K(t')) - \Delta_0\,. \tag{149}$$

On the other hand, at round t, it must be the case that

$$\hat{\mu}_i(t-1) + 3\sigma\sqrt{\frac{\log t}{N_i(t-1)}} \geqslant \hat{\mu}_K(t-1) + 3\sigma\sqrt{\frac{\log t}{N_K(t-1)}}\,,$$

which is equivalent to

$$\hat{\mu}_i(t') + 3\sigma\sqrt{\frac{\log t}{N_i(t')}} \geqslant \hat{\mu}_K(t-1) + 3\sigma\sqrt{\frac{\log t}{N_K(t-1)}}\,.$$

Therefore,

$$\begin{aligned} 3\sigma\sqrt{\frac{\log t}{N_i(t')}} - 3\sigma\sqrt{\frac{\log t}{N_K(t-1)}} &\geqslant \hat{\mu}_K(t-1) - \hat{\mu}_i(t') \\ &\geqslant \hat{\mu}_K(t-1) - \hat{\mu}_K(t') + 2\beta(N_K(t')) + \Delta_0 \\ &\geqslant \Delta_0\,, \end{aligned}$$

where we have used Eqn. 149 in the second inequality, the condition in event E as well as Lemma A.2 in the third. Since $\Delta_0 > 0$, we can see that $N_i(t') < N_K(t-1)$, and thus

$$N_i(t) = N_i(t') + 1 \leqslant N_K(t-1) = N_K(t)\,. \tag{150}$$

Furthermore, since $3\sigma\sqrt{\frac{\log t}{N_K(t-1)}} > 0$, we have $3\sigma\sqrt{\frac{\log t}{N_i(t')}} > \Delta_0$, which implies

$$N_i(t) = 1 + N_i(t') \leqslant 1 + \frac{9\sigma^2}{\Delta_0^2}\log t\,. \tag{151}$$

Combining (150) and (151) gives the desired bound (8).

Proof of Lemma 2.8 Fix any $i < K$. As the desired upper bound is increasing in t, we only need to prove the result for t where $I_t = i$ and $\alpha_t > 0$. It follows from (7) that,

$$\frac{1}{N_i(t)}\sum_{s\in\tau_i(t)}\alpha_s \;=\; \hat{\mu}_i^0(t) - \hat{\mu}_K(t-1) + 2\beta(N_K(t-1)) + \Delta_0\,.$$

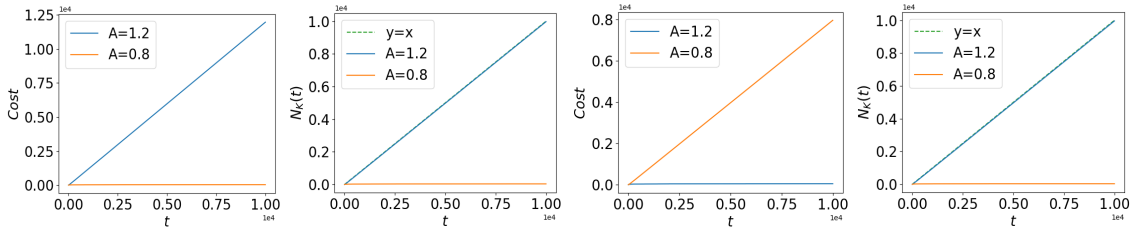
Since event E holds, we have

$$\frac{1}{N_i(t)}\sum_{s\in\tau_i(t)}\alpha_s \leqslant \Delta_i + \Delta_0 + \beta(N_i(t)) + 3\beta(N_K(t-1))\,.$$

The proof is completed by observing $N_K(t-1) = N_K(t)$, $N_i(t) \leqslant N_K(t)$ (Lemma 2.7) and Lemma A.2.

A.4 Simulations on Heuristic Constant Attack

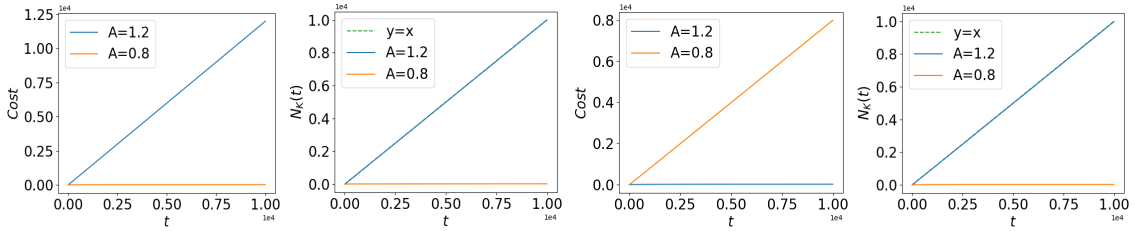
We run simulations on ϵ-greedy and UCB to illustrate the heuristic constant attack algorithm. The bandit has two arms, where the reward distributions are $\mathcal{N}(1, 0.1^2)$ and $\mathcal{N}(0, 0.1^2)$ respectively, thus $\max_i \Delta_i = \mu_1 - \mu_2 = 1$. Alice's target arm is arm 2. In our experiment, Alice tried two different constants for attack: $A = 1.2$ and $A = 0.8$, one being greater and the other being smaller than $\max_i \Delta_i$. We run the attack for $T = 10^4$ rounds. Fig. 31 and Fig. 32 show Alice's cumulative attack cost and Bob's number of target arm pulls $N_K(t)$ for ϵ-greedy and UCB. Note that if $A > \max_i \Delta_i$, then $N_K(t) \approx t$, which verifies that Alice succeeds with the heuristic constant attack. At the same time, pushing up the target arm would incur linear cost; while dragging down the non-target arm achieves logarithmic cost. In summary, Alice should use an A value larger than Δ, and should drag down the expected reward of the non-target arm by amount A.



(a) push up the target arm: $\alpha_t = \mathbb{1}\{I_t = 2\} \cdot (-A)$

(b) drag down the non-target arm: $\alpha_t = \mathbb{1}\{I_t \neq 2\} \cdot A$

Figure 31: Constant attack on ϵ-greedy



(a) push up the target arm

(b) drag down the non-target arm

Figure 32: Constant attack on UCB1

B APPENDIX FOR ADAPTIVE REWARD-POISONING ATTACKS AGAINST REINFORCEMENT LEARNING

B.1 Proof of Theorem 4.3

Proof. Consider two MDPs with reward functions defined as $R + \Delta$ and $R - \Delta$, denote the Q table corresponding to them as $Q_{+\Delta}$ and $Q_{-\Delta}$, respectively. Let $\{(s_t, a_t)\}$ be any instantiated trajectory of the learner corresponding to the attack policy ϕ. By assumption, $\{(s_t, a_t)\}$ visits all (s, a) pairs infinitely often and α_t's satisfy $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. Assuming now that we apply Q-learning on this particular trajectory with reward given by $r_t + \Delta$, standard Q-learning convergence applies and we have that $Q_{t,+\Delta} \to Q_{+\Delta}$ and similarly, $Q_{t,-\Delta} \to Q_{-\Delta}$ (Melo, 2001).

Next, we want to show that $Q_t(s, a) \leqslant Q_{t,+\Delta}(s, a)$ for all $s \in S, a \in A$ and for all t. We prove by induction. First, we know $Q_0(s, a) = Q_{0,+\Delta}(s, a)$. Now, assume that $Q_k(s, a) \leqslant Q_{k,+\Delta}(s, a)$. We have

$$
\begin{aligned}
& Q_{k+1,+\Delta}(s_{k+1}, a_{k+1}) \\
= \; & (1 - \alpha_{k+1}) Q_{k,+\Delta}(s_{k+1}, a_{k+1}) + \\
& \alpha_{k+1} \left(r_{k+1} + \Delta + \gamma \max_{a' \in A} Q_{k,+\Delta}(s'_{k+1}, a') \right) \\
\geqslant \; & (1 - \alpha_{k+1}) Q_k(s_{k+1}, a_{k+1}) + \\
& \alpha_{k+1} \left(r_{k+1} + \delta_{k+1} + \gamma \max_{a' \in A} Q_k(s'_{k+1}, a') \right) \\
= \; & Q_{k+1}(s_{k+1}, a_{k+1}),
\end{aligned}
$$

which established the induction. Similarly, we have $Q_t(s, a) \geqslant Q_{t,-\Delta}(s, a)$. Since $Q_{t,+\Delta} \to Q_{+\Delta}$, $Q_{t,-\Delta} \to Q_{-\Delta}$, we have that for large enough t,

$$
Q_{-\Delta}(s, a) \leqslant Q_t(s, a) \leqslant Q_{+\Delta}, \forall s \in S, a \in A. \tag{152}
$$

Finally, it's not hard to see that $Q_{+\Delta}(s, a) = Q^*(s, a) + \frac{\Delta}{1-\gamma}$ and $Q_{-\Delta}(s, a) =$

$Q^*(s,a) - \frac{\Delta}{1-\gamma}$. This concludes the proof. ■

B.2 Proof of Theorem 4.6

Proof. We provide a constructive proof. We first design an attack policy ϕ, and then show that ϕ is a *strong attack*. For the purpose of finding a strong attack, it suffices to restrict the constructed ϕ to depend only on (s,a) pairs, which is a special case of our general attack setting. Specifically, for any $\Delta > \Delta_3$, we define the following Q':

$$Q'(s,a) = \begin{cases} Q^*(s,a) + \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \in \pi^\dagger(s), \\ Q^*(s,a) - \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \notin \pi^\dagger(s), \\ Q^*(s,a), \forall s \notin S^\dagger, a, \end{cases}$$

where $Q^*(s,a)$ is the original optimal value function without attack. We will show $Q' \in \mathcal{Q}^\dagger$, i.e., the constructed Q' induces the target policy. For any $s \in S^\dagger$, let $a^\dagger \in \arg\max_{a \in \pi^\dagger(s)} Q^*(s,a)$, a best target action desired by the attacker under the original value function Q^*. We next show that $a^\dagger$ becomes the optimal action under Q'. Specifically, $\forall a' \notin \pi^\dagger(s)$, we have

$$\begin{aligned} Q'(s,a^\dagger) &= Q^*(s,a^\dagger) + \frac{\Delta}{(1+\gamma)} \\ &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q^*(s,a') - \frac{\Delta}{(1+\gamma)} \\ &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a'), \end{aligned}$$

Next note that

$$\begin{aligned}\Delta > \Delta_3 &\geqslant \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - \max_{a \in \pi^\dagger(s)} Q^*(s,a)] \\ &= \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - Q^*(s,a^\dagger)] \\ &\geqslant \frac{1+\gamma}{2}[Q^*(s,a') - Q^*(s,a^\dagger)],\end{aligned}$$

which is equivalent to

$$Q^*(s,a^\dagger) - Q^*(s,a') > -\frac{2\Delta}{1+\gamma},$$

thus we have

$$\begin{aligned}Q'(s,a^\dagger) &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a') \\ &> 0 + Q'(s,a') = Q'(s,a').\end{aligned}$$

This shows that under Q', the original best target action $a^\dagger$ becomes better than all non-target actions, thus $a^\dagger$ is optimal and $Q' \in \mathcal{Q}^\dagger$. According to Proposition 4 in (Ma et al., 2019), the Bellman optimality equation induces a unique reward function $R'(s,a)$ corresponding to Q':

$$R'(s,a) = Q'(s,a) - \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q'(s',a').$$

We then construct our attack policy $\phi_{\Delta_3}^{sas}$ as:

$$\phi_{\Delta_3}^{sas}(s,a) = R'(s,a) - R(s,a), \forall s,a.$$

The $\phi_{\Delta_3}^{sas}(s,a)$ results in that the reward function after attack appears to be $R'(s,a)$ from the learner's perspective. This in turn guarantees that the learner will eventually learn Q', which achieves the target policy. Next we show that under $\phi_{\Delta_3}^{sas}(s,a)$, the objective value (40) is finite, thus the attack is feasible. To prove feasibility, we

consider adapting Theorem 4 in (Even-Dar and Mansour, 2003), re-stated as below.

Lemma B.1 (Even-Dar & Mansour). *Assume the attack is* $\phi_{\Delta_3}^{sas}(s, a)$ *and let* Q_t *be the value of the Q-learning algorithm using polynomial learning rate* $\alpha_t = (\frac{1}{1+t})^\omega$ *where* $\omega \in (\frac{1}{2}, 1]$. *Then with probability at least* $1-\delta$, *we have* $\|Q_T - Q'\|_\infty \leqslant \tau$ *with*

$$T = \Omega\left(L^{3+\frac{1}{\omega}} \frac{1}{\tau^2} (\ln \frac{1}{\delta\tau})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}} \ln \frac{1}{\tau}\right),$$

Note that $Q^\dagger$ is an open set and $Q' \in \mathcal{Q}^\dagger$. This implies that one can pick a small enough $\tau_0 > 0$ such that $\|Q_T - Q'\|_\infty \leqslant \tau_0$ implies $Q_T \in Q^\dagger$. From now on we fix this τ_0, thus the bound in the above theorem becomes

$$T = \Omega\left(L^{3+\frac{1}{\omega}} (\ln \frac{1}{\delta})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}}\right).$$

As the authors pointed out in (Even-Dar and Mansour, 2003), the ω that leads to the tightest lower bound on T is around 0.77. Here for our purpose of proving feasibility, it is simpler to let $\omega \approx \frac{1}{2}$ to obtain a loose lower bound on T as below

$$T = \Omega\left(L^5 (\ln \frac{1}{\delta})^2\right).$$

Now we represent δ as a function of T to obtain that $\forall T > 0$,

$$P[\|Q_T - Q'\|_\infty > \tau_0] \leqslant C \exp(-L^{-\frac{5}{2}} T^{\frac{1}{2}}).$$

Let $e_t = \mathbb{1}[\|Q_t - Q'\|_\infty > \tau_0]$, then we have

$$\begin{aligned}
&\mathbf{E}_{\phi_{\Delta_3}^{sas}}\left[\sum_{t=1}^{\infty} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]\right] \leqslant \mathbf{E}_{\phi_{\Delta_3}^{sas}}\left[\sum_{t=1}^{\infty} e_t\right] \\
&= \sum_{t=1}^{\infty} P[\|Q_T - Q'\|_\infty > \tau_0] \leqslant \sum_{t=1}^{\infty} C \exp(-L^{-\frac{5}{2}} t^{\frac{1}{2}}) \\
&\leqslant \int_{t=0}^{\infty} C \exp(-L^{-\frac{5}{2}} t^{\frac{1}{2}}) dt = 2CL^5,
\end{aligned}$$

which is finite. Therefore the attack is feasible.

It remains to validate that $\phi^{sas}_{\Delta_3}$ is a legitimate attack, i.e., $|\delta_t| \leqslant \Delta$ under attack policy $\phi^{sas}_{\Delta_3}$. By Lemma 7 in (Ma et al., 2019), we have

$$\begin{aligned}
&|\delta_t| = |R'(s_t, a_t) - R(s_t, a_t)| \\
&\leqslant \max_{s,a}[R'(s,a) - R(s,a)] = \|R' - R\|_\infty \\
&\leqslant (1+\gamma)\|Q' - Q^*\| = (1+\gamma)\frac{\Delta}{(1+\gamma)} = \Delta.
\end{aligned}$$

Therefore the attack policy $\phi^{sas}_{\Delta_3}$ is valid. ■

Discussion on a number of non-adaptive attacks: Here, we discuss and contrast 3 non-adaptive attack polices developed in this and prior work:

1. (Huang and Zhu, 2019) produces the non-adaptive attack that is feasible with the smallest Δ. In particular, it solves for the following optimization problem:

$$\begin{aligned}
&\min_{\delta, Q \in \mathbb{R}^{S\times A}} \|\delta\|_\infty \\
&\quad \text{s.t. } Q(s,a) = \delta(s,a) + \\
&\qquad \mathbf{E}_{P(s'|s,a)}\left[R(s,a,s) + \gamma \max_{a' \in A} Q(s', a')\right] \\
&\qquad Q \in \mathcal{Q}^\dagger
\end{aligned}$$

   where the optimal objective value implicitly defines a $\Delta'_3 < \Delta_3$. However, it's a fixed policy independent of the actual Δ . In other word, It's either feasible if $\Delta > \Delta'_3$, or not.

2. $\phi^{sas}_{\Delta_3}$ is a closed-form non-adaptive attack that depends on Δ. $\phi^{sas}_{\Delta_3}$ is guaranteed to be feasible when $\Delta > \Delta_3$. However, this is sufficient but not necessary. Implicitly, there exists a Δ''_3 which is the necessary condition for the feasibility of $\phi^{sas}_{\Delta_3}$. Then, we know $\Delta''_3 > \Delta'_3$, because Δ'_3 is the sufficient and necessary condition for the feasibility of any non-adaptive attacks, whereas Δ''_3 is

the condition for the feasibility of non-adaptive attacks of the specific form constructed above.

3. ϕ^{sas}_{TD3} (assume perfect optimization) produces the most efficient non-adaptive attack that depends on Δ.

In terms of efficiency, ϕ^{sas}_{TD3} achieves smaller $J_\infty(\phi)$ than $\phi^{sas}_{\Delta_3}$ and (Huang and Zhu, 2019). It's not clear between $\phi^{sas}_{\Delta_3}$ and (Huang and Zhu, 2019) which one is better. We believe that in most cases, especially when Δ is large and learning rate α_t is small, $\phi^{sas}_{\Delta_3}$ will be faster, because it takes advantage of that large Δ, whereas (Huang and Zhu, 2019) does not. But there probably exist counterexamples on which (Huang and Zhu, 2019) is faster than $\phi^{sas}_{\Delta_3}$.

B.3 The Covering Time L is $O(\exp(|S|))$ for the chain MDP

Proof. While the ϵ-greedy exploration policy constantly change according to the agent's current policy π_t, since L is a uniform upper bound over the whole sequence, and we know that π_t will eventually converge to $\pi^\dagger$, it suffice to show that the covering time under $\pi^\dagger_\epsilon$ is $O(\exp(|S|))$.

Recall that $\pi^\dagger$ prefers going right in all but the left most grid. The covering time in this case is equivalent to the expected number of steps taken for the agent to get from s_0 to the left-most grid, because to get there, the agent necessarily visited all states along the way. Denote the non-absorbing states from right to left as $s_0, s_1, ..., s_{n-1}$, with $|S| = n$. Denote V_k the expected steps to get from state s_k to s_{n-1}. Then, we have the following recursive relation:

$$\begin{aligned} V_{n-1} &= 0 \\ V_k &= 1 + (1 - \frac{\epsilon}{2})V_{k-1} + \frac{\epsilon}{2}V_{k+1}, \\ &\qquad\qquad \text{for } k = 1, ..., n-2 \\ V_0 &= 1 + (1 - \frac{\epsilon}{2})V_0 + \frac{\epsilon}{2}V_1 \end{aligned}$$

Solving the recursive gives

$$V_0 = \frac{p(1+p(1-2p))}{(1-2p)^2}\left[(\frac{1-p}{p})^{n-1} - 1\right] \tag{153}$$

where $p = \frac{\epsilon}{2} < \frac{1}{2}$ and thus $V_0 = O(\exp(n))$. ■

B.4 Proof of Theorem 4.9

Lemma B.2. *For any state $s \in S$ and target actions $A(s) \subset A$, it takes FAA at most $\frac{|A|}{1-\epsilon}$ visits to s in expectation to enforce the target actions $A(s)$.*

Proof. Denote V_t the expected number of visits s to teach $A(s)$ given that under the current Q_t, $\max_{a \in A(s)}$ is ranked t among all actions, where $t \in 1, ..., |A|$. Then, we can write down the following recursion:

$$\begin{aligned} V_1 &= 0 && (154)\\ V_t &= 1 + (1-\epsilon)V_{t-1} + \\ &\quad \epsilon\left[\frac{t-1}{|A|}V_{t-1} + \frac{1}{A}V_1 + \frac{|A|-t}{|A|}V_t\right] && (155) \end{aligned}$$

Equation (155) can be simplified to

$$\begin{aligned} V_t &= \frac{1-\epsilon+\epsilon\frac{t-1}{|A|}}{1-\epsilon\frac{|A|-t}{|A|}}V_{t-1} + \frac{1}{1-\epsilon\frac{|A|-t}{|A|}} \\ &\leqslant V_{t-1} + \frac{1}{1-\epsilon} \end{aligned}$$

Thus, we have

$$V_t \leqslant \frac{t-1}{1-\epsilon} \leqslant \frac{|A|}{1-\epsilon}$$

as needed. ■

Now, we prove Theorem 4.9.

Proof. Let $i \in [1, n]$ be given. First, consider the number of episodes, on which the agent was found in at least one state s_t and is equipped with a policy π_t, s.t. $\pi_t(s_t) \notin \nu_i(s_t)$. Since each of these episodes contains at least one state s_t on which ν_i has not been successfully taught, and according to Lemma 2, it takes at most $\frac{|A|}{1-\epsilon}$ visits to each state to successfully teach any actions $A(s)$, there will be at most $\frac{|S||A|}{1-\epsilon}$ such episodes. These episodes take at most $\frac{|S||A|H}{1-\epsilon}$ iterations for all target states. Out of these episodes, we can safely assume that the agent has successfully picked up ν_i for all the states visited.

Next, we want to show that the expected number of iterations taken by $\pi_i^\dagger$ to get to s_i is upper bounded by $\left[\frac{|A|}{\epsilon}\right]^{i-1} D$, where $\pi_i^\dagger$ is defined as

$$\pi_i^\dagger = \underset{\pi \in \Pi, \pi(s_j) \in \pi^\dagger(s_j), \forall j \leqslant i-1}{\arg\min} \mathbf{E}_{s_0 \sim \mu_0} \left[d_\pi(s_0, s_i) \right]. \tag{156}$$

First, we define another policy

$$\hat{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_1, ..., s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{157}$$

Clearly $\mathbf{E}_{s_0 \sim \mu_0} \left[d_{\pi_i^\dagger}(s_0, s_i) \right] \leqslant \mathbf{E}_{s_0 \sim \mu_0} \left[d_{\hat{\pi}_i^\dagger}(s_0, s_i) \right]$ for all i.

We now prove by induction that $d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant \left[\frac{|A|}{\epsilon}\right]^{i-1} D$ for all i and $s \in S$.

First, let $i = 1$, $\hat{\pi}_i^\dagger = \pi_{s_1}$, and thus $d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant D$.

Next, we assume that when $i = k$, $d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant D_k$, and would like to show that when $i = k + 1$, $d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant \left[\frac{|A|}{\epsilon}\right] D_k$. Define another policy

$$\tilde{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_2, ..., s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{158}$$

which respect the target policies on $s_2, ..., s_{i-1}$, but ignore the target policy on s_1. By the inductive hypothesis, we have that $d_{\tilde{\pi}_i^\dagger}(s, s_i) \leqslant D_k$. Consider the difference between $d_{\hat{\pi}_i^\dagger(s)}(s_1, s_k)$ and $d_{\tilde{\pi}_i^\dagger}(s_1, s_k)$. Since $\hat{\pi}_i^\dagger(s)$ and $\tilde{\pi}_i^\dagger$ only differs by their first

action at s_1, we can derive Bellman's equation on each policy, which yield

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s_1, s_k) &= (1-\epsilon)Q(s_1, \pi^\dagger(s_1)) + \epsilon\bar{Q}(s_1, a) \\ &\leqslant \max_{a\in A} Q(s_1, a) \\ d_{\tilde{\pi}_i^\dagger}(s_1, s_k) &= (1-\epsilon)Q(s_1, \pi_{s_1}(s_1)) + \epsilon\bar{Q}(s_1, a) \\ &\geqslant \frac{\epsilon}{|A|}\max_{a\in A} Q(s_1, a) \end{aligned}$$

where $Q(s_1, a)$ denotes the expected distance to s_k from s_1 by performing action a in the first step, and follow $\hat{\pi}_i^\dagger$ thereafter, and $\bar{Q}(s_1, a)$ denote the expected distance by performing a uniformly random action in the first step. Thus,

$$d_{\hat{\pi}_i^\dagger}(s, s_k) \leqslant \frac{|A|}{\epsilon} d_{\tilde{\pi}_i^\dagger}(s_1, s_k) \tag{159}$$

With this, we can perform the following decomposition:

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s, s_k) &= \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right] \\ &\quad \left(d_{\hat{\pi}_i^\dagger}(s, s_1) + d_{\hat{\pi}_i^\dagger}(s_1, s_k)\right) \\ &\quad + \mathbb{P}\left[\text{not visit } s_1\right] \\ &\quad \left(d_{\hat{\pi}_i^\dagger}(s, s_1) | \text{not visit } s_1\right) \\ &\leqslant \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right] \\ &\quad \left(d_{\tilde{\pi}_i^\dagger}(s, s_1) + \frac{|A|}{\epsilon} d_{\tilde{\pi}_i^\dagger}(s_1, s_k)\right) \\ &\quad + \mathbb{P}\left[\text{not visit } s_1\right] \\ &\quad \left(d_{\tilde{\pi}_i^\dagger}(s, s_k) | \text{not visit } s_1\right) \\ &= d_{\tilde{\pi}_i^\dagger}(s, s_k) + \left(\frac{|A|}{\epsilon} - 1\right) d_{\tilde{\pi}_i^\dagger}(s_1, s_k) \\ &\leqslant D_k + \left(\frac{|A|}{\epsilon} - 1\right) D_k = \frac{|A|}{\epsilon} D_k. \end{aligned}$$

This completes the induction. Thus, we have

$$d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant \left(\frac{|A|}{\epsilon}\right)^{i-1} D,$$

and the total number of iterations taken to arrive at all target states sequentially sums up to

$$\sum_{i=1}^{n} d_{\hat{\pi}_i^\dagger}(s, s_i) \leqslant \left(\frac{|A|}{\epsilon}\right)^{n} D. \tag{160}$$

Finally, each target states need to visited for $\frac{|A|}{1-\epsilon}$ number of times to successfully enforce $\pi^\dagger$. Adding the numbers for enforcing each $\pi_i^\dagger$ gives the correct result. ■

B.5 Detailed Explanation of Fast Adaptive Attack Algorithm

In this section, we try to give a detailed walk-through of the Fast Adaptive Attack Algorithm (FAA) with the goal of providing intuitive understanding of the design principles behind FAA. For the sake of simplisity, in this section we assume that the Q-learning agent is $\epsilon = 0$, such that the attacker is able to fully control the agent's behavior. The proof of correctness and sufficiency in the general case when $\epsilon \in [0, 1]$ is provided in section B.4.

The Greedy Attack: To begin with, let's talk about *the greedy attack*, a fundamental subroutine that is called in every step of FAA to generate the actual attack. Given a desired (partial) policy ν, the greedy attack aims to teach ν to the agent in a greedy fashion. Specifically, at time step t, when the agent performs action a_t at state s_t, the greedy attack first look at whether a_t is a desired action at $s + t$ according to $s\nu$, i.e. whether $a_t \in \nu(s_t)$. If a_t is a desired action, the greedy attack will produce a large enough δ_t, such that after the Q-learning update, a_t becomes strictly more preferred than all undesired actions, i.e. $Q_{t+1}(s_t, a_t) > \max_{a \notin \nu(s_t)} Q_{t+1}(s_t, a)$.

On the other hand, if a_t is not a desired action, the greedy attack will produce a negative enough δ_t, such that after the Q-learning update, a_t becomes strictly less preferred than all desired actions, i.e. $Q_{t+1}(s_t, a_t) < \max_{a \in \nu(s_t)} Q_{t+1}(s_t, a)$. It can be shown that with $\epsilon = 0$, it takes the agent at most $|A| - 1$ visit to a state s, to force the desired actions $\nu(s)$.

Given the greedy attack procedure, one could directly apply the greedy attack with respect to $\pi^\dagger$ throughout the attack procedure. The problem, however, is efficiency. The attack is not considered success without the attacker achieving the target actions in ALL target states, not just the target states visited by the agent. If a target state is never visited by the agent, the attack never succeed. $\pi^\dagger$ itself may not efficiently lead the agent to all the target states. A good example is the chain MDP used as the running example in the main chapter. In section B.3, we have shown that if an agent follows $\pi^\dagger$, it will take exponentially steps to reach the left-most state. In fact, if $\epsilon = 0$, the agent will never reach the left-most state following $\pi^\dagger$, which implies that the naive greedy attack w.r.t. $\pi^\dagger$ is in fact infeasible. Therefore, explicit navigation is necessary. This bring us to the second component of FAA, *the navigation polices*.

The navigation polices: Instead of trying to achieve all target actions at once by directly appling the greedy attack w.r.t. $\pi^\dagger$, FAA aims at one target state at a time. Let $s^\dagger_{(1)}, ..., s^\dagger_{(k)}$ be an order of target states. We will discuss the choice of ordering in the next paragraph, but for now, we will assume that an ordering is given. The agent starts off aiming at forcing the target actions in a single target state $s^\dagger_{(1)}$. To do so, the attacer first calculate the corresponding navigation policy ν_1, where $\nu_1(s_t) = \pi_{s^\dagger_{(1)}}(s_t)$ when $s_t \neq s^\dagger_{(1)}$, and $\nu_1(s_t) = \pi^\dagger(s_t)$ when $s_t = s^\dagger_{(1)}$. That is, ν_1 follows the shortest path policy w.r.t. $s^\dagger_{(1)}$ when the agent has not arrived at $s^\dagger_{(1)}$, And when the agent is in $s^\dagger_{(1)}$, ν_1 follows the desired target actions. Using the greedy attack w.r.t. ν_1 allows the attacker to effectively lure the agent into $s^\dagger_{(1)}$ and force the target actions $\pi^\dagger(s^\dagger_{(1)})$. After successfully forcing the target actions in $s^\dagger_{(1)}$, the attacker moves on to $s^\dagger_{(2)}$. This time, the attacker defines the navigation policy ν_2 similiar to ν_1, except that we don't want the already forced $\pi^\dagger(s^\dagger_{(1)})$ to be

untaught. As a result, in ν_2, we define $\nu_2(s^\dagger_{(1)}) = \pi^\dagger(s^\dagger_{(1)})$, but otherwise follows the corresponding shortest-path policy $\pi_{s^\dagger_{(2)}}$. Follow the greedy attack w.r.t. ν_2, the attacker is able to achieve $\pi^\dagger(s^\dagger_{(2)})$ efficiently without affecting $\pi^\dagger(s^\dagger_{(1)})$. This process is carried on throughout the whole ordered list of target states, where the target actions for already achieved target states are always respected when defining the next ν_i. If each target states $s^\dagger_{(i)}$ can be reachable with the corresponding ν_i, then the whole process will terminate at which point all target actions are guaranteed to be achieved. However, the reachability is not always guaranteed with any ordering of target states. Take the chain MDP as an example. if the 2nd left target state is ordered before the left-most state, then after teaching the target action for the 2nd left state, which is moving right, it's impossible to arrive at the left-most state when the navigation policy resepct the moving-right action in the 2nd left state. Therefore, the *ordering* of target states matters.

The ordering of target states: FAA orders the target states descendingly by their shortest distance to the starting state s_0. Under such an ordering, the target states achieved first are those that are farther away from the starting state, and they necessarily do not lie on the shortest path of the target states later in the sequence. In the chain MDP example, the target states are ordered from left to right. This way, the agent is always able to get to the currently focused target state from the starting state s_0, without worrying about violating the already achieved target states to the left. However, note that the bound provided in theorem 4.9 do not utilize this particular ordering choice and applies to any ordering of target states. As a result, the bound diverges when $\epsilon \to 0$, matching with the pathological case described at the end of the last paragraph.

Parameters	Values	Description
exploration noise	0.5	Std of Gaussian exploration noise.
batch size	100	Batch size for both actor and critic
discount factor	0.99	Discounting factor for the attacker problem.
policy noise	0.2	Noise added to target policy during critic update.
noise clip	$[-0.5, 0.5]$	Range to clip target policy noise.
action L2 weight	50	Weight for L2 regularization
buffer size	10^7	Replay buffer size
optimizer	Adam	Use the Adam optimizer.
learning rate critic	10^{-3}	Learning rate for the critic network.
learning rate actor	5^{-4}	Learning rate for the actor network.
τ	0.002	Target network update rate.
policy frequency	2	Frequency of delayed policy update.

Table 8: Hyperparameters for TD3.

B.6 Experiment Setting and Hyperparameters for TD3

Throughout the experiments, we use the following set of hyperparameters for TD3, described in Table 8. The hyperparameters are selected via grid search on the Chain MDP of length 6. Each experiment is run for 5000 episodes, where each episode is of 1000 iteration long. The learned policy is evaluated for every 10 episodes, and the policy with the best evaluation performance is used for e evaluations in the experiment section.

B.7 Additional Plot for the rate comparison experiment

See Figure 33.

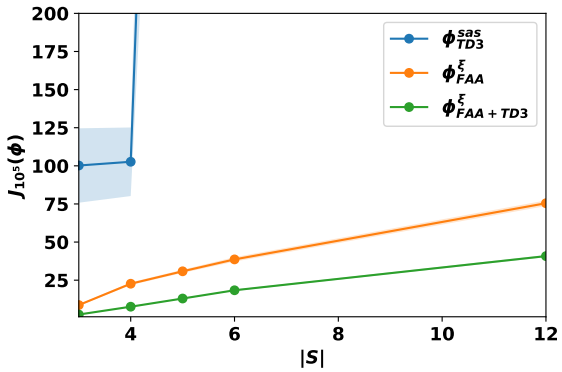


Figure 33: Attack performances on the chain MDP of different length in the normal scale. As can be seen in the plot, both $\phi^{\xi}_{\mathrm{FAA}}$ + $\phi^{\xi}_{\mathrm{TD3+FAA}}$ achieve linear rate.

B.8 Additional Experiments: Attacking DQN

Throughout the chapter, we have been focusing on attacking the tabular Q-learning agent. However, the attack MDP also applies to arbitrary RL agents. We describe the general interaction protocol in Alg. 9. Importantly, we assume that the RL agent can be fully characterized by an **internal state**, which determines the agent's current behavior policy as well as the learning update. For example, if the RL

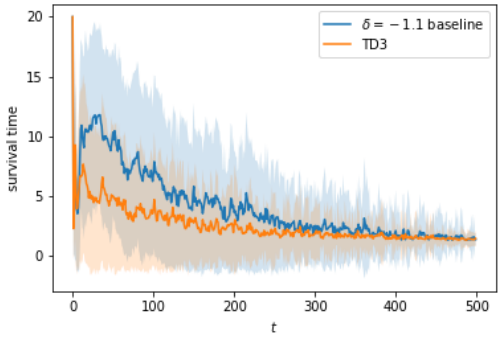


Figure 34: Result for attacking DQN on the Cartpole environment. The left figure plots the cumulative attack cost $J_T(\phi)$ as a function of T. The right figure plot the performance of the DQN agent $J(\theta_t)$ under the two attacks.

agent is a Deep Q-Network (DQN), the internal state will consist of the Q-network

Protocol 9 Reward Poisoning against general RL agent

Parameters: MDP (S, A, R, P, μ_0), RL agent hyperparameters.

1: **for** $t = 0, 1, ...$ **do**
2: agent at state s_t, has internal state θ_0.
3: agent acts according to a behavior policy:
$a_t \leftarrow \pi_{\theta_t}(s_t)$
4: environment transits $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, produces reward $r_t = R(s_t, a_t, s_{t+1})$ and an end-of-episode indicator EOE.
5: attacker perturbs the reward to $r_t + \delta_t$
6: agent receives $(s_{t+1}, r_t + \delta_t, \text{EOE})$, performs one-step of internal state update:

$$\theta_{t+1} = f(\theta_t, s_t, a_t, s_{t+1}, r_t + \delta_t, \text{EOE}) \tag{161}$$

7: environment resets if $\text{EOE} = 1$: $s_{t+1} \sim \mu_0$.
8: **end for**

parameters as well as the transitions stored in the replay buffer.

In the next example, we demonstrate an attack against DQN in the cartpole environment. In the cartpole environment, the agent can perform 2 actions, moving left and moving right, and the goal is to keep the pole upright without moving the cart out of the left and right boundary. The agent receives a constant $+1$ reward in every iteration, until the pole falls or the cart moves out of the boundary, which terminates the current episode and the cart and pole positions are reset.

In this example, the attacker's goal is to poison a well-trained DQN agent to perform as poorly as possible. The corresponding attack cost $\rho(\xi_t)$ is defined as $J(\theta_t)$, the expected total reward received by the current DQN policy in evaluation. The DQN is first trained in the clean cartpole MDP and obtains the optimal policy that successfully maintains the pole upright for 200 iterations (set maximum length of an episode). The attacker is then introduced while the DQN agent continues to train in the cartpole MDP. We freeze the Q-network except for the last layer to reduce the size of the attack state representation. We compare TD3 with a naive

attacker that perform $\delta_t = -1.1$ constantly. The results are shown in Fig. 34.

One can see that under the TD3 found attack policy, the performance of the DQN agent degenerates much faster compared to the naive baseline. While still being a relatively simple example, this experiment demonstrates the potential of applying our adaptive attack framework to general RL agents.

C APPENDIX FOR POLICY POISONING IN BATCH REINFORCEMENT LEARNING AND CONTROL

C.1 Proof of Proposition 5.2

The proof of feasibility relies on the following result, which states that there is a bijection mapping between reward space and value function space.

Proposition C.1. *Given an MDP with transition probability function* P *and discounting factor* $\gamma \in [0, 1)$*, let* $\mathcal{R} = \{R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}\}$ *denote the set of all possible reward functions, and let* $\mathcal{Q} = \{Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}\}$ *denote the set of all possible Q tables. Then, there exists a bijection mapping between* $\mathcal{R}$ *and* $\mathcal{Q}$*, induced by Bellman optimality equation.*

Proof. $\Rightarrow$ Given any reward function $R(s, a) \in \mathcal{R}$, define the Bellman operator as

$$H_R(Q)(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{162}$$

Since $\gamma < 1$, $H_R(Q)$ is a contraction mapping, i.e., $\|H_R(Q_1) - H_R(Q_2)\|_\infty \leqslant \gamma \|Q_1 - Q_2\|_\infty, \forall Q_1, Q_2 \in \mathcal{Q}$. Then by Banach Fixed Point Theorem, there is a unique $Q \in \mathcal{Q}$ that satisfies $Q = H_R(Q)$, which is the Q that R maps to.

$\Leftarrow$ Given any $Q \in \mathcal{Q}$, one can define the corresponding $R \in \mathcal{R}$ by

$$R(s, a) = Q(s, a) - \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{163}$$

Thus the mapping is one-to-one. ∎

Proposition C.2. *The attack problem* (59)-(62) *is always feasible for any target policy* $\pi^\dagger$*.*

Proof. For any target policy $\pi^\dagger : \mathcal{S} \mapsto \mathcal{A}$, we construct the following Q:

$$Q(s, a) = \begin{cases} \epsilon & \forall s \in \mathcal{S}, a = \pi^\dagger(s), \\ 0, & \text{otherwise.} \end{cases} \tag{164}$$

The Q values in (164) satisfy the constraint (62). Note that we construct the Q values so that for all $s \in \mathcal{S}$, $\max_a Q(s,a) = \epsilon$. By proposition C.1, the corresponding reward function induced by Bellman optimality equation is

$$\hat{R}(s,a) = \begin{cases} (1-\gamma)\epsilon & \forall s \in \mathcal{S}, a = \pi^\dagger(s), \\ -\gamma\epsilon, & \text{otherwise.} \end{cases} \tag{165}$$

Then one can let $r_t = \hat{R}(s_t, a_t)$ so that $\mathbf{r} = (r_0, ..., r_{T-1})$, $\hat{R}$ in (165), together with Q in (164) is a feasible solution to (59)-(62). ■

C.2 Proof of Theorem 5.3

The proof of Theorem 5.3 relies on a few lemmas. We first prove the following result, which shows that given two vectors that have equal element summation, the vector whose elements are smoother will have smaller ℓ_α norm for any $\alpha \geqslant 1$. This result is used later to prove Lemma C.4.

Lemma C.3. *Let $x, y \in \mathbb{R}^T$ be two vectors. Let $\mathcal{I} \subset \{0, 1, ..., T-1\}$ be a subset of indexes such that*

$$i). \quad x_i = \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} y_j, \forall i \in \mathcal{I}, \qquad ii). \quad x_i = y_i, \forall i \neq \mathcal{I}. \tag{166}$$

Then for any $\alpha \geqslant 1$, we have $\|x\|_\alpha \leqslant \|y\|_\alpha$.

Proof. Note that the conditions $i)$ and $ii)$ suggest the summation of elements in x and y are equal, and only elements in $\mathcal{I}$ differ for the two vectors. However, the elements in $\mathcal{I}$ of x are smoother than that of y, thus x has smaller norm. To prove the result, we consider three cases separately.

Case 1: $\alpha = 1$. Then we have

$$\|x\|_\alpha - \|y\|_\alpha = \sum_i |x_i| - \sum_j |y_j| = \sum_{i \in \mathcal{I}} |x_i| - \sum_{j \in \mathcal{I}} |y_j| = |\sum_{j \in \mathcal{I}} y_j| - \sum_{j \in \mathcal{I}} |y_j| \leqslant 0. \tag{167}$$

Case 2: $1 < \alpha < \infty$. We show $\|x\|_\alpha^\alpha \leqslant \|y\|_\alpha^\alpha$. Note that

$$\begin{aligned}\|x\|_\alpha^\alpha - \|y\|_\alpha^\alpha &= \sum_i |x_i|^\alpha - \sum_j |y_j|^\alpha = \sum_{i\in\mathcal{I}} |x_i|^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha \\ &= \frac{1}{|\mathcal{I}|^{\alpha-1}} |\sum_{j\in\mathcal{I}} y_j|^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha \leqslant \frac{1}{|\mathcal{I}|^{\alpha-1}} (\sum_{j\in\mathcal{I}} |y_j|)^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha.\end{aligned} \tag{168}$$

Let $\beta = \frac{\alpha}{\alpha-1}$. By Holder's inequality, we have

$$\sum_{j\in\mathcal{I}} |y_j| \leqslant (\sum_{j\in\mathcal{I}} |y_j|^\alpha)^{\frac{1}{\alpha}} (\sum_{j\in\mathcal{I}} 1^\beta)^{\frac{1}{\beta}} = (\sum_{j\in\mathcal{I}} |y_j|^\alpha)^{\frac{1}{\alpha}} |\mathcal{I}|^{1-\frac{1}{\alpha}}. \tag{169}$$

Plugging (169) into (168), we have

$$\|x\|_\alpha^\alpha - \|y\|_\alpha^\alpha \leqslant \frac{1}{|\mathcal{I}|^{\alpha-1}} (\sum_{j\in\mathcal{I}} |y_j|^\alpha) |\mathcal{I}|^{\alpha-1} - \sum_{j\in\mathcal{I}} |y_j|^\alpha = 0. \tag{170}$$

Case 3: $\alpha = \infty$. We have

$$\begin{aligned}\|x\|_\alpha = \max_i |x_i| &= \max\{\frac{1}{|\mathcal{I}|} |\sum_{j\in\mathcal{I}} y_j|, \max_{i\notin\mathcal{I}} |x_i|\} \leqslant \max\{\frac{1}{|\mathcal{I}|} \sum_{j\in\mathcal{I}} |y_j|, \max_{i\notin\mathcal{I}} |x_i|\} \\ &\leqslant \max\{\max_{j\in\mathcal{I}} |y_j|, \max_{i\notin\mathcal{I}} |x_i|\} = \max\{\max_{j\in\mathcal{I}} |y_j|, \max_{j\notin\mathcal{I}} |y_j|\} = \max_j |y_j| = \|y\|_\alpha.\end{aligned} \tag{171}$$

Therefore $\forall \alpha \geqslant 1$, we have $\|x\|_\alpha \leqslant \|y\|_\alpha$. ■

Next we prove Lemma C.4, which shows that one possible optimal attack solution to (59)-(62) takes the following form: shift all the clean rewards in $T_{s,a}$ by the same amount $\psi(s,a)$. Here $\psi(s,a)$ is a function of state s and action a. That means, rewards belonging to different $T_{s,a}$ might be shifted a different amount, but those corresponding to the same (s,a) pair will be identically shifted.

Lemma C.4. *There exists a function* $\psi(s,a)$ *such that* $r_t = r_t^0 + \psi(s_t, a_t)$*, together with some* $\hat{R}$ *and* Q*, is an optimal solution to our attack problem* (59)-(62)*.*

We point out that although there exists an optimal attack taking the above form, it is not necessarily the only optimal solution. However, all those optimal solutions must have exactly the same objective value (attack cost), thus it suffices to consider the solution in Lemma C.4.

Proof. Let $\mathbf{r}^* = (r_0^*, ..., r_{T-1}^*)$, $\hat{R}^*$ and Q^* be any optimal solution to (59)-(62). Fix a particular state-action pair (s, a), we have

$$\hat{R}^*(s,a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t^*. \tag{172}$$

Let $\hat{R}^0(s,a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t^0$ be the reward function for the (s,a) pair estimated from clean data $\mathbf{r}^0$. We then define a different poisoned reward vector $\mathbf{r}' = (r_0', ..., r_{T-1}')$, where

$$r_t' = \begin{cases} r_t^0 + \hat{R}^*(s,a) - \hat{R}^0(s,a), & t \in T_{s,a}, \\ r_t^*, & t \notin T_{s,a}. \end{cases} \tag{173}$$

Now we show $\mathbf{r}'$, $\hat{R}^*$ and Q^* is another optimal solution to (59)-(62). We first verify that $\mathbf{r}'$, $\hat{R}^*$, and Q^* satisfy constraints (60)-(62). To verify (60), we only need to check $\hat{R}^*(s,a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t'$, since $\mathbf{r}'$ and $\mathbf{r}^*$ only differ on those rewards in $T_{s,a}$. We have

$$\begin{aligned} \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t' &= \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} \left(r_t^0 + \hat{R}^*(s,a) - \hat{R}^0(s,a) \right) \\ &= \hat{R}^0(s,a) + \hat{R}^*(s,a) - \hat{R}^0(s,a) = \hat{R}^*(s,a), \end{aligned} \tag{174}$$

Thus $\mathbf{r}'$ and $\hat{R}^*$ satisfy constraint (60). $\hat{R}^*$ and Q^* obviously satisfy constraints (61) and (62) because $\mathbf{r}^*$, $\hat{R}^*$ and Q^* is an optimal solution.

Let $\delta' = \mathbf{r}' - \mathbf{r}^0$ and $\delta^* = \mathbf{r}^* - \mathbf{r}^0$, then one can easily show that δ' and δ^* satisfy the conditions in Lemma C.3 with $\mathcal{I} = T_{s,a}$. Therefore by Lemma C.3, we have

$$\|\mathbf{r}' - \mathbf{r}^0\|_\alpha = \|\delta'\|_\alpha \leqslant \|\delta^*\|_\alpha = \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha. \tag{175}$$

But note that by our assumption, $\mathbf{r}^*$ is an optimal solution, thus $\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leqslant \|\mathbf{r}' - \mathbf{r}^0\|_\alpha$, which gives $\|\mathbf{r}' - \mathbf{r}^0\|_\alpha = \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha$. This suggests $\mathbf{r}'$, $\hat{R}^*$, and Q^* is

another optimal solution. Compared to $\mathbf{r}^*$, $\mathbf{r}'$ differs in that $r'_t - r^0_t$ now becomes identical for all $t \in T_{s,a}$ for a particular (s, a) pair. Reusing the above argument iteratively, one can make $r'_t - r^0_t$ identical for all $t \in T_{s,a}$ for all (s, a) pairs, while guaranteeing the solution is still optimal. Therefore, we have

$$r'_t = r^0_t + \hat{R}^*(s, a) - \hat{R}^0(s, a), \forall t \in T_{s,a}, \forall s, a, \tag{176}$$

together with $\hat{R}^*$ and Q^* is an optimal solution to (59)-(62). Let $\psi(s, a) = \hat{R}^*(s, a) - \hat{R}^0(s, a)$ conclude the proof. ■

Finally, Lemma C.5 provides a sensitive analysis on the value function Q as the reward function changes.

Lemma C.5. *Let $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}', \gamma)$ and $\hat{M}^0 = (\mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}^0, \gamma)$ be two MDPs, where only the reward function differs. Let Q' and Q^0 be action values satisfying the Bellman optimality equation on $\hat{M}$ and $\hat{M}^0$ respectively, then*

$$(1 - \gamma)\|Q' - Q^0\|_\infty \leqslant \|\hat{R} - \hat{R}^0\|_\infty \leqslant (1 + \gamma)\|Q' - Q^0\|_\infty. \tag{177}$$

Proof. Define the Bellman operator as

$$H_{\hat{R}}(Q)(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{P}(s' \mid s, a) \max_{a'} Q(s', a'). \tag{178}$$

From now on we suppress variables s and a for convenience. Note that due to the Bellman optimality, we have $H_{\hat{R}^0}(Q^0) = Q^0$ and $H_{\hat{R}'}(Q') = Q'$, thus

$$\begin{aligned}
\|Q' - Q^0\|_\infty &= \|H_{\hat{R}'}(Q') - H_{\hat{R}^0}(Q^0)\|_\infty \\
&= \|H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0) + H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \\
&\leqslant \|H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0)\|_\infty + \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \\
&\leqslant \gamma\|Q' - Q^0\|_\infty + \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \text{ (by contraction of } H_{\hat{R}'}(\cdot)) \\
&= \gamma\|Q' - Q^0\|_\infty + \|\hat{R}' - \hat{R}^0\|_\infty \text{ (by } H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0) = \hat{R}' - \hat{R}^0)
\end{aligned} \tag{179}$$

Rearranging we have $(1-\gamma)\|Q' - Q^0\|_\infty \leqslant \|\hat{R}' - \hat{R}^0\|_\infty$. Similarly we have

$$\begin{aligned}
\|Q' - Q^0\|_\infty &= \|H_{\hat{R}'}(Q') - H_{\hat{R}^0}(Q^0)\|_\infty \\
&= \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0) + H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0)\|_\infty \\
&\geqslant \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty - \|H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0)\|_\infty \\
&\geqslant \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty - \gamma\|Q' - Q^0\|_\infty \\
&= \|\hat{R}' - \hat{R}^0\|_\infty - \gamma\|Q' - Q^0\|_\infty
\end{aligned} \tag{180}$$

Rearranging we have $\|\hat{R}' - \hat{R}^0\|_\infty \leqslant (1+\gamma)\|Q' - Q^0\|_\infty$, concluding the proof. ∎

Now we are ready to prove our main result.

Theorem 5.3. *Assume* $\alpha \geqslant 1$ *in* (59)*. Let* $\mathbf{r}^*$, $\hat{R}^*$ *and* Q^* *be an optimal solution to* (59)-(62)*, then*

$$\frac{1}{2}(1-\gamma)\Delta(\epsilon)\left(\min_{s,a}|T_{s,a}|\right)^{\frac{1}{\alpha}} \leqslant \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leqslant \frac{1}{2}(1+\gamma)\Delta(\epsilon)T^{\frac{1}{\alpha}}. \tag{63}$$

Proof. We construct the following value function Q'.

$$Q'(s,a) = \begin{cases} Q^0(s,a) + \dfrac{\Delta(\epsilon)}{2}, & \forall s \in \mathcal{S}, a = \pi^\dagger(s), \\ Q^0(s,a) - \dfrac{\Delta(\epsilon)}{2}, & \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s). \end{cases} \tag{181}$$

Note that $\forall s \in \mathcal{S}$ and $\forall a \neq \pi^\dagger(s)$, we have

$$\begin{aligned}
\Delta(\epsilon) &= \max_{s' \in \mathcal{S}}[\max_{a' \neq \pi^\dagger(s')} Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon]_+ \\
&\geqslant \max_{a' \neq \pi^\dagger(s)} Q^0(s, a') - Q^0(s, \pi^\dagger(s)) + \epsilon \geqslant Q^0(s,a) - Q^0(s, \pi^\dagger(s)) + \epsilon,
\end{aligned} \tag{182}$$

which leads to

$$Q^0(s,a) - Q^0(s, \pi^\dagger(s)) - \Delta(\epsilon) \leqslant -\epsilon, \tag{183}$$

thus we have $\forall s \in \mathcal{S}$ and $\forall a \neq \pi^\dagger(s)$,

$$\begin{aligned} Q'(s,\pi^\dagger(s)) &= Q^0(s,\pi^\dagger(s)) + \frac{\Delta(\epsilon)}{2} \\ &= Q^0(s,a) - [Q^0(s,a) - Q^0(s,\pi^\dagger(s)) - \Delta(\epsilon)] - \frac{\Delta(\epsilon)}{2} \\ &\geqslant Q^0(s,a) + \epsilon - \frac{\Delta(\epsilon)}{2} = Q'(s,a) + \epsilon. \end{aligned} \tag{184}$$

Therefore Q' satisfies the constraint (62). By proposition C.1, there exists a unique function R' such that Q' satisfies the Bellman optimality equation of MDP $\hat{\mathcal{M}}' = (\mathcal{S}, \mathcal{A}, \hat{P}, R', \gamma)$. We then construct the following reward vector $\mathbf{r}' = (r'_0, ..., r'_{T-1})$ such that $\forall(s,a)$ and $\forall t \in T_{s,a}$, $r'_t = r^0_t + R'(s,a) - \hat{R}^0(s,a)$, where $\hat{R}^0(s,a)$ is the reward function estimated from $\mathbf{r}^0$. The reward function estimated on $\mathbf{r}'$ is then

$$\begin{aligned} \hat{R}'(s,a) &= \frac{1}{|T_{s,a}|}\sum_{t\in T_{s,a}} r'_t = \frac{1}{|T_{s,a}|}\sum_{t\in T_{s,a}} \left(r^0_t + R'(s,a) - \hat{R}^0(s,a)\right) \\ &= \hat{R}^0(s,a) + R'(s,a) - \hat{R}^0(s,a) = R'(s,a). \end{aligned} \tag{185}$$

Thus $\mathbf{r}'$, $\hat{R}'$ and Q' is a feasible solution to (59)-(62). Now we analyze the attack cost for $\mathbf{r}'$, which gives us a natural upper bound on the attack cost of the optimal solution $\mathbf{r}^*$. Note that Q' and Q^0 satisfy the Bellman optimality equation for reward function $\hat{R}'$ and $\hat{R}^0$ respectively, and

$$\|Q' - Q^0\|_\infty = \frac{\Delta(\epsilon)}{2}, \tag{186}$$

thus by Lemma C.5, we have $\forall t$,

$$\begin{aligned} |r'_t - r^0_t| = |\hat{R}'(s_t,a_t) - \hat{R}^0(s_t,a_t)| &\leqslant \max_{s,a} |\hat{R}'(s,a) - \hat{R}^0(s,a)| = \|\hat{R}' - \hat{R}^0\|_\infty \\ &\leqslant (1+\gamma)\|Q' - Q^0\|_\infty = \frac{1}{2}(1+\gamma)\Delta(\epsilon). \end{aligned} \tag{187}$$

Therefore, we have

$$\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leqslant \|\mathbf{r}' - \mathbf{r}^0\|_\alpha = (\sum_{t=0}^{T-1} |r'_t - r^0_t|^\alpha)^{\frac{1}{\alpha}} \leqslant \frac{1}{2}(1+\gamma)\Delta(\epsilon)T^{\frac{1}{\alpha}}. \tag{188}$$

Now we prove the lower bound. We consider two cases separately.

Case 1: $\Delta(\epsilon) = 0$. We must have $Q^0(s, \pi^\dagger(s)) \geqslant Q^0(s, a) + \epsilon, \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s)$. In this case no attack is needed and therefore the optimal solution is $\mathbf{r}^* = \mathbf{r}^0$. The lower bound holds trivially.

Case 2: $\Delta(\epsilon) > 0$. Let s' and a' ($a' \neq \pi^\dagger(s')$) be a state-action pair such that

$$\Delta(\epsilon) = Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon. \tag{189}$$

Let $\mathbf{r}^*$, $\hat{R}^*$ and Q^* be an optimal solution to (59)-(62) that takes the form in Lemma C.4, i.e.,

$$r^*_t = r^0_t + \hat{R}^*(s, a) - \hat{R}^0(s, a), \forall t \in T_{s,a}, \forall s, a. \tag{190}$$

Constraint (62) ensures that $Q^*(s', \pi^\dagger(s')) \geqslant Q^*(s', a') + \epsilon$, in which case either one of the following two conditions must hold:

$$\text{i)}. \quad Q^*(s', \pi^\dagger(s')) - Q^0(s', \pi^\dagger(s')) \geqslant \frac{\Delta(\epsilon)}{2}, \qquad \text{ii)}. \quad Q^0(s', a') - Q^*(s', a') \geqslant \frac{\Delta(\epsilon)}{2}, \tag{191}$$

since otherwise we have

$$\begin{aligned} &Q^*(s', \pi^\dagger(s')) < Q^0(s', \pi^\dagger(s')) + \frac{\Delta(\epsilon)}{2} = Q^0(s', \pi^\dagger(s')) + \frac{1}{2}[Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon] \\ &= \frac{1}{2}Q^0(s', a') + \frac{1}{2}Q^0(s', \pi^\dagger(s')) + \frac{\epsilon}{2} = Q^0(s', a') - \frac{1}{2}[Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon] + \epsilon \\ &= Q^0(s', a') - \frac{\Delta(\epsilon)}{2} + \epsilon < Q^*(s', a') + \epsilon. \end{aligned} \tag{192}$$

Next note that if either i) or ii) holds, we have $\|Q^* - Q^0\|_\infty \geqslant \frac{\Delta(\epsilon)}{2}$. By Lemma C.5,

we have

$$\max_{s,a} |\hat{R}^*(s,a) - \hat{R}^0(s,a)| = \|\hat{R}^* - \hat{R}^0\|_\infty \geqslant (1-\gamma)\|Q^* - Q^0\|_\infty \geqslant \frac{1}{2}(1-\gamma)\Delta(\epsilon). \quad (193)$$

Let $s^*, a^* \in \arg\max_{s,a} |\hat{R}^*(s,a) - \hat{R}^0(s,a)|$, then we have

$$|\hat{R}^*(s^*,a^*) - \hat{R}^0(s^*,a^*)| \geqslant \frac{1}{2}(1-\gamma)\Delta(\epsilon). \quad (194)$$

Therefore, we have

$$\begin{aligned} \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha^\alpha &= \sum_{t=0}^{T-1} |r_t^* - r_t^0|^\alpha = \sum_{s,a} \sum_{t \in T_{s,a}} |r_t^* - r_t^0|^\alpha \geqslant \sum_{t \in T_{s^*,a^*}} |r_t^* - r_t^0|^\alpha \\ &= \sum_{t \in T_{s^*,a^*}} |\hat{R}^*(s^*,a^*) - \hat{R}^0(s^*,a^*)|^\alpha \geqslant \left(\frac{1}{2}(1-\gamma)\Delta(\epsilon)\right)^\alpha |T_{s^*,a^*}| \quad (195) \\ &\geqslant \left(\frac{1}{2}(1-\gamma)\Delta(\epsilon)\right)^\alpha \min_{s,a} |T_{s,a}|. \end{aligned}$$

Therefore $\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \geqslant \frac{1}{2}(1-\gamma)\Delta(\epsilon)\left(\min_{s,a} |T_{s,a}|\right)^{\frac{1}{\alpha}}$.

We finally point out that while an optimal solution $\mathbf{r}^*$ may not necessarily take the form in Lemma C.4, it suffices to bound the cost of an optimal attack which indeed takes this form (as we did in the proof) since all optimal attacks have exactly the same objective value. ■

C.3 Convex Surrogate for LQR Attack Optimization

By pulling the positive semi-definite constraints on Q and R out of the lower level optimization (79), one can turn the original attack optimization (74)-(80) into the

following surrogate optimization:

$$\min_{\mathbf{r},\hat{Q},\hat{R},\hat{q},\hat{c},X,x} \quad \|\mathbf{r}-\mathbf{r}_0\|_\alpha \tag{196}$$

$$\text{s.t.} \quad -\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}=K^\dagger, \tag{197}$$

$$-\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top x=k^\dagger, \tag{198}$$

$$X=\gamma\hat{A}^\top X\hat{A}-\gamma^2\hat{A}^\top X\hat{B}\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}+\hat{Q} \tag{199}$$

$$x=\hat{q}+\gamma(\hat{A}+\hat{B}K^\dagger)^\top x \tag{200}$$

$$(\hat{Q},\hat{R},\hat{q},\hat{c})=\arg\min\sum_{t=0}^{T-1}\left\|\frac{1}{2}s_t^\top Qs_t+q^\top s_t+a_t^\top Ra_t+c+r_t\right\|_2^2 \tag{201}$$

$$\hat{Q}\succeq 0,\hat{R}\succeq \epsilon I,X\succeq 0. \tag{202}$$

The feasible set of (196)-(202) is a subset of the original problem, thus the surrogate attack optimization is a more stringent formulation than the original attack optimization, that is, successfully solving the surrogate optimization gives us a (potentially) sub-optimal solution to the original problem. To see why the surrogate optimization is more stringent, we illustrate with a much simpler example as below. A formal proof is straight forward, thus we omit it here. The original problem is (203)-(204). The feasible set for $\hat{a}$ is a singleton set $\{0\}$, and the optimal objective value is 0.

$$\min_{\hat{a}} \quad 0 \tag{203}$$

$$\text{s.t.} \quad \hat{a}=\arg\min_{a\geqslant 0}(a+3)^2, \tag{204}$$

Once we pull the constraint out of the lower-level optimization (204), we end up with a surrogate optimization (205)-(207). Note that (206) requires $\hat{a}=-3$, which does not satisfy (207). Therefore the feasible set of the surrogate optimization is $\emptyset$,

meaning it is more stringent than (203)-(204).

$$\min_{\hat{a}} \quad 0 \tag{205}$$
$$\text{s.t.} \quad \hat{a} = \arg\min(a+3)^2, \tag{206}$$
$$\hat{a} \geqslant 0 \tag{207}$$

Back to our attack optimization (196)-(202), this surrogate attack optimization comes with the advantage of being convex, thus can be solved to global optimality.

Proposition C.6. *The surrogate attack optimization* (196)-(202) *is convex.*

Proof. First note that the sub-level optimization (201) is itself a convex problem, thus is equivalent to the corresponding KKT condition. We write out the KKT condition of (201) to derive an explicit form of our attack formulation as below:

$$\min_{\mathbf{r},\hat{Q},\hat{R},\hat{q},\hat{c},X,x} \quad \|\mathbf{r}-\mathbf{r}_0\|_\alpha \tag{208}$$
$$\text{s.t.} \quad -\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A} = K^\dagger, \tag{209}$$
$$-\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top x = k^\dagger, \tag{210}$$
$$X = \gamma\hat{A}^\top X\hat{A} - \gamma^2\hat{A}^\top X\hat{B}\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A} + \hat{Q} \tag{211}$$
$$x = \hat{q} + \gamma(\hat{A}+\hat{B}K^\dagger)^\top x \tag{212}$$
$$\sum_{t=0}^{T-1}\left(\frac{1}{2}s_t^\top\hat{Q}s_t + \hat{q}^\top s_t + a_t^\top\hat{R}a_t + \hat{c} + r_t\right)s_t s_t^\top = 0, \tag{213}$$
$$\sum_{t=0}^{T-1}\left(\frac{1}{2}s_t^\top\hat{Q}s_t + \hat{q}^\top s_t + a_t^\top\hat{R}a_t + \hat{c} + r_t\right)a_t a_t^\top = 0, \tag{214}$$
$$\sum_{t=0}^{T-1}\left(\frac{1}{2}s_t^\top\hat{Q}s_t + \hat{q}^\top s_t + a_t^\top\hat{R}a_t + \hat{c} + r_t\right)s_t = 0, \tag{215}$$
$$\sum_{t=0}^{T-1}\left(\frac{1}{2}s_t^\top\hat{Q}s_t + \hat{q}^\top s_t + a_t^\top\hat{R}a_t + \hat{c} + r_t\right) = 0, \tag{216}$$
$$\hat{Q} \succeq 0, \hat{R} \succeq \epsilon I, X \succeq 0. \tag{217}$$

The objective is obviously convex. (209)-(211) are equivalent to

$$-\gamma\hat{B}^\top X\hat{A} = \left(\hat{R} + \gamma\hat{B}^\top X\hat{B}\right) K^\dagger. \tag{218}$$

$$-\gamma\hat{B}^\top x = \left(\hat{R} + \gamma\hat{B}^\top X\hat{B}\right) k^\dagger. \tag{219}$$

$$X = \gamma\hat{A}^\top X(\hat{A} + \hat{B}K^\dagger) + \hat{Q}, \tag{220}$$

Note that these three equality constraints are all linear in X, $\hat{R}$, x, and $\hat{Q}$. (212) is linear in x and $\hat{q}$. (213)-(216) are also linear in $\hat{Q}$, $\hat{R}$, $\hat{q}$, $\hat{c}$ and $\mathbf{r}$. Finally, (217) contains convex constraints on $\hat{Q}$, $\hat{R}$, and X. Given all above, the attack problem is convex. ■

Next we analyze the feasibility of the surrogate attack optimization.

Proposition C.7. *Let* $\hat{A}$, $\hat{B}$ *be the learner's estimated transition kernel. Let*

$$L^\dagger(s, a) = \frac{1}{2}s^\top Q^\dagger s + (q^\dagger)^\top s + a^\top R^\dagger a + c^\dagger \tag{221}$$

be the attacker-defined loss function. Assume $R^\dagger \succeq \epsilon I$. *If the target policy* $K^\dagger$, $k^\dagger$ *is the optimal control policy induced by the LQR with transition kernel* $\hat{A}$, $\hat{B}$, *and loss function* $L^\dagger(s, a)$, *then the surrogate attack optimization* (196)-(202) *is feasible. Furthermore, the optimal solution can be achieved.*

Proof. To prove feasibility, it suffices to construct a feasible solution to optimization (196)-(202). Let

$$r_t = \frac{1}{2}s_t^\top Q^\dagger s_t + {q^\dagger}^\top s_t + a_t^\top R^\dagger a_t + c^\dagger \tag{222}$$

and $\mathbf{r}$ be the vector whose t-th element is r_t. We next show that $\mathbf{r}$, $Q^\dagger$, $R^\dagger$, $q^\dagger$, $c^\dagger$, together with some X and x is a feasible solution. Note that since $K^\dagger$, $k^\dagger$ is induced by the LQR with transition kernel $\hat{A}$, $\hat{B}$ and cost function $L^\dagger(s, a)$, constraints (197)-(200) must be satisfied with some X and x. The poisoned reward vector $\mathbf{r}$ obviously satisfies (201) since it is constructed exactly as the minimizer. By our assumption, $R^\dagger \succeq \epsilon I$, thus (202) is satisfied. Therefore, $\mathbf{r}$, $Q^\dagger$, $R^\dagger$, $q^\dagger$, $c^\dagger$, together with the

corresponding X, x is a feasible solution, and the optimization (196)-(202) is feasible. Furthermore, since the feasible set is closed, the optimal solution can be achieved. ■

C.4 Conditions for The LQR Learner to Have Unique Estimate

The LQR learner estimates the cost function by

$$(\hat{Q}, \hat{R}, \hat{q}, \hat{c}) = \underset{(Q \succeq 0, R \succeq \epsilon I, q, c)}{\arg\min} \frac{1}{2} \sum_{t=0}^{T-1} \left\| \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t \right\|_2^2. \tag{223}$$

We want to find a condition that guarantees the uniqueness of the solution.

Let $\psi \in \mathbb{R}^T$ be a vector, whose t-th element is

$$\psi_t = \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c, 0 \leqslant t \leqslant T-1. \tag{224}$$

Note that we can view ψ as a function of D, Q, R, q, and c, thus we can also denote $\psi(D, Q, R, q, c)$. Define $\Psi(D) = \{\psi(D, Q, R, q, c) \mid Q \succeq 0, R \succeq \epsilon I, q, c\}$, i.e., all possible vectors that are achievable with form (224) if we vary Q, R, q and c subject to positive semi-definite constraints on Q and R. We can prove that Ψ is a closed convex set.

Proposition C.8. *$\forall D, \Psi(D) = \{\psi(D, Q, R, q, c) \mid Q \succeq 0, R \succeq \epsilon I, q, c\}$ is a closed convex set.*

Proof. Let $\psi_1, \psi_2 \in \Psi(D)$. We use $\psi_{i,t}$ to denote the t-th element of vector ψ_i. Then we have

$$\psi_{1,t} = \frac{1}{2} s_t^\top Q_1 s_t + q_1^\top s_t + a_t^\top R_1 a_t + c_1 \tag{225}$$

for some $Q_1 \succeq 0$, $R_1 \succeq \epsilon I$, q_1 and c_1, and

$$\psi_{2,t} = \frac{1}{2} s_t^\top Q_2 s_t + q_2^\top s_t + a_t^\top R_2 a_t + c_2 \tag{226}$$

for some $Q_2 \succeq 0$, $R_2 \succeq \epsilon I$, q_2 and c_2. $\forall k \in [0,1]$, let $\psi_3 = k\psi_1 + (1-k)\psi_2$. Then the t-th element of ψ_3 is

$$\begin{aligned}\psi_{3,t} =& \frac{1}{2} s_t^\top [kQ_1 + (1-k)Q_2] s_t + [kq_1 + (1-k)q_2]^\top s_t \\ &+ a_t^\top [kR_1 + (1-k)R_2] a_t + kc_1 + (1-k)c_2\end{aligned} \tag{227}$$

Since $kQ_1 + (1-k)Q_2 \succeq 0$ and $kR_1 + (1-k)R_2 \succeq \epsilon I$, $\psi_3 \in \Psi(D)$, concluding the proof. ■

The optimization (223) is intrinsically a least-squares problem with positive semi-definite constraints on Q and R, and is equivalent to solving the following linear equation:

$$\frac{1}{2} s_t^\top \hat{Q} s_t + \hat{q}^\top s_t + a_t^\top \hat{R} a_t + \hat{c} = \psi_t^*, \forall t, \tag{228}$$

where $\psi^* = \arg\min_{\psi \in \Psi(D)} \|\psi + \mathbf{r}\|_2^2$ is the projection of the negative reward vector $-\mathbf{r}$ onto the set $\Psi(D)$. The solution to (228) is unique if and only if the following two conditions both hold

i). The projection ψ^* is unique.

ii). (228) has a unique solution for ψ^*.

Condition i) is satisfied because $\Psi(D)$ is convex, and any projection (in ℓ_2 norm) onto a convex set exists and is always unique (see Hilbert Projection Theorem). We next analyze when condition ii) holds. (228) is a linear function in $\hat{Q}$, $\hat{R}$, $\hat{q}$, and $\hat{c}$, thus one can vectorize $\hat{Q}$ and $\hat{R}$ to obtain a problem in the form of linear regression. Then the uniqueness is guaranteed if and only if the design matrix has full column rank. Specifically, let $\hat{Q} \in \mathbb{R}^{n \times n}$, $\hat{R} \in \mathbb{R}^{m \times m}$, and $\hat{q} \in \mathbb{R}^n$. Let $s_{t,i}$ and $a_{t,i}$ denote

the i-th element of s_t and a_t respectively. Define

$$\mathbf{A} = \left[\begin{array}{ccccc|ccccc|c|c} \frac{s_{0,1}^2}{2} & \dots & \frac{s_{0,i}s_{0,j}}{2} & \dots & \frac{s_{0,n}^2}{2} & a_{0,1}^2 & \dots & a_{0,i}a_{0,j} & \dots & a_{0,m}^2 & s_0^\top & 1 \\ \frac{s_{1,1}^2}{2} & \dots & \frac{s_{1,i}s_{1,j}}{2} & \dots & \frac{s_{1,n}^2}{2} & a_{1,1}^2 & \dots & a_{1,i}a_{2,j} & \dots & a_{1,m}^2 & s_1^\top & 1 \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \frac{s_{t,1}^2}{2} & \dots & \frac{s_{t,i}s_{t,j}}{2} & \dots & \frac{s_{t,n}^2}{2} & a_{t,1}^2 & \dots & a_{t,i}a_{t,j} & \dots & a_{t,m}^2 & s_t^\top & 1 \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \frac{s_{T-1,1}^2}{2} & \dots & \frac{s_{T-1,i}s_{T-1,j}}{2} & \dots & \frac{s_{T-1,n}^2}{2} & a_{T-1,1}^2 & \dots & a_{T-1,i}a_{T-1,j} & \dots & a_{T-1,m}^2 & s_{T-1}^\top & 1 \end{array}\right],$$

$$\mathbf{x}^\top = \left[\begin{array}{ccccc|ccccc|cccc|c} \hat{Q}_{11} & \dots & \hat{Q}_{ij} & \dots & \hat{Q}_{nn} & \hat{R}_{11} & \dots & \hat{R}_{ij} & \dots & \hat{R}_{mm} & \hat{q}_1 & \dots & \hat{q}_i & \dots & \hat{q}_n & \hat{c} \end{array}\right],$$

then (228) is equivalent to $\mathbf{A}\mathbf{x} = \psi^*$, where $\mathbf{x}$ contains the vectorized variables $\hat{Q}$, $\hat{R}$, $\hat{q}$ and $\hat{c}$. $\mathbf{A}\mathbf{x} = \psi^*$ has a unique solution if and only if $\mathbf{A}$ has full column rank.

C.5 Sparse Attacks on TCE and LQR

In this section, we present experimental details for both TCE and LQR victims when the attacker uses ℓ_1 norm to measure the attack cost, i.e. $\alpha = 1$. The other experimental parameters are set exactly the same as in the main text.

We first show the result for MDP experiment 2 with $\alpha = 1$, see Figure 35. The attack cost is $\|\mathbf{r} - \mathbf{r}^0\|_1 = 3.27$, which is small compared to $\|\mathbf{r}^0\|_1 = 105$. We note that the reward poisoning is extremely sparse: only the reward corresponding to action "go up" at the terminal state G is increased by 3.27, and all other rewards remain unchanged. To explain this attack, first note that we set the target action for the terminal state to "go up", thus the corresponding reward must be increased. Next note that after the attack, the terminal state becomes a sweet spot, where the agent can keep taking action "go up" to gain large amount of discounted future reward. However, such future reward is discounted more if the agent reaches the terminal state via a longer path. Therefore, the agent will choose to go along the red trajectory to get into the terminal state earlier, though at a price of two discounted -10 rewards.

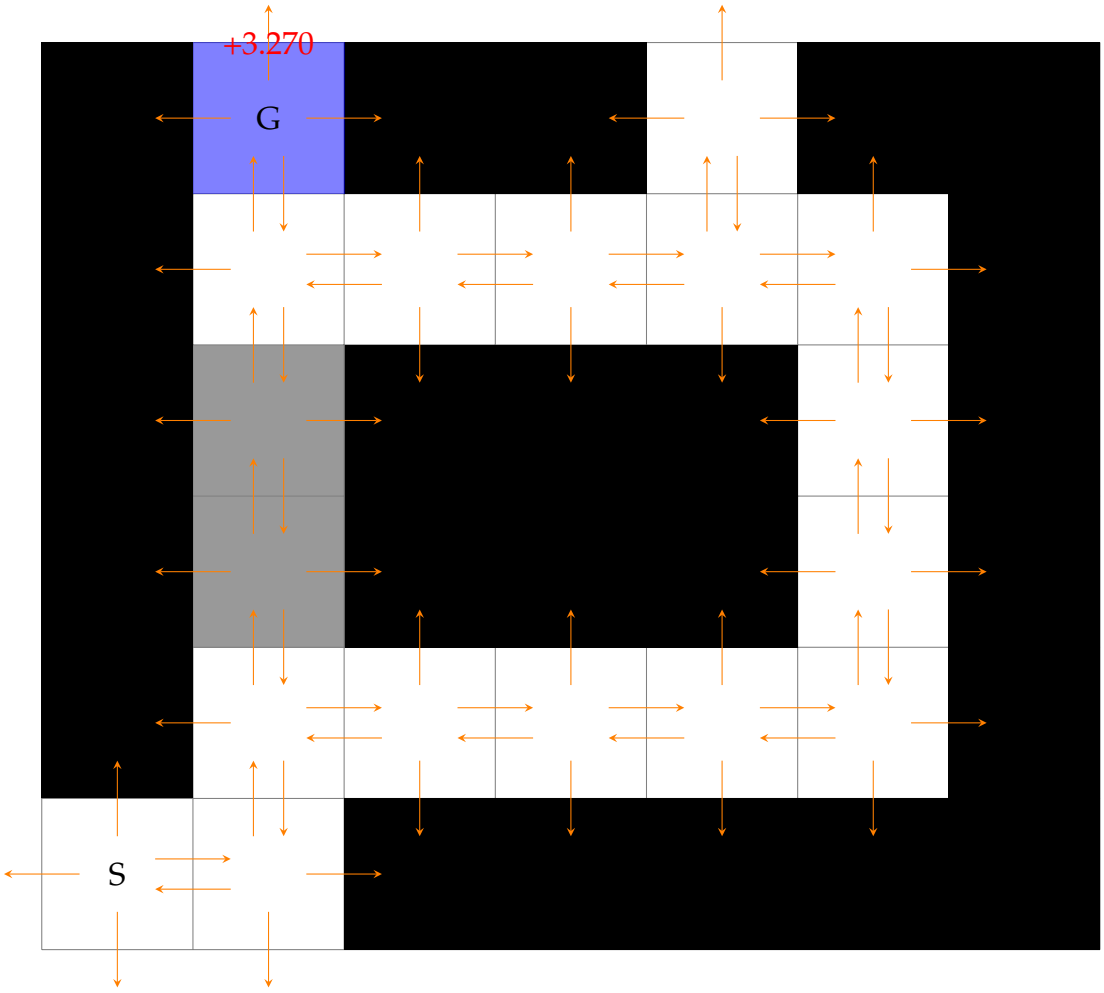


Figure 35: Sparse reward modification for MDP experiment 2.

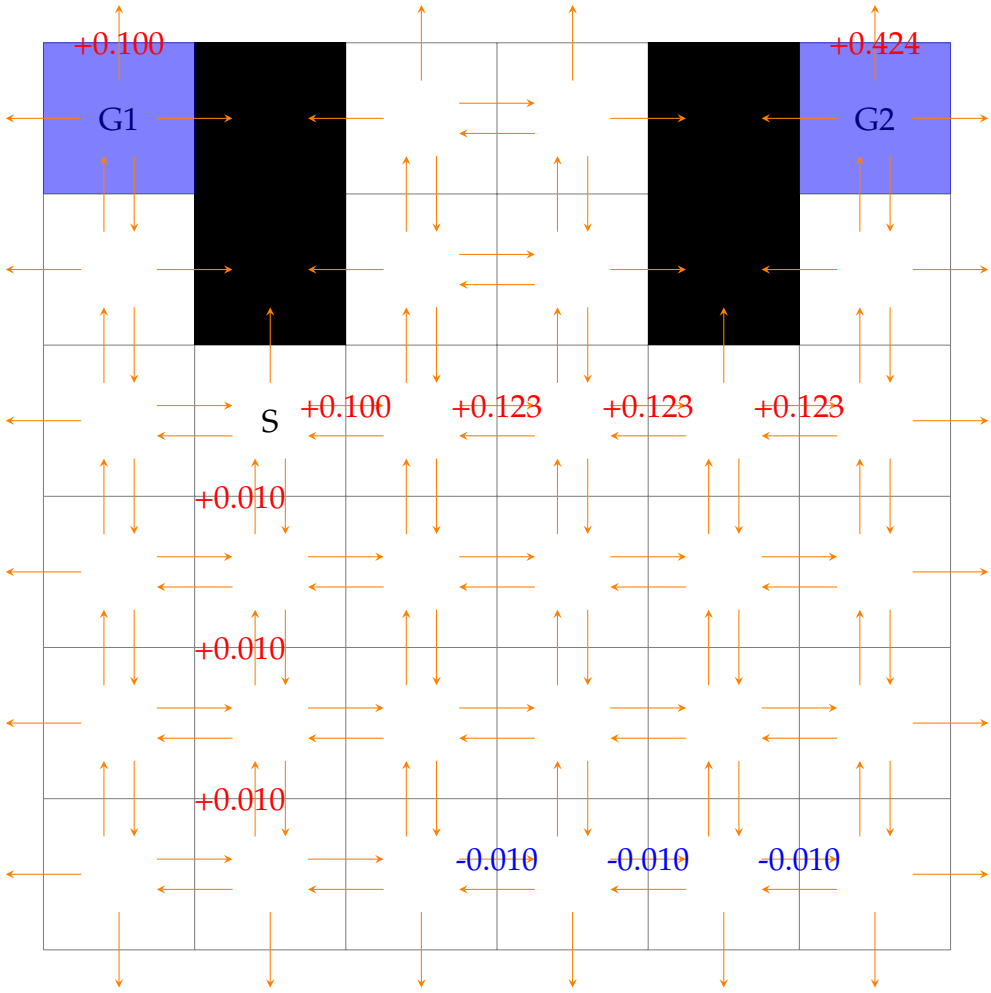
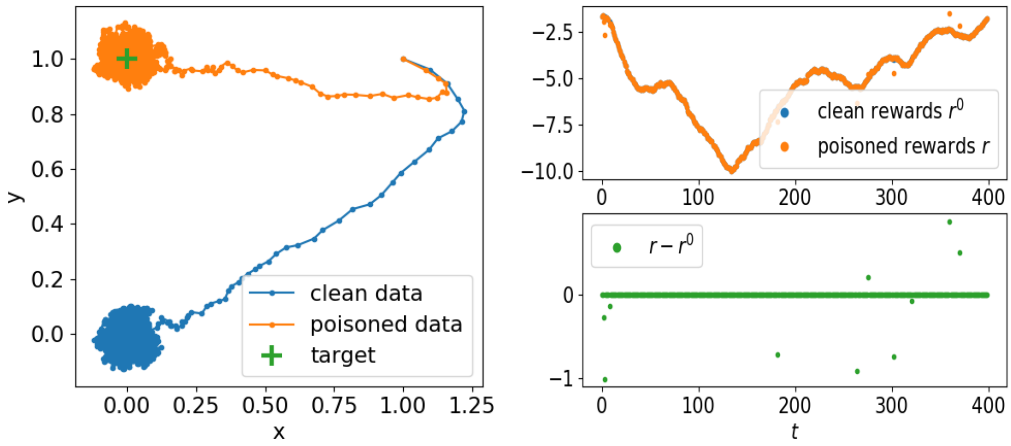


Figure 36: Sparse reward modification for MDP experiment 3.

The result is similar for MDP experiment 3. The attack cost is $\|\mathbf{r} - \mathbf{r}^0\|_1 = 1.05$, compared to $\|\mathbf{r}^0\|_1 = 121$. In Figure 36, we show the reward modification for each state action pair. Again, the attack is very sparse: only rewards of 12 state-action pairs are modified out of a total of 124.

Finally, we show the result on attacking LQR with $\alpha = 1$. The attack cost is $\|\mathbf{r} - \mathbf{r}^0\|_1 = 5.44$, compared to $\|\mathbf{r}^0\|_1 = 2088.57$. In Figure 37, we plot the clean and poisoned trajectory of the vehicle, together with the reward modification in each time step. The attack is as effective as with a dense 2-norm attack in Figure 21.

However, the poisoning is highly sparse: only 10 out of 400 rewards are changed.



(a) Clean and poisoned vehicle trajectory. (b) Clean and poisoned rewards.

Figure 37: Sparse-poisoning a vehicle running LQR in 4D state space.

C.6 Derivation of Discounted Discrete-time Algebraic Riccati Equation

We provide a derivation for the discounted Discrete-time Algebraic Riccati Equation. For simplicity, we consider the noiseless case, but the derivation easily generalizes to noisy case. We consider the loss function is a general quadratic function w.r.t. s as follows:

$$L(s, a) = \frac{1}{2} s^\top Q s + q^\top s + c + a^\top R a. \tag{229}$$

When $q = 0, c = 0$, we recover the classic LQR setting. Assume the general value function takes the form $V(s) = \frac{1}{2} s^\top X s + s^\top x + v$. Let $Q(s, a)$ (note that this is different notation from the Q matrix in $L(s, a)$) be the corresponding action value

function. We perform dynamics programming as follows:

$$\begin{aligned} Q(s,a) &= \frac{1}{2}s^\top Q s + q^\top s + c + a^\top R a + \gamma V(As+Ba) \\ &= \frac{1}{2}s^\top Q s + q^\top s + c + a^\top R a + \gamma\left(\frac{1}{2}(As+Ba)^\top X(As+Ba) + (As+Ba)^\top x + v\right) \\ &= \frac{1}{2}s^\top(Q+\gamma A^\top X A)s + \frac{1}{2}a^\top(R+\gamma B^\top X B)a + s^\top(\gamma A^\top X B)a \\ &\quad + s^\top(q+\gamma A^\top x) + a^\top(\gamma B^\top x) + (c+\gamma v). \end{aligned} \tag{230}$$

We minimize a above:

$$\begin{aligned} &(R+\gamma B^\top X B)a + \gamma B^\top X A s + \gamma B^\top x = 0 \\ &\Rightarrow a = -\gamma(R+\gamma B^\top X B)^{-1}B^\top X A s - \gamma(R+\gamma B^\top X B)^{-1}B^\top x \triangleq Ks + k. \end{aligned} \tag{231}$$

Now we substitute it back to $Q(s,a)$ and regroup terms, we get:

$$\begin{aligned} V(s) =& \frac{1}{2}s^\top(Q+\gamma A^\top X A + K^\top(R+\gamma B^\top X B)K + 2\gamma A^\top X B K)s \\ &+ s^\top(K^\top(R+\gamma B^\top X B)k + \gamma A^\top X B k + q + \gamma A^\top x + \gamma K^\top B^\top x) + C \end{aligned} \tag{232}$$

for some constant C, which gives us the following recursion:

$$\begin{aligned} X &= \gamma A^\top X A - \gamma^2 A^\top X B(R+\gamma B^\top X B)^{-1}B^\top X A + Q, \\ x &= q + \gamma(A+BK)^\top x. \end{aligned} \tag{233}$$

D.1 Simulated Raw Data Processing

CARLA outputs a single RGB image, a depth map image, and variable number of RADAR points for each 0.05 second time step of the simulation. We analyze this data at each time step to produce object detections in the same format of MATLAB FCW (MATLAB, 2020b):

$$\begin{bmatrix} d^1 & v^1 & d^2 & v^2 \end{bmatrix}$$

where d^1 and d^2 are the distance, in meters, from the vehicle sensor in directions parallel and perpendicular to the vehicle's motion, respectively. v^1 and v^2 are the detected object velocities, in m/s, relative to the ego along these parallel and perpendicular axes.

To produce these detections from vision data, we first find bounding boxes around probable vehicles in each RGB image frame using an implementation of a YOLOv2 network in MATLAB which has been pre-trained on vehicle images (MATLAB, 2020a). Each bounding box is used to create a distinct object detection. The d^1 value, or depth, of each object is taken to be the depth recorded by the depth map at the center pixel of each bounding box.

The d^2 value of each detection is then computed as

$$d^2 = u * \frac{d^1}{l_{foc}} \tag{234}$$

where u is the horizontal pixel coordinate of the center of a bounding box in a frame, and l_{foc} is the focal length of the RGB camera in pixels (R. Collins, 2007). l_{foc} is not directly specified by CARLA, but can be computed using the image length, 800 pixels, and the camera field of vision, 90 degrees (Edmund Optics, 2020).

To compute v^1 and v^2 for detections of the current time step, we also consider detections from the previous time step. First, we attempt to match each bounding box from the current time step to a single bounding box from the previous step.

Box pairs are evaluated based on their Intersection-Over-Union (IoU) (A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, sep. 2016). Valued between 0 and 1, a high IoU indicates high similarity of size and position of two boxes, and we enforce a minimum threshold of 0.4 for any two boxes to be paired. For two adjacent time steps, A and B, we take the IoU of all possible pairs of bounding boxes with one box from step A, and one from B. These IoU values form the cost matrix for the Hungarian matching algorithm (Murray, 2017), which produces the best possible pairings of bounding boxes from the current time step to the previous.

This matching process results in a set of detections with paired bounding boxes, and a set with unpaired boxes. For each detection with a paired box, we calculate its velocity simply as the difference between respective d^1 and d^2 values of the current detection and its paired observation from the previous time step, multiplied by the frame rate, fps_{cam}. For a detection, a, paired with a previous detection, b:

$$< v_a^1, v_a^2 >=< d_b^1 - d_a^1, d_b^2 - d_a^2 > * fps_{cam} \tag{235}$$

For each detection left unpaired after Hungarian matching, we make no conclusions about v^1 or v^2 for that detection, and treat each as zero.

Each RADAR measurement output by CARLA represents an additional object detection. RADAR measurements contain altitude (al) and azimuth (az) angle measurements, as well as depth (d) and velocity (v), all relative to the RADAR sensor. We convert these measurements into object detection parameters as follows

$$d^1 = d * \cos az * \cos al \qquad v^1 = v * \cos az * \cos al$$
$$d^2 = d * \sin az * \cos al \qquad v^2 = v * \sin az * \cos al$$

D.2 Derivation of Surrogate Constraints

The original attack optimization (91)-(98) may not be convex due to that (97) and (98) could be nonlinear. Our goal in this section is to derive convex surrogate constraints that are good approximations to (97) and (98). Furthermore, we require the surrogate constraints to be tighter than the original constraints, so that solving the attack under the surrogate constraints will always give us a feasible solution to the original attack. Concretely, we want to obtain surrogate constraints to $F(x) = \ell$, where $\ell \in \{\text{green, yellow, red}\}$. We analyze each case of ℓ separately:

- $\ell = \text{green}$

  In this case, $F(x) = \ell$ is equivalent to $v \geqslant 0$ according to (90). While this constraint is convex, when we actually solve the optimization, it might be violated due to numerical inaccuracy. To avoid such numerical issues, we tighten it by adding a margin parameter $\epsilon > 0$, and the derived surrogate constraint is $v \geqslant \epsilon$.

- $\ell = \text{red}$

  In this case, $F(x) = \ell$ is equivalent to

  $$v < 0 \tag{236}$$
  $$d \leqslant -1.2v + \frac{1}{0.8g}v^2. \tag{237}$$

  Similar to case 1, we tighten the first constraint as

  $$v \leqslant -\epsilon. \tag{238}$$

  Note that by the first constraint, we must have $v < 0$. The second constraint is $d \leqslant -1.2v + \frac{1}{0.8g}v^2$. Given $v < 0$, this is equivalent to

  $$v \leqslant 0.48g - \sqrt{(0.48g)^2 + 0.8gd}. \tag{239}$$

We next define the following function

$$U(d) = 0.48g - \sqrt{(0.48g)^2 + 0.8gd}. \tag{240}$$

The first derivative is

$$U'(d) = -\frac{0.4g}{\sqrt{(0.48g)^2 + 0.8gd}}, \tag{241}$$

which is increasing when $d \geqslant 0$. Therefore, the function $U(d)$ is convex. We now fit a linear function that lower bounds $U(d)$. Specifically, since $U(d)$ is convex, for any $d_0 \geqslant 0$, we have

$$U(d) \geqslant U'(d_0)(d - d_0) + U(d_0). \tag{242}$$

Therefore, $v \leqslant U'(d_0)(d - d_0) + U(d_0)$ is a tighter constraint than $v \leqslant U(d)$. The two constraints are equivalent at $d = d_0$. Again, we need to add a margin parameter to avoid constraint violation due to numerical inaccuracy. With this in mind, the surrogate constraint becomes

$$v \leqslant U'(d_0)(d - d_0) + U(d_0) - \epsilon, \tag{243}$$

Or, equivalently:

$$v \leqslant -\epsilon, \tag{244}$$

$$v \leqslant U'(d_0)(d - d_0) + U(d_0) - \epsilon, \tag{245}$$

This concludes the proof of our Proposition 6.1.

However, we still need to pick an appropriate d_0. In our scenario, the distance d has physical limitation $d \in [\underline{d}, \bar{d}]$ with $\underline{d} = 0$ and $\bar{d} = 75$. The $U(d)$ curve for $d \in [0, 75]$ is shown in Fig 38. Based on the figure, we select d_0 such that $U'(d_0)$ is equal to the slope of the segment connecting the two end points of

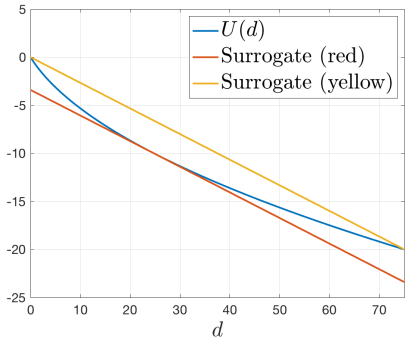


Figure 38: Surrogate light constraints.

the curve, i.e.,

$$U'(d_0) = \frac{U(75) - U(0)}{75} = \frac{U(75)}{75}. \tag{246}$$

We now derive the concrete surrogate constraints used in our experiment section. We begin with the following equation:

$$0.48g + \frac{0.4g}{U'(d)} = U(d). \tag{247}$$

From which, we can derive d_0:

$$d_0 = \frac{1}{0.8g}\left((\frac{30g}{U(75)})^2 - (0.48g)^2\right) \tag{248}$$

and

$$U(d_0) = 0.48g + \frac{30g}{U(75)}. \tag{249}$$

By substituting d_0 and $U(d_0)$ into (243), we find that the surrogate constraint is

$$v \leqslant \frac{U(75)}{75}(d - d_0) + 0.48g + \frac{30g}{U(75)} + \epsilon. \tag{250}$$

- ℓ = yellow

In this case, $F(x) = \ell$ is equivalent to

$$v < 0 \tag{251}$$

$$d \geqslant -1.2v + \frac{1}{0.8g}v^2. \tag{252}$$

Similarly, we tighten the first constraint to

$$v \leqslant -\epsilon. \tag{253}$$

For the second constraint, the situation is similar to $\ell = \text{red}$. $\forall d_0 > 0$. We have

$$v \geqslant \frac{U(d_0)}{d_0} d, \forall d \in [0, d_0] \tag{254}$$

The above inequality is derived by fitting a linear function that is always above the $U(d)$ curve. Next, we select $d_0 = 75$ and add a margin parameter ϵ to derive the surrogate constraint:

$$v \leqslant -\epsilon \tag{255}$$

$$v \geqslant \frac{U(75)}{75} d + \epsilon. \tag{256}$$

To summarize, we have derived surrogate constraints for $F(x) = \ell$, where $l \in \{\text{green, yellow, red}\}$. When we solve the attack optimization, we replace each individual constraint of (97) and (98) by one of the above three surrogate constraints. In Fig 38, we show the surrogate constraints for red and yellow lights with $\epsilon = 10^{-3}$.

D.3 Preprocessing of CARLA Measurements

In this section, we describe how we preprocess the measurements obtained from CARLA simulation. The measurement in each time step takes the form of $t_t = [y_t^1, y_t^2] \in \{\mathbb{R} \cup \text{NaN}\}^8$, where $y_t^1 \in \{\mathbb{R} \cup \text{NaN}\}^4$ is the vision detection produced by

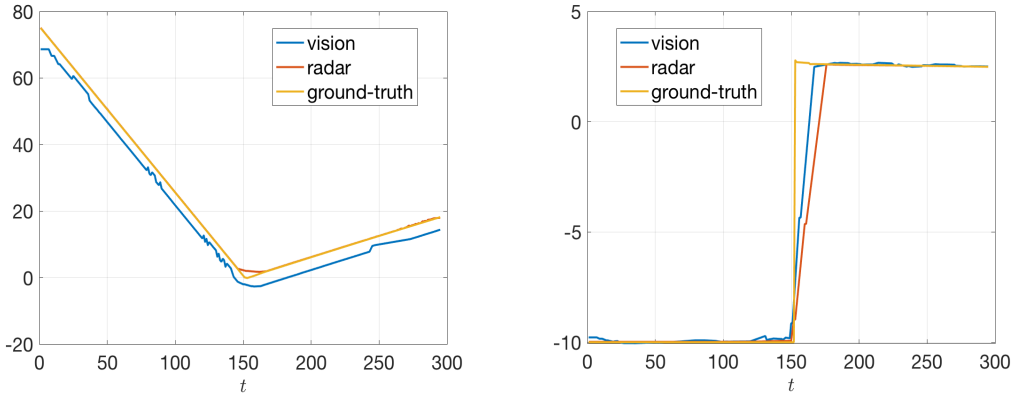
ML-based objection detection algorithm YOLOv2, and y_t^2 is the detection generated by radar (details in Appendix D.1). Both vision and radar measurements contain four components: (1) the distance to MIO along driving direction, (2) the velocity of MIO along driving direction, (3) the distance to MIO along lateral direction, and (4) the velocity of MIO along lateral direction. The radar measurements are relatively accurate, and do not have missing data or outliers. However, there are missing data (NaN) and outliers in vision measurements. The missing data problem arises because the MIO sometimes cannot be detected, e.g., in the beginning of the video sequence when the MIO is out of the detection range of the camera. Outliers occur because YOLOv2 may not generate an accurate bounding box of the MIO, causing it to correspond to a depth map reading of an object at a different physical location. As such, a small inaccuracy in the location of the bounding box could lead to dramatic change to the reported distance and velocity of the MIO.

In our experiment, we preprocess detections output from CARLA to address missing data and outlier issues. First, we identify the outliers by the Matlab "fill-outliers" method, where we choose "movmedian" as the detector and use linear interpolation to replace the outliers. The concrete Matlab command is:

$$\text{filloutliers}(Y, \text{'linear'}, \text{'movmedian'}, 0, \text{'ThresholdFactor'}, 0.5),$$

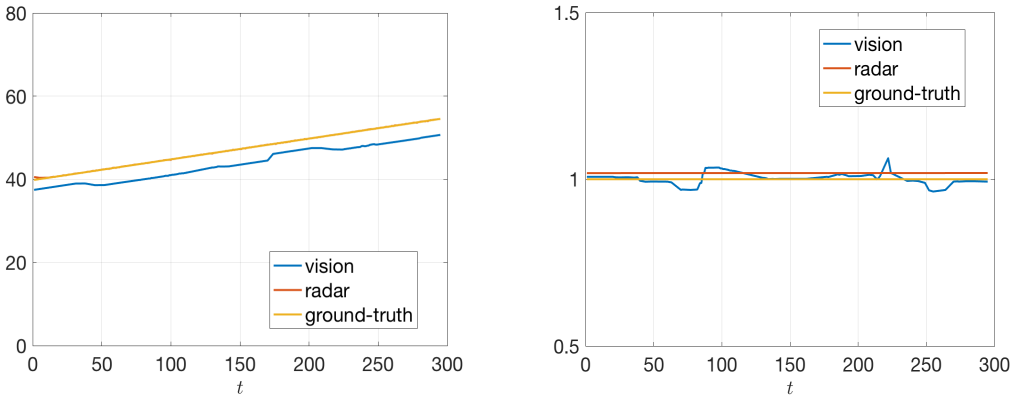
where $Y \in \mathbb{R}^{T\times 8}$ is the matrix of measurements and T is the total number steps. In our experiment $T = 295$. We perform the above outlier detection and replace operation twice to smooth the measurements.

Then, we apply the Matlab "impute" function to interpolate the missing vision measurements. In Fig 39 and 40, we show the preprocessed distance and velocity measurements from vision and radar compared with the ground-truth for both MIO-10 and MIO+1 datasets. Note that after preprocessing, both radar and vision measurements match with the ground-truth well.



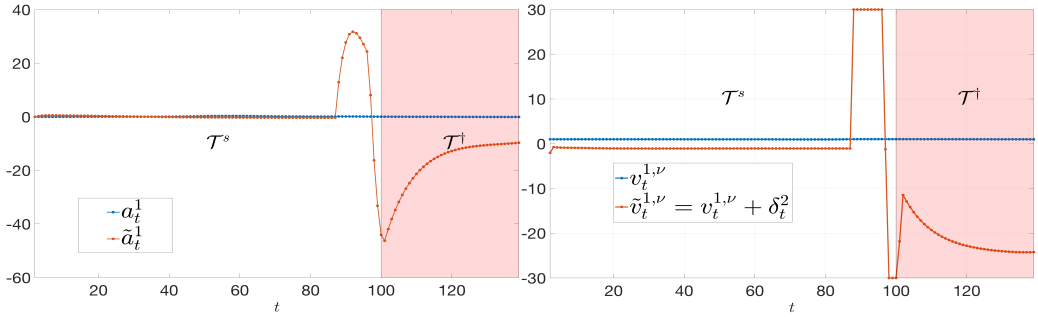
(a) Distance measurements. (b) Velocity measurements.

Figure 39: On the MIO-10 dataset, the preprocessed vision measurements and the radar measurements match the ground-truth reasonably well.



(a) Distance measurements. (b) Velocity measurements.

Figure 40: On the MIO+1 dataset, the preprocessed vision measurements and the radar measurements match the ground-truth reasonably well.



(a) Acceleration estimation. (b) Manipulation on velocity.

Figure 41: Acceleration reduces significantly as the velocity measurement drops after step 96. This in turn causes the KF velocity estimation to decrease fast.

D.4 Velocity Increase in Figure 24c

In Figure 41b, we show again the manipulation on the velocity measurement for the MIO+1 dataset. The attacker's goal is to cause the FCW to output red warnings in the target interval $[100, 139]$. Intuitively, the attacker should decrease the distance and velocity. However, in Figure 41b, the attacker instead chooses to increase the velocity during interval [88,96]. We note that this is because the attacker hopes to force a very negative KF acceleration estimation. To accomplish that, the attacker first strategically increases the velocity from step 88 to 96, and then starting from step 97, the attacker suddenly decreases the velocity dramatically. This misleads the KF to believe that the MIO has a very negative acceleration. In Fig 41a, we show the acceleration estimation produced by KF. At step 96, the estimated acceleration is 8.1m/s^2. However, at step 97, the estimated acceleration suddenly drops to -16m/s^2, and then stays near -30m/s^2 until the target interval. The very negative acceleration in turn causes the KF velocity estimation to decrease quickly. The resulting velocity estimation reached around -10m/s during the target interval, which causes FCW to output red lights.

D.5 Human Behavior Algorithm

In this section, we provide an algorithmic description of the human behavior model.

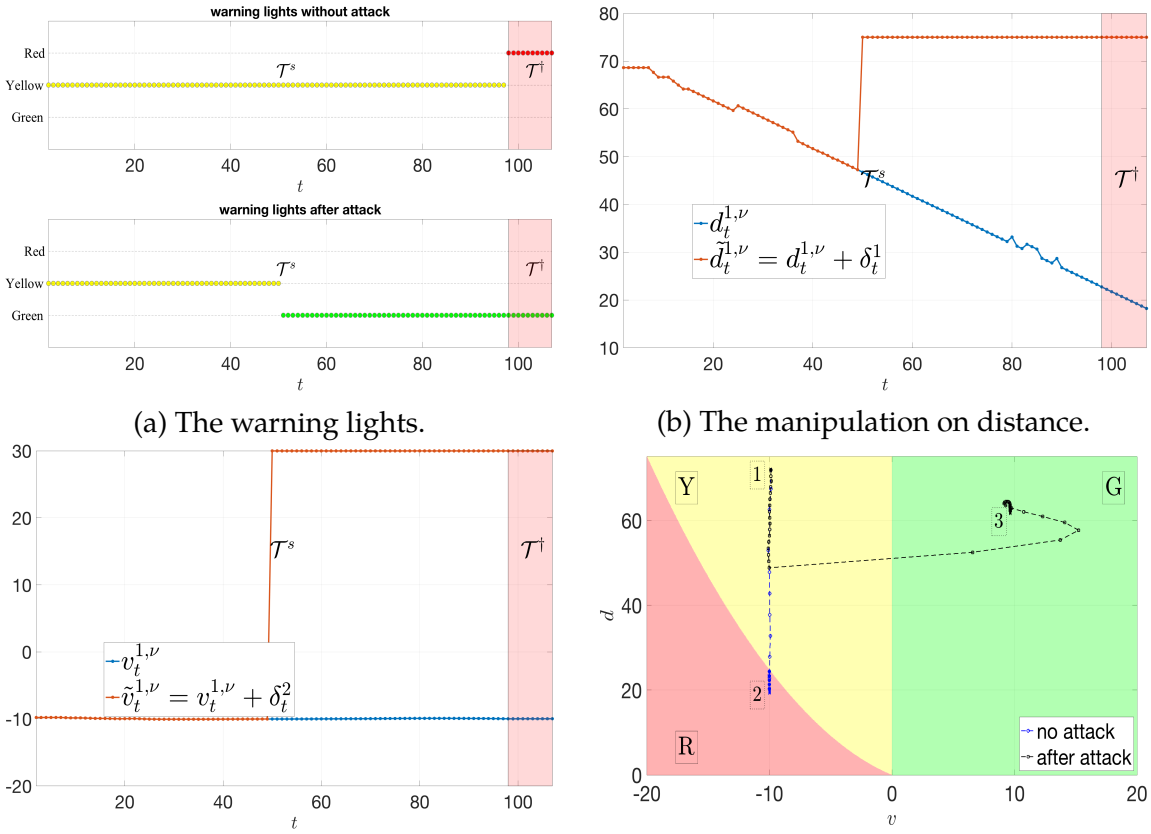
Protocol 10 Human Behavior Algorithm.

Input: light sequence $\ell_t (1 \leqslant t \leqslant T)$, reaction time h^*.
Initialize $s = 0$.
for $t \leftarrow 1, ..., T$ **do**
  if $t! = 1$ and $\ell_t ! = \ell_{t-1}$ **then**
    $s = 0$.
  else if $\ell_t =$ red **then**
    $s = s + 1$
  else
    $s = s - 1$
  end if
  if $\ell_t =$ red **then**
    human applies pressure on pedal
  else if $s \leqslant -h^*$ **then**
    human releases brake
  else
    human stays in the previous state
  end if
end for

D.6 Detailed Results of Greedy Attack

In this section, we provide more detailed results of the greedy attack, including warning lights before and after attack, manipulations on measurements, and the trajectory of KF state predictions. We notice that the results are very similar for different lengths of the stealthy interval $\mathcal{T}^s$. Therefore, here we only show the results for $\mathcal{T}^s = 2.5$ seconds (i.e., half of the full length) as an example.

Fig. 42 shows the greedy attack on MIO-10, where the stealthy interval $\mathcal{T}^s = [50, 97]$. By manipulating the distance and velocity to the maximum possible value, the attacker successfully causes the FCW to output green lights in the target interval



(a) The warning lights.

(b) The manipulation on distance.

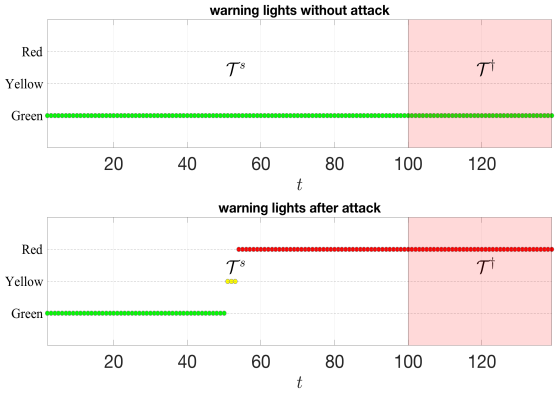
(c) The manipulation on velocity.

(d) The state trajectory.

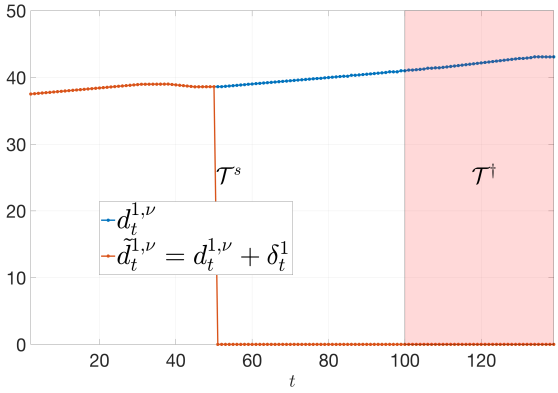
Figure 42: Greedy attack on the MIO-10 dataset.

$\mathcal{T}^\dagger$. However, the attack induces side effect in $\mathcal{T}^s$, where the original yellow lights are changed to green. In contrast, our MPC-based attack does not have any side effect during $\mathcal{T}^s$. Also note that the trajectory of the KF state prediction enters "into" the desired green region during $\mathcal{T}^\dagger$. This is more than necessary and requires larger total manipulation (J_1) than forcing states just on the boundary of the desired region, as does our attack.

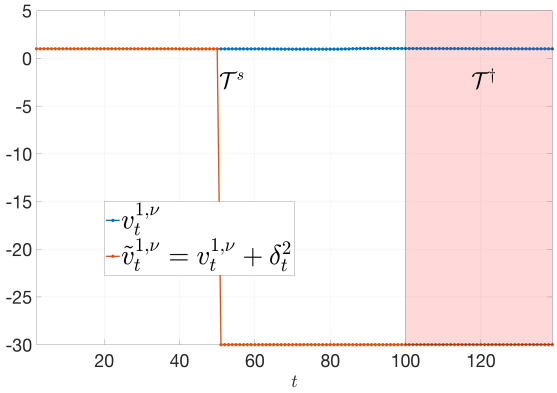
In Fig. 43, we show the greedy attack on MIO+1. The stealthy interval is $\mathcal{T}^s = [51, 99]$. Again, the attack results in side effect during the stealthy interval $\mathcal{T}^s$. Furthermore, the side effect is much more severe (green to red) than that of our MPC-based attack (green to yellow). The KF state trajectory enters "into" the desired



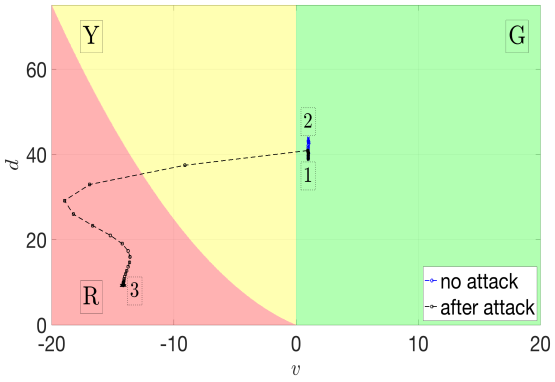
(a) The warning lights.



(b) The manipulation on distance.



(c) The manipulation on velocity.



(d) The state trajectory.

Figure 43: Greedy attack on the MIO+1 dataset.

red region, and requires larger total manipulation (J_1) than our attack.

REFERENCES

A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Sep. 2016. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing*.

Abbasi-Yadkori, Yasin, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. In *Advances in neural information processing systems (nips)*, 2312–2320.

Agarwal, Alekh, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li, Dan Melamed, Gal Oshri, Oswaldo Ribas, Siddhartha Sen, and Alex Slivkins. 2016. Making contextual decisions with low technical debt. CoRR abs/1606.03966.

Agarwal, Alekh, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st international conference on machine learning (icml)*, 1638–1646.

Alfeld, Scott, Xiaojin Zhu, and Paul Barford. 2016. Data poisoning attacks against autoregressive models. In *The 30th aaai conference on artificial intelligence*.

Altschuler, Jason, Victor-Emmanuel Brunel, and Alan Malek. 2019. Best arm identification for contaminated bandits. *Journal of Machine Learning Research* 20(91): 1–39.

Asmuth, John, Michael L Littman, and Robert Zinkov. 2008. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the 23rd national conference on artificial intelligence-volume 2*, 604–609. AAAI Press.

Athalye, Anish, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2–3):235–256.

Auer, Peter, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2002b. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing* 32(1): 48–77.

Auer, Peter, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002c. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32(1):48–77.

Bai, Cheng-Zong, Vijay Gupta, and Fabio Pasqualetti. 2017. On kalman filtering with compromised sensors: Attack stealthiness and performance bounds. *IEEE Transactions on Automatic Control* 62(12):6641–6648.

Banihashem, Kiarash, Adish Singla, and Goran Radanovic. 2021. Defense against reward poisoning attacks in reinforcement learning. *arXiv preprint arXiv:2102.05776*.

Barto, Andrew G. 2013. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, 17–47. Springer.

Behzadan, Vahid, and Arslan Munir. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. In *International conference on machine learning and data mining in pattern recognition*, 262–275. Springer.

Bellemare, Marc, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, 1471–1479.

Biggio, Battista, B Nelson, and P Laskov. 2012. Poisoning attacks against support vector machines. In *29th international conference on machine learning*, 1807–1814. ArXiv e-prints.

Biggio, Battista, and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84:317–331.

Blum, Avrim, and Yishay Monsour. 2007. Learning, regret minimization, and equilibria.

Brown, Daniel S, and Scott Niekum. 2019. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 7749–7758.

Brown, Tom B., Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665*.

Bubeck, Sébastien, and Nicolò Cesa-Bianchi. 2012a. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning* 5:1–122.

Bubeck, Sébastien, and Nicolo Cesa-Bianchi. 2012b. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*.

Cakmak, Maya, and Manuel Lopes. 2012. Algorithmic and human teaching of sequential decision tasks. In *Twenty-sixth aaai conference on artificial intelligence*.

Carlini, Nicholas, and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

Chakraborty, Anirban, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

Chapelle, Olivier, Eren Manavoglu, and Romer Rosales. 2014. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology* 5(4):61:1–61:34.

Chen, Minmin, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the twelfth acm international conference on web search and data mining*, 456–464. ACM.

Chen, Xinyun, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017a. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Chen, Yiding, and Xiaojin Zhu. 2019. Optimal attack against autoregressive models by manipulating the environment. *arXiv preprint arXiv:1902.00202*.

Chen, Yuan, Soummya Kar, and José MF Moura. 2016. Cyber physical attacks with control objectives and detection constraints. In *2016 ieee 55th conference on decision and control (cdc)*, 1125–1130. IEEE.

———. 2017b. Optimal attack strategies subject to detection constraints against cyber-physical systems. *IEEE Transactions on Control of Network Systems* 5(3):1157–1168.

Dean, Sarah, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. 2017. On the sample complexity of the linear quadratic regulator. *arXiv preprint arXiv:1710.01688*.

Devlin, Sam Michael, and Daniel Kudenko. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems*, 433–440. IFAAMAS.

Dhingra, Bhuwan, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Diamond, Steven, and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17(83): 1–5.

Ding, Qin, Cho-Jui Hsieh, and James Sharpnack. 2021. Robust stochastic linear contextual bandits under adversarial attacks. *arXiv preprint arXiv:2106.02978*.

Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. In *CARLA: An open urban driving simulator*, ed. Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, vol. 78 of *Proceedings of Machine Learning Research*, 1–16. PMLR.

Dudek, Gregory, Michael RM Jenkin, Evangelos Milios, and David Wilkes. 1996. A taxonomy for multi-agent robotics. *Autonomous Robots* 3(4):375–397.

Edmund Optics. 2020. Understanding focal length and field of view.

European New Car Assessment Programme. 2018. Euro NCAP LSS Test Protocol. Version 2.0.1.

Even-Dar, Eyal, and Yishay Mansour. 2003. Learning rates for q-learning. *Journal of machine learning Research* 5(Dec):1–25.

Eykholt, Kevin, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018a. Physical adversarial examples for object detectors. In *Proceedings of the 12th usenix conference on offensive technologies*. WOOTâŁ™18.

Eykholt, Kevin, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018b. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Computer vision and pattern recognition (cvpr)*.

Figura, Martin, Krishna Chaitanya Kosaraju, and Vijay Gupta. 2021. Adversarial attacks in consensus-based multi-agent reinforcement learning. *arXiv preprint arXiv:2103.06967*.

Fu, Hu. 2018. Notes on (coarse) correlated equilibrium and swap regret.

Fujimoto, Scott, Herke van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Garcelon, Evrard, Baptiste Roziere, Laurent Meunier, Olivier Teytaud, Alessandro Lazaric, and Matteo Pirotta. 2020. Adversarial attacks on linear contextual bandits. *arXiv preprint arXiv:2002.03839*.

Gleave, Adam, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2019. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Greenewald, Kristjan, Ambuj Tewari, Susan A. Murphy, and Predrag V. Klasnja. 2017. Action centered contextual bandits. In *Advances in neural information processing systems 30 (nips)*, 5979–5987.

Gu, Shixiang, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 ieee international conference on robotics and automation (icra)*, 3389–3396. IEEE.

Guan, Ziwei, Kaiyi Ji, Donald J Bucci Jr, Timothy Y Hu, Joseph Palombo, Michael Liston, and Yingbin Liang. 2020. Robust stochastic bandit algorithms under probabilistic unbounded adversarial attack. In *Proceedings of the aaai conference on artificial intelligence*, vol. 34, 4036–4043.

Huang, Ling, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. 2011. Adversarial machine learning. In *Proceedings of the 4th acm workshop on security and artificial intelligence*, 43–58. ACM.

Huang, Sandy, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

Huang, Yunhan, and Quanyan Zhu. 2019. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International conference on decision and game theory for security*, 217–237. Springer.

Jagielski, Matthew, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. *arXiv preprint arXiv:1804.00308*.

Jia, Yunhan, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei. 2020. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. *2020 International Conference on Learning Representations (ICLR)*.

Jin, Chi, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is q-learning provably efficient? In *Advances in neural information processing systems*, 4863–4873.

Joseph, Anthony D., Blaine Nelson, Benjamin I. P. Rubinstein, and J.D. Tygar. 2018. *Adversarial machine learning*. Cambridge University Press.

Joseph A. Gregor. 2017. Tesla driver assistance system.

Jun, Kwang-Sung, Lihong Li, Yuzhe Ma, and Jerry Zhu. 2018. Adversarial attacks on stochastic bandits. In *Advances in neural information processing systems*, 3640–3649.

Kamalaruban, P, R Devidze, V Cevher, and A Singla. 2019. Interactive teaching algorithms for inverse reinforcement learning. In *28th international joint conference on artificial intelligence*, 604–609.

Kearns, Michael, and Satinder Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning* 49(2-3):209–232.

Koh, Pang Wei, Jacob Steinhardt, and Percy Liang. 2018. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*.

Kos, Jernej, and Dawn Song. 2017. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*.

Kuhn, H. W. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly, vol. 2, no. 1âĽ“2*.

Kuleshov, Volodymyr, and Doina Precup. 2014. Algorithms for multi-armed bandit problems. CoRR abs/1402.6028.

Kung, Enoch, Subhrakanti Dey, and Ling Shi. 2016. The performance and limitations of ϵ-stealthy attacks on higher order systems. *IEEE Transactions on Automatic Control* 62(2):941–947.

Kutschinski, Erich, Thomas Uthmann, and Daniel Polani. 2003. Learning competitive pricing strategies by multi-agent reinforcement learning. *Journal of Economic Dynamics and Control* 27(11-12):2207–2218.

Kveton, Branislav, Csaba Szepesvári, Zheng Wen, and Azin Ashkan. 2015. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd international conference on machine learning (icml)*, 767–776.

Lawler, Gregory F. 1986. Expected hitting times for a random walk on a connected graph. *Discrete mathematics* 61(1):85–92.

Li, Bo, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016a. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, 1885–1893.

Li, Jiwei, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Li, Lihong, Wei Chu, John Langford, and Robert E Schapire. 2010a. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on world wide web*, 661–670.

Li, Lihong, Wei Chu, John Langford, and Robert E. Schapire. 2010b. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the nineteenth international conference on world wide web (www)*, 661–670.

Lin, Yen-Chen, 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. *arXiv preprint arXiv:1703.06748*.

Liu, Fang, and Ness Shroff. 2019. Data poisoning attacks on stochastic bandits. In *International conference on machine learning*, 4042–4050.

Liu, Guanlin, and Lifeng Lai. 2020. Action-manipulation attacks on stochastic bandits. In *Icassp 2020-2020 ieee international conference on acoustics, speech and signal processing (icassp)*, 3112–3116. IEEE.

Lu, Shiyin, Guanghui Wang, and Lijun Zhang. 2021. Stochastic graphical bandits with adversarial corruptions. In *Proceedings of the aaai conference on artificial intelligence*, vol. 35, 8749–8757.

Ma, Yuzhe, Kwang-Sung Jun, Lihong Li, and Xiaojin Zhu. 2018. Data poisoning attacks in contextual bandits. In *International conference on decision and game theory for security*, 186–204. Springer.

Ma, Yuzhe, Jon Sharp, Ruizhe Wang, Earlence Fernandes, and Xiaojin Zhu. 2020. Sequential attacks on kalman filter-based forward collision warning systems. *arXiv preprint arXiv:2012.08704*.

Ma, Yuzhe, Xuezhou Zhang, Wen Sun, and Jerry Zhu. 2019. Policy poisoning in batch reinforcement learning and control. In *Advances in neural information processing systems*, 14570–14580.

Mannion, Patrick, Karl Mason, Sam Devlin, Jim Duggan, and Enda Howley. 2016. Dynamic economic emissions dispatch optimisation using multi-agent reinforcement learning. In *Proceedings of the adaptive and learning agents workshop (at aamas 2016)*.

MATLAB. 2020a. Detect vehicles using yolo v2 network - matlab vehicledetectoryolov2.

———. 2020b. Forward collision warning using sensor fusion.

Mei, Shike, and Xiaojin Zhu. 2015a. The security of latent Dirichlet allocation. In *The 18th international conference on artificial intelligence and statistics (aistats)*.

———. 2015b. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-ninth aaai conference on artificial intelligence*.

Melo, Francisco S. 2001. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep* 1–4.

Murray, S. 2017. Real-time multiple object tracking - a study on the importance of speed. *arXiv:1709.03572 [cs]*.

National Highway Traffic Safety Administration. 2011. A test track protocol for assessing forward collision warning driver-vehicle interface effectiveness.

———. 2020. Common driver assistance technologies.

Neff, Gina, and Peter Nagy. 2016. Talking to bots: Symbiotic agency and the case of tay. *International Journal of Communication* 10:17.

Ng, Andrew Y, Daishi Harada, and Stuart Russell. 1999a. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, vol. 99, 278–287.

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell. 1999b. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th international conference on machine learning (icml)*, 278–287.

Niss, Laura, and Ambuj Tewari. 2019. What you see may not be what you get: Ucb bandit algorithms robust to ?-contamination. *CoRR, abs/1910.05625*.

Oudeyer, Pierre-Yves, and Frederic Kaplan. 2009. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics* 1:6.

Papernot, Nicolas, 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)*, 372–387. IEEE.

Peltola, Tomi, Mustafa Mert Çelikok, Pedram Daee, and Samuel Kaski. 2019. Machine teaching of active sequential learners. In *Advances in neural information processing systems*, 11202–11213.

Peters, Jan, Sethu Vijayakumar, and Stefan Schaal. 2003. Reinforcement learning for humanoid robotics. In *Proceedings of the third ieee-ras international conference on humanoid robots*, 1–20.

R. Collins. 2007. Lecture 12: Camera projection.

Rakhsha, Amin, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2020. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. *arXiv preprint arXiv:2003.12909*.

Redmon, Joseph, S. Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 779–788.

Schmidhuber, Jürgen. 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, 222–227.

Shafahi, Ali, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in neural information processing systems*, 6103–6113.

Sharif, Mahmood, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition.

In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 1528âŁ“1540. CCS âŁ™16.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550(7676): 354–359.

Slivkins, Aleksandrs. 2019. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*.

Smart, William D, and L Pack Kaelbling. 2002. Effective reinforcement learning for mobile robots. In *Proceedings 2002 ieee international conference on robotics and automation (cat. no. 02ch37292)*, vol. 4, 3404–3410. IEEE.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575(7782):350–354.

Vorotnikov, Sergey, Konstantin Ermishin, Anaid Nazarova, and Arkady Yuschenko. 2018. Multi-agent robotic systems in collaborative robotics. In *International conference on interactive collaborative robotics*, 270–279. Springer.

Wang, Yizhen, and Kamalika Chaudhuri. 2018. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*.

Wiewiora, Eric. 2003. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research* 19:205–208.

Wikipedia contributors. 2021. Volunteer's dilemma — Wikipedia, the free encyclopedia. [Online; accessed 17-September-2021].

Xiao, Huang, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning? In *International conference on machine learning*, 1689–1698.

Yang, Lin, Mohammad Hajiesmaili, Mohammad Sadegh Talebi, John Lui, and Wing Shing Wong. 2021. Adversarial bandits with corruptions: Regret lower bound and no-regret algorithm. In *Advances in neural information processing systems (neurips)*.

Yang, Qingyu, Liguo Chang, and Wei Yu. 2016. On false data injection attacks against kalman filtering in power system dynamic state estimation. *Security and Communication Networks* 9(9):833–849.

Yom-Tov, Elad, Guy Feraru, Mark Kozdoba, Shie Mannor, Moshe Tennenholtz, and Irit Hochberg. 2017. Encouraging physical activity in patients with diabetes: intervention using a reinforcement learning system. *Journal of medical Internet research* 19(10):e338.

Yu, Chao, Jiming Liu, and Shamim Nemati. 2019. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*.

Zhang, Haoqi, and David C Parkes. 2008. In *Value-based policy teaching with active indirect elicitation.*

Zhang, Haoqi, David C Parkes, and Yiling Chen. 2009. Policy teaching through reward function learning. In *Proceedings of the 10th acm conference on electronic commerce*, 295–304.

Zhang, Ruochi, and Parv Venkitasubramaniam. 2016. Stealthy control signal attacks in vector lqg systems. In *2016 american control conference (acc)*, 1179–1184. IEEE.

Zhang, Xuezhou, Yiding Chen, Jerry Zhu, and Wen Sun. 2021a. Corruption-robust offline reinforcement learning. *arXiv preprint arXiv:2106.06630*.

Zhang, Xuezhou, Yiding Chen, Xiaojin Zhu, and Wen Sun. 2021b. Robust policy gradient against strong data corruption. *arXiv preprint arXiv:2102.05800*.

Zhang, Xuezhou, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. 2020. Adaptive reward-poisoning attacks against reinforcement learning. *arXiv preprint arXiv:2003.12613*.

Zhang, Xuezhou, and Xiaojin Zhu. 2019. Online data poisoning attack. *arXiv preprint arXiv:1903.01666*.

Zhao, Mengchen, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. 2018a. Data poisoning attacks on multi-task relationship learning. In *Proceedings of the 32nd aaai conference on artificial intelligence*, 2628–2635.

Zhao, Xiangyu, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018b. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th acm conference on recommender systems*, 95–103. ACM.

Zheng, Stephan, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C Parkes, and Richard Socher. 2020. The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332*.

Zhong, Zixin, Wang Chi Cheung, and Vincent Tan. 2021. Probabilistic sequential shrinking: A best arm identification algorithm for stochastic bandits with corruptions. In *International conference on machine learning*, 12772–12781. PMLR.

Zhu, Xiaojin. 2015. 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)*.

Zhu, Xiaojin, Adish Singla, Sandra Zilles, and Anna N. Rafferty. 2018. An Overview of Machine Teaching. *ArXiv e-prints*. Https://arxiv.org/abs/1801.05927, 1801.05927.

Zuo, Shiliang. 2020. Near optimal adversarial attack on ucb bandits. *arXiv preprint arXiv:2008.09312*.