ADVERSARIAL LEARNING IN SEQUENTIAL DECISION MAKING

by

Xuezhou Zhang

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: 06/24/2021

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

Xiaojin Zhu, Professor, Computer Sciences
Stephen J. Wright, Professor, Computer Sciences
Robert D. Nowak, Professor, Electrical Engineering
Laurent Lessard, Associate Professor, Northeastern University
Lihong Li, Senior Principal Scientist, Amazon

To my parents, Xiudong Zhang and Xiaohong Zhang.

LIST OF TABLES

1 INTRODUCTION

This thesis provides an overview of recent results in adversarial online learning due to myself and my collaborators. The key question is the following: Consider an online learning setting, e.g. online supervised learning, bandits, or reinforcement learning, in which we define 3 entities: the underlying environment, the adversary, and the learning agent. During the learning process, the agent tries to achieve the learning goal by interacting with the environment, whereas the adversary desires to mislead the learner to achieve an attack goal by contaminating the interaction between the agent and the environment.

This general setting of adversarial learning can be viewed as a two-player game between the attacker and the learner. The learner plays first by choosing a learning strategy. Observing the learning strategy and the environment, the attacker plays second by choosing an attack strategy. Notice that there is an information gap between the attacker and the learner. A (white-box) attacker observes both the strategy of the learner and the mechanics of the environment and therefore can find the corresponding optimal strategy directly. On the other hand, the learner observes neither the environment's mechanics (otherwise there is no need to learn) nor the attacker's strategy/utility, and therefore has no better choice than deploying a min-max strategy, hoping to maximize its utility against the worst-case adversary.

Traditionally, this problem has been extensively studied in the offline learning setting. Taking supervised learning as an example, where the environment generates a pool of i.i.d. data from the underlying distribution D. The attacker has the power to contaminate the data by injection/deletion/modification, and the learner must perform learning on the contaminated data. Prior work has studied this problem from both the attacker's side and the learner's side. On the attacker's side, this is sometimes referred to as the *data poisoning problem*. It is shown that the optimal attack problem can be formulated as a bi-level optimization problem [122] and a computationally efficient attack is possible when the learning problem is convex. On the learner's side, the majority of the work focuses on designing robust learning algorithms that aim to recover the underlying distribution of D despite the contamination. This is traditionally studied in the field of *robust statistics* going back at least to Tukey [162]. Recently, it has been shown that a computationally efficient robust estimator exists in high-dimensional settings [51].

However, many real-world machine learning systems, such as recommendation systems, stock market forecasting, and automatic logistics planning, need to constantly adapt to the changing environment in an online fashion. Looking into the future, the grand goal of artificial intelligence is to design learning systems that can automatically learn and adapt in the ever-changing open world. These all call for the study of adversarial learning and robust learning in the online learning context, for which very little prior work exists. When it comes to online learning, instead of performing one contamination and learning action, now both the attacker and the learner need to make a sequence

of decisions throughout the learning process. On the attacker's side, we will show that instead of a bilevel optimization problem, now the optimal attack problem can be formulated as an optimal control problem. On the learner's side, this sequential nature gives rise to several unique challenges to robust learning compared to the offline setting:

1. **Robust estimation under time-varying distribution**: In online learning, typically the underlying distribution is also evolving, sometimes depending on the agent's action. Performing robust estimation under this moving distribution and with adversarial noise can be challenging;

2. **Contamination robust exploration**: An important theme unique to online learning is the need for exploration, where the agent must learn to collect data adaptively to more efficiently learn the optimal policy;

In the first half of this work, we study the optimal attack problem from the attacker's perspective, showing that the optimal attack problem can be formulated as an optimal control problem in both online supervised learning (Chapter 2, 3) and reinforcement learning (Chapter 4, 5) settings. On the theoretical side, we derive necessary and sufficient conditions under which attacks can succeed. In the second half, we switch to the learner's perspective and aim at designing robust reinforcement learning algorithms against adversarial corruptions in both offline (Chapter 6) and online RL (Chapter 7) settings.

In our first chapter, we study the strategy of an unconstrained omnipotent attacker who can provide arbitrary legitimate data to the learner, completely bypassing the environment. While such assumptions is obvious strong and make defenses hopeless, our main goal in this chapter is to lay down the mathematical foundation of the optimal attack problem, and we leave the discussion of more realistic and constrained adversary to the next chapter. As a demonstrating example, in this chapter, we will focus on a learner that performs online gradient descent with a squared loss. Such omnipotent attacker is mathematically equivalent to a teacher the *machine teaching* setting. Under this setting, we show that one can use the Pontryagin principle from optimal control theory to derive a closed-form solution to the optimal attack/teaching strategy.

2.1 An optimal control view of sequential machine teaching

Machine teaching studies optimal control on machine learners [196, 194]. In controls language the plant is the learner, the state is the model estimate, and the input is the (not necessarily $i.i.d.$) training data. The controller wants to use the least number of training items—a concept known as the teaching dimension [67]—to force the learner to learn a target model. For example, in adversarial learning, an attacker may minimally poison the training data to force a learner to learn a nefarious model [24, 123]. Conversely, a defender may immunize the learner by injecting adversarial training examples into the training data [68]. In education systems, a teacher may optimize the training curriculum to enhance student (modeled as a learning algorithm) learning [152, 137].

Machine teaching problems are either batch or sequential depending on the learner. The majority of prior work studied batch machine teaching, where the controller performs one-step control by giving the batch learner an input training *set*. Modern machine learning, however, extensively employs sequential learning algorithms. We thus study sequential machine teaching: what is the shortest training sequence to force a learner to go from an initial model $\mathbf{w}_0$ to some target model $\mathbf{w}_\star$? Formally, at time $t = 0, 1, \ldots$ the controller chooses input $(\mathbf{x}_t, y_t)$ from an input set $\mathcal{U}$. The learner then updates the model according to its learning algorithm. This forms a dynamical system f:

$$\mathbf{w}_{t+1} = f(\mathbf{w}_t, \mathbf{x}_t, y_t). \tag{2.1a}$$

The controller has full knowledge of $\mathbf{w}_0, \mathbf{w}_\star, f, \mathcal{U}$, and wants to minimize the terminal time T subject to $\mathbf{w}_T = \mathbf{w}_\star$. As a concrete example, we focus on teaching a gradient descent learner of least squares:

$$f(\mathbf{w}_t, \mathbf{x}_t, y_t) = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t \tag{2.1b}$$

with $\mathbf{w} \in \mathbb{R}^n$ and the input set $\|\mathbf{x}\| \leq R_x, |y| \leq R_y$. We caution the reader not to trivialize the

problem: (2.1) is a *nonlinear* dynamical system due to the interaction between $\mathbf{w}_t$ and $\mathbf{x}_t$. A previous best attempt to solve this control problem by [115] employs a greedy control policy, which at step t optimizes $\mathbf{x}_t, y_t$ to minimize the distance between $\mathbf{w}_{t+1}$ and $\mathbf{w}_\star$. One of our observations is that this greedy policy can be substantially suboptimal. Figure 2.1 shows three teaching problems and the number of steps T to arrive at $\mathbf{w}_\star$ using different methods. Our optimal control method formulated as Nonlinear Programming (NLP) found shorter teaching sequences compared to the greedy policy (lengths 151, 153, 259 for NLP vs 219, 241, 310 for GREEDY, respectively). This and other experiments are discussed in Section 2.3.

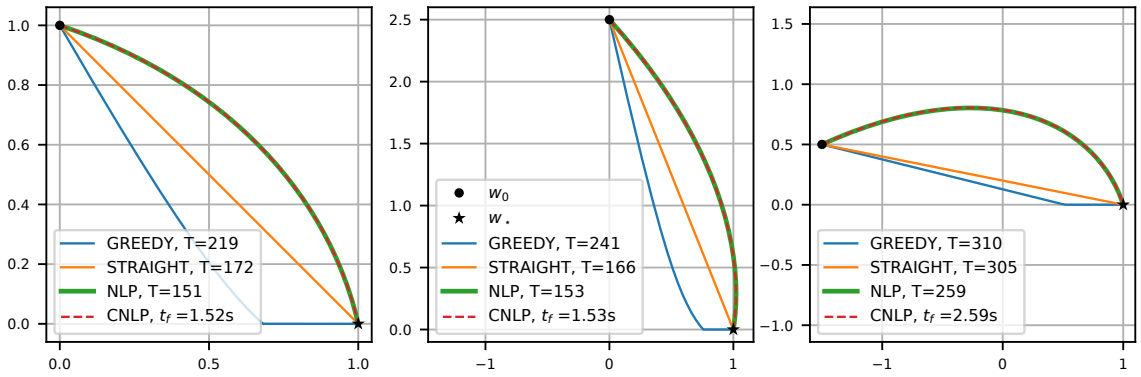


Figure 2.1: The shortest teaching trajectories found by different methods. All teaching tasks use the terminal point $\mathbf{w}_\star = (1, 0)$. The initial points used are $\mathbf{w}_0 = (0, 1)$ (left panel), $\mathbf{w}_0 = (0, 2.5)$ (middle panel), and $\mathbf{w}_0 = (-1.5, 0.5)$ (right panel). The learner is the least squares gradient descent algorithm (2.1) with $\eta = 0.01$ and $R_x = R_y = 1$. Total steps T to arrive at $\mathbf{w}_\star$ is indicated in the legends.

Time-optimal control

To study the structure of optimal control we consider the continuous *gradient flow* approximation of gradient descent, which holds in the limit of diminishing step size, i.e. $\eta \to 0$. In this section, we present the corresponding canonical time-optimal control problem and summarize some of the key theoretical and computational tools in optimal control that address it. For a more detailed exposition on the theory, we refer the reader to modern references on the topic [93, 110, 13].

This section is self-contained and we will use notation consistent with the control literature (x instead of $\mathbf{w}$, u instead of $(\mathbf{x}, y)$, t_f instead of T). We revert back to machine learning notation in section 2.2. Consider the following boundary value problem:

$$\dot{x} = f(x, u) \qquad \text{with } x(0) = x_0 \text{ and } x(t_f) = x_f. \tag{2.2}$$

The function $x : \mathbb{R}_+ \to \mathbb{R}^n$ is called the *state* and $u : \mathbb{R}_+ \to \mathcal{U}$ is called the *input*. Here, $\mathcal{U} \subseteq \mathbb{R}^m$ is

a given constraint set that characterizes admissible inputs. The initial and terminal states x_0 and x_f are fixed, but the terminal time t_f is free. If an admissible u together with a state x satisfy the boundary value problem (2.2) for some choice of t_f, we call (x, u) a *trajectory* of the system. The objective in a time-optimal control problem is to find an *optimal trajectory*, which is a trajectory that has minimal t_f.

Established approaches for solving time-optimal control problems can be grouped in three broad categories: dynamic programming, indirect methods, and direct methods. We now summarize each approach.

Dynamic Programming Consider the value function $V : \mathbb{R}^n \to \mathbb{R}_+$, where $V(x)$ is the minimum time required to reach x_f starting at the initial state x. The Hamilton–Jacobi–Bellman (HJB) equation gives necessary and sufficient conditions for optimality and takes the form:

$$\min_{\tilde{u} \in \mathcal{U}} \ \nabla V(x)^{\mathsf{T}} f(x, \tilde{u}) + 1 = 0 \qquad \text{for all } x \in \mathbb{R}^n \tag{2.3}$$

together with the boundary condition $V(x_f) = 0$. If the solution to this differential equation is $V_\star$, then the optimal input is given by the minimizer:

$$u(x) \in \operatorname*{arg\,min}_{\tilde{u} \in \mathcal{U}} \ \nabla V_\star(x)^{\mathsf{T}} f(x, \tilde{u}) \quad \text{for all } x \in \mathbb{R}^n \tag{2.4}$$

A nice feature of this solution is that the optimal input u depends on the current state x. In other words, HJB produces an optimal *feedback policy*.

Unfortunately, the HJB equation (2.3) is generally difficult to solve. Even if the minimization has a closed form solution, the resulting differential equation is often intractable. We remark that the optimal $V_\star$ may not be differentiable. For this reason, one looks for so-called *viscosity solutions*, as described by [110, 159] and references therein.

Numerical approaches for solving HJB include the fast-marching method [161] and Lax–Friedrichs sweeping [90]. The latter reference also contains a detailed survey of other numerical schemes.

Indirect Methods Also known as "optimize then discretize", indirect approaches start with necessary conditions for optimality obtained via the Pontryagin Maximum Principle (PMP). The PMP may be stated and proved in several different ways, most notably using the Hamiltonian formalism from physics or using the calculus of variations. Here is a formal statement.

Theorem 2.1.1 (PMP)**.** *Consider the boundary value problem* (2.2) *where f and its Jacobian with respect to x are continuous on $\mathbb{R}^n \times \mathcal{U}$. Define the Hamiltonian $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}$ as $H(x, p, u) := p^{\mathsf{T}} f(x, u) + 1$. If $(x^\star, u^\star)$ is an optimal trajectory, then there exists some function $p^\star : \mathbb{R}_+ \to \mathbb{R}^n$ (called the "co-state") such that the following conditions hold.*

1. *$x^\star$ and $p^\star$ satisfy the following system of differential equations for $t \in [0, t_f]$ with boundary conditions $x^\star(0) = x_0$ and $x^\star(t_f) = x_f$.*

$$\dot{x}^\star(t) = \frac{\partial H}{\partial p}(x^\star(t), p^\star(t), u^\star(t)), \tag{2.5a}$$

$$\dot{p}^\star(t) = -\frac{\partial H}{\partial x}(x^\star(t), p^\star(t), u^\star(t)). \tag{2.5b}$$

2. *For all $t \in [0, t_f]$, an optimal input $u^\star(t)$ satisfies:*

$$u^\star(t) \in \arg\min_{\tilde{u} \in \mathcal{U}} \; H(x^\star(t), p^\star(t), \tilde{u}). \tag{2.6}$$

3. *Zero Hamiltonian along optimal trajectories:*

$$H(x^\star(t), p^\star(t), u^\star(t)) = 0 \quad \textit{for all } t \in [0, t_f]. \tag{2.7}$$

In comparison to HJB, which needs to be solved for all $x \in \mathbb{R}^n$, the PMP only applies along optimal trajectories. Although the differential equations (2.5) may still be difficult to solve, they are simpler than the HJB equation and therefore tend to be more amenable to both analytical and numerical approaches. Solutions to HJB and PMP are related via $\nabla V^\star(x^\star(t)) = p^\star(t)$.

PMP is only necessary for optimality, so solutions of (2.5)–(2.7) are not necessarily optimal. Moreover, PMP does not produce a feedback policy; it only produces optimal trajectory *candidates*. Nevertheless, PMP can provide useful insight, as we will explore in Section 2.2.

If PMP cannot be solved analytically, a common numerical approach is the *shooting method*, where we guess $p^\star(0)$, propagate the equations (2.5)–(2.6) forward via numerical integration. Then $p^\star(0)$ is refined and the process is repeated until the trajectory reaches x_f.

Direct Methods Also known as "discretize then optimize", a sparse nonlinear program is solved, where the variables are the state and input evaluated at a discrete set of timepoints. An example is *collocation methods*, which use different basis functions such as piecewise polynomials to interpolate the state between timepoints. For contemporary surveys of direct and indirect numerical approaches, see [143, 22].

If the dynamics are already discrete as in (2.1), we may directly formulate a nonlinear program. We refer to this approach as NLP. Alternatively, we can take the continuous limit and then discretize, which we call CNLP. We discuss the advantages and disadvantages of both approaches in Section 2.3.

2.2 Teaching least squares: insight from Pontryagin

In this section, we specialize time-optimal control to least squares. To recap, our goal is to find the minimum number of steps T such that there exists a control sequence $(\mathbf{x}_t, y_t)_{0:T-1}$ that drives the learner (2.1) with initial state $\mathbf{w}_0$ to the target state $\mathbf{w}_\star$. The constraint set is

$$\mathcal{U} = \{(\mathbf{x}, y) \mid \|\mathbf{x}\| \leq R_x, \, |y| \leq R_y\}$$

This is an *nonlinear discrete-time time-optimal control problem*, for which no closed-form solution is available. On the corresponding continuous-time control problem, applying Theorem 2.1.1 we obtain the following necessary conditions for optimality[1] for all $t \in [0, t_f]$.

$$\mathbf{w}(0) = \mathbf{w}_0, \quad \mathbf{w}(t_f) = \mathbf{w}_\star \tag{2.8a}$$

$$\dot{\mathbf{w}}(t) = (y(t) - \mathbf{w}(t)^\mathsf{T}\mathbf{x}(t))\,\mathbf{x}(t) \tag{2.8b}$$

$$\dot{\mathbf{p}}(t) = (\mathbf{p}(t)^\mathsf{T}\mathbf{x}(t))\,\mathbf{x}(t) \tag{2.8c}$$

$$\mathbf{x}(t), y(t) \in \underset{\|\hat{\mathbf{x}}\| \leq R_x, \, |\hat{y}| \leq R_y}{\arg\min} (\hat{y} - \mathbf{w}(t)^\mathsf{T}\hat{\mathbf{x}})(\mathbf{p}(t)^\mathsf{T}\hat{\mathbf{x}}) \tag{2.8d}$$

$$0 = (y(t) - \mathbf{w}(t)^\mathsf{T}\mathbf{x}(t))(\mathbf{p}(t)^\mathsf{T}\mathbf{x}(t)) + 1 \tag{2.8e}$$

We can simplify (2.8) by setting $y(t) = R_y$, as described in Proposition 2.2.1 below.

Proposition 2.2.1. *For any trajectory $(\mathbf{w}, \mathbf{p}, \mathbf{x}, y)$ satisfying (2.8), there exist another trajectory of the form $(\mathbf{w}, \mathbf{p}, \tilde{\mathbf{x}}, R_y)$. So we may set $y(t) = R_y$ without any loss of generality.*

Proof. Since (2.8d) is linear in $\hat{y}$, the optimal $\hat{y}$ occurs at a boundary and $\hat{y} = \pm R_y$. Changing the sign of $\hat{y}$ is equivalent to changing the sign of $\hat{\mathbf{x}}$, so we may assume without loss of generality that $\hat{y} = R_y$. These changes leave (2.8b)–(2.8c) and (2.8e) unchanged so $\mathbf{w}$ and $\mathbf{p}$ are unchanged as well. ■

In fact, Proposition 2.2.1 holds if we consider trajectories of (2.1) as well. For a proof, see the appendix. Applying Proposition 2.2.1, the conditions (2.8d) and (2.8e) may be combined to yield the following quadratically constrained quadratic program (QCQP) equation.

$$\min_{\|\mathbf{x}\| \leq R_x} (R_y - \mathbf{w}^\mathsf{T}\mathbf{x})(\mathbf{p}^\mathsf{T}\mathbf{x}) = -1 \tag{2.9}$$

where we have omitted the explicit time specification (t) for clarity. Note that (2.9) constrains the possible tuples $(\mathbf{w}, \mathbf{p}, \mathbf{x})$ that can occur as part of an optimal trajectory. So in addition to solving

[1] State, co-state, and input in Theorem 2.1.1 are (x, p, u), which is conventional controls notation. For this problem, we use $(\mathbf{w}, \mathbf{p}, (\mathbf{x}, y))$, which is machine learning notation.

the left-hand side to find $\mathbf{x}$, we must also ensure that it's equal to -1. We will now characterize the solutions of (2.9) by examining five distinct regimes of the solution space that depend on the relationship between $\mathbf{w}$ and $\mathbf{p}$ as well as which regime transitions are admissible.

Regime I (Origin): $\mathbf{w} = 0$ and $\mathbf{p} \neq 0$. This regime happens when the teaching trajectory pass through the origin. In this regime, one can obtain closed-form solutions. In particular, $\mathbf{x} = -\frac{R_x}{\|\mathbf{p}\|}\mathbf{p}$ and $\|\mathbf{p}\| = \dfrac{1}{R_x R_y}$. In this regime, both $\dot{\mathbf{w}}$ and $\dot{\mathbf{p}}$ are positively aligned with $\mathbf{p}$. Therefore, Regime I necessarily *transitions* from Regime II and into Regime III, given that it is not at the beginning or the end of the teaching trajectory.

Regime II (positive alignment): $\mathbf{w} = \alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$. This regime happens when $\mathbf{w}$ and $\mathbf{p}$ are positively aligned. Again we have closed form solutions. In particular, $\mathbf{x}^\star = -\dfrac{R_x}{\|\mathbf{w}\|}\mathbf{w}$ and $\alpha = R_x\|\mathbf{w}\|(R_y + R_x\|\mathbf{w}\|)$. In this regime, both $\dot{\mathbf{w}}$ and $\dot{\mathbf{p}}$ are negatively aligned with $\mathbf{w}$, thus Regime II necessarily transitions into Regime I and can never transition from any other regimes.

Regime III (negative alignment inside the origin-centered ball): $\mathbf{w} = -\alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$ and $\|\mathbf{w}\| \leq \frac{R_y}{2R_x}$. This regime happens when $\mathbf{w}$ and $\mathbf{p}$ are negatively aligned and $\mathbf{w}$ is inside the ball centered at the origin with radius $R = \frac{R_y}{2R_x}$. Again, closed form solutions exists: $\mathbf{x}^\star = \dfrac{R_x}{\|\mathbf{w}\|}\mathbf{w}$ and $\alpha = R\|\mathbf{w}\|(1 - R\|\mathbf{w}\|)$. Regime III necessarily transitions from Regime I and into Regime IV.

Regime IV (negative alignment out of the origin-centered ball): $\mathbf{w} = -\alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$ and $\|\mathbf{w}\| > \frac{R_y}{2R_x}$. In this case, the solutions satisfies $\alpha = \dfrac{R_y^2}{4}$ so that $\mathbf{p}$ is uniquely determined by $\mathbf{w}$. However, the optimal $\mathbf{x}^\star$ is **not** unique. Any solution to $\mathbf{w}^\mathsf{T}\mathbf{x} = \dfrac{R_y}{2}$ with $\|\mathbf{x}\| \leq R_x$ can be chosen. Regime IV can only transition from Regime III and cannot transition into any other regime. In other word, once the teaching trajectory enters Regime IV, it cannot escape. Another interesting property of Regime IV is that we know exactly how fast the norm of $\mathbf{w}$ is changing. In particular, knowing $\mathbf{w}^\mathsf{T}\mathbf{x} = \dfrac{R_y}{2}$, one can derive that $\frac{\mathrm{d}\|\mathbf{w}\|^2}{\mathrm{d}t} = \frac{R_y^2}{2}$. As a result, once the trajectory enters regime IV, we know exact how long it will take for the trajectory to reach $\mathbf{w}_\star$, if it is able to reach it.

Regime V (general positions): $\mathbf{w}$ and $\mathbf{p}$ are linearly independent. This case covers the remaining possibilities for the state and co-state variables. To characterize the solutions in this regime, we'll first introduce some new coordinates. Define $\{\hat{\mathbf{w}}, \hat{\mathbf{u}}\}$ to be the orthonormal basis for $\mathrm{span}\{\mathbf{w}, \mathbf{p}\}$ such that $\mathbf{w} = \gamma\hat{\mathbf{w}}$ and $\mathbf{p} = \alpha\hat{\mathbf{w}} + \beta\hat{\mathbf{u}}$ for some $\alpha, \beta, \gamma \in \mathbb{R}$. Note that $\beta \neq 0$ because we assume $\mathbf{w}$ and $\mathbf{p}$ are assumed to be linearly independent in this regime. We can therefore express any input uniquely as $\mathbf{x} = w\hat{\mathbf{w}} + u\hat{\mathbf{u}} + z\hat{\mathbf{z}}$ where $\hat{\mathbf{z}}$ is an *out-of-plane* unit vector orthogonal to both $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$,

and $w, u, z \in \mathbb{R}$ are suitably chosen. Substituting these definitions, (2.9) becomes

$$\min_{w^2+u^2+z^2 \leq R_x^2} \quad (R_y - \gamma w)(\alpha w + \beta u) = -1. \tag{2.10}$$

Now observe that the objective is linear in u and does not depend on z. The objective is linear in u because $\beta \neq 0$ and $(1-\gamma w) \neq 0$ otherwise the entire objective would be zero. Since the feasible set is convex, the optimal u must occur at the boundary of the feasible set of variables w and u. Therefore, $z = 0$. This is profound, because it implies that in Regime V, the optimal solution necessarily lies on the 2D plane $\text{span}\{\mathbf{w}, \mathbf{p}\}$. In light of this fact, we can pick a more convenient parametrization. Let $w = R_x \cos\theta$ and $u = R_x \sin\theta$. Equation (2.10) becomes:

$$\min_{\theta} \quad R_x(R_y - \gamma R_x \cos\theta)(\alpha \cos\theta + \beta \sin\theta) = -1. \tag{2.11}$$

This objective function has at most four critical points, of which there is only one global minimum, and we can find it numerically. Last but not least, Regime V does not transition from or into any other Regime.

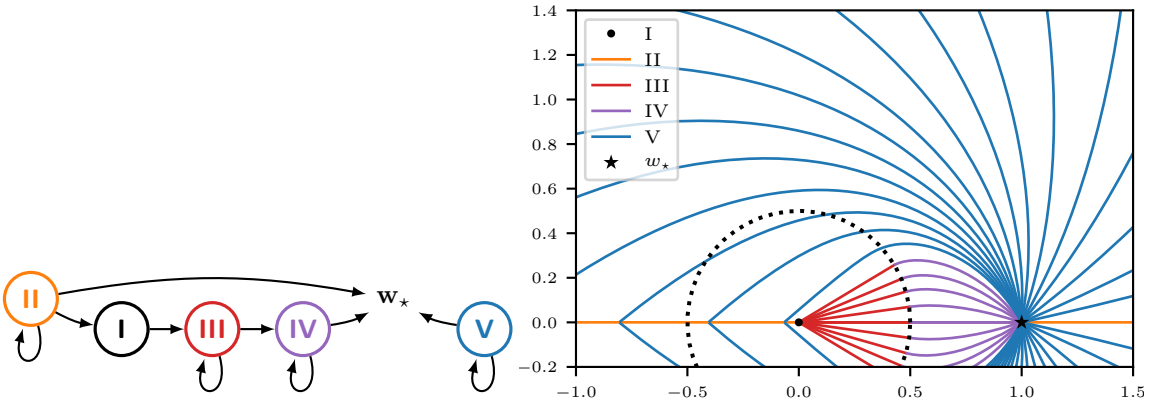


Figure 2.2: Optimal trajectories for $\mathbf{w}_\star = (1, 0)$ for different choices of $\mathbf{w}_0$. Trajectories are colored according to the regime to which they belong and the directed graph above shows all possible transitions. The optimal trajectories are symmetric about the x-axis. For implementation details, see Section 2.3.

Intrinsic low-dimensional structure of the optimal control solution.

As is hinted in the analysis of Regime V, the optimal control $\mathbf{x}$ sometimes lies in the 2D subspace spanned by $\mathbf{w}$ and $\mathbf{p}$. In fact, this holds not only for Regime V but for the whole problem. In particular, we make the following observation.

Theorem 2.2.2. *There always exists a global optimal trajectory of* (2.8) *that lies in a 2D subspace of* $\mathbb{R}^n$.

The detailed proof can be found in the appendix. An immediate consequence of Theorem 2.2.2 is that if $\mathbf{w}_0$ and $\mathbf{w}_\star$ are linearly independent, we only need to consider trajectories that are confined to the subspace $\mathrm{span}\{\mathbf{w}_0, \mathbf{w}_\star\}$. When $\mathbf{w}_0$ and $\mathbf{w}_\star$ are aligned, trajectories are still 2D, and any subspace containing $\mathbf{w}_0$ and $\mathbf{w}_\star$ is equivalent and arbitrary choice can be made.

This insight is extremely important because it enables us to restrict our attention to 2D trajectories even though the dimensionality of the original problem (n) may be huge. This allows us to not only obtain a more elegant and accurate solution in solving the necessary condition induced by PMP, but also to parametrize direct and indirect approaches (see Sections 2.1 and 2.1) to solve this intrinsically 2D problem more efficiently.

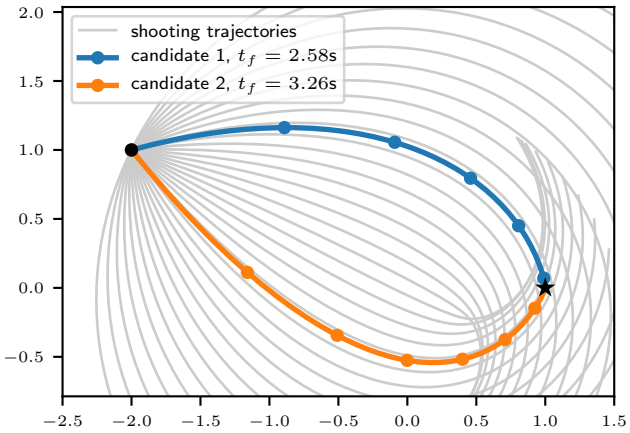


Figure 2.3: Trajectories found using a shooting approach (Section 2.1) with $\mathbf{w}_0 = (-2, 1)$ and $\mathbf{w}_\star = (1, 0)$. Gray curves show different shooting trajectories while the blue and orange curves show two trajectories that satisfy the necessary conditions for optimality (2.8). Markers show intervals of 0.5 seconds, which is roughly 50 steps when using a stepsize of $\eta = 0.01$.

Multiplicity of Solution Candidates.

The PMP conditions are only *necessary* for optimality. Therefore, the optimality conditions (2.8) need not have a unique solution. We illustrate this phenomenon in Figure 2.3. We used a shooting approach (Section 2.1) to propagate different choices of $\mathbf{p}^\star(0)$ forward in time. It turns out two choices lead to trajectories that end at $\mathbf{w}_\star$, and they do not have equal total times. So in general, PMP identifies *optimal trajectory candidates*, which can be thought of as local minima for this highly nonlinear optimization problem.

2.3 Numerical methods

While the PMP yields necessary conditions for time-optimal control as detailed in Section 2.2, there is no closed-form solution in general. We now present and discuss four numerical methods: CNLP and NLP are different implementations of time-optimal control, while GREEDY and STRAIGHT are heuristics.

CNLP: This approach solves the continuous gradient flow limit of the machine teaching problem using a direct approach (Section 2.1). Specifically, we used the NLOptControl package [63], which is an implementation of the hp-pseudospectral method GPOPS-II [138] written in the Julia programming language using the JuMP modeling language [58] and the IPOPT interior-point solver [164]. The main tuning parameters for this software are the integration scheme and the number of mesh points. We selected the trapezoidal integration rule with 100 mesh points for most simulations. We used CNLP to produce the trajectories in Figures 2.1 and 2.2.

NLP: A naïve approach to optimal control is to find the minimum T for which there is a feasible input sequence to drive the learner to $\mathbf{w}_\star$. Fixing T, the feasibility subproblem is a nonlinear program over $2T$ n-dimensional variables $\mathbf{x}_0, \dots, \mathbf{x}_{T-1}$ and $\mathbf{w}_1, \dots, \mathbf{w}_T$ constrained by learner dynamics. Recall $\mathbf{w}_0$ is given, and one can fix $y_t = R_y$ for all t by Proposition 2.2.1. For our learner (2.1), the feasibility problem is

$$\begin{aligned} \min_{\mathbf{w}_{1:T},\, \mathbf{x}_{0:T-1}} \quad & 0 \\ \text{s.t.} \quad & \mathbf{w}_T = \mathbf{w}_\star \\ & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - R_y)\mathbf{x}_t \\ & \|\mathbf{x}_t\| \le R_x, \quad \forall t = 0, \dots, T-1. \end{aligned} \tag{2.12}$$

As in the CNLP case, we modeled and solved the subproblems (2.12) using JuMP and IPOPT. We also tried Knitro, a state-of-the-art commercial solver [33], and it produced similar results. We stress that such feasibility problems are difficult; IPOPT and Knitro can handle moderately sized T. For our specific learner (2.1) there are 2D optimal control and state trajectories in $\mathrm{span}\{\mathbf{w}_0, \mathbf{w}_\star\}$ as discussed in Section 2.2. Therefore, we reparameterized (2.12) to work in 2D.

On top of this, we run a binary search over positive integers to find the minimum T for which the subproblem (2.12) is feasible. Subject to solver numerical stability, the minimum T and its feasibility solution $\mathbf{x}_0, \dots, \mathbf{x}_{T-1}$ is the time-optimal control. While NLP is conceptually simple and correct, it requires solving many subproblems with $2T$ variables and $2T$ constraints, making it less stable and scalable than CNLP.

GREEDY: We restate the greedy control policy initially proposed by [115]. It has the advantage of being computationally more efficient and readily applicable to different learning algorithms (i.e. dynamics). Specifically for the least squares learner (2.1) and given the current state $\mathbf{w}_t$, GREEDY solves the following optimization problem to determine the next teaching example $(\mathbf{x}_t, y_t)$:

$$\begin{aligned} \min_{(\mathbf{x}_t, y_t) \in \mathcal{U}} \quad & \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 \\ \text{s.t.} \quad & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t. \end{aligned} \tag{2.13}$$

The procedure repeats until $\mathbf{w}_{t+1} = \mathbf{w}_\star$. We used the MATLAB function `fmincon` to solve the above quadratic program iteratively. We point out that the optimization problem is not convex. Moreover, $\mathbf{w}_{t+1}$ does not necessarily point in the direction of $\mathbf{w}_\star$. This is evident in Figure 2.1 and Figure 2.6.

STRAIGHT: We describe an intuitive control policy: at each step, move $\mathbf{w}$ in straight line toward $\mathbf{w}_\star$ as far as possible subject to the constraint $\mathcal{U}$. This policy is less greedy than GREEDY because it may not reduce $\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2$ as much at each step. The per-step optimization in $\mathbf{x}$ is a 1D line search:

$$\begin{aligned} \min_{a, y_t \in \mathbb{R}} \quad & \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 \\ \text{s.t.} \quad & \mathbf{x}_t = a(\mathbf{w}_\star - \mathbf{w}_t)/\|\mathbf{w}_\star - \mathbf{w}_t\| \\ & (\mathbf{x}_t, y_t) \in \mathcal{U} \\ & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t. \end{aligned} \tag{2.14}$$

The line search (2.14) can be solved in closed-form. In particular, one can obtain that

$$a = \begin{cases} \min\{R_x, \frac{R_y\|\mathbf{w}_\star - \mathbf{w}\|}{2(\mathbf{w}_\star - \mathbf{w})^\mathsf{T}\mathbf{w}}\}, & \text{if } (\mathbf{w}_\star - \mathbf{w})^\mathsf{T}\mathbf{w} > 0 \\ R_x, & \text{otherwise.} \end{cases}$$

Comparison of Methods We ran a number of experiments to study the behavior of these numerical methods. In all experiments, the learner is gradient descent on least squares (2.1), and the control constraint set is $\|\mathbf{x}\| \leq 1, |y| \leq 1$. Our first observation is that CNLP has a number of advantages:

1. CNLP's continuous optimal state trajectory matches NLP's discrete state trajectories, especially on learners with small η. This is expected, since the continuous optimal control problem is obtained asymptotically from the discrete one as $\eta \to 0$. Figure 2.4 shows the teaching task $\mathbf{w}_0 = (1, 0) \Rightarrow \mathbf{w}_* = (1, 0)$. Here we compare CNLP with NLP's optimal state trajectories on four gradient descent learners with different η values. The NLP optimal teaching sequences

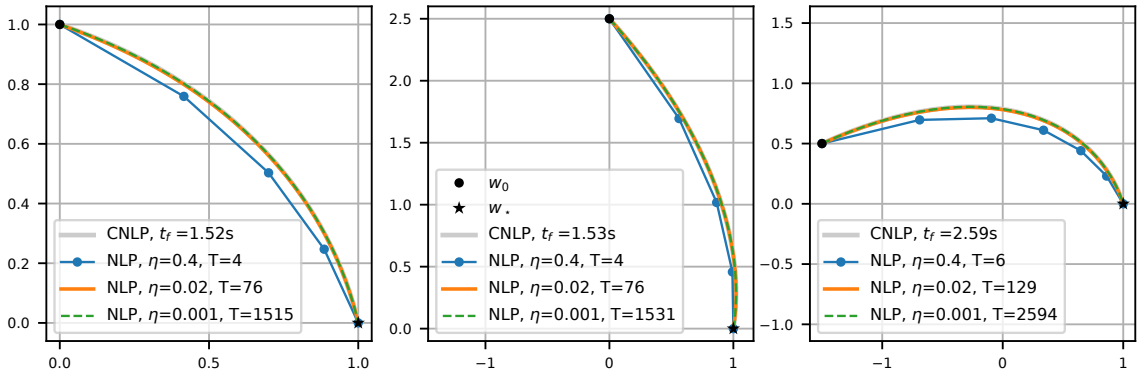


Figure 2.4: Comparison of CNLP vs NLP. All teaching tasks use the terminal point $\mathbf{w}_\star = (1, 0)$. The initial points used are $\mathbf{w}_0 = (0, 1)$ (left panel), $\mathbf{w}_0 = (0, 2.5)$ (middle panel), and $\mathbf{w}_0 = (-1.5, 0.5)$ (right panel). We observe that the NLP trajectories on learners with smaller η's quickly converges to the CNLP trajectory.

vary drastically in length T, but their state trajectories quickly overlap with CNLP's optimal trajectory.

2. CNLP is quick to compute, while NLP runtime grows as the learner's η decreases. Table 2.1 presents the wall clock time. With a small η, the optimal control takes more steps (larger T). Consequently, NLP must solve a nonlinear program with more variables and constraints. In contrast, CNLP's runtime does not depend on η.

3. CNLP can be used to approximately compute the "teaching dimension", i.e. the minimum number of sequential teaching steps T for the discrete problem. Recall CNLP produces an optimal terminal time t_f. When the learner's η is small, the discrete "teaching dimension" T is related by $T \approx t_f/\eta$. This is also supported by Table 2.1.

That said, it is not trivial to extract a discrete control sequence from CNLP's continuous control function. This hinders CNLP's utility as an optimal teacher.

Our second observation is that NLP, being the discrete-time optimal control, produces shorter teaching sequences than GREEDY or STRAIGHT. This is not surprising, and we have already presented three teaching tasks in Figure 2.1 where NLP has the smallest T. In fact, there exist teaching tasks on which GREEDY and STRAIGHT can perform arbitrarily worse than the optimal teaching sequence found by NLP. A case study is presented in Table 2.2. In this set of experiments, we set $\mathbf{w}_0 = (a, 0)$ and $\mathbf{w}_\star = (0, 2a)$. As a increases, the ratio of teaching sequence length between STRAIGHT and NLP and between GREEDY and NLP grow at an exponential rate.

We now dig deeper and present an intuitive explanation of why GREEDY requires more teaching steps than NLP. The fundamental issue is the nonlinearity of the learner dynamics (2.1) in $\mathbf{x}$. For any $\mathbf{w}$ let us define the one-step reachable set $\left\{\mathbf{w} - \eta(\mathbf{w}^\mathsf{T}\mathbf{x} - y)\mathbf{x} \;\middle|\; (\mathbf{x}, y) \in \mathcal{U}\right\}$. Figure 2.5 shows

Table 2.1: Teaching sequence length and wall clock time comparison. NLP teaches three learners with different η's. Target is always $\mathbf{w}_\star = (1, 0)$. All experiments were performed on a conventional laptop.

	NLP			**CNLP**
$\mathbf{w}_0$	$\eta = 0.4$	0.02	0.001	
$(0, 1)$	$T = 3$	75	1499	$t_f = 1.52$s
	0.013s	0.14s	59.37s	4.1s
$(0, 2.5)$	$T = 5$	76	1519	$t_f = 1.53$s
	0.008s	0.11s	53.28s	2.37s
$(-1.5, 0.5)$	$T = 6$	128	2570	$t_f = 2.59$s
	0.012s	0.63s	310.08s	2.11s

Table 2.2: Comparison of teaching sequence length T. We fixed $\eta = 0.01$ in all cases.

$\mathbf{w}_0$	$\mathbf{w}_\star$	**NLP**	**STRAIGHT**	**GREEDY**
$(0, 1)$	$(2, 0)$	148	161	233
$(0, 2)$	$(4, 0)$	221	330	721
$(0, 4)$	$(8, 0)$	292	867	2667
$(0, 8)$	$(16, 0)$	346	2849	10581

a sample of such reachable sets. The key observation is that the starting $\mathbf{w}$ is quite close to the boundary of most reachable sets. In other words, there is often a compressed direction—from $\mathbf{w}$ to the closest boundary of $\mathcal{U}$—along which $\mathbf{w}$ makes minimal progress. The GREEDY scheme falls victim to this phenomenon.

Figure 2.6 compares NLP and GREEDY on a teaching task chosen to have short teaching sequences in order to minimize clutter. GREEDY starts by eagerly descending a slope and indeed this quickly brings it closer to $\mathbf{w}_\star$. Unfortunately, it also arrived at the x-axis. For $\mathbf{w}$ on the x-axis, the compressed direction is horizontally outward. Therefore, subsequent GREEDY moves are relatively short, leading to a large number of steps to reach $\mathbf{w}_\star$. Interestingly, STRAIGHT is often better than GREEDY because it also avoids the x-axis compressed direction for general $\mathbf{w}_0$.

We illustrate the optimal inputs in Figure 2.7, which compares $\{\mathbf{x}_t\}$ produced by STRAIGHT, GREEDY, and NLP and the $\mathbf{x}(t)$ produced by CNLP. The heuristic approaches eventually take smaller-magnitude steps as they approach $\mathbf{w}_\star$ while NLP and CNLP maintain a maximal input norm the whole way.

Concluding remarks

Techniques from optimal control are under-utilized in machine teaching, yet they have the power to provide better quality solutions as well as useful insight into their structure.

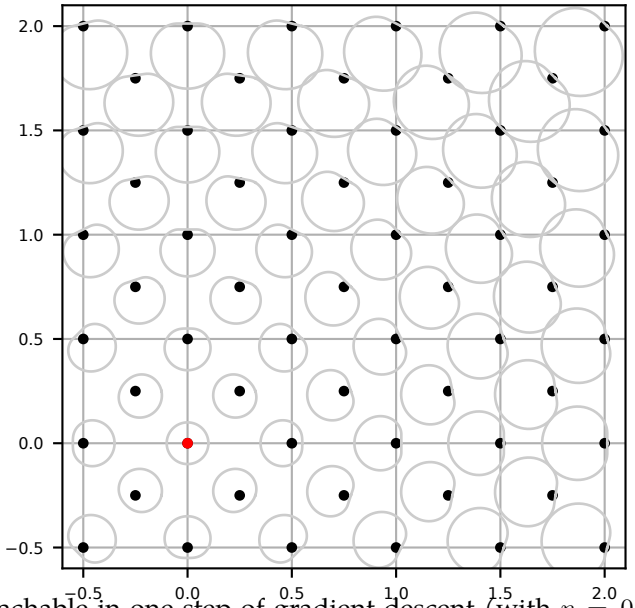


Figure 2.5: Points reachable in one step of gradient descent (with $\eta = 0.1$) on a least-squares objective starting from each of the black dots. There is circular symmetry about the origin (red dot).

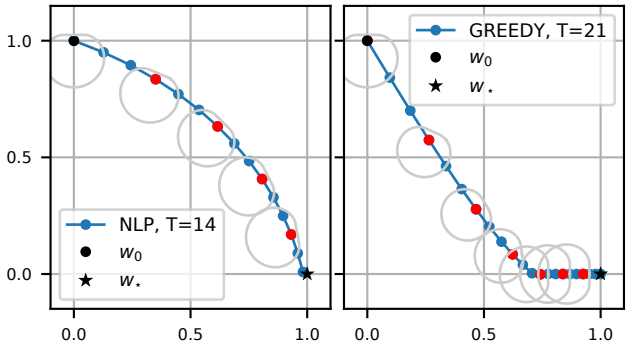


Figure 2.6: Reachable sets along the trajectory of NLP (left panel) and GREEDY (right panel). To minimize clutter, we only show every 3^{rd} reachable set. For this simulation, we used $\eta = 0.1$. The greedy approach makes fast progress initially, but slows down later on.

As seen in Section 2.2, optimal trajectories for the least squares learner are fundamentally 2D. Moreover, there is a taxonomy of regimes that dictates their behavior. We also saw in Section 2.3 that the continuous CNLP solver can provide a good approximation to the true discrete trajectory when η is small. CNLP is also more scalable than simply solving the discrete NLP directly because NLP becomes computationally intractable as T gets large (or η gets small), whereas the runtime of CNLP is independent of η.

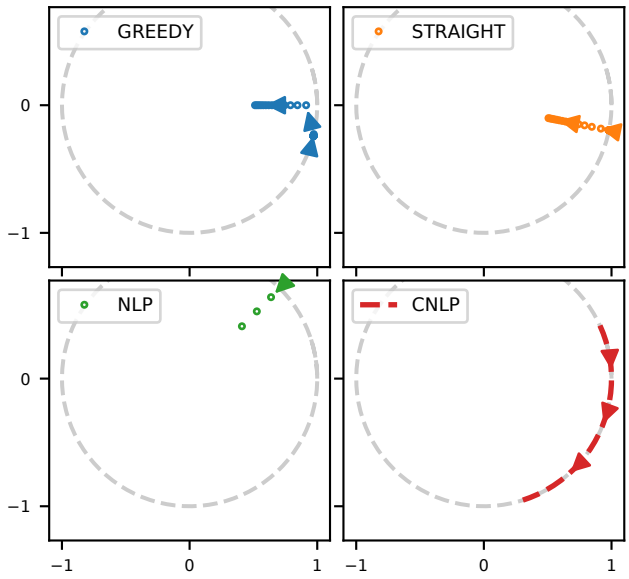


Figure 2.7: Trajectories of the input sequence $\{\mathbf{x}_t\}$ for GREEDY, STRAIGHT, and NLP methods and the corresponding $\mathbf{x}(t)$ for CNLP. The teaching task is $\mathbf{w}_0 = (-1.5, 0.5)$, $\mathbf{w}_\star = (1, 0)$, and $\eta = 0.01$. Markers show every 10 steps. Input constraint is $\|\mathbf{x}\| \leq 1$.

A drawback of both NLP and CNLP is that they produce *trajectories* rather than *policies*. In practice, using an open-loop teaching sequence $(\mathbf{x}_t, y_t)$ will not yield the $\mathbf{w}_t$ we expect due to the accumulation of small numerical errors as we iterate. In order to find a control policy, which is a map from state $\mathbf{w}_t$ to input $(\mathbf{x}_t, y_t)$, we discussed the possibility of solving HJB (Section 2.1) which is computationally expensive.

An alternative to solving HJB is to pre-compute the desired trajectory via CNLP and then use *model-predictive control* (MPC) to find a policy that tracks the reference trajectory as closely as possible. Such an approach is used in [113], for example, to design controllers for autonomous race cars, and would be an interesting avenue of future work for the machine teaching problem.

Finally, this chapter presents only a glimpse at what is possible using optimal control. For example, the PMP is not restricted to merely solving time-optimal control problems. It is possible to analyze problems with state- and input-dependent running costs, state and input pointwise or integral constraints, conditional constraints, and even problems where the goal is to reach a target *set* rather than a target point.

3 ONLINE DATA POISONING ATTACK.

In this chapter, we generalize our analysis to a more general online learning setting, where no assumption is made on the learning algorithm or the environment. In this more general setting, we show that, again, optimal control techniques can be used to formulate and solve for the optimal attack strategy. Theoretically, we show that even when the underlying distribution is unknown to the attacker, it can perform reinforcement learning on the joint system of the learner and environment, and achieve a regret that is sublinear in time and polynomial only in the complexity of the underlying distribution and independent of the complexity of the learning system.

3.1 Motivations and Problem definitions

Protecting machine learning from adversarial attacks is of paramount importance [163, 84, 195]. To do so one much first understand various types of adversarial attacks. Data poisoning is a type of attack where an attacker contaminates the training data in order to force a nefarious model on the learner [171, 122, 32, 40, 85, 108]. Prior work on data poisoning focused almost exclusively on the batch setting, where the attacker poisons a batch training set, and then the victim learns from the batch [25, 125, 171, 122, 151, 39]. However, the batch setting misses the threats posed by the attacker on sequential learners. For example, in e-commerce applications user-generated data arrives at the learner sequentially. Such applications are particularly susceptible to poisoning attacks, because it is relatively easy for the attacker to manipulate data items before they arrive at the learner. Furthermore, the attacker may observe the effect of previous poisoning on the learner and *adaptively* decides how to poison next. This adaptivity makes online data poisoning a potentially more severe threat compared to its batch counterpart.

This paper presents a principled study of online data poisoning attacks. Our key contribution is an optimal control formulation of such attacks. We provide theoretical analysis to show that the attacker can attack near-optimally even without full knowledge of the underlying data generating distribution. We then propose two practical attack algorithms—one based on traditional model-based optimal control, and the other based on deep reinforcement learning—and show that they achieve near-optimal attack performance in synthetic and real-data experiments. Taken together, this paper builds a foundation for future studies of defense against online data poisoning.

The online data poisoning problem in this paper is shown in Figure 3.1. There are three entities: a stochastic ***environment***, a sequential learning ***victim***, and the online ***attacker***. In the absence of attacks, at time t the environment draws a training data point $\mathbf{z}_t \in \mathcal{Z}$ i.i.d. from a time-invariant distribution P: $\mathbf{z}_t \overset{\text{i.i.d.}}{\sim} P$. For example, $\mathbf{z}_t$ can be a feature-label pair $\mathbf{z}_t := (\mathbf{x}_t, y_t)$ in supervised learning or just the features $\mathbf{z}_t := \mathbf{x}_t$ in unsupervised learning. The victim maintains a model $\theta_t \in \Theta$.

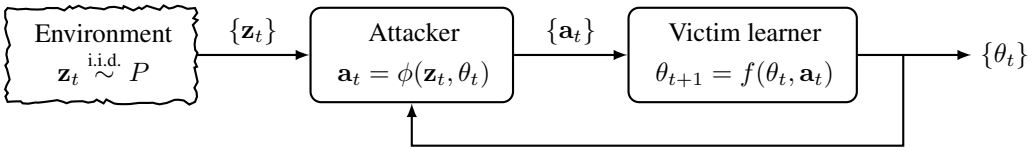


Figure 3.1: Online data poisoning attack diagram. The attacker observes the training samples $\{\mathbf{z}_t\}$ and the learner's model $\{\theta_t\}$ in an online fashion, and injects poisoned samples $\{\mathbf{a}_t\}$.

Upon receiving $\mathbf{z}_t$, the victim performs one step of the sequential update defined by the function f:

$$\theta_{t+1} = f(\theta_t, \mathbf{z}_t), \tag{3.1}$$

For example, f can be gradient descent $f(\theta_t, \mathbf{z}_t) := \theta_t - \eta \nabla \ell(\theta_t, \mathbf{z}_t)$ under learner loss ℓ and step size η. We now introduce the attacker by defining its knowledge, allowed actions, and goals:

- The attacker has knowledge of the victim's update function f, the victim's initial model θ_0, data $\mathbf{z}_{0:t}$ generated by the environment so far, and optionally n "pre-attack" data points $\mathbf{z}_{-n:-1} \overset{\text{i.i.d.}}{\sim} P$. **However, at time t the attacker does not have the clairvoyant knowledge of future data points $\mathbf{z}_{t+1}, \mathbf{z}_{t+2}, \ldots$, nor does it have the knowledge of the environment distribution P.**

- The attacker can perform only one type of action: once the environment draws a data point $\mathbf{z}_t$, the attacker perturbs the data point into a potentially different point $\mathbf{a}_t \in \mathcal{Z}$. The attacker incurs a perturbation cost $g_{\text{per}}(\mathbf{z}_t, \mathbf{a}_t)$, which reflects the price to attack. For example, $g_{\text{per}}(\mathbf{z}_t, \mathbf{a}_t) := \|\mathbf{a}_t - \mathbf{z}_t\|_p$ if $\mathcal{Z}$ is endowed with an appropriate p-norm. The attacker then gives $\mathbf{a}_t$ to the victim, who proceeds with model update (3.1) using $\mathbf{a}_t$ instead of $\mathbf{z}_t$.

- The attacker's goal, informally, is to force the victim's learned models θ_t to satisfy certain nefarious properties at each step while paying a small cumulative perturbation cost. These "nefarious properties" (rather the inability to achieve them) are captured by a nefarious cost $g_{\text{nef}}(\theta)$. It can encode a variety of attack goals considered in the literature such as: (*i*) *targeted attack* $g_{\text{nef}}(\theta) := \|\theta - \theta^\dagger\|$ to drive the learned model toward an attacker-defined target model $\theta^\dagger$ (the dagger is a mnemonic for attack); (*ii*) *aversion attack* $g_{\text{nef}}(\theta) := -\|\theta - \hat{\theta}\|$ (note the sign) to push the learned model away from a good model $\hat{\theta}$, such as the one estimated from pre-attack data; (*iii*) *backdoor attack* $g_{\text{nef}}(\theta) := \ell(\theta, \mathbf{z}^\dagger)$, in which the goal is to plant a backdoor such that the learned model behaves unexpectedly on special examples $\mathbf{z}^\dagger$ [108, 151, 39]. To balance nefarious properties with perturbation cost, the attacker defines a *running cost* g at time t:

$$g(\theta_t, \mathbf{z}_t, \mathbf{a}_t) := \lambda g_{\text{nef}}(\theta_t) + g_{\text{per}}(\mathbf{z}_t, \mathbf{a}_t), \tag{3.2}$$

where λ is a weight chosen by the attacker to balance the two. The attacker desires small cumulative running costs, which is the topic of Section 3.2.

Related Work

Data poisoning attacks have been studied against a wide range of learning systems. However, this body of prior work has almost exclusively focused on the offline or batch settings, where the attacker observes and can poison the whole training set or an entire batch of samples at once, respectively. In contrast, our paper focuses on the online setting, where the attacker has to act multiple times and sequentially during training. Examples of offline or batch poisoning attacks against SVM include [25, 32, 171]. Such attacks are generalized into a bilevel optimization framework against general offline learners with a convex objective function in [122]. A variety of attacks against other learners have been developed, including neural networks [95, 125], autoregressive models [8, 40], linear and stochastic bandits [85, 119], collaborative filtering [108], and models for sentiment analysis [130].

There is an intermediate attack setting between offline and online, which we call **clairvoyant online attacks**, where the attacker performs actions sequentially but **has full knowledge of all future input data** $\mathbf{z}_{t+1}, \mathbf{z}_{t+2}, \ldots$ Examples include heuristic attacks against SVM learning from data streams [32] and binary classification with an online gradient descent learner [166]. Our paper focuses instead on the perhaps more realistic setting where the attacker has no knowledge of the future data stream. More broadly, our paper advocates for a general optimal control viewpoint that is not restricted to specific learners such as SVM.

The parallel line of work studying **online teaching** also considers the sequential control problem of machine learners, where the goal is to choose a sequence of training examples that accelerates learning [115, 106]. However, [115] solves the problem using a greedy heuristic that we show in Section 3.5 performs poorly compared to our optimal control approach. On the other hand, [106] finds optimal teaching sequences but is restricted to an ordinary linear regression learner.

The problem of optimal feedback control in the presence of noise, uncertain disturbances, or uncertain dynamics has been an area of study for the better part of the past century. Major subfields include *stochastic control*, when disturbances are stochastic in nature [11, 100], *adaptive control*, when unknown parameters must be learned in an online fashion [12, 146], and *robust control*, when a single controller is designed to control a family of systems within some uncertainty set [193, 155].

More recently, these classical problems have been revisited in the context of modern statistics, with the goal of obtaining tight sample complexity bounds. Examples include unknown dynamics [45], adversarial dynamics [5], adversarial cost [42] or unknown dynamics *and* cost [64]. These works typically restrict their attention to linear systems with quadratic or convex losses, which is a common and often reasonable assumption for control systems. However, for our problem of interest

described in Section 3.1, the system dynamics (3.1) are the learner's dynamics, which are nonlinear for all cases of practical interest, including simple cases like gradient descent. In the sections that follow, we develop tools, algorithms, and analysis for handling this more general nonlinear setting.

3.2 An Optimal Control Formulation

We now precisely define the notion of optimal online data poisoning attacks. To do so, we cast the online data poisoning attack setting in Section 3.1 as a ***Markov Decision Process*** (***MDP***) $\mathcal{M} = (S, A, T, g, \gamma, \mathbf{s}_0)$ explained below.

- The **state** $\mathbf{s}_t$ at time t is the stacked vector $\mathbf{s}_t := [\theta_t, \mathbf{z}_t]^\mathsf{T}$ consisting of the victim's current model θ_t and the incoming environment data point $\mathbf{z}_t$. The **state space** is $S := \Theta \times \mathcal{Z}$.
- The attacker's **action** is the perturbed training point, i.e. $\mathbf{a}_t \in \mathcal{Z}$. The **action space** is $A := \mathcal{Z}$.
- From the attacker's perspective, the **state transition probability** $T : S \times A \to \Delta_S$, where Δ_S is the probability simplex over S, describes the conditional probability on the next state given current state and attack action. Specifically, $T(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) = T([\theta_{t+1}, \mathbf{z}_{t+1}]^\mathsf{T} \mid \mathbf{s}_t, \mathbf{a}_t) = Pr(f(\theta_t, \mathbf{a}_t) = \theta_{t+1}) \cdot P(\mathbf{z}_{t+1})$. For concreteness, in this paper, we assume that the victim learning update f is **deterministic**, and thus the stochasticity is solely in the $\mathbf{z}_{t+1}$ component inside $\mathbf{s}_{t+1}$, which has a marginal distribution P, i.e.

$$T\big([f(\theta_t, \mathbf{a}_t), \mathbf{z}_{t+1}]^\mathsf{T} \;\big|\; \mathbf{s}_t, \mathbf{a}_t\big) = P(\mathbf{z}_{t+1}). \tag{3.3}$$

- The quality of control at time t is specified by the **running cost** $g(\theta_t, \mathbf{z}_t, \mathbf{a}_t)$ in (3.2), to be minimized. From now on, we overload the notation and write the running cost equivalently as $g(\mathbf{s}_t, \mathbf{a}_t)$. Note that this is the opposite of the reward maximization setup commonly seen in reinforcement learning.
- We present online data poisoning attack with an infinite time horizon (the finite horizon case is similar but omitted due to space). We introduce a **discounting factor** $\gamma \in (0, 1)$ to define a discounted cumulative cost.
- The **initial probability** $\mu_0 : S \to \Delta_S$ is the probability distribution of the initial state $\mathbf{s}_0$. In particular, we assume that the initial model θ_0 is fixed while the first data point $\mathbf{z}_0$ is sampled from P, i.e. $\mu_0(\theta_0, \mathbf{z}_0) = P(\mathbf{z}_0)$.

A **policy** is a function $\phi : S \to A$ that the attacker uses to choose the attack action $\mathbf{a}_t := \phi(\mathbf{s}_t) = \phi([\theta_t, \mathbf{z}_t]^\mathsf{T})$ based on the current victim model θ_t and the current environment input $\mathbf{z}_t$. The **value**

$V^{\phi}_{\mathcal{M}}(\mathbf{s})$ of a state $\mathbf{s}$ is the expected discounted cumulative cost starting at $\mathbf{s}$ and following policy ϕ:

$$V^{\phi}_{\mathcal{M}}(\mathbf{s}) := \mathbb{E}_{\mathcal{M}}\left[\sum_{t=0}^{\infty}\gamma^t g(\mathbf{s}_t,\phi(\mathbf{s}_t))\Big|_{\mathbf{s}_0=\mathbf{s}}\right] \tag{3.4}$$

where the expectation is over the transition probability T. Overall, the attacker wants to perform optimal control over the MDP, that is, to find an **optimal control policy** $\phi^{\star}_{\mathcal{M}}$ that minimizes the expected value at the initial state. Define the attacker's objective as $J_{\mathcal{M}}(\phi) := \mathbb{E}_{\mathbf{s}\sim\mu_0}\left[V^{\phi}_{\mathcal{M}}(\mathbf{s})\right]$, and the attacker's optimal attack policy as $\phi^{\star}_{\mathcal{M}} = \arg\min_{\phi} J_{\mathcal{M}}(\phi)$.

Fortunately for the victim, the attacker cannot directly solve this optimal attack problem because it does not know the environment data distribution P and thus cannot evaluate the expectation. However, as we show next, the attacker can use model predictive control to approximately and incrementally solve for the optimal attack policy while it gathers information about P as the attack happens.

3.3 Practical Attack Algorithms via Model Predictive Control

The key obstacle that prevents the attacker from obtaining an optimal attack is the unknown data distribution P. However, the attacker can build an increasingly accurate **empirical distribution** $\hat{P}_t$ from $\mathbf{z}_{0:t}$ and optionally the pre-attack data sampled from P. Specifically, at time t with $\hat{P}_t$ in place of P and with the model θ_t in place of θ_0, the attacker can construct a surrogate MDP $\hat{\mathcal{M}}_t = (S, A, \hat{T}_t, g, \gamma, \hat{\mu}_t)$, solve for the optimal policy $\phi^{\star}_{\hat{\mathcal{M}}_t} = \arg\min_{\phi} J_{\hat{\mathcal{M}}_t}(\phi)$ on $\hat{\mathcal{M}}_t$, and use $\phi^{\star}_{\hat{\mathcal{M}}_t}$ to perform a **one-step attack**: $\mathbf{a}_t = \phi^{\star}_{\hat{\mathcal{M}}_t}(\mathbf{s}_t)$.

As time t goes on, the attacker repeats the process of estimating $\phi^{\star}_{\hat{\mathcal{M}}_t}$ and applying the one-step attack $\phi^{\star}_{\hat{\mathcal{M}}_t}(\mathbf{s}_t)$. This repeated procedure of (re)-planning ahead but only executing one action is called ***Model Predictive Control*** **(*MPC*)** [27, 121], and is widely used across the automotive, aerospace, and petrochemical industries, to name a few. At each time step t, MPC would plan a sequence of attacks using the surrogate model (in our case $\hat{P}_t$ instead of P), apply the first attack $\mathbf{a}_t$, update $\hat{P}$, and repeat. This allows the controller to continually adapt without committing to an inaccurate model.

Next, we present two algorithms that practically solve the surrogate MDP, one based on model-based planning and the other based on model-free reinforcement learning.

Algorithm NLP: Planning with Nonlinear Programming In the NLP algorithm, the attacker further approximates the surrogate objective as

$$J_{\hat{\mathcal{M}}_t}(\phi) \approx \mathbb{E}_{\hat{P}_t}\left[\sum_{\tau=t}^{t+h-1}\gamma^{\tau-t}g(\mathbf{s}_\tau,\phi(\mathbf{s}_\tau))\right] \approx \sum_{\tau=t}^{t+h-1}\gamma^{\tau-t}g(\mathbf{s}_\tau,\mathbf{a}_\tau)\Big|_{\mathbf{z}_{t:t+h-1}}$$

The first approximation truncates at h steps after t, making it a finite-horizon control problem. The second approximation does two things: (i) It replaces the expectation by one sampled trajectory of the future input sequence, i.e. $\mathbf{z}_{t:t+h-1} \sim \hat{P}_t$. It is of course possible to use the average of multiple trajectories to better approximate the expectation, though empirically we found that one trajectory is sufficient. (ii) Instead of optimizing over a policy ϕ, it locally searches for the action sequence $\mathbf{a}_{t:t+h-1} \in \mathcal{Z}$. The attacker now solves the following optimization problem at every time t:

$$\begin{aligned} \min_{\mathbf{a}_{t:t+h-1}} \quad & \sum_{\tau=t}^{t+h-1} \gamma^{\tau-t} g(\mathbf{s}_\tau, \mathbf{a}_\tau) \\ \text{s.t.} \quad & \mathbf{s}_{\tau+1} = [f(\mathbf{s}_\tau, \mathbf{a}_\tau), \mathbf{z}_\tau]^\mathsf{T}, \forall \tau = t, ..., t+h-1 \\ & \mathbf{z}_{t:t+h-1} \text{ and } \mathbf{s}_t \text{ fixed and given.} \end{aligned} \tag{3.5}$$

Let $\mathbf{a}^\star_{t:t+h-1}$ be a solution. The NLP algorithm *defines* $\phi^\star_{\hat{M}_t}(\mathbf{s}_t) := \mathbf{a}^\star_t$, then moves on to $t+1$. The resulting attack problem in general has a nonlinear objective stemming from $g_{\text{nef}}()$ and $g_{\text{per}}()$ in (3.2), and nonconvex equality constraints stemming from the victim's learning rule $f()$ in (3.1). Nonetheless, the attacker can solve modest-sized problems using modern nonlinear programming solvers such as IPOPT [165].

Algorithm DDPG: Deep Deterministic Policy Gradient Instead of truncating and sampling to approximate the surrogate attack problem with a nonlinear program, one can directly solve for the optimal parametrized policy ϕ using reinforcement learning. In this paper, we utilize **deep deterministic policy gradient (DDPG)** [111] to handle a continuous action space. DDPG learns a deterministic policy with an actor-critic framework. Roughly speaking, it simultaneously learns an **actor network** $\mu(s)$ parametrized by θ^μ and a **critic network** $Q(s,a)$ parametrized by θ^Q. The actor network represents the currently learned policy while the critic network estimate the action-value function of the current policy, whose functional gradient guides the actor network to improve its policy. Specifically, the policy gradient can be written as: $\nabla_{\theta^\mu} J = \mathbb{E}_{\mathbf{s}\sim\rho^\mu}\left[\nabla_a Q(s, \mu(s)|\theta^Q) \nabla_{\theta^\mu}\mu(s|\theta^\mu)\right]$ in which the expectation is taken over ρ^μ, the *discounted state visitation distribution* for the current policy μ. The critic network is updated using Temporal-Difference (TD) learning. We refer the reader to the original paper [111] for a more detailed discussion of this algorithm and other deep learning implementation details.

There are two advantages of this policy learning approach to the direct approach NLP. Firstly, it actually learns a policy which can **generalize** to more than one step of attack. Secondly, it is a **model-free** method and doesn't require knowledge of the analytical form of the system dynamic f, which is necessary for the direct approach. Therefore, DDPG also applies to the black-box attack setting, where the learner's dynamic f is unknown. To demonstrate the generalizability of the learned policy, in our experiments described later, we only allow the DDPG method to train once

at the beginning of the attack on the surrogate MDP $\hat{\mathcal{M}}_0$ based on $(\theta_0, \mathbf{z}_0)$ and the pre-attack data $\mathbf{z}_{-n:-1}$. The learned policy $\phi_{\hat{\mathcal{M}}_0}$ is then applied to all later attack rounds without retraining.

3.4 Theoretical Analysis

The fundamental statistical limit to a realistic attacker is its lack of knowledge on the environment data distribution P. An idealized attacker with knowledge of P can find the optimal control policy $\phi^\star_{\mathcal{M}}$ that achieves the optimal attack objective $J_{\mathcal{M}}$. In contrast, a realistic attacker only has an estimated $\hat{P}$, hence an estimated state transition $\hat{T}$, and ultimately an estimated MDP $\hat{\mathcal{M}} = (S, A, \hat{T}, g, \gamma, \hat{\mu}_0)$. The realistic attacker will find an optimal policy with respect to its estimated MDP $\hat{\mathcal{M}}$: $\phi^\star_{\hat{\mathcal{M}}} = \arg\min_\phi J_{\hat{\mathcal{M}}}(\phi)$, but $\phi^\star_{\hat{\mathcal{M}}}$ is in general suboptimal with respect to the true MDP $\mathcal{M}$. We are interested in the **optimality gap** $V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s)$. Note both are evaluated on the true MDP.

We present a theoretical analysis relating the optimality gap to the quality of estimated $\hat{P}$. Our analysis is a natural extension to the Simulation Lemma in tabular reinforcement learning [91] and that of [16]. We assume that both $\mathcal{Z}$ and Θ are compact, and the running cost g is continuous and thus bounded on its compact domain. WLOG, we assume $g \in [0, C_{\max}]$. It is easy to see that then the range of value is bounded: $V \in [0, \frac{C_{\max}}{1-\gamma}]$ for both $\mathcal{M}, \hat{M}$, any policy, and any state. Note the value function (3.4) satisfies the **Bellman equation**: $V_{\mathcal{M}}^{\phi}(\mathbf{s}) = g(\mathbf{s}, \phi(\mathbf{s})) + \gamma \mathbb{E}_{T(\mathbf{s}'|\mathbf{s},\phi(\mathbf{s}))}\left[V_{\mathcal{M}}^{\phi}(\mathbf{s}')\right]$.

Proposition 3.4.1. *Consider two MDPs $\mathcal{M}, \hat{\mathcal{M}}$ that differ only in state transition, induced by P and $\hat{P}$, respectively. Assume that $\|\hat{P} - P\|_1 := \int_{\mathcal{Z}} |\hat{P}(\mathbf{z}) - P(\mathbf{z})| \, \mathrm{d}\mathbf{z} \le \epsilon$. Let $\phi^\star_{\mathcal{M}}$ denote the optimal policy on $\mathcal{M}$ and $\phi^\star_{\hat{\mathcal{M}}}$ the optimal policy on $\hat{\mathcal{M}}$. Then, $\sup_{s\in S} V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s) \le \frac{\gamma C_{\max}\epsilon}{(1-\gamma)^2}$.*

Proposition 3.4.1 implies that optimality gap is at most linear in $\epsilon := \|\hat{P} - P\|_1$. Classic Results on Kernel Density Estimation (KDE) suggest that the L_1 distance between P and the kernel density estimator $\hat{P}_n$ based on n samples converges to zero asymptotically in a rate of $O(n^{-s/d+2s})$ for some constant s (e.g. Theorem 9 in [71]).

In the experiment section below, the environment data stream is generated from a uniform distribution on a finite data set, in which case P is a multinomial distribution. Under this special setting, we are able to provide a finite-sample bound of order $O(n^{-1/2})$ that matches with the best achievable asymptotic rate above, i.e. as $s \to \infty$.

Theorem 3.4.2. *Consider an MDP $\mathcal{M}$ induced by a multinomial distribution P with support cardinality N, and a surrogate MDP $\hat{\mathcal{M}}$ induced by the empirical distribution $\hat{P}$ on n i.i.d. samples, i.e. $\hat{P}(i) = \frac{1}{n}\sum_{j=1}^n I_{x_j=i}$. Denote $\phi^\star_{\mathcal{M}}$ the optimal policy on $\mathcal{M}$ and $\phi^\star_{\hat{\mathcal{M}}}$ the optimal policy on $\hat{\mathcal{M}}$. Then, with probability at least $1-\delta$, we have $\sup_{s\in S} V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s) \le \frac{2\gamma C_{\max}}{(1-\gamma)^2}\sqrt{\frac{1}{2n}\ln\frac{2^{N+1}}{\delta}} = O(n^{-1/2})$.*

3.5 Experiments

In this section, we empirically evaluate our attack algorithms NLP and DDPG in Section 3.3 against several baselines on synthetic and real data. As an empirical measure of attack efficacy, we compare the attack methods by their **empirical discounted cumulative cost** $\tilde{J}(t) := \sum_{\tau=0}^{t} \gamma^\tau g(\theta_\tau, \mathbf{z}_\tau, \mathbf{a}_\tau)$, where the attack actions $\mathbf{a}_\tau$ are chosen by each method. Note that $\tilde{J}(t)$ is computed on the actual instantiation of the environment data stream $\mathbf{z}_0, \ldots, \mathbf{z}_t$. Better attack methods tend to have smaller $\tilde{J}(t)$. We compare our algorithms with the following **Baseline Attackers**:

Null Attack: This is the baseline without attack, namely $\mathbf{a}_t^{\text{Null}} = \mathbf{z}_t$ for all t. We expect the null attack to form an upper bound on any attack method's empirical discounted cumulative cost $\tilde{J}(t)$.

Greedy Attack: The greedy strategy is applied widely as a practical heuristic in solving sequential decision problems ([115, 106]). For our problem at time step t the greedy attacker uses a time-invariant attack policy which minimizes the current step's running cost g. Specifically, the **greedy attack policy** can be written as $\mathbf{a}_t^{\text{Greedy}} = \arg\min_{\mathbf{a}} g(\theta_t, \mathbf{z}_t, \mathbf{a})$. If we instantiate $g_{\text{nef}} = \|\theta_t - \theta^\dagger\|_2^2$ for a target model $\theta^\dagger$ and $g_{\text{per}} = 0$, we exactly recover the algorithm in [115]. Both null attack and greedy attack can be viewed as time-invariant policies that do not utilize the information in $\hat{P}_t$.

Clairvoyant Attack: A clairvoyant attacker is an idealized attacker who knows the time horizon T and the whole data sequence $\mathbf{z}_{0:T-1}$ upfront. In most realistic online data poisoning settings an attacker only know $\mathbf{z}_{0:t}$ at time t. Therefore, the clairvoyant attacker has strictly more information, and we expect it to form a lower bound on realistic attack methods in terms of $\tilde{J}(t)$. The clairvoyant attacker solves a finite time-horizon optimal control problem, equivalent to the formulation in [166] but without terminal cost: $\min_{\mathbf{a}_{0:T-1}} \sum_{t=0}^{T-1} \gamma^t g(\theta_t, \mathbf{z}_t, \mathbf{a}_t)$ subject to θ_0 given, $\mathbf{z}_{0:T-1}$ given (clairvoyant), and $\theta_{t+1} = f(\theta_t, \mathbf{a}_t), t = 0 \ldots T-1$.

Poisoning Task Specification To specify a poisoning task is to define the victim learner f in (3.1) and the attacker's running cost g in (3.2). We evaluate all attacks on two types of victim learners: online logistic regression, a supervised learning algorithm, and online soft k-means clustering, an unsupervised learning algorithm.

Online logistic regression: Online logistic regression performs a binary classification task. The incoming data takes the form of $\mathbf{z}_t = (\mathbf{x}_t, y_t)$, where $\mathbf{x}_t \in \mathbb{R}^d$ is the feature vector and $y_t \in \{-1, 1\}$ is the binary label. In the experiments, we focus on attacking the feature part of the data, as is done in a number of prior works [122, 166, 95]. The learner's update rule is one step of gradient descent on the log likelihood with step size η: $f(\theta, (\mathbf{x}, y)) = \theta + \eta \frac{y\mathbf{x}}{1+\exp(y\theta^\intercal \mathbf{x})}$. The attacker wants to force the victim learner to stay close to a target parameter $\theta^\dagger$, i.e. this is a targeted attack. The attacker's cost function g is a weighted sum of two terms: the nefarious cost g_{nef} is the negative cosine similarity between the victim's parameter and the target parameter, and the perturbation cost g_{per} is the L_2 distance between the perturbed feature vector and the clean one, i.e. $g(\theta_t, (\mathbf{x}_t, y), (\mathbf{x}'_t, y)) =$

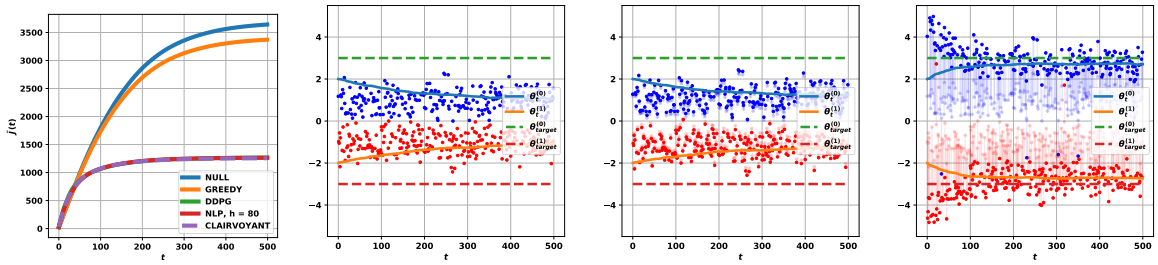
$-\lambda \cos\left(\theta_t, \theta^\dagger\right) + \|\mathbf{x}'_t - \mathbf{x}_t\|^2$. Recall $\cos(a, b) := \frac{a^\mathsf{T} b}{\|a\|\|b\|}$.

Online soft k-means: Online soft k-means performs a k-means clustering task. The incoming data contains only the feature vector, i.e. $\mathbf{z}_t = \mathbf{x}_t$. Its only difference from traditional k-means is that instead of updating only the centroid closest to the current data point, it updates all the centroids but the updates are weighted by their squared distances to the current data point using the softmax function [23]. Specifically, the learner's update rule is one step of soft k-means update with step size η on all centroids, i.e. $f(\theta^{(j)}, \mathbf{a}) = \theta^{(j)} + \eta r_j(\mathbf{a} - \theta^{(j)})$, $j = 1, \ldots, k$, where $\mathbf{r} = \text{softmax}(-\|\mathbf{a} - \theta^{(1)}\|^2, \ldots, -\|\mathbf{a} - \theta^{(k)}\|^2)$. Recall $\text{softmax}(x_1, \ldots, x_k) := [\frac{e^{x_1}}{\sum_j^k e^{x_j}}, \ldots, \frac{e^{x_k}}{\sum_j^k e^{x_j}}]^\mathsf{T}$. Similar to online logistic regression, we consider a targeted attack objective. The attacker wants to force the learned centroids to each stay close to the corresponding target centroid $\theta^{\dagger(j)}$. The attacker's cost function g is a weighted sum of two terms: the nefarious cost function g_{nef} is the sum of the squared distance between each of the victim's centroid and the corresponding target centroid, and the perturbation cost g_{per} is the ℓ_2 distance between the perturbed feature vector and the clean one, i.e. $g(\theta_t, \mathbf{z}_t, \mathbf{a}_t) = \lambda \sum_{j=1}^k \|f(\theta, \mathbf{a})^{(j)} - \theta^{\dagger(j)}\|^2 + \|\mathbf{a}_t - \mathbf{z}_t\|^2$.

Synthetic Data Experiments We first show a synthetic data experiment where the attack policy can be visualized. The environment is a mixture of two 1D Gaussian: $P = \frac{1}{2}N(\theta^{(1)}, 1) + \frac{1}{2}N(\theta^{(2)}, 1)$ with $\theta^{(1)} = -1$ and $\theta^{(2)} = +1$. The victim learner is online soft k-means with $k = 2$ and initial parameter $\theta_0^{(1)} = -2, \theta_0^{(2)} = +2$. The attack target is $\theta^{\dagger(1)} = -3$ and $\theta^{\dagger(2)} = +3$, namely the opposite of how the victim's parameters should move. We set the learning rate $\eta = 0.01$, cost regularizer $\lambda = 10$, discounting factor $\gamma = 0.99$, evaluation length $T = 500$ and look-ahead horizon for MPC $h = 100$. For attack methods that requires solving a nonlinear program, including GREEDY, NLP and Clairvoyant, we use the JuMP modeling language [59] and the IPOPT interior-point solver [165]. Following the above specification, we run each attack method on the same data stream and compare their behavior.

Results: Figure 3.2a shows the empirical discounted cumulative cost $\tilde{J}(t)$ as the attacks go on. On this toy example, the null attack baseline achieves $\tilde{J}(T) = 3643$ at $T = 500$. The greedy attacker is only slight more effective at $\tilde{J}(T) = 3372$. NLP and DDPG (curve largely overlap and hidden under NLP) achieve 1265 and 1267, respectively, almost matching Clairvoyant's 1256. As expected, the null and clairvoyant attacks form upper and lower bounds on $\tilde{J}(t)$.

Figure 3.2b-f shows the victim's θ_t trajectory as attacks go on. Without attack (null), θ_t converges to the true parameter -1 and $+1$. The greedy attack only perturbs each data point slightly, failing to force θ_t toward attack targets. This failure is due to its greedy nature: the immediate cost g_t at each round is indeed minimized, but not enough to move the model parameters close to the target parameters. In contrast, NLP and DDPG (trajectory similar to NLP, not shown) exhibit a different strategy in the earlier rounds. They inject larger perturbations to the data points and sacrifice larger



(a) Cumulative Attack Costs (b) NULL Attack (c) GREEDY Attack (d) NLP Attack

Figure 3.2: Synthetic data experiments. In (b)-(f), transparent blue and red dots indicate clean positive and negative data point $\mathbf{z}_t$ at time step t, solid dots indicate attacker-perturbed data point $\mathbf{a}_t$, vertical lines in between indicate the amount of perturbation.

immediate costs in order to drive the victim's model parameters quickly towards the target. In later rounds they only need to stabilize the victim's parameters near the target with smaller per-step cost.

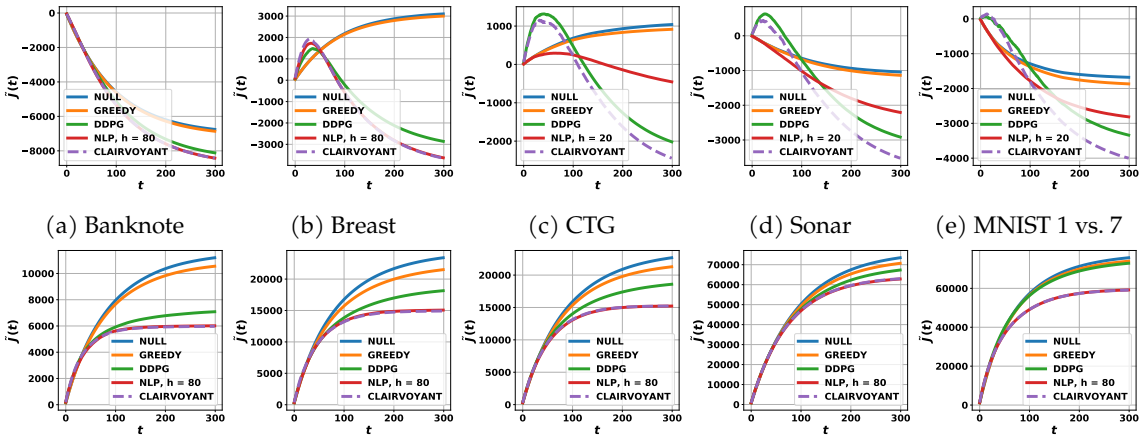
Real Data Experiments In the real data experiments, we run each attack method on 10 data sets across two victim learners.

Datasets: We use 5 datasets for online logistic regression: Banknote Authentication (with feature dimension $d = 4$), Breast Cancer ($d = 9$), Cardiotocography ($d = 25$), Sonar ($d = 60$), and MNIST 1 vs. 7 ($d = 784$), and 5 datasets for online k-means clustering: User Knowledge ($d = 6, k = 2$), Breast Cancer ($d = 10, k = 2$), Seeds ($d = 8, k = 3$), posture ($d = 11, k = 5$), MNIST 1 vs. 7 ($d = 784, k = 2$). All datasets except for MNIST can be found in the UCI Machine Learning Repository [57]. Note that two datasets, Breast Cancer and MNIST, are shared across both tasks.

Preprocessing: To reduce the running time, for datasets with dimensionality $d > 30$, we reduce the dimension to 30 via PCA projection. Then, all datasets are normalized so that each feature has mean 0 and variance 1. Each dataset is then turned into a data stream by random sampling. Specifically, each training data point $\mathbf{z}_t$ is sampled uniformly from the dataset with replacement.

Experiment Setup: In order to demonstrate the general applicability of our methods, we draw both the victim's initial model θ_0 and the attacker's target $\theta^\dagger$ at random from a standard Gaussian distribution of the appropriate dimension, for both online logistic regression and online k-means in all 10 datasets. Across all datasets, we use the following hyperparameters: $\eta = 0.01, \gamma = 0.99, T = 300$. For online logistic regression $\lambda = 100$ while for online k-means $\lambda = 10$.

For DDPG attacker we only perform policy learning at the beginning to obtain $\phi_{\hat{\mathcal{M}}_0}$; the learned policy is then fixed and used to perform all the attack actions in later rounds. In order to give it a fair chance, we give it a pre-attack dataset $\mathbf{z}_{-n:-1}$ of size $n = 1000$. For the sake of fair comparisons, we give the same pre-attack dataset to NLP as well. For NLP attack we set the look-ahead horizon h



(a) Banknote (b) Breast (c) CTG (d) Sonar (e) MNIST 1 vs. 7

(f) Knowledge (g) Breast (h) Seeds (i) Posture (j) MNIST 1 vs. 7

Figure 3.3: The empirical discounted cumulative reward $\tilde{J}(t)$ for the five attack methods across 10 real datasets. The first row is on online logistic regression and the second row is on online k-means. Note that $g(\cdot)$ for online logistic regression can be negative, and thus the $\tilde{J}(t)$ curve can be decreasing.

such that the total runtime to perform $T = 300$ attacks does not exceed the DDPG training time, which is 24 hours on an Intel Core i7-6800K CPU 3.40GHz with 12 cores. This results in $h = 20$ for online logistic regression on CTG, Sonar and MNIST, and $h = 80$ in all other experiments.

Results: The experiment results are shown in figure 3.3. Interestingly, several consistent patterns emerge from the experiments: The clairvoyant attacker consistently achieves the lowest cumulative cost $\tilde{J}(T)$ across all 10 datasets. This is not surprising, as the clairvoyant attacker has extra information of the future. The NLP attack achieves clairvoyant-matching performance on all 7 datasets in which it is given a large enough look-ahead horizon, i.e. $h = 80$. DDPG follows closely next to MPC and Clairvoyant on most of the datasets, indicating that the pre-trained policy $\phi_{\hat{\mathcal{M}}_0}$ can achieve reasonable attack performance in most cases. On the 3 datasets where $h = 20$ for NLP, DDPG exceeds the short-sighted NLP, indicating that when the computational resource is limiting, DDPG has an advantage by avoiding the iterative retraining that NLP cannot bypass. GREEDY does not do well on any of the 10 datasets, achieving only a slightly lower cost than the NULL baseline. This matches our observations in the synthetic experiment.

Each of the attack methods also exhibits strategic behavioral patterns similar to what we observe in the synthetic experiment. In particular, the optimal-control based methods NLP and DDPG sacrifice larger immediate costs in earlier rounds in order to achieve smaller attack costs in later rounds. This is especially obvious in the online logistic regression plots 3.3b-e, where the cumulative costs $\tilde{J}(t)$ rise dramatically in the first 50 rounds, becoming higher than the cost of NULL and GREEDY around that time. This early sacrifice pays off after $t = 50$ where the cumulative cost starts

to fall much faster. In 3.3c-e, however, the short-sighted NLP (with $h = 20$) fails to fully pick up this long-term strategy, and exhibits a behavior close to an interpolation of greedy and optimal. This is not surprising, as NLP with horizon $h = 1$ is indeed equivalent to the GREEDY method. Thus, there is a spectrum of methods between GREEDY and NLP that can achieve various levels of performance with different computational costs.

Next, we extend our study of optimal attack to the reinforcement learning setting. This chapter studies the adversarial attack problem in the batch reinforcement learning where the data is pre-collected and provided as a set of transitions. The attacker aims at mislead the learner to learn a particular target policy by contaminate the collected dataset before presenting it to the learner. This is in fact an offline learning problem, but we will see that several insights obtained here will help us understand the online attack problem later.

4.1 Motivation

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 [26, 72]. 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 [122, 96, 166, 189] and multi-armed bandits [85, 119, 114]. 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.

Related Work

Of particular interest is the work on *test-time attacks* against RL [73]. 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 [25, 170, 108, 122] or online mode [166, 189]. Recently data-poisoning attacks have been extended to multi-armed bandit victims [85, 119, 114], but not yet to RL victims.

There are two related but distinct concepts in RL research. One concept is reward shaping [131, 10, 47, 167] 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 [36, 29, 89]. 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.

4.2 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:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \max_{a' \in \mathcal{A}} Q^*(s', a'), \tag{4.1}$$

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 $\hat{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta_{\mathcal{S}}$

$$\hat{P} \quad \in \quad \arg\max_{P} \sum_{t=0}^{T-1} \log P(s'_t | s_t, a_t), \tag{4.2}$$

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

$$\hat{R} \quad = \quad \arg\min_{R} \sum_{t=0}^{T-1} (r_t - R(s_t, a_t))^2. \tag{4.3}$$

Note that we do not require (4.2) 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.,

$$\hat{\pi} \in \arg\max_{\pi : \mathcal{S} \mapsto \mathcal{A}} \mathbb{E}_{\hat{P}} \sum_{\tau=0}^{\infty} \gamma^{\tau} \hat{R}(s_\tau, \pi(s_\tau)), \tag{4.4}$$

where s_0 is a specified or random initial state. Note that there could be multiple optimal policies, thus we use $\in$ in (4.4). Later we will specialize (4.4) to two specific victim learners: the tabular certainty equivalence learner (TCE) and the certainty-equivalent linear quadratic regulator (LQR).

4.3 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., (4.2) and (4.3). In the case (4.2) 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[1]:

$$\min_{\mathbf{r},\hat{R}} \quad \|\mathbf{r}-\mathbf{r}^0\|_\alpha \tag{4.5}$$

$$\text{s.t.} \quad \hat{R} = \arg\min_{R} \sum_{t=0}^{T-1} (r_t - R(s_t,a_t))^2 \tag{4.6}$$

$$\{\pi^\dagger\} = \arg\max_{\pi:\mathcal{S}\mapsto\mathcal{A}} \mathbb{E}_{\hat{P}} \sum_{\tau=0}^{\infty} \gamma^\tau \hat{R}(s_\tau, \pi(s_\tau)). \tag{4.7}$$

The $\hat{P}$ in (4.7) does not involve $\mathbf{r}$ and is precomputed from D^0. The singleton set $\{\pi^\dagger\}$ on the LHS of (4.7) 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 $T_{s,a} \geq 1, \forall s, a$, 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 $\hat{R}$ exist. Remember that we require (4.3) to have a unique solution. For the TCE learner, $\hat{R}$ is unique as long as it exists. Therefore, $T_{s,a} \geq 1, \forall s, a$ is sufficient to guarantee a unique solution to (4.3). Let the poisoned data be $D = (s_t, a_t, r_t, s_t')_{0:T-1}$. Instantiating model estimation (4.2), (4.3) for TCE, we have

$$\hat{P}(s' \mid s, a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} \mathbb{1}\left[s_t' = s'\right], \tag{4.8}$$

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

$$\hat{R}(s,a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t. \tag{4.9}$$

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 (4.1). The attack goal (4.7) can be naively characterized by

$$Q(s, \pi^\dagger(s)) > Q(s,a), \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s). \tag{4.10}$$

[1]As we will show, the constraint (4.7) could lead to an open feasible set (e.g., in (4.10)) for the attack optimization (4.5)-(4.7), on which the minimum of the objective function (4.5) may not be well-defined. In the case (4.7) 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.

However, due to the strict inequality, (4.10) induces an open set in the Q space, on which the minimum of (4.5) 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 4.3.1. (ϵ-*robust target Q polytope*) *The set of ϵ-robust Q functions induced by a target policy $\pi^\dagger$ is the polytope*

$$\mathcal{Q}_\epsilon(\pi^\dagger) = \{Q : Q(s, \pi^\dagger(s)) \geq Q(s, a) + \epsilon, \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s)\} \tag{4.11}$$

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 $\mathcal{Q}_\epsilon(\pi^\dagger)$:

$$\min_{\mathbf{r}\in\mathbb{R}^T, \hat{R}, Q\in\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}} \quad \|\mathbf{r} - \mathbf{r}^0\|_\alpha \tag{4.12}$$

$$\text{s.t.} \quad \hat{R}(s,a) = \frac{1}{|T_{s,a}|} \sum_{t\in T_{s,a}} r_t \tag{4.13}$$

$$Q(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{P}\left(s'|s,a\right) Q\left(s', \pi^\dagger(s')\right), \forall s, \forall a \tag{4.14}$$

$$Q(s, \pi^\dagger(s)) \geq Q(s,a) + \epsilon, \forall s \in \mathcal{S}, \forall a \neq \pi^\dagger(s). \tag{4.15}$$

The constraint (4.14) 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 (4.15). Note that problem (4.12)-(4.15) is a convex program with linear constraints given that $\alpha \geq 1$, thus could be solved to global optimality. However, we point out that (4.12)-(4.15) is a more stringent formulation than (4.5)-(4.7) due to the additional margin parameter ϵ we introduced. The feasible set of (4.12)-(4.15) is a subset of (4.5)-(4.7). Therefore, the optimal solution to (4.12)-(4.15) could in general be a sub-optimal solution to (4.5)-(4.7) 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 4.3.2. *The attack problem* (4.12)-(4.15) *is always feasible for any target policy $\pi^\dagger$.*

Proposition 4.3.2 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 4.3.1. *Assume $\alpha \geq 1$ in (4.12). Let $\mathbf{r}^*$, $\hat{R}^*$ and Q^* be an optimal solution to (4.12)-(4.15), then*

$$\frac{1}{2}(1-\gamma)\Delta(\epsilon)\left(\min_{s,a}|T_{s,a}|\right)^{\frac{1}{\alpha}} \leq \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leq \frac{1}{2}(1+\gamma)\Delta(\epsilon)T^{\frac{1}{\alpha}}. \tag{4.16}$$

Corollary 4.3.3. *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 [45]. Let the linear dynamical system be

$$s_{t+1} = As_t + Ba_t + w_t, \forall t \geq 0, \tag{4.17}$$

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

$$L(s,a) = \frac{1}{2}s^\top Qs + q^\top s + a^\top Ra + c \tag{4.18}$$

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 4.2) 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 [45]. Let the poisoned data be $D = (s_t, a_t, r_t, s_{t+1})_{0:T-1}$. Instantiating model estimation (4.2), (4.3), the learner performs system

identification on the poisoned data:

$$(\hat{A}, \hat{B}) \in \arg\min_{(A,B)} \frac{1}{2} \sum_{t=0}^{T-1} \|As_t + Ba_t - s_{t+1}\|_2^2 \tag{4.19}$$

$$(\hat{Q}, \hat{R}, \hat{q}, \hat{c}) = \arg\min_{(Q \succeq 0, R \succeq \epsilon I, q, c)} \frac{1}{2} \sum_{t=0}^{T-1} \left\| \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t \right\|_2^2. \tag{4.20}$$

Note that in (4.20), 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 (4.20) 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.

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

$$\max_{\pi: \mathcal{S} \mapsto \mathcal{A}} \quad -\mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau \left(\frac{1}{2} s_\tau^\top \hat{Q} s_\tau + \hat{q}^\top s_\tau + \pi(s_\tau)^\top \hat{R} \pi(s_\tau) + \hat{c} \right) \right] \tag{4.21}$$

$$\text{s.t.} \quad s_{\tau+1} = \hat{A} s_\tau + \hat{B} \pi(s_\tau) + w_\tau, \forall \tau \geq 0. \tag{4.22}$$

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

$$K = -\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B} \right)^{-1} \hat{B}^\top X \hat{A}, \quad k = -\gamma (\hat{R} + \gamma \hat{B}^\top X \hat{B})^{-1} \hat{B}^\top x. \tag{4.23}$$

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

$$X = \gamma \hat{A}^\top X \hat{A} - \gamma^2 \hat{A}^\top X \hat{B} \left(\hat{R} + \gamma \hat{B}^\top X \hat{B} \right)^{-1} \hat{B}^\top X \hat{A} + \hat{Q}, \tag{4.24}$$

and x is a vector that satisfies

$$x = \hat{q} + \gamma (\hat{A} + \hat{B} K)^\top x. \tag{4.25}$$

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:

$$\{(K,k) : \exists Q \succeq 0, R \succeq \epsilon I, q \in \mathbb{R}^n, c \in \mathbb{R}, \text{such that (4.23), (4.24), and (4.25) are satisfied}\}. \quad (4.26)$$

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:

$$\begin{aligned}
\min_{\mathbf{r},\hat{Q},\hat{R},\hat{q},\hat{c},X,x} \quad & \|\mathbf{r} - \mathbf{r}^0\|_\alpha && (4.27)\\
\text{s.t.} \quad & -\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top X \hat{A} = K^\dagger && (4.28)\\
& -\gamma \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top x = k^\dagger && (4.29)\\
& X = \gamma \hat{A}^\top X \hat{A} - \gamma^2 \hat{A}^\top X \hat{B} \left(\hat{R} + \gamma \hat{B}^\top X \hat{B}\right)^{-1} \hat{B}^\top X \hat{A} + \hat{Q} && (4.30)\\
& x = \hat{q} + \gamma (\hat{A} + \hat{B} K^\dagger)^\top x && (4.31)\\
& (\hat{Q}, \hat{R}, \hat{q}, \hat{c}) = \underset{(Q \succeq 0, R \succeq \epsilon I, q, c)}{\arg\min} \sum_{t=0}^{T-1} \left\| \frac{1}{2} s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t \right\|_2^2 && (4.32)\\
& X \succeq 0. && (4.33)
\end{aligned}$$

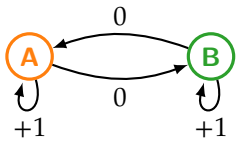
Note that the estimated transition matrices $\hat{A}$, $\hat{B}$ are not optimization variables because the attacker can only modify the rewards, which will not change the learner's estimate on $\hat{A}$ and $\hat{B}$. The attack optimization (4.27)-(4.33) is hard to solve due to the constraint (4.32) 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). Solving the surrogate attack problem, whose feasible region is a subset of the original problem, in general gives a suboptimal solution to (4.27)-(4.33). But it comes with one advantage: convexity.

4.4 Experiments

Throughout the experiments, we use CVXPY [54] 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 4.1a. The discounting factor is $\gamma = 0.9$. The MDP's Q values are shown in table 4.1b. Note that the optimal policy will always pick action *stay*. The



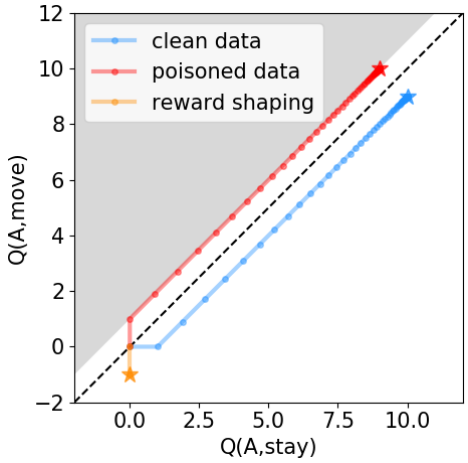
(a) A toy MDP with two states.

	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 4.1: Poisoning TCE in a two-state MDP.

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 $\pi^\dagger(s) = \textit{move}$, 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 (4.12)-(4.15) produces the poisoned data:

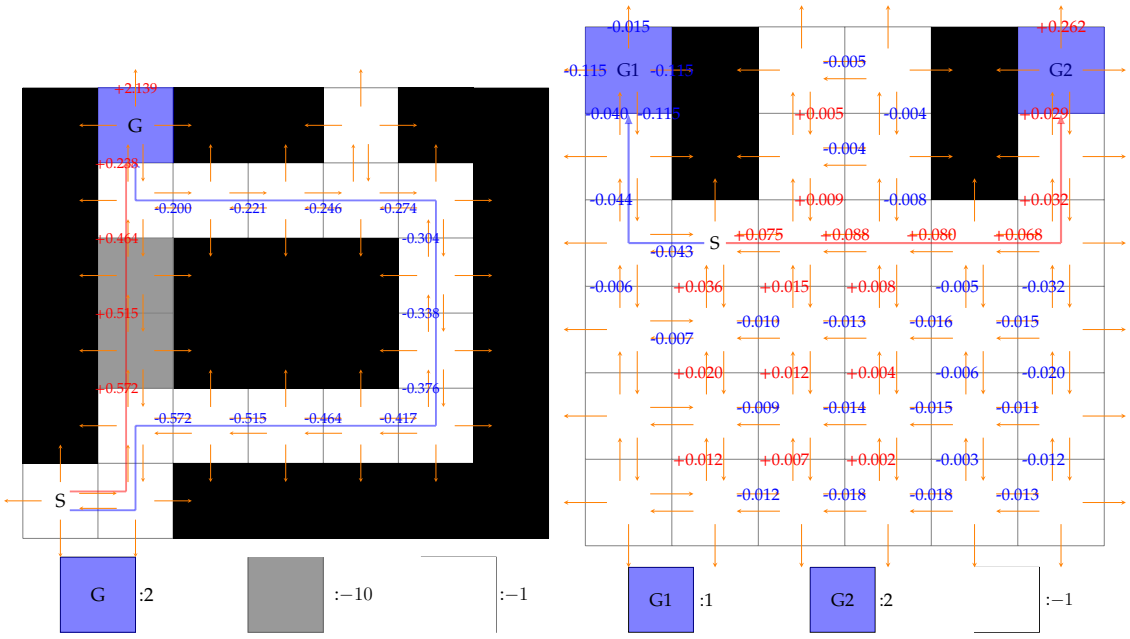
$$(A, \textit{stay}, 0, A) \quad (A, \textit{move}, 1, B) \quad (B, \textit{stay}, 0, B) \quad (B, \textit{move}, 1, A)$$

with attack cost $\|\mathbf{r} - \mathbf{r}^0\|_2 = 2$. The resulting poisoned Q values are shown in table 4.1c. 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 in (4.8) and (4.9) on each data set, and then run value iteration until the Q values converge. In Figure 4.1d, we show the trajectory of Q values for state A, where the x and y axes denote $Q(A, \textit{stay})$ and $Q(A, \textit{move})$ 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 $\textit{stay}$ 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 $\textit{move}$ becomes optimal. This validates our attack.

We also compare our attack with reward shaping [131]. 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 4.1d, 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 [131]. 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 4.2: Poisoning TCE in grid-world tasks.

Experiment 2. As another example, we consider the grid world tasks in [36]. In particular, we focus on two tasks shown in figure 4.2a and 4.2b. In figure 4.2a, 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.

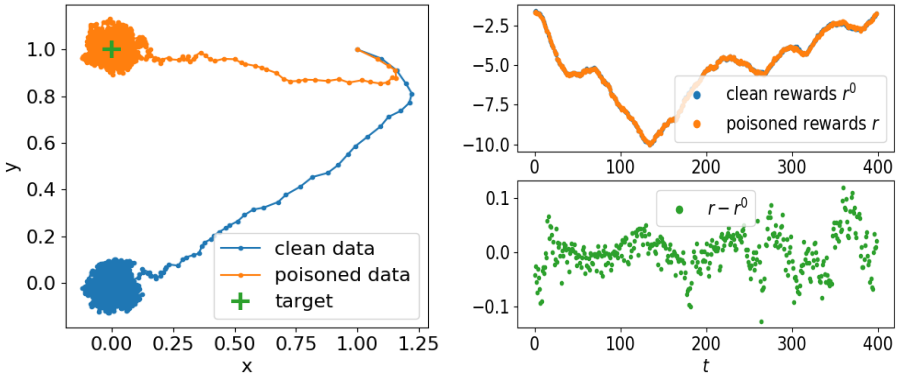
The clean training data contains a single item for every state-action pair. We run the attack with $\epsilon = 0.1$ and $\alpha = 2$. Our attack is successful: with the poisoned data, TCE generates a policy that produces the red trajectory in Figure 4.2a, which is the desired behavior. The attack cost is $\|\mathbf{r} - \mathbf{r}^0\|_2 \approx 2.64$, which is small compared to $\|\mathbf{r}^0\|_2 = 21.61$. In Figure 4.2a, 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.

Experiment 3. In Figure 4.2b 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 4.2b, 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

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 (4.17) where

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 - h\eta/m & 0 \\ 0 & 0 & 0 & 1 - h\eta/m \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m & 0 \\ 0 & h/m \end{bmatrix}. \tag{4.34}$$



(a) Clean and poisoned vehicle trajectory. (b) Clean and poisoned rewards.

Figure 4.3: Poisoning a vehicle running LQR in 4D state space.

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 (4.18)). Throughout the experiment, we let $\gamma = 0.9$ for solving the optimal control policy in (4.21). With the true dynamics and loss function, the computed optimal control policy is

$$K^* = \begin{bmatrix} -1.32 & 0 & -2.39 & 0 \\ 0 & -1.32 & 0 & -2.39 \end{bmatrix}, k^* = \begin{bmatrix} 0 & 0 \end{bmatrix}, \tag{4.35}$$

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

$$\hat{K}^0 = \begin{bmatrix} -1.31 & 1.00\text{e}{-2} & -2.41 & 2.03\text{e}{-3} \\ -1.97\text{e}{-2} & -1.35 & -1.14\text{e}{-2} & -2.42 \end{bmatrix}, \hat{k}^0 = \begin{bmatrix} -4.88\text{e}{-5} & 4.95\text{e}{-6} \end{bmatrix}. \tag{4.36}$$

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 $\frac{1}{2}(s - s^\dagger)^\top Q(s - s^\dagger) + a^\top Ra$ 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 (4.18). 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

$$K^\dagger = \begin{bmatrix} -1.31 & 9.99\text{e}{-3} & -2.41 & 2.02\text{e}{-3} \\ -1.97\text{e}{-2} & -1.35 & -1.14\text{e}{-2} & -2.42 \end{bmatrix}, k^\dagger = \begin{bmatrix} -0.01 & 1.35 \end{bmatrix}. \tag{4.37}$$

We run our attack (4.27)-(4.33) with $\alpha = 2$ and $\epsilon = 0.01$ in (4.32). Figure 4.3 shows the result of our attack. In Figure 4.3a, 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

$$\hat{K} = \begin{bmatrix} -1.31 & 9.99\text{e}{-3} & -2.41 & 2.02\text{e}{-3} \\ -1.97\text{e}{-2} & -1.35 & -1.14\text{e}{-2} & -2.42 \end{bmatrix}, \hat{k} = \begin{bmatrix} -0.01 & 1.35 \end{bmatrix}, \tag{4.38}$$

which matches exactly the target policy $K^\dagger$, $k^\dagger$. In Figure 4.3b, 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 (4.5), 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.

In this chapter, we move to the online renforcement learning setting, where the agent must learn by interacting with the environment. However, we restrict our attention to a particular form of adversarial attack called reward poisoning, where the adversary perturbs the rewards generated by the environment but does not have the power to contaminate the state transition.

5.1 Motivation

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 [190, 38]; in conversational AI, the rewards can be user sentiment or conversation length [48, 109]. In such scenarios, an adversary can manipulate user feedback to influence the RL agent in nefarious ways. Figure 5.1 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 [126]. Such attacks reveal significant security threats in the application of reinforcement learning.

Figure 5.1: 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 paper, 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 5.2 shows a running example that we use throughout the paper. The episodic MDP is a linear chain with five states, with left or right actions and no movement if it hits the boundary. Each

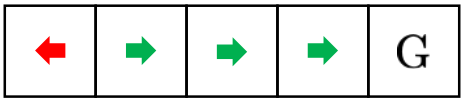


Figure 5.2: A chain MDP with attacker's target policy $\pi^\dagger$

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 5.2. 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:

leftmirgin=*, nolistsep We characterize conditions under which such attacks are guaranteed to fail (thus RL is safe), and vice versa;

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

leftmiiirgiiin=*, noliiistsep We show that effective attacks can be found empirically using deep RL techniques.

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 [73, 112, 97, 19]. For example, in video games the attacker can make small pixel perturbation to a frame [69]) 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 [85, 139, 9, 114, 119], 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* [182, 183, 120] 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 paper instead focuses on the *online* RL attack setting where reward poisoning must be done on the fly.

[74] 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 paper 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 5.3.

Reward Shaping: While this paper 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 [131] has been shown able to speed up learning while preserving the optimal policy. [47] extend potential-based reward shaping to be time-varying while remains policy-preserving. More recently, intrinsic motivations[147, 136, 18, 20] 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.

5.2 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 1.

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 D.4, 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:

$$Q^*(s, a) = \mathbb{E}_{P(s'|s,a)} \left[R(s, a, s') + \gamma \max_{a' \in A} Q^*(s', a') \right] \tag{5.1}$$

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, since most of our analyses happen in the space of value functions.

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

$$a_t \leftarrow \begin{cases} \arg\max_a Q_t(s_t, a), & \text{w.p. } 1-\epsilon \\ \text{uniform from } A, & \text{w.p. } \epsilon. \end{cases} \tag{5.2}$$

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:

$$\begin{aligned} &Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha_t) Q_t(s_t, a_t) + \\ &\alpha_t \left(r_t + \delta_t + \gamma \max_{a' \in A} Q_t(s_{t+1}, a') \right) \end{aligned} \tag{5.3}$$

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

The Threat Model The attacker sits between the environment and the RL agent. In this paper 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: $|\delta_t| \leq \Delta, \forall t$.

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 paper, 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 5.2.1. *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 5.2.2. *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 but itself can be in general non-convex.

5.3 Theoretical Guarantees

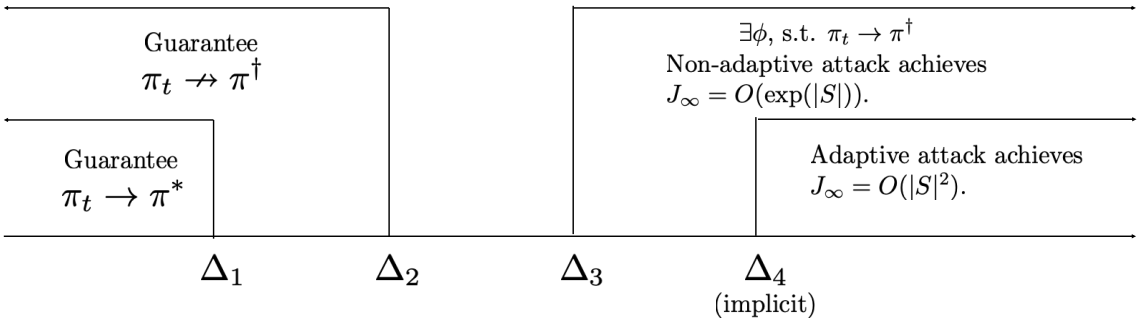


Figure 5.3: A summary diagram of the theoretical results.

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):

$$\xi_t := (s_t, a_t, s_{t+1}, r_t, Q_t) \in \Xi \tag{5.4}$$

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 $\mathcal{Q}^\dagger$:

$$\min_{\phi} \quad \mathbb{E}_\phi \sum_{t=0}^{\infty} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger], \tag{5.5}$$

where the expectation accounts for randomness in Alg 1. We denote $J_\infty(\phi) = E_\phi \sum_{t=0}^{\infty} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]$ the total attack cost, and $J_T(\phi) = E_\phi \sum_{t=0}^{T} \mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]$ the finite-horizon cost. We say the attack is *feasible* if (5.5) is finite.

Next, we characterize attack feasibility in terms of poison magnitude constraint Δ, as summarized in Figure 5.3. 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 (5.5) 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 5.3.1 (Boundedness of Q-learning)**.** *Assume that $\delta_t < \Delta$ for all t, and the stepsize α_t's satisfy that $\alpha_t \leq 1$ for all t, $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. Let Q^* be defined as* (5.1)*. Then, for any attack sequence $\{\delta_t\}$, there exists $N \in \mathbb{N}$ such that, with probability* 1*, for all $t \geq N$, we have*

$$Q^*(s,a) - \frac{\Delta}{1-\gamma} \leq Q_t(s,a) \leq Q^*(s,a) + \frac{\Delta}{1-\gamma}. \tag{5.6}$$

Remark 5.3.2. *The bounds in Theorem 5.3.1 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 5.3.3 (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 5.3.4 (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 5.3.5. *Given a target policy $\pi^\dagger$, define*

$$\Delta_3 = \frac{1+\gamma}{2} \max_{s \in S^\dagger} [\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - \max_{a \in \pi^\dagger(s)} Q^*(s,a)]_+ \tag{5.7}$$

where $[x]_+ \coloneqq \max(x, 0)$. Assume the same conditions on α_t as in Theorem 5.3.1. If $\Delta > \Delta_3$, there is a feasible attack policy $\phi^{sas}_{\Delta_3}$. Furthermore, $J_\infty(\phi^{sas}_{\Delta_3}) \leq O(L^5)$, where L is the covering number.

Algorithm 2 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

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

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

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

def Attack(ξ_t):

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

Theorem 5.3.5 is proved by constructing an attack policy $\phi^{sas}_{\Delta_3}(s_t, a_t)$, detailed in Alg. 2. 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 [61]. It is well-known that ϵ-greedy exploration has a covering time $L \leq O(e^{|S|})$ [91]. Prior work has constructed examples on which this bound is tight [78]. We show in appendix D.1 that on our toy example ϵ-greedy indeed has a covering time $O(e^{|S|})$. Therefore, the objective value of (5.5) for non-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: $k = |S^\dagger| \leq O(\log|S|)$. The FAA attack policy ϕ^{ξ}_{FAA} is defined in Alg. 3.

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. 3 with examples in appendix D.2.

Definition 5.3.6. *Define the shortest ϵ-**distance** from s to s' as*

$$d_\epsilon(s, s') = \min_{\pi \in \Pi} \mathbb{E}_{\pi_\epsilon}[T] \tag{5.9}$$
$$s.t.\ s_0 = s, s_T = s', s_t \neq s', \forall t < T$$

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

$$D_\epsilon = \max_{s, s' \in S} d_\epsilon(s, s') \tag{5.10}$$

Theorem 5.3.8. *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. $|S^\dagger| = k$. If Δ is large enough that the $Clip_\Delta()$ function in Alg. 3 never takes effect, then ϕ^{ξ}_{FAA} is feasible, and we have*

$$J_\infty(\phi^{\xi}_{FAA}) \leq k\frac{|S||A|H}{1-\epsilon} + \frac{|A|}{1-\epsilon}\left[\frac{|A|}{\epsilon}\right]^k D_\epsilon, \tag{5.11}$$

Algorithm 3 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 $Q'_{t+1}(s_t, a_t) = (1-\alpha_t)Q_t(s_t, a_t) + \alpha_t\left(r_t + \gamma(1-EOE)\max_{a' \in A} Q_t(s_{t+1}, a')\right)$.
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

$$\begin{cases} \frac{1}{\alpha_t}[\max_{a \notin A_{s_t}} Q_t(s_t, a) - \\ \qquad Q'_{t+1}(s_t, a_t) + \eta]_+ & \text{if } a_t \in A_{s_t} \\ \frac{1}{\alpha_t}[\max_{a \in A_{s_t}} Q_t(s_t, a) - \\ \qquad Q'_{t+1}(s_t, a_t) + \eta]_- & \text{if } a_t \notin A_{s_t}. \end{cases} \tag{5.8}$$

3: Define $\text{Clip}_\Delta(\delta) = \min(\max(\delta, -\Delta), \Delta)$.
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 $\mathbb{E}_{s \sim \mu_0}\left[d^\epsilon(s, s_{(i)})\right]$.
5: **for** $i = 1, ..., k$ **do**
6: Define the temporary target policy ν_i as

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

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 $\delta_t \leftarrow \text{Clip}_\Delta(g(\{\nu_i(s_t)\}))$.
4:
5: Return $\delta_t \leftarrow \text{Clip}_\Delta(g(\{\pi^\dagger(s_t)\}))$.

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 random walk is bounded by $O(|S|^2)$[104]. 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 5.3.9. *If in addition to the assumptions of Theorem 5.3.8, the maximal episode length $H = O(|S|)$, then $J_\infty(\phi^\xi_{FAA}) \le O(e^k|S|^2|A|)$ in Grid World environments. When the number of target states is small, i.e. $k \le O(\log|S|)$, $J_\infty(\phi^\xi_{FAA}) \le O(\text{poly}(|S|))$.*

Remark 5.3.10. *Theorem 5.3.8 and Corollary 5.3.9 can be thought of as defining an implicit Δ_4, such that*

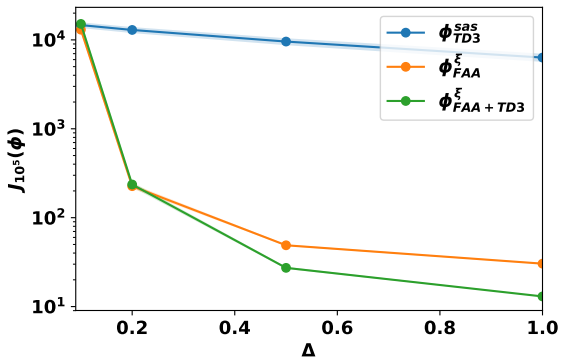


Figure 5.4: Attack cost $J_{10^5}(\phi)$ on different Δ's. Each curve shows mean ± 1 standard error over 1000 independent test runs.

for any $\Delta > \Delta_4$, the clip function in Alg. 3 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 5.3. We use the chain MDP in Figure 5.2 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 5.3.8 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 5.3.5 and Corollary 5.3.9.

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

5.4 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 (5.5) 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 $\{\delta_t \in \mathbb{R} : |\delta_t| \leq \Delta\}$.
- 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:

$$\rho(\xi_t) = \sum_{s \in S^\dagger} \left[\max_{a \notin \pi^\dagger(s)} Q_t(s,a) - \max_{a \in \pi^\dagger(s)} Q_t(s,a) + \eta \right]_+ \tag{5.12}$$

  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:

leftmargin=* s_{t+1} is copied from ξ_t if not the end of episode, else $s_{t+1} \sim \mu_0$.

leftmargin=* a_{t+1} is the RL agent's exploration action drawn according to (5.2), note it involves Q_{t+1}.

leftmargin=* s_{t+2} is the RL agent's new state drawn according to the MDP transition probability $P(\cdot \mid s_{t+1}, a_{t+1})$.

leftmargin=* r_{t+1} is the new (not yet poison) reward according to MDP $R(s_{t+1}, a_{t+1}, s_{t+2})$.

leftmargin=* The attack δ_t happens. The RL agent updates Q_{t+1} according to (5.3).

With the higher-level attack MDP $\mathcal{N}$, we relax the optimal attack problem (5.5) into

$$\phi^* = \arg\min_{\phi} \mathbb{E}_\phi \sum_{t=0}^{\infty} \rho(\xi_t) \tag{5.13}$$

One can now solve (5.13) using Deep RL algorithms. In this paper, we choose Twin Delayed DDPG (TD3) [65], 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 D.3.

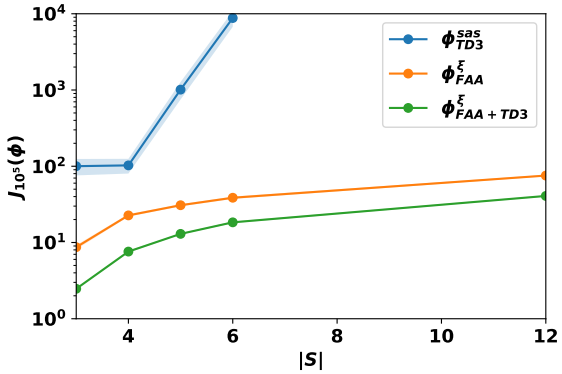


Figure 5.5: Attack performances on the chain MDPs of different lengths. Each curve shows mean ± 1 standard error over 1000 independent test runs.

5.5 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 [120, 74] 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 (5.13) with TD3; $\phi^{\xi}_{FAA+TD3}$ is the attack policy found by TD3 initialized from FAA (Algorithm 3), 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 (5.5).

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 5.2. 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 5.4.

We are able to make several interesting observations:

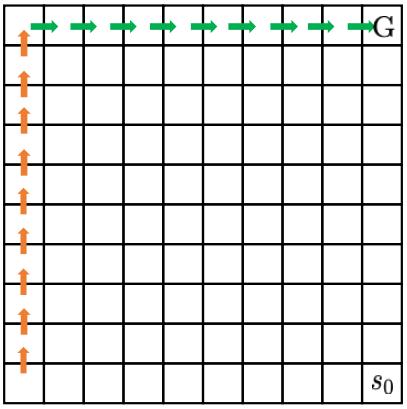
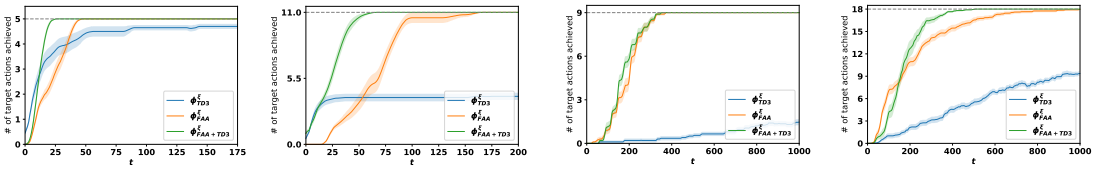


Figure 5.6: The 10×10 Grid World. s_0 is the starting state and G the terminal goal. Each move has a -0.1 negative reward, and a $+1$ reward for arriving at the goal. We consider two partial target policies: $\pi_1^\dagger$ marked by the green arrows, and $\pi_2^\dagger$ by *both* the green and the orange arrows.

(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_{TD3}^{sas} 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 $\phi_{FAA+TD3}^{\xi}$ 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 5.2 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 5.5. The best-found non-adaptive attack ϕ_{TD3}^{sas} is approximately straight on the log-scale plot, suggesting attack cost J growing exponentially with MDP size $|S|$. In contrast, the two adaptive attack polices ϕ_{FAA}^{ξ} and $\phi_{FAA+TD3}^{\xi}$ actually achieves attack cost linear in $|S|$. This is not easy to see from this log-scaled plot; We reproduce Figure 5.5 without the log scale in the appendix D.4, where the linear rate can be clearly verified. This suggests that the upperbound developed in Theorem 5.3.8 and Corollary 5.3.9 can be potentially improved.



(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 5.7: 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.

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 $\phi^{\xi}_{FAA+TD3}$ 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 5.6.

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 5.7.

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 equiped 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 $\phi^{\xi}_{FAA+TD3}$ 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 5.6, 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 $\phi^{\xi}_{FAA+TD3}$

couldn't achieve much improvement, as shown in 5.7c. 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, $\phi^{\xi}_{FAA+TD3}$ 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 $\phi^{\xi}_{FAA+TD3}$.

In this and the next chapters, we move on to the defense side of the problem and investigate whether learning algorithms can be robust against data poisoning attacks, and if so, what is the information-theoretical limits of the learning performance, in online and offline settings respectively. In the rest of this chapter, we start with a general introduction of the RL under data corruption problem along with a thorough overview of prior works. We then follow by formally define the contamination model and learning objectives. For both online and offline reinforcement learning settings, we derive near-matching information-theoretical bounds on the best achievable optimality gap, showing the unique benefit of online interaction to robust learning that is previously unknown to the robust learning community.

6.1 Introduction

While reinforcement learning algorithms have seen tremendous success in recent years, one main drawback of existing RL algorithms is their lack of robustness to data corruption, which severely limits their applications to high-stack decision-making domains with highly noisy data, such as autonomous driving, quantitative trading, or medical diagnosis. In fact, data corruption can be a larger threat in the RL paradigm than in traditional supervised learning, because supervised learning is often applied in a controlled environment where data are collected and cleaned by highly-skilled data scientists and domain experts, whereas RL agents are developed to learn in the wild using raw feedbacks from the environment. While the increasing autonomy and less supervision mark a step closer to the goal of general artificial intelligence, they also make the learning system more susceptible to data corruption: autonomous vehicles can misread traffic signs when the signs are contaminated by adversarial stickers [62]; chatbot can be mistaught by a small group of tweeter users to make misogynistic and racist remarks [126]; recommendation systems can be fooled by a small number of fake clicks/reviews/comments to rank products higher than they should be. Despite the many vulnerabilities, *robustness* against data corruption in RL has not been extensively studied only until recently.

The existing works on *robust* RL are mostly theoretical and can be viewed as a successor of the adversarial bandit literature. However, several drawbacks of this line of approach make them insufficient to modern real-world threats faced by RL agents. We elaborate them below:

1. **Reward vs. transition contamination**: The majority of prior works on adversarial RL focus on reward contamination [60, 127, 128, 197, 145, 79], while in reality the adversary often has stronger control during the adversarial interactions. For example, when a chatbot interacts with an adversarial user, the user has full control over both the rewards and transitions during that conversation episode.

2. **Density of contamination**: The existing works that do handle adversarial/time-varying transitions can only tolerate *sublinear* number of interactions being corrupted [118, 41, 132, 133]. These methods would fail when the adversary's attack budget also grows linearly with time, which is often the case in practice.

3. **Practicability**: The majority of these work focuses on the setting of tabular MDPs and cannot be applied to real-world RL problems that have large state and action spaces and require function approximations.

In this chapter, we address the above shortcomings by developing a variant of natural policy gradient (NPG) methods that, under the linear value function assumption, are provably robust against strongly adaptive adversaries, who can **arbitrarily contaminate** both rewards and transitions in ϵ fraction of all learning episodes. Our algorithm does not need to know ϵ, and is adaptive to the contamination level. Specifically, it guarantees to find an $\tilde{O}(\epsilon^{1/4})$-optimal policy in a polynomial number of steps. Complementarily, we also present a corresponding lower-bound, showing that no algorithm can consistently find a better than $\Omega(\epsilon)$ optimal policy, even with infinite data. In addition to the theoretical results, we also develop a neural network implementation of our algorithm which is shown to achieve strong robustness performance on the MuJoCo continuous control benchmarks [158], proving that our algorithm can be applied to real-world, high-dimensional RL problems.

Related Work

RL in standard MDPs. Learning MDPs with stochastic rewards and transitions is relatively well-studied for the tabular case (that is, a finite number of states and actions). For example, in the episodic setting, the UCRL2 algorithm [14] achieves $O(\sqrt{H^4S^2AT})$ regret, where H is the episode length, S is the state space size, A is the action space size, and T is the total number of steps. Later the UCBVI algorithm [16, 44] achieves the optimal $O(\sqrt{H^2SAT})$ regret matching the lower-bound [135, 43]. Recent work extends the analysis to various linear setting [81, 174, 173, 180, 15, 191, 34, 56, 86] with known linear feature. For unknown feature, [3] proposes a sample efficient algorithm that explicitly learns feature representation under the assumption that the transition matrix is low rank. Beyond the linear settings, there are works assuming the function class has low Eluder dimension which so far is known to be small only for linear functions and generalized linear models [134]. For more general function approximation, [77, 156] showed that polynomial sample complexity is achievable as long as the MDP and the given function class together induce low Bellman rank and Witness rank, which include almost all prior models such as tabular MDP, linear MDPs [174, 81], Kernelized nonlinear regulators [86], low rank MDP [3], and Bellman completion under linear functions [180].

Policy Gradient and Policy Optimization Policy Gradient [168, 157] and Policy optimization methods are widely used in practice [87, 149, 150] and have demonstrated amazing performance on challenging applications [21, 7]. Unlike model-based approach or Bellman-backup based approaches, PG methods directly optimize the objective function and are often more robust to model-misspecification [2]. In addition to being robust to model-misspecification, we show in this chapter that vanilla NPG is also robust to constant fraction and bounded adversarial corruption on both rewards and transitions.

RL with adversarial rewards. Almost all prior works on adversarial RL study the setting where the reward functions can be adversarial but the transitions are still stochastic and remain unchanged throughout the learning process. Specifically, at the beginning of each episode, the adversary must decide on a reward function for this episode, and can not change it for the rest of the episode. Also, the majority of these works focus on tabular MDPs. Early works on adversarial MDPs assume a known transition function and full-information feedback. For example, [60] proposes the algorithm MDP-E and proves a regret bound of $\tilde{O}(\tau\sqrt{T\log A})$ in the non-episodic setting, where τ is the mixing time of the MDP; Later, [197] consider the episodic setting and propose the O-REPS algorithm which applies Online Mirror Descent over the space of occupancy measures, a key component adopted by [145] and [79]. O-REPS achieves the optimal regret $\tilde{O}(\sqrt{H^2T\log(SA)})$ in this setting. Several works consider the harder bandit feedback model while still assuming known transitions. The work [127] achieves regret $\tilde{O}(\sqrt{H^3AT}/\alpha)$ assuming that all states are reachable with some probability α under all policies. Later, [127] eliminates the dependence on α but only achieves $O(T^{2/3})$ regret. The O-REPS algorithm of [197] again achieves the optimal regret $\tilde{O}(\sqrt{H^3SAT})$. To deal with unknown transitions, [128] proposes the Follow the Perturbed Optimistic Policy algorithm and achieves $\tilde{O}(\sqrt{H^2S^2A^2T})$ regret given full-information feedback. Combining the idea of confidence sets and Online Mirror Descent, the UC-O-REPS algorithm of [145] improves the regret to $\tilde{O}(\sqrt{H^2S^2AT})$. A few recent works start to consider the hardest setting assuming unknown transition as well as bandit feedback. [145] achieves $O(T^{3/4})$ regret, which is improved by [79] to $\tilde{O}(\sqrt{H^2S^2AT})$, matching the regret of UC-O-REPS in the full information setting. Also, note that the lower bound of $\Omega(\sqrt{H^2SAT})$ [78] still applies. In summary, it is found that on tabular MDPs with oblivious reward contamination, an $O(\sqrt{T})$ regret can still be achieved. Recent improvements include best-of-both-worlds algorithms [82], data-dependent bound [105] and extension to linear function approximation [129].

RL with adversarial transitions and rewards. Very few prior works study the problem of both adversarial transitions and adversarial rewards, in fact, only one that we are aware of [118]. They study a setting where only a constant C number of episodes can be corrupted by the adversary, and most of their technical effort dedicate to designing an algorithm that is agnostic to C, i.e. the

algorithm doesn't need to know the contamination level ahead of time. As a result, their algorithm takes a multi-layer structure and cannot be easily implemented in practice. Their algorithm achieves a regret of $O(C\sqrt{T})$ for tabular MDPs and $O(C^2\sqrt{T})$ for linear MDPs, which unfortunately becomes vacuous when $C \geq \Omega(\sqrt{T})$ and $C \geq \Omega(T^{1/4})$, respectively. Note that the contamination ratio C/T approaches zero when T increases, and hence their algorithm cannot handle constant fraction contamination. Notably, in all of the above works, the adversary can *partially adapt* to the learner's behavior, in the sense that the adversary can pick an adversary MDP $\mathcal{M}_k$ or reward function r_k at the start of episode k based on the history of interactions so far. However, it can no longer adapt its strategy after the episode starts, and therefore, the learner can still use a randomization strategy to trick the adversary.

A separate line of work studies the *online MDP* setting, where the MDP is not adversarial but *slowly* change over time, and the amount of change is bounded under a total-variation metric [41, 132, 133, 55]. Due to the slow-changing nature of the environment, algorithms in these works typically uses a sliding window approach where the algorithm keeps throwing away old data and only learns a policy from recent data, assuming that most of them come from the MDP that the agent is currently experiencing. These methods typically achieve a regret in the form of $O(\Delta^c K^{1-c})$, where Δ is the total variation bound. It is worth noting that all of these regrets become vacuous when the amount of variation is linear in time, i.e. $\Delta \geq \Omega(T)$. Separately, it is shown that when both the transitions and the rewards are adversarial in every episode, the problem is at least as hard as stochastic parity problem, for which no computationally efficient algorithm exists [172].

Learning robust controller. A different type of robustness has also been considered in RL [142, 46] and robust control [192, 141], where the goal is to learn a control policy that is robust to potential misalignment between the training and deployment environment. Such approaches are often conservative, i.e. the learned polices are sub-optimal even if there is no corruption. In comparison, our approach can learn as effectively as standard RL algorithms without corruption. Interestingly, parallel to our work, a line of concurrent work in the robust control literature [185, 184, 186] has also found that policy optimization method enjoys some implicit regularization/robustness property that can automatically converge to robust control policies. An interesting future direction could be to understand the connection between these two kind of robustness.

Robust statistics. One of the most important discoveries in modern robust statistics is that there exists computationally efficient and robust estimator that can learn near-optimally even under the strongest adaptive adversary. For example, in the classic problem of Gaussian mean estimation, the recent works [49, 101] present the first computational and sample-efficient algorithms. The algorithm in [49] can generate a robust mean estimate $\hat{\mu}$, such that $\|\hat{\mu} - \mu\|_2 \leq O(\epsilon\sqrt{\log(1/\epsilon)})$ under ϵ corruption. Crucially, the error bound does not scale with the dimension d of the problem,

suggesting that the estimator remains robust even in high dimensional problems. Similar results have since been developed for robust mean estimation under weaker assumptions [51], and for supervised learning and unsupervised learning tasks [37, 50]. We refer readers to [52] for a more thorough survey of recent advances in high-dimensional robust statistics.

6.2 Problem Definitions

A Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu_0)$ is specified by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition model $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ (where $\Delta(\mathcal{S})$ denotes a distribution over $\mathcal{S}$), a (stochastic and possibly unbounded) reward function $r : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$, a discounting factor $\gamma \in [0, 1)$, and an initial state distribution $\mu_0 \in \Delta(\mathcal{S})$, i.e. $s_0 \sim \mu_0$. In this chapter, we assume that $\mathcal{A}$ is a small and finite set, and denote $A = |\mathcal{A}|$. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ specifies a decision-making strategy in which the agent chooses actions based on the current state, i.e., $a \sim \pi(\cdot|s)$.

The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ is defined as the expected discounted sum of future rewards, starting at state s and executing π, i.e. $V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|\pi, s_0 = s\right]$, where the expectation is taken with respect to the randomness of the policy and environment $\mathcal{M}$. Similarly, the *state-action* value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as $Q^\pi(s, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|\pi, s_0 = s, a_0 = a\right]$.

We define the discounted state-action distribution d_s^π of a policy π: $d_{s'}^\pi(s, a) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(s_t = s, a_t = a|s_0 = s')$, where $\mathbb{P}^\pi(s_t = s, a_t = a|s_0 = s')$ is the probability that $s_t = s$ and $a_t = a$, after we execute π from $t = 0$ onwards starting at state s' in model $\mathcal{M}$. Similarly, we define $d_{s',a'}^\pi(s, a)$ as: $d_{s',a'}^\pi(s, a) := (1 - \gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(s_t = s, a_t = s|s_0 = s', a_0 = a')$. For any state-action distribution ν, we write $d_\nu^\pi(s, a) := \sum_{(s',a') \in \mathcal{S}\times\mathcal{A}} \nu(s', a') d_{s',a'}^\pi(s, a)$. For ease of presentation, we assume that the agent can reset to $s_0 \sim \mu_0$ at any point in the trajectory. We denote $d_\nu^\pi(s) = \sum_a d_\nu^\pi(s, a)$.

The goal of the agent is to find a policy π that maximizes the expected value from the starting state s_0, i.e. the optimization problem is: $\max_\pi V^\pi(\mu_0) := \mathbb{E}_{s\sim\mu_0} V^\pi(s)$, where the $\max$ is over some policy class.

For completeness, we specify a d_ν^π-sampler and an unbiased estimator of $Q^\pi(s, a)$ in Algorithm 4, which are standard in discounted MDPs [4, 2]. The d_ν^π sampler samples (s, a) i.i.d from d_ν^π, and the Q^π sampler returns an unbiased estimate of $Q^\pi(s, a)$ for a given pair (s, a) by a single roll-out from (s, a). Later, when we define the contamination model and the sample complexity of learning, we treat each call of d_ν^π-sampler (optionally followed by a $Q^\pi(s, a)$-estimator) as a *single episode*, as in practice both of these procedures can be achieved in a single roll-out from μ_0.

Assumption 6.2.1 (Linear Q function)**.** *For the theoretical analysis, we focus on the setting of linear value function approximation. In particular, we assume that there exists a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, such that for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ and any policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$, we have*

$$Q^\pi(s, a) = \phi(s, a)^\top w^\pi, \text{ for some } \|w^\pi\| \leq W \tag{6.1}$$

We also assume that the feature is bounded, i.e. $\max_{s,a}\|\phi(s,a)\|_2 \leq 1$*, and the reward function has bounded first and second moments, i.e.* $\mathbb{E}\left[r(s,a)\right] \in [0,1]$ *and* $Var(r(s,a)) \leq \sigma^2$ *for all* (s,a)*.*

Remark 6.2.1. Assumption 7.2.1 is satisfied, for example, in tabular MDPs and linear MDPs of [81] or [173]. Unlike most theoretical RL literature, we allow the reward to be stochastic and unbounded. Such a setup aligns better with applications with a low signal-to-noise ratio and motivates the requirement for nontrivial robust learning techniques.

Notation. When clear from context, we write $d^\pi(s,a)$ and $d^\pi(s)$ to denote $d^\pi_{\mu_0}(s,a)$ and $d^\pi_{\mu_0}(s)$ respectively. For iterative algorithms which obtain policies at each episode, we let V^i,Q^i and A^i denote the corresponding quantities associated with episode i. For a vector v, we denote $\|v\|_2 = \sqrt{\sum_i v_i^2}$, $\|v\|_1 = \sum_i |v_i|$, and $\|v\|_\infty = \max_i |v_i|$. We use Uniform$(\mathcal{A})$ (in short Unif$_\mathcal{A}$) to represent a uniform distribution over the set $\mathcal{A}$.

The Contamination Model

In this chapter, we study the performance of algorithms under the ϵ-*contamination model*, a widely studied adversarial model in the robust statistics literature, e.g. see [49]. In the classic robust mean estimation problem, given a dataset D and a learning algorithm f, the ϵ-contamination model assumes that the adversary has full knowledge of the dataset D and the learning algorithm f, and can arbitrarily change ϵ-fraction of the data in the dataset and then send the contaminated data to the learner. The goal of the learner is to identify an $O(\text{poly}(\epsilon))$-optimal estimator of the mean despite the ϵ-contamination.

Unfortunately, the original ϵ-contamination model is defined for the offline learning setting and does not directly generalize to the online setting, because it doesn't specify the availability of knowledge and the order of actions between the adversary and the learner in the time dimension. In this paper, we define the ϵ-contamination model for online learning as follows:

Definition 6.2.1 (ϵ-contamination model for Reinforcement Learning)**.** Given ϵ and the clean MDP $\mathcal{M}$, an ϵ-contamination adversary operates as follows:

1. The adversary has full knowledge of the MDP $\mathcal{M}$ and the learning algorithm, and observes all the historical interactions.I
2. At any time step t, the adversary observes the current state-action pair (s_t, a_t), as well as the reward and next state returned by the environment, (r_t, s_{t+1}). He then can decide whether to replace (r_t, s_{t+1}) with an arbitrary reward and next state $(r_t^\dagger, s_{t+1}^\dagger) \in \mathbb{R} \times \mathcal{S}$.
3. The only constraint on the adversary is that if the learning process terminates after K episodes, he can contaminate in at most ϵK episodes.

Compared to the standard adversarial models studied in online learning [153], adversarial bandits [30, 117, 70] and adversarial RL [118, 79], the ϵ-contamination model in Definition 6.2.1 is stronger in several ways: (1) The adversary can adaptively attack after observing the action of the learner as well as the feedback from the clean environments; (2) the adversary can perturb the data arbitrarily (any real-valued reward and any next state from the state space) rather than sampling it from a pre-specified bounded adversarial reward function or adversarial MDP.

Given the contamination model, our first result is a lower-bound, showing that under the ϵ-contamination model, one can only hope to find an $O(\epsilon)$-optimal policy. Exact optimal policy identification is not possible even with infinite data.

Theorem 6.2.1 (lower bound)**.** *For any algorithm, there exists an MDP such that the algorithm fails to find an* $\left(\frac{\epsilon}{2(1-\gamma)}\right)$*-optimal policy under the* ϵ*-contamination model with a probability of at least* $1/4$*.*

The high-level idea is that we can construct two MDPs, M and M', with the following properties: 1. No policy can be $O(\epsilon/(1-\gamma))$ optimal on both MDP simultaneously. 2. An ϵ-contamination adversary can with large probability mimic one MDP via contamination in the other, regardless of the learner's behavior. Therefore, under contamination, the learner will not be able to distinguish M and M' and must suffer $\Omega(\epsilon/(1-\gamma))$ gap on at least one of them.

Background on NPG

Given a differentiable parameterized policy $\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A})$, NPG can be written in the following actor-critc style update form. With the dataset $\{s_i, a_i, \widehat{Q}^{\pi_\theta}(s_i, a_i)\}_{i=1}^N$ where $s_i, a_i \sim d_\nu^{\pi_\theta}$, and $\widehat{Q}^{\pi_\theta}(s_i, a_i)$ is unbiased estimate of $Q^{\pi_\theta}(s, a)$ (e.g., via Q^π-estimator), we have

$$\widehat{w} \in \mathop{\arg\min}_{w:\|w\|_2 \le W} \sum_{i=1}^N \left(w^\top \nabla \log \pi_\theta(a_i|s_i) - \widehat{Q}^{\pi_\theta}(s_i, a_i)\right)^2$$
$$\theta' = \theta + \eta \widehat{w}. \tag{6.2}$$

In theoretical part of this chapter, we focus on softmax linear policy, i.e., $\pi_\theta(a|s) \propto \exp(\theta^\top \phi(s, a))$. In this case, note that $\nabla \log \pi_\theta(a|s) = \phi(s, a)$, and it is not hard to verify that the policy update procedure is equivalent to:

$$\pi_{\theta'}(a|s) \propto \pi_\theta(a|s) \exp\left(\eta \widehat{w}^\top \phi(s, a)\right), \quad \forall s, a,$$

which is equivalent to running Mirror Descent on each state with a reward vector $\widehat{w}^\top \phi(s, \cdot) \in \mathbb{R}^{|\mathcal{A}|}$. We refer readers to [4] for more detailed explanation of NPG and the equivalence between the form in Eq. (6.2) and the classic form that uses Fisher information matrix. Similar to [4], we make the

following assumption of having access to an exploratory reset distribution, under which it has been shown that NPG can converge to the optimal policy without contamination.

Assumption 6.2.2 (Relative condition number)**.** *With respect to any state-action distribution υ, define:*

$$\Sigma_\upsilon = \mathbb{E}_{s,a\sim\upsilon}\left[\phi_{s,a}\phi_{s,a}^\top\right],$$

and define

$$\sup_{w\in\mathbb{R}^d} \frac{w^\top \Sigma_{d^*} w}{w^\top \Sigma_\nu w} = \kappa, \text{ where } d^*(s,a) = d_{\mu_0}^{\pi^*}(s) \circ \mathit{Unif}_{\mathcal{A}}(a)$$

We assume κ is finite and small w.r.t. a reset distribution ν available to the learner at training time.

6.3 The Natural Robustness of NPG Against Bounded corruption

Our first result shows that, surprisingly, NPG can already be robust against ϵ-contamination, if the adversary can only generate small and bounded rewards. In particular, we assume that the adversarial rewards is bounded in $[0,1]$ (the feature $\phi(s,a)$ is already bounded).

Theorem 6.3.1 (Natural robustness of NPG)**.** *Under assumptions 7.2.1 and 6.2.2, given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2 guarantees with a $poly(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination with adversarial rewards bounded in $[0,1]$, we have*

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \le \tilde{O}\left(\max\left[\alpha, W\sqrt{\frac{|\mathcal{A}|\kappa\epsilon}{(1-\gamma)^3}}\right]\right)$$

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

A few remarks are in order.

Remark 6.3.1 (Agnostic to the contamination level ϵ)**.** It is worth emphasizing that to achieve the above bound, the hyperparameters of NPG are agnostic to the value of ϵ, and so the algorithm can be applied in the more realistic setting where the agent does not have knowledge of the contamination level ϵ, similar to what's achieved in [118] with a complicated nested structure. The same property is also achieved by the FPG algorithm in the next section.

Remark 6.3.2 (Dimension-independent robustness guarantee)**.** Theorem 6.3.1 guarantees that NPG can find an $O(\epsilon^{1/2})$-optimal policy after polynomial number of episodes, provided that $|\mathcal{A}|$ and κ are small. Conceptually, the relative condition number κ indicates how well-aligned the initial state distribution is to the occupancy distribution of the optimal policy. A good initial distribution can

have a κ as small as 1, and so κ is independent of d. Interested readers can refer to [4] (Remark 6.3) for additional discussion on the relative condition number. Here, importantly, the optimality gap does not directly scale with d, and so the guarantee will not blow up on high-dimensional problems. This is an important attribute of robust learning algorithms heavily emphasized in the traditional robust statistics literature.

The proof of Theorem 6.3.1 relies on the following NPG regret lemma, first developed by [60] for the MDP-Expert algorithm and later extend to NPG by [4, 2]:

Lemma 6.3.1 (NPG Regret Lemma)**.** *Suppose Assumption 7.2.1 and 6.2.2 hold and Algorithm 2 starts with $\theta^{(0)} = 0$, $\eta = \sqrt{2\log|\mathcal{A}|/(W^2T)}$. Suppose in addition that the (random) sequence of iterates satisfies the assumption that*

$$\mathbb{E}\left[\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a) - \phi(s,a)^\top w^{(t)}\right)^2\right]\right] \le \epsilon_{stat}^{(t)}.$$

Then, we have that

$$\mathbb{E}\left[\sum_{t=1}^{T}\{V^*(\mu_0) - V^{(t)}(\mu_0)\}\right] \le \frac{W}{1-\gamma}\sqrt{2\log|\mathcal{A}|T} + \sum_{t=1}^{T}\sqrt{\frac{4|\mathcal{A}|\kappa\epsilon_{stat}^{(t)}}{(1-\gamma)^3}}.$$

Intuitively, Lemma 6.3.1 decompose the regret of NPG into two terms. The first term corresponds to the regret of standard mirror descent procedure, which scales with $\sqrt{T}$. The second term corresponds to the estimation error on the Q value, which acts as the reward signal for mirror descent. When not under attack, estimation error $\epsilon_{stat}^{(t)}$ goes to zero as the number of samples M gets larger, which in turn implies the global convergence of NPG. However, when under bounded attack, the generalization error $\epsilon_{stat}^{(t)}$ will not go to zero even with infinite data. Nevertheless, we can show that it is bounded by $O(\epsilon^{(t)})$ when the sample size M is large enough, where $\epsilon^{(t)}$ denotes the fraction of episodes being corrupted in iteration t. Note that by definition, we have $\sum_t \varepsilon^{(t)} \le \varepsilon T$.

Lemma 6.3.2 (Robustness of linear regression under bounded contamination)**.** *Suppose the adversarial rewards are bounded in $[0,1]$, and in a particular iteration t, the adversary contaminates $\epsilon^{(t)}$ fraction of the episodes, then given M episodes, it is guaranteed that with probability at least $1-\delta$,*

$$\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a) - \phi(s,a)^\top w^{(t)}\right)^2\right] \le 4\left(W^2 + WH\right)\left(\epsilon^{(t)} + \sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right).$$

where $H = (\log\delta - \log M)/\log\gamma$ is the effective horizon.

This along with the NPG regret lemma guarantees that the expected regret of NPG is bounded by $O(\sqrt{T} + M^{-1/4} + \sqrt{\epsilon}T)$ which in turn guarantees to identify an $O(\sqrt{\epsilon})$-optimal policy.

Algorithm 1 d_ν^π sampler and Q^π estimator

function d_ν^π-SAMPLER
  Input: A reset distribution $\nu \in \Delta(\mathcal{S} \times \mathcal{A})$.
  Sample $s_0, a_0 \sim \nu$.
  Execute π from s_0, a_0; at any step t with (s_t, a_t), return (s_t, a_t) with probability $1 - \gamma$.
function Q^π-ESTIMATOR
  Input: current state-action (s, a), a policy π.
  Execute π from $(s_0, a_0) = (s, a)$; at step t with (s_t, a_t), terminate with probability $1 - \gamma$.
  Return: $\widehat{Q}^\pi(s, a) = \sum_{i=0}^{t} r(s_i, a_i)$.

[In an adversarial episode, the adversary can hijack the d_ν^π sampler to return any (s, a) pair and the Q^π-estimator to return any $\widehat{Q}^\pi(s, a) \in \mathbb{R}$.]

In the special case of tabular MDPs, $\phi(s, a)$ will all be one-hot vectors and W will in general by on the order of $O(\sqrt{SA})$, which means that the bound given by Theorem 6.3.1 still scales with the size of the state space. In the following corollary, we show that this dependency can be removed through a tighter analysis.

Corollary 6.3.1 (Dimension-free Robustness of NPG in tabular MDPs)**.** *Given a tabular MDP and assumption 6.2.2, given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2 guarantees with a $poly(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination with adversarial rewards bounded in $[0, 1]$, we have*

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \leq \tilde{O}\left(\max\left[\alpha, \sqrt{\frac{|\mathcal{A}|\kappa\epsilon}{(1-\gamma)^5}}\right]\right)$$

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

In the more general case of linear MDP, W will not necessarily scale with d in an obvious way and thus we leave Theorem 6.3.1 untouched.

6.4 FPG: Robust NPG Against Unbounded Corruption

Our second result is the Filtered Policy Gradient (FPG) algorithm, a robust variant of the NPG algorithm [88, 4] that can be robust against arbitrary and *potentially unbounded* data corruption. Specifically, FPG replace the standard linear regression solver in NPG with a statistically robust alternative. In this chapter, we use the SEVER algorithm [50]. In practice, one can substitute it with any computationally efficient robust linear regression solver. We show that FPG can find an $O(\epsilon^{1/4})$-optimal policy under ϵ-contamination with a polynomial number of samples.

Algorithm 2 Natural Policy Gradient (NPG)

Require: Learning rate η; number of episodes per iteration M

1: Initialize $\theta^{(0)} = 0$.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Call Algorithm 4 M times with $\pi^{(t)}$ to obtain a dataset that consist of $s_i, a_i \sim d_\nu^{(t)}$ and $\widehat{Q}^{(t)}(s_i, a_i)$, $i \in [M]$.
4: Solve the linear regression problem

$$w^{(t)} = \arg\min_{\|w\|_2 \le W} \sum_{i=1}^{M} \left(\widehat{Q}^{(t)}(s_i, a_i) - w^\top \nabla_\theta \phi(s_i, a_i) \right)^2$$

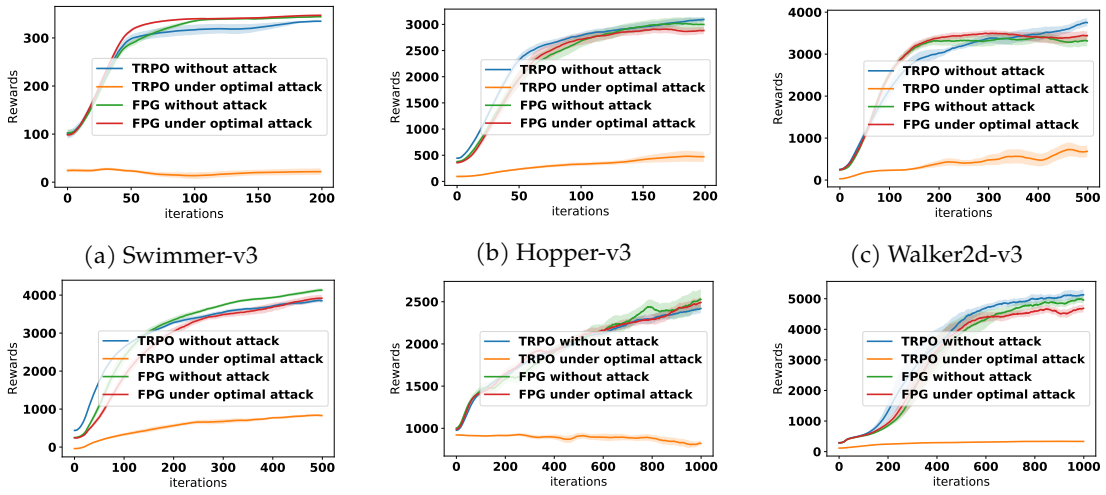
5: Update $\theta^{(t+1)} = \theta^{(t)} + \eta w^{(t)}$.

Algorithm 3 Robust Linear Regression via `SEVER`

Input: Dataset $\{(x_i, y_i)\}_{i=1:M}$, a standard linear regression solver $\mathcal{L}$, and parameter $\sigma' \in \mathbb{R}_+$.
Initialize $S \leftarrow \{1, \ldots, M\}$, $f_i(w) = \|y_i - w^\top x_i\|^2$.
repeat
$w \leftarrow \mathcal{L}(\{(x_i, y_i)\}_{i \in S})$. ▷ Run learner on S.
Let $\widehat{\nabla} = \frac{1}{|S|} \sum_{i \in S} \nabla f_i(w)$.
Let $G = [\nabla f_i(w) - \widehat{\nabla}]_{i \in S}$ be the $|S| \times d$ matrix of centered gradients.
Let v be the top right singular vector of G.
Compute the vector τ of *outlier scores* defined via $\tau_i = \left((\nabla f_i(w) - \widehat{\nabla}) \cdot v \right)^2$.
$S' \leftarrow S$
if $\frac{1}{|S|} \sum_{i \in S} \tau_i \le c_0 \cdot \sigma'^2$, for some constant $c_0 > 1$ **then**
$S = S'$ ▷ We only filter out points if the variance is larger than an appropriately chosen threshold.
else
Draw T from Uniform$[0, \max_i \tau_i]$.
$S = \{i \in S : \tau_i < T\}$.
until $S = S'$.
Return w.

Theorem 6.4.1. *Under assumptions 7.2.1 and 6.2.2, given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2, using Algorithm 3 as the linear regression solver, guarantees with a poly$(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination, we have*

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \le \tilde{O}\left(\max\left[\alpha, \sqrt{\frac{|\mathcal{A}|\kappa\,(W^2 + \sigma W)}{(1-\gamma)^4}} \epsilon^{1/4} \right] \right). \tag{6.3}$$



(a) Swimmer-v3 (b) Hopper-v3 (c) Walker2d-v3

(d) HalfCheetah-v3 (e) Ant-v3 (f) Humanoid-v3

Figure 6.1: Experiment Results on the 6 MuJoTo benchmarks.

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

The proof of Theorem 6.4.1 relies on a similar result to Lemma 6.3.2, which shows that if we use Algorithm 3 as the linear regression subroutine, then $\epsilon_{stat}^{(t)}$ can be bounded by $O(\sqrt{\epsilon^{(t)}})$ when the sample size M is large enough, even under unbounded ϵ-contamination.

Lemma 6.4.1 (Robustness of SEVER under unbounded contamination)**.** *Suppose the adversarial rewards are unbounded, and in a particular iteration t, the adversarial contaminate $\epsilon^{(t)}$ fraction of the episodes, then given M episodes, it is guaranteed that if $\epsilon^{(t)} \leq c$, for some absolute constant c, and any constant $\tau \in [0, 1]$, we have*

$$\mathbb{E}\left[\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a) - \phi(s,a)^\top w^{(t)}\right)^2\right]\right] \leq O\left(\left(W^2 + \frac{\sigma W}{1-\gamma}\right)\left(\sqrt{\epsilon^{(t)}} + f(d,\tau)M^{-\frac{1}{2}} + \tau\right)\right). \tag{6.4}$$

where $f(d,\tau) = \sqrt{d\log d} + \sqrt{\log(1/\tau)}$.

In Lemma 6.4.1, c is the break point of SEVER and is an absolute constant that does not depend on the data, and $(1-\tau)$ is the probability that the clean data satisfies a certain stability condition which suffices for robust learning.

Algorithm 4 Robust Policy Cover-Policy Gradient (PC-PG)

1: **Input**: iterations N, threshold β, regularizer λ
2: Initialize $\pi^0(a|s)$ to be uniform.
3: **for** episode $n = 0, \dots N-1$ **do**
4: Define the policy cover's state-action distribution ρ^n_{cov} as

$$\rho^n_{\text{cov}}(s,a) = \sum_{i=0}^{n} d^i(s,a)/(n+1)$$

5: Sample $\{s_i, a_i\}_{i=1}^K \sim \rho^n_{\text{cov}}(s,a)$ and estimate the covariance of π^n as

$$\widehat{\Sigma}^n = (n+1)\left(\sum_{i=1}^{K} \phi(s_i,a_i)\phi(s_i,a_i)^\top/K\right) + \lambda I$$

6: Set the exploration bonus b^n to reward infrequently visited state-action under ρ^n_{cov}

$$b^n(s,a) = \frac{\mathbf{1}\{(s,a) \ : \ \phi(s,a)^\top(\widehat{\Sigma}^n_{\text{cov}})^{-1}\phi(s,a) \geq \beta\}}{1-\gamma}.$$

7: Update $\pi^{n+1} =$ Robust-NPG-Update$(\rho^n_{\text{cov}}, b^n)$ [Alg. 9 in the appendix, similar to Alg. 2].
8: **return** $\hat{\pi} :=$ Uniform$\{\pi^0, ..., \pi^{N-1}\}$.

6.5 Robust NPG with Exploration via Policy Cover

The Policy Cover-Policy Gradient (`PC-PG`) algorithm, defined in Algorithm 4, is an exploratory policy gradient methods recently developed by [2]. Intuitively, PC-PG is a spiritually inheritor of the `RMax` algorithm [28], and encourages exploration by adding reward bonuses in directions of the feature space that past polices (stored in the policy cover) haven't visited sufficiently. Similar to the NPG algorithm, we show that `PC-PG` enjoys a (weaker) natural robustness against bounded data corruption. This gives us the following robustness guarantee:

Theorem 6.5.1 (Best hyperparameters, assuming known ϵ)**.** *There exists a set of hyperparameters, such that Algorithm 4 guarantees with probability at least* $1-\delta$

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \leq \tilde{O}\left(d^2\epsilon^{1/7}\right) \tag{6.5}$$

with poly $(d, W, \sigma, \kappa, |\mathcal{A}|, 1/(1-\gamma), 1/\alpha)$ *number of episodes.*

Remark 6.5.1 (The scaling with dimension d)**.** Compared to the guarantee of vanilla NPG, `PC-PG` alleviate the requirement of a good initial distribution with small relative conditional number. However, this process introduce a dependency on d. In particular, the gap in Theorem 6.5.1 is on the order of $\tilde{O}\left(d^2\epsilon^{1/7}\right)$. This implies that for any fixed ϵ, the bound becomes vacuous for high

dimensional problems where $d \geq \Omega(\epsilon^{-2/3})$. Intuitively, the dependency on d is introduced because PC-PG is trying to find a initial state-action distribution with good coverage, i.e. a distribution whose covariance matrix has a lower-bounded smallest eigenvalue. Under the assumption that $\|\phi(s,a)\|_2 \leq 1$, such a distribution will have a covariance matrix whose eigenvalues are all on the order of $O(1/d)$. and so the value of κ will be on the order of $O(d)$, which by Theorem 6.4.1 will similarly introduce a d dependency. We expect that for a robust RL algorithm to avoid the d dependency, it must gradually find a state-action distribution approaching d^*. How to design such an algorithm is left as an open problem.

6.6 Experiments

In the theoretical analysis, we rely on the assumption of linear Q function, finite action space and exploratory initial state distribution to prove the robustness guarantees for NPG and FPG. In this section, we present a practical implementation of FPG, based on the *Trusted Region Policy Optimization* (TRPO) algorithm [148], in which the conjugate gradient step (equivalent to the linear regression step in Alg. 2) is robustified with SEVER. The pseudo-code and implementation details are discussed in appendix E.7[1]. In this section, we demonstrate its empirical performance on the MuJoCo benchmarks [158], a set of high-dimensional continuous control domains where both assumptions no longer holds, and show that FPG can still consistently performs near-optimally with and without attack.

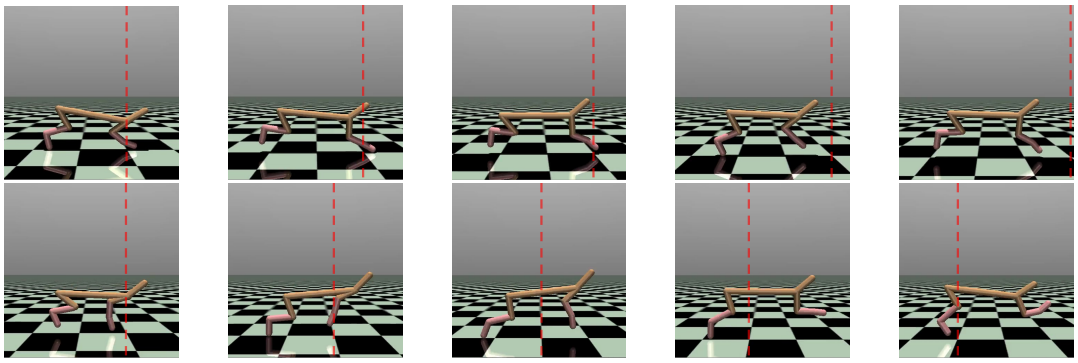
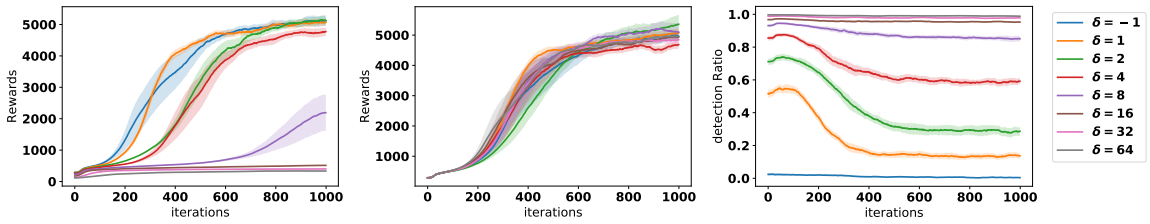


Figure 6.2: Consecutive Frames of Half-Cheetah trained with TRPO (top row) and FPG (bottom row) respectively under $\delta = 100$ attack. The dashed red line serves as a stationary reference object. TRPO was fooled to learn a "running backward" policy, contrasted with the normal "running forward" policy learned by FPG.

[1] A Pytorch Implementation of FPG-TRPO can be found at https://github.com/zhangxz1123/FilteredPolicyGradient



(a) TRPO Rewards (b) FPG Rewards (c) FPG Detection Ratio

Figure 6.3: Detailed Results on Humanoid-v3.

Attack mechanism: While designing and calculating the *optimal* attack strategy against a deep RL algorithm is still a challenging problem and active area of research [120, 188], here we describe the poisoning strategy used in our empirical evaluation, which, despite being simple, can fool non-robust RL algorithms with ease. Conceptually, policy gradient methods can be viewed as a stochastic gradient ascent method, where each iteration can be simplified as:

$$\theta^{(t+1)} = \theta^{(t)} + g^{(t)} \tag{6.6}$$

where $g^{(t)}$ is a gradient step that ideally points in the direction of fastest policy improvement. Assuming that $g^{(t)}$ is a good estimate of the gradient direction, then a simple attack strategy is to try to perturb $g^{(t)}$ to point in the $-g^{(t)}$ direction, in which case the policy, rather than improving, will deteriorate as learning proceed. A straightforward way to achieve this is to flip the rewards and multiply them by a big constant δ in the adversarial episodes. In the linear regression subproblem of Alg. 2, this would result in a set of (x, y) pairs whose y becomes $-\delta y$. This in expectation will make the best linear regressor w point to the opposite direction, which is precisely what we want.

This attack strategy is therefore parameterized by a single parameter δ, which guides the magnitude of the attack, and is **adaptively tuned** against each learning algorithm in the experiments: Throughout the experiment, we set the contamination level $\epsilon = 0.01$, and tune δ among the values of $[1, 2, 4, 8, 16, 32, 64]$ to find the most effective magnitude against each learning algorithm. All experiments are repeated with 3 random seeds and the mean and standard deviations are plotted in the figures.

Results: The experiment results are shown in Figure 7.1. Consistent patterns can be observed across all environments: vanilla `TRPO` performs well without attack but fails completely under the adaptive attack (which choose $\delta = 64$ in all environments). `FPG`, on the other hand, matches the performance of vanilla `TRPO` with or without attack. Figure 6.2 showcase two half-cheetah control policies learned by `TRPO` and `FPG` under attack with $\delta = 100$. Interestingly, due to the large negative adversarial rewards, TRPO actually learns the "running backward" policy, showing that our attack

strategy indeed achieves what it's designed for. In contrast, FPG is still able to learn the "running forward" policy despite the attack.

Figure 6.3 shows the detailed performances of TRPO and FPG across different δ's on the hardest *Humanoid* environment. One can observe that TRPO actually learns robustly under attacks of small magnitude ($\delta = 1, 2, 4$) and achieves similar performances to itself in clean environments, verifying our theoretical result in Theorem 6.3.1. In contrast, FPG remains robust across all values of δ's. Figure 6.3c shows the proportion of adversary data detected and removed by FPG's filtering subroutine throughout the learning process. One can observe that as the attack norm δ increases, the filtering algorithm also does a better job detecting the adversarial data and thus protect the algorithm from getting inaccurate gradient estimates. Similar patterns can be observed in all the other environments, and we defer the additional figures to the appendix.

6.7 Discussions

To summarize, in this chapter we present an $O(\epsilon)$ lower-bound for robust learning, an $O(\sqrt{\kappa\epsilon})$ upper-bound for algorithms with the help of an exploration policy with finite relative condition number κ, showing that online robust RL can be dimension-free. When no such helper policy is available, we show that one can still achieve $O(d^2\epsilon^{1/7})$, which, despite being suboptimal, is the first robust RL result that achieves non-vanishing bound under constant fraction adaptive contamination. In the next section, we shall look at the offline RL setting, in which robust learning becomes strictly harder due to the lack of online interaction.

7.1 Introduction

Offline Reinforcement Learning (RL) [102, 107] has received increasing attention recently due to its appealing property of avoiding online experimentation and making use of offline historical data. In applications such as assistive medical diagnosis and autonomous driving, historical data is abundant and keeps getting generated by high-performing policies (from human doctors/drivers). However, it is expensive to allow an online RL algorithm to take over and start experimenting with potentially suboptimal policies, as often human lives are at stake. Offline RL provides a powerful framework where the learner aims at finding the optimal policy based on historical data alone. Exciting advances have been made in designing stable and high-performing empirical offline RL algorithms [66, 103, 169, 98, 99, 6, 92, 154, 116, 175, 178]. On the theoretical front, recent works have proposed efficient algorithms with theoretical guarantees, based on the principle of *pessimism in face of uncertainty* [116, 31, 179, 83, 144], or variance reduction [176, 177]. Interesting readers are encouraged to check out these works and the references therein.

In this chapter, we investigate a different aspect of the offline RL framework that have not been explored previously, namely the statistical robustness in the presence of data corruption. To the best of our knowledge, *all* prior works on corruption-robust RL study the online RL setting.

In the batch learning setting, existing works mostly come from the robust statistics community and focuses on statistical estimation and lately supervised and unsupervised learning. We refer interesting readers to a comprehensive survey [52] of recent advances along these directions. In robust statistics, a prevailing problem setting is to perform a statistical estimation, e.g. mean estimation of an unknown distribution, assuming that a small fraction of the collected data is arbitrarily corrupted by an adversary. This is also referred to as the *Huber's contamination model* [75]. Motivated by these prior works, in this chapter we ask the following question:

Given an offline RL dataset with ϵ-fraction of corrupted data, what is the information-theoretic limit of robust identification of the optimal policy?

Towards answering this question, we summarize the following results in this chapter:

1. We provide the formal definition of ϵ-contamination model in offline RL, and establish an information-theoretical lower-bound of $\Omega(Hd\epsilon)$ in the setting of linear MDP with dimension d.

2. We design a robust variant of the Least-Square Value Iteration (LSVI) algorithm utilizing robust supervised learning oracles with a novel pessimism bonus term, and show that it achieves near-optimal performance in cases with (Theorem 7.3.2) or without data coverage (Theorem 7.3.3).

3. In the without coverage case, we establish a sufficient condition for learning based on the relative condition number with respect to any comparator policy — not necessary the optimal one. When specialized to offline RL without corruption, our partial coverage assumption is much weaker than the full coverage assumption in [83] for linear MDP.

4. In contrast to existing robust online RL results, we show that agnostic learning, i.e. learning without the knowledge of ϵ, is generally impossible in the offline RL setting, establishing a separation in hardness between online and offline RL in face of data corruption.

While our main focus are on corruption robust offline RL, it is worth noting when specialized to the classic offline RL setting, i.e., $\epsilon = 0$, our work also gives two interesting new results: (1) under linear MDP setting, we achieve an optimality gap with respect to any comparator policy (not necessarily the optimal one) in the order of $O(d^{3/2}/\sqrt{N})$ with N being the number of offline samples, by simply randomly splitting the dataset (this does sacrifices H dependence), (2) our analysis works for the setting where offline data only has partial coverage which is formalized using the concept of relative condition number with respect to the comparator policy.

7.2 Preliminaries

To begin with, let us formally introduce the episodic linear MDP setup we will be working with, the data collection and contamination protocol, as well as the robust linear regression oracle.

Environment. We consider an episodic finite-horizon Markov decision process (MDP), $\mathcal{M}(\mathcal{S}, \mathcal{A}, P, R, H, \mu_0)$, 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 function, such that $P(\cdot|s, a)$ gives the distribution over the next state if action a is taken from state s, $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is a stochastic and potentially unbounded reward function, H is the time horizon, and $\mu_0 \in \Delta_{\mathcal{S}}$ is an initial state distribution. The value functions $V_h^\pi : \mathcal{S} \to \mathbb{R}$ is the expected sum of future rewards, starting at time h in state s and executing π, i.e. $V_h^\pi(s) := \mathbb{E}\left[\sum_{t=h}^{H} R(s_t, a_t)|\pi, s_0 = s\right]$, where the expectation is taken with respect to the randomness of the policy and environment $\mathcal{M}$. Similarly, the *state-action* value function $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as $Q_h^\pi(s, a) := \mathbb{E}\left[\sum_{t=h}^{H} R(s_t, a_t)|\pi, s_0 = s, a_0 = a\right]$. We use π_h^*, Q_h^*, V_h^* to denote the optimal policy, Q-function and value function, respectively. For any function $f : \mathcal{S} \to \mathbb{R}$, we define the Bellman operator as

$$(\mathbb{B}f)(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[R(s, a) + f(s')]. \tag{7.1}$$

We then have the Bellman equation

$$V_h^\pi(s) = \langle Q_h^\pi(s, \cdot), \pi_h(\cdot|s)\rangle_{\mathcal{A}},\ Q_h^\pi(s, a) = (\mathbb{B}V_{h+1}^\pi)(s, a)$$

and the Bellman optimality equation

$$V_h^*(s) = \max_a Q_h^*(s,a), \quad Q_h^*(s,a) = (\mathbb{B}V_{h+1}^*)(s,a)$$

We define the averaged state-action distribution d^π of a policy π: $d^\pi(s,a) := \frac{1}{H}\sum_{h=1}^H \mathbb{P}^\pi(s_t = s, a_t = a | s_0 \sim \mu_0)$. We aim to learn a policy that maximizes the expected cumulative reward and thus define the performance metric as the suboptimality of the learned policy π compared to a *comparator policy* $\tilde{\pi}$:

$$\text{SubOpt}(\pi, \tilde{\pi}) = \mathbb{E}_{s\sim\mu_0}[V_1^{\tilde{\pi}}(s) - V_1^\pi(s)]. \tag{7.2}$$

Notice that $\tilde{\pi}$ doesn't necessarily have to be the optimal policy π^*, in contrast to most recent results in pessimistic offline RL, such as [83, 31]. For the majority of this chapter, we focus on the linear MDP setting [173, 81].

Assumption 7.2.1 (Linear MDP)**.** *There exists a known feature map $\phi : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^d$, d unknown signed measures $\mu = (\mu^{(1)}, ..., \mu^{(d)})$ over $\mathcal{S}$ and an unknown vector $\theta \in \mathbb{R}^d$, such that for all $(s,a,s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$,*

$$P(s'|s,a) = \phi(s,a)^\top \mu(s'),\ R(s,a) = \phi(s,a)^\top\theta + \omega$$

where ω is a zero-mean and σ^2-subgaussian distribution. Here we also assume that the parameters are bounded, i.e.$\|\phi(s,a)\| \leq 1$, $\mathbb{E}[R(s,a)] \in [0,1]$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ and $\max(\|\mu(\mathcal{S})\|, \|\theta\|) \leq \sqrt{d}$.

Note that compared to the last chapter, here our assumption on the MDP is stronger. We not only require linear value function but in fact, linear transition and reward functions, a strictly stronger condition.

Clean Data Collection. We consider the offline setting, where a clean dataset $\tilde{D} = \{(\tilde{s}_i, \tilde{a}_i, \tilde{r}_i, \tilde{s}_i')\}_{i=1:N}$ of transitions is collected a priori by an unknown experimenter. We assume the stochasticity of the clean data collecting process, i.e. there exists an offline state-action distribution $\nu \in \Delta(\mathcal{S}\times\mathcal{A})$, s.t. $(\tilde{s}_i, \tilde{a}_i) \sim \nu(s,a)$, $\tilde{r}_i \sim R(\tilde{s}_i, \tilde{a}_i)$ and $\tilde{s}_i' \sim P(\tilde{s}_i, \tilde{a}_i)$. When there is no corruption, $\tilde{D}$ will be observed by the learner. However, we study the setting where the data is contaminated by an adversary before revealed to the learner.

Contamination model. We define an adversarial model that can be viewed as a direct extension to the *ϵ-contamination model* studied in the traditional robust statistics literature.

Assumption 7.2.2 (ϵ-Contamination in offline RL)**.** *Given $\epsilon \in [0,1]$ and a set of clean tuples $\tilde{D} = \{(\tilde{s}_i, \tilde{a}_i, \tilde{r}_i, \tilde{s}_i')\}_{i=1:N}$, the adversary is allowed to inspect the tuples and replace any ϵN of them with arbitrary transition tuples $(s,a,r,s') \in \mathcal{S}\times\mathcal{A}\times\mathbb{R}\times\mathcal{S}$. The resulting set of transitions is then revealed to the learner. We*

will call such a set of samples ϵ-corrupted, and denote the contaminated dataset as $D = \{(s_i, a_i, r_i, s_i')\}_{i=1:N}$. In other words, there are at most ϵN number of indices i, on which $(\tilde{s}_i, \tilde{a}_i, \tilde{r}_i, \tilde{s}_i') \neq (s_i, a_i, r_i, s_i')$.

Under ϵ-contamination, we assume access to a robust linear regression oracle.

Assumption 7.2.3 (Robust least-square oracle (RLS))**.** *Given a set of ϵ-contaminated samples $S = \{(x_i, y_i)\}_{1:N}$, where the clean data is generated as: $\tilde{x}_i \sim \nu$, $P(\|x\| \leq 1) = 1$, $\tilde{y}_i = \tilde{x}_i \top w^* + \gamma_i$, where γ_i's are subgaussian noise with zero-mean and γ^2-variance. Then, a robust least-square oracle returns an estimator $\hat{w}$, such that*

leftmirgin= If $\mathbb{E}_\nu[xx^\top] \succeq \xi$, then with probability at least $1 - \delta$,*

$$\|\hat{w} - w^*\|_2 \leq c_1(\delta) \cdot \left(\sqrt{\frac{\gamma^2 poly(d)}{\xi^2 N}} + \frac{\gamma}{\xi} \epsilon \right)$$

leftmiirgiin= With probability at least $1 - \delta$,*

$$\mathbb{E}_\nu \left(\|x^\top (\hat{w} - w^*)\|_2^2 \right) \leq c_2(\delta) \cdot \left(\frac{\gamma^2 poly(d)}{N} + \gamma^2 \epsilon \right)$$

where c_1 and c_2 hide absolute constants and polylog$(1/\delta)$.

Such guarantees are common in the robust statistics literature, see e.g. [17, 140, 94]. While we focus on oracles with such guarantees, our algorithm and analysis are modular and allow one to easily plug in oracles with stronger or weaker guarantees.

7.3 Algorithms and Main Results

Here, we focus on a Robust variant of Least-Squares Value Iteration (LSVI)-style algorithms [83], which directly calls a robust least-square oracle to estimate the Bellman operator $\hat{\mathbb{B}}\hat{V}_h(s, a)$. Optionally, it may also subtract a pessimistic bonus $\Gamma_h(s, a)$ during the Bellman update. A template of such an algorithm is defined in Algorithm 5. In section 7.3 and 7.3, we present two variants of the LSVI algorithm designed for two different settings, depending on whether the data has full coverage over the whole state-action space or not. However, before that, we first present an algorithm-independent minimax lower-bound that illustrates the hardness of the robust learning problem in offline RL, in contrast to classic results in statistical estimation and supervised learning.

Algorithm 5 Robust Least-Square Value Iteration (R-LSVI)

1: Input: Dataset $D = \{(s_i, a_i, r_i, s_i')\}_{1:N}$; pessimism bonus $\Gamma_h(s,a) \geq 0$, robust least-squares Oracle: $RLS(\cdot)$.
2: Split the dataset randomly into H subset: $D_h = \{(s_i^h, a_i^h, r_i^h, s_i'^h)\}_{1:(N/H)}$, for $h \in [H]$.
3: Initialization: Set $\hat{V}_{H+1}(s) \leftarrow 0$.
4: **for** step $h = H, H-1, \ldots, 1$ **do**
5: Set $\hat{w}_h \leftarrow RLS\left(\left\{(\phi(s_i^h, a_i^h), (r_i^h + \hat{V}_{h+1}(s_i'^h)))\right\}_{1:(N/H)}\right)$.
6: Set $\hat{Q}_h(s,a) \leftarrow \phi(s,a)^\top \hat{w}_h - \Gamma_h(s,a)$, clipped within $[0, H-h+1]$.
7: Set $\hat{\pi}_h(a|s) \leftarrow \arg\max_a \hat{Q}_h(s,a)$ and $\hat{V}_h(s) \leftarrow \max_a \hat{Q}_h(s,a)$.
8: Output: $\{\hat{\pi}_h\}_{h=1}^H$.

Minimax Lower-bound

Theorem 7.3.1 (Minimax Lower bound)**.** *Under assumptions 7.2.1 (linear MDP) and 7.2.2 (ϵ-contamination), for any fixed data-collecting distribution ν, no algorithm $L : (\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{A})^N \to \Pi$ can find a better than $O(dH\epsilon)$-optimal policy with probability more than $1/4$ on all MDPs. Specifically,*

$$\min_{L,\nu} \max_{\mathcal{M}, f_c} \textit{SubOpt}(\hat{\pi}, \pi^*) = \Omega\left(dH\epsilon\right) \tag{7.3}$$

where f_c denotes an ϵ-contamination strategy that corrupts the data based on the MDP $\mathcal{M}$ and clean data $\tilde{D}$ and returns a contaminated dataset, and L denotes an algorithm that takes the contaminated dataset and return a policy $\hat{\pi}$, i.e. $\hat{\pi} = L(f_c(\mathcal{M}, \tilde{D}))$.

The detailed proof is presented in appendix F.2, but the high-level idea is simple. Consider the tabular MDP setting which is a special case of linear MDP with $d = SA$. For any data generating distribution ν, by the pigeonhole principle, there must exists a least-sampled (s,a) pair, for which $\nu(s,a) \leq 1/SA$. If the adversary *concentrate* all its attack budget on this least sampled (s,a) pair, it can perturb the empirical reward on this (s,a) pair to be as much as $\hat{r}(s,a) = r(s,a) + SA\epsilon$. Further more, assume that there exists another (s^*, a^*) such that $r(s^*, a^*) = r(s,a) + SA\epsilon/2$. Then, the learner has no way to tell if truly $r(s,a) > r(s^*, a^*)$ (i.e., the learner believes what she observes and believes there is no contamination) or if the data is contaminated and in fact $r(s,a) < r(s^*, a^*)$. Either could be true and whichever alternative the learner chooses to believe, it will suffer at least $SAH\epsilon/2$ optimality gap in one of the two scenarios.

Remark 7.3.1 (dimension scaling)**.** Theorem 7.3.1 says that even if the algorithm has control over the data collecting distribution ν (without knowing $\mathcal{M}$ a priori), it can still do no better than $\Omega(dH\epsilon)$ in the worst-case, which implies that robustness is fundamentally impossible in high-dimensional problems where $d \geq 1/\epsilon$. This is in sharp contrast to the classic results in the robust

statistics literature, where estimation errors are found to not scale with the problem dimension, in settings such as robust mean estimation [49, 101] and robust supervised learning [37, 50]. From the construction we can see that the dimension scaling appears fundamentally due to a *multi-task learning* effect: the learner must perform SA separate reward mean estimation problems for each (s, a) pair, while the data is provided as a mixture for all these tasks. As a result, the adversary can concentrate on one particular task, raising the contamination level to effectively $d\epsilon$.

Remark 7.3.2 (Offline vs. Online RL)**.** We note that the construction in Theorem 7.3.1 remains valid even if the adversary only contaminates the rewards, and if the adversary is oblivious and perform the contamination based only on the data generating distribution ν rather than the instantiated dataset $\tilde{D}$. In contrast, in the last chapter, our lower-bound for robust online RL is $\Omega(H\epsilon)$. It remains unknown whether $\Omega(H\epsilon)$ is tight, as no algorithm yet can achieve a matching upper-bound without additional information. We will come back to this discussion in section 7.3.

In what follows, we show that the above lower-bound is tight in both d and ϵ, by presenting two upper-bound results nearly matching the lower-bound.

Robust Learning with Data Coverage

To begin with, we study the simple setting where the offline data has sufficient coverage over the whole state-action distribution. This is often considered as a strong assumption. However, results in this setting will establish meaningful comparison to the above lower-bound and the no-coverage results later. In the context of linear MDP, we say that a data generating distribution has coverage if it satisfies the following assumption.

Assumption 7.3.1 (Uniform Coverage)**.** *Under assumption 7.2.1, define $\Sigma_\nu := \mathbb{E}_\nu[\phi(s,a)\phi(s,a)^\top]$ as the covariance matrix of ν. We say that the data generating distribution ξ-covers the state-action space for $\xi > 0$, if*

$$\Sigma_\nu \succeq \xi \tag{7.4}$$

i.e. the smallest eigenvalue of Σ_ν is strictly positive and at least ξ.

Under such an assumption, we show that the R-LSVI without pessimism bonus can already be robust to data contamination.

Theorem 7.3.2 (Robust Learning under ξ-Coverage)**.** *Under assumption 7.2.1, 7.2.2 and 7.3.1, for any $\xi, \epsilon > 0$, given a dataset of size N, Algorithm 5 with bonus $\Gamma_h(s,a) = 0$ achieves*

$$\mathit{SubOpt}(\hat{\pi}, \pi^*) \leq c_1(\delta/H) \cdot \left(\sqrt{\frac{(\sigma+H)^2 H^3 \mathit{poly}(d)}{\xi^2 N}} + \frac{(\sigma+H)H^2}{\xi}\epsilon \right) \tag{7.5}$$

with probability at least $1-\delta$.

The proof of Theorem 7.3.2 follows readily from the standard analysis of approximated value iterations and rely on the following classic result connecting the Bellman error to the suboptimality of the learned policy, see e.g. Section 2.3 of [76].

Lemma 7.3.1 (Optimality gap of VI)**.** *Under assumption 7.2.1, Algorithm 5 with* $\Gamma_h(s,a) = 0$ *satisfies*

$$\textit{SubOpt}(\hat{\pi}, \pi^*) \leq 2H \max_{s,a,h} |\hat{Q}_h(s,a) - (\mathbb{B}_h \hat{V}_{h+1})(s,a)| \leq 2H \max_{s,a,h} \|\phi(s,a)\|_2 \cdot \|\hat{w}_h - w_h^*\|_2 \tag{7.6}$$

where $w_h^* := \theta + \int_{\mathcal{S}} \hat{V}_{h+1}(s')\mu_h(s')ds'$ *is the best linear predictor.*

The result then follows immediately using property 1 of the robust least-square oracle and the fact that $\mathbb{E}[((r(s,a) + \hat{V}(s')) - (\mathbb{B}_h \hat{V})(s,a))^2 | s,a] \leq (\sigma + H)^2$ (Lemma F.1.2).

Remark 7.3.3 (Data Splitting and tighter d-dependency)**.** The data splitting in step 2 of Algorithm 5 is mainly for the sake of theoretical analysis and is not required for practical implementations. Nevertheless, it directly contributes to our tighter bounds. Specifically, the data splitting makes $\hat{V}_{h+1}$, which is learned based on D_{h+1}, independent from D_h, at the cost of an additional H multiplicative factor. In contrast, the typical covering argument used in online RL will introduce another $O(d^{1/2})$ multiplicative factor, and naively applying it to the offline RL setting will make the finally sample complexity scales as $O(d^{3/2})$, see e.g. Corollary 4.5 of [83]. Our result above, when specialized to offline RL without corruption (i.e., $\epsilon = 0$), achieves the following results.

Corollary 7.3.1 (Uncorrupted Learning under ξ-Coverage)**.** *Under assumption 7.2.1 and 7.3.1, for any* $\xi > 0$*, given a clean dataset of size* N*, with bonus* $\Gamma_h(s,a) = 0$ *and ridge regression with regularizer coefficient* $\lambda = 1$ *as the RLS solver, Algorithm 5 achieves with probability at least* $1 - \delta$

$$\textit{SubOpt}(\hat{\pi}, \pi^*) \leq \tilde{O}\left(\frac{H^3 d}{\xi\sqrt{N}}\right). \tag{7.7}$$

Remark 7.3.4 (Tolerable ϵ)**.** Notice that Theorem 7.3.2 requires $\epsilon \leq \xi$ to provide a non-vacuous bound. This is because if $\epsilon > \xi$, then similar to the lower-bound construction in Theorem 7.3.1, the adversary can corrupt all the data along the eigenvector direction corresponding to the smallest eigenvalue, in which case the empically estimated reward along that direction can be arbitrarily far away from the true reward even with a robust mean estimator, and thus the estimation error becomes vacuous.

Remark 7.3.5 (Unimprovable gap)**.** Notice in contrast to classic RL results, Theorem 7.3.2 implies that in the presence of data contamination, there exists an unimprovable optimality gap $(\sigma+H)H^2\epsilon/\xi$ even if the learner has access to infinite data. Also note that because $\|\phi(s,a)\| \leq 1$, ξ is at most $1/d$. This implies that asymptotically, $V^* - V^{\hat{\pi}} \leq O(H^3 d\epsilon)$ when ξ is on the order of $1/d$, matching the lower-bound upto H factors.

Remark 7.3.6 (Agnosticity to problem parameters)**.** It is worth noting that in theorem 7.3.2, the algorithm does not require the knowledge of ϵ or ξ, and thus works in the agnostic setting where these parameters are not available to the learner (given that the robust least-square oracle is agnostic). In other words, the algorithm and the bound are adaptive to both ϵ and ξ. This point will be revisited in the next section.

Robust Learning without Coverage

Next, we consider the harder setting where assumption 7.3.1 does not hold, as often in practice, the offline data will not cover the whole state-action space. Instead, we provide a much weaker sufficient condition under which offline RL is possible.

Assumption 7.3.2 (relative condition number)**.** *For any given comparator policy $\tilde{\pi}$, under assumption 7.2.1 and 7.2.2, define the relative condition number as*

$$\kappa = \sup_{w} \frac{w^\top \tilde{\Sigma} w}{w^\top \Sigma_\nu w} \tag{7.8}$$

where $\tilde{\Sigma}$ denotes $\Sigma_{d^{\tilde{\pi}}}$ and we take the convention that $\frac{0}{0} = 0$. We assume that $\kappa < \infty$.

The relative condition number is recently introduced in the policy gradient literature [4, 187]. Intuitively, the relative condition number measures the worst-case density ratio between the occupancy distribution of comparator policy and the data generating distribution. For example, in a tabular MDP, $\kappa = \max_{s,a} \frac{d^{\tilde{\pi}}(s,a)}{\nu(s,a)}$. Here, we show that a finite relative condition number with respect to an *arbitrary* comparator policy is already sufficient for offline RL, for both clean and contaminated setting.

Without data coverage, we now rely on pessimism to retain reasonable behavior. However, the challenge, in this case, is to design a valid confidence bonus using only the corrupted data. We now present our constructed pessimism bonus that allows Algorithm 5 to handle ϵ-corruption, albeit requiring the knowledge of ϵ.

Theorem 7.3.3 (Robust Learning without Coverage)**.** *Under assumption 7.2.1, 7.2.2 and 7.3.2, with $\epsilon > 0$, given any comparator policy $\tilde{\pi}$ with $\kappa < \infty$, define the ϵ-robust empirical covariance as*

$$\Lambda_h = \frac{3}{5}\left(\frac{H}{N}\sum_{i=1}^{N/H} \phi(s_i^h, a_i^h)\phi(s_i^h, a_i^h)^\top + (\epsilon + \lambda)\cdot I\right), \quad \lambda = c' \cdot dH\log(N/\delta)/N$$

where c' is an absolute constant. Then, Algorithm 5 with pessimism bonus

$$\Gamma_h(s,a) = \left(\frac{(\sigma + H)\sqrt{H}\mathit{poly}(d)}{\sqrt{N}} + ((\sigma + H) + 2H\sqrt{d})\sqrt{\epsilon} + \sqrt{d\lambda}\right)\sqrt{c_2(\delta/H)}\|\phi(s,a)\|_{\Lambda_h^{-1}} \tag{7.9}$$

will with probability at least $1-\delta$ *achieve*

$$\textit{SubOpt}(\hat{\pi},\tilde{\pi}) \leq \tilde{O}\left(\frac{(\sigma+H)\sqrt{H^3\kappa}\textit{poly}(d)}{\sqrt{N}} + ((\sigma+H)H + H^2\sqrt{d})\sqrt{d\kappa\epsilon}\right) \tag{7.10}$$

Remark 7.3.7 (Arbitrary comparator policy)**.** Notice that in comparison to Theorem 4.2 of [83], Lemma 7.3.2 allows the comparator policy to be arbitrary, and the implication is profound. Specifically, Lemma 7.3.2 indicates that a pessimism-style algorithm *always* retains reasonable behavior, in the sense that, given enough data, it will eventually find the best policy among all the policies covered by the data generating distribution, i.e. $\arg\max_\pi V^\pi(\mu)$, s.t. $\kappa(\pi) < \infty$. Similar to the ξ-coverage, when specialized to standard offline RL, our analysis provides a tighter bound.

Corollary 7.3.2 (Uncorrupted Learning without Coverage)**.** *Under assumption 7.2.1 and 7.3.2, given any comparator policy* $\tilde{\pi}$ *with* $\kappa < \infty$*, define the empirical covariance as*

$$\Lambda_h = \frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i^h,a_i^h)\phi(s_i^h,a_i^h)^\top + \lambda\cdot I, \quad \lambda = c'\cdot dH\log(N/\delta)/N$$

where c' *is an absolute constant. Then, with pessimism bonus*

$$\Gamma_h(s,a) = H\left(\sqrt{d\cdot\lambda} + \sqrt{\frac{Hd\log(N/\delta\lambda)}{N}}\right)\cdot\|\phi(s,a)\|_{\Lambda_h^{-1}} \tag{7.11}$$

and ridge regression with regularizer coefficient λ *as the RLS solver, Algorithm 5 will with probability at least* $1-\delta$ *achieve*

$$\textit{SubOpt}(\hat{\pi},\tilde{\pi}) \leq \tilde{O}\left(\left(H^2d + H^{2.5}\sqrt{d}\right)\sqrt{\frac{d\kappa}{N}}\right) \tag{7.12}$$

We note that the leading term (first term) $O(d^{3/2})$ is directly due to the assumption that the linear MDP parameter $\max(\|\mu(\mathcal{S})\|,\|\theta\|) \leq \sqrt{d}$. If instead $\max(\|\mu(\mathcal{S})\|,\|\theta\|) \leq \rho$ for some ρ indepdent of d, then the above bound will become linear in d. In contrast, the covering-number style analysis will generate $d^{3/2}$ regardless of the parameter norm, since its second term will become $O(d^{3/2})$ and dominate (as one needs to perform a covering argument to cover the quadratic penalty term $\Gamma_h(s,a)$).

The proof of Theorem 7.3.3 is technical but largely follows the analysis framework of pessimism-based offline RL and consists of two main steps. The first step establishes $\Gamma_h(s,a)$ as a valid bonus by showing

$$|\hat{Q}_h(s,a) - (\mathbb{B}_h\hat{V}_{h+1})(s,a)| \leq \Gamma_h(s,a), \text{ w.p. } 1-\delta/H. \tag{7.13}$$

The second step applies the following Lemma connectingthe optimality gap with the expectation of $\Gamma_h(s,a)$ under visitation distribution of the comparator policy.

Lemma 7.3.2 (Suboptimality for Pessimistic Value Iteration)**.** *Under assumption 7.2.1, if Algorithm 5 has a proper pessimism bonus, i.e.* $|\hat{Q}_h(s,a) - (\mathbb{B}_h\hat{V}_{h+1})(s,a)| \leq \Gamma_h(s,a), \forall h \in [H]$*, then against any comparator policy* $\tilde{\pi}$*, it achieves*

$$\mathit{SubOpt}(\hat{\pi}, \tilde{\pi}) \leq 2\sum_{h=1}^{H} \mathbb{E}_{d^{\tilde{\pi}}}[\Gamma_h(s,a)] \tag{7.14}$$

We then further upper-bound the expectation through the following inequality, which bounds the distribution shift effect using the relative condition number κ:

$$\mathbb{E}_{d^{\tilde{\pi}}}\left[\sqrt{\phi(s,a)^\top \Lambda^{-1}\phi(s,a)}\right] \leq \sqrt{5d\kappa} \tag{7.15}$$

The detailed proof can be found in Appendix F.3. Note that the prior work [83] only establishes results in terms of the suboptimality comparing with the optimal policy, and when specializes to linear MDPs, they assume the offline data has global full coverage. We replace these redundant assumptions with a single assumption of partial coverage with respect to any comparator policy, in the form of a finite relative condition number.

Remark 7.3.8 (Novel bonus term)**.** One of our main algorithmic contributions is the new bonus term that upper-bound the effect of data contamination on the Bellman error. Ignoring ϵ-independent additive terms and absolute constants, our bonus term has the form

$$H\sqrt{\epsilon} \cdot \sqrt{\phi(s,a)^\top \Lambda^{-1}\phi(s,a)}. \tag{7.16}$$

In comparison, below is the one used in [118] for online corruption-robust RL:

$$H\epsilon \cdot \sqrt{\phi(s,a)^\top \Lambda^{-2}\phi(s,a)}. \tag{7.17}$$

In the tabular case, (7.17) evaluates to $H\epsilon/\nu(s,a)$ and (7.16) evaluates to $H\sqrt{\epsilon/\nu(s,a)}$, and thus (7.17) is actually tighter than (7.16) for $\nu(s,a) \geq \epsilon$. However, in the linear MDP case, the relation between the two is less obvious. As we shall see, when offline distribution has good coverage, i.e. Λ is well-conditioned, (7.17) appears to be tighter. However, as the smallest eigenvalue of Λ goes to zero, a.k.a. lack of coverage, (7.17) actually blows up rapidly, whereas both (7.16) and the actual achievable gap remain bounded.

We demonstrate these behaviors with a numerical simulation, shown in Figure 7.1. In the

simulation, we compare the size of three terms

$$\text{maximum possible gap} = \max_{\|y\|_\infty \le 2H, \|y\|_0 \le \epsilon N} \phi(s,a)^\top \Lambda^{-1} \left(\frac{1}{N} \sum_{i=1}^{N} \phi(s_i, a_i) \cdot y_i \right) \tag{7.18}$$

$$\text{bonus 1} = H\epsilon \cdot \sqrt{\phi(s,a)^\top \Lambda^{-2} \phi(s,a)}$$

$$\text{bonus 2} = H\sqrt{\epsilon} \cdot \sqrt{\phi(s,a)^\top \Lambda^{-1} \phi(s,a)}$$

The maximum possible gap is defined as above since for any (s,a) pair and in any step h, the bias introduced to its Bellman update due to corruption takes the form of

$$\phi(s,a)^\top \Lambda^{-1} \left(\frac{1}{N} \sum_{i=1}^{N} \phi(s_i, a_i) \cdot \left((\tilde{r}_i + \hat{V}_{h+1}(\tilde{s}'_i)) - (r_i + \hat{V}_{h+1}(s'_i)) \right) \right) \tag{7.19}$$

where $\tilde{r}_i$ and $\tilde{x}'_i$ are the clean reward and transitions. For the sake of clarity, here we assume that the adversary only contaminate the reward and transitions in a bounded fashion while keeping the current (s,a)-pairs unchanged. (7.19) can then be upper-bounded by (7.18), because there are at most ϵN tuples on which $\tilde{r}_i \neq r_i$ or $\tilde{s}'_i \neq s'_i$, and for any such tuple $(\tilde{r}_i + \hat{V}_{h+1}(\tilde{s}'_i)) - (r_i + \hat{V}_{h+1}(s'_i)) \le 2H$.

In the simulation, we set $H = 1$ to ignore the scaling on time horizon and let $\lambda = 1$; We let both the test data $\phi(s,a)$ and the training data $\phi(s_i, a_i)$ to be sampled from a truncated standard Gaussian distribution in $\mathbb{R}^3$, denoted by ν, with mean 0, and covariance eigenvalues $1, 1, \lambda_{\min}$. We set the training data size set to $N = 10^6$ and contamination level set to $\epsilon = 0.01$. The x-axis tracks $-\log(\lambda_{\min})$, while the y-axis tracks $\mathbb{E}_{s,a\sim\nu}\text{bonus}(s,a)$, with expectation being approximated by 1000 test samples from ν. It can be seen that bonus 1 starts off closely upper-bounding the maximum possible gap when the data has good coverage, but increases rapidly as $\lambda_{\min}$ decreases. Note that for a fixed N, bonus 1 will eventually plateau at $HN\epsilon/\lambda$, but this term scales with N, so the error blows up as the number of samples grows, which certainly is not desirable. Bonus 2, on the other hand, is not as tight as bonus 1 when there is good data coverage, but remains intact regardless of the value of $\lambda_{\min}$, which is essential for the more challenging setting with poor data coverage.

This new bonus term can be of independent interest in other robust RL contexts. For example, in the online corruption-robust RL problem, as a result of using the looser bonus term (7.16), the algorithm in [118] can only handle $\epsilon = T^{-3/4}$ amount of corruptions in the linear MDP setting, while being able to handle $\epsilon = T^{-1/2}$ amount of corruptions in the tabular setting, due to the tabular bonus being tighter. Our bonus term can be directly plugged into their algorithm, allowing it to handle up to $\epsilon = T^{-1/2}$ amount of corruption even in the linear MDP setting, achieving an immediate improvement over previous results.[1]

[1] Though our bound improve their result, the tolerable corruption amount is still sublinear, which is due to the multi-layer scheduling procedure used in their algorithm.

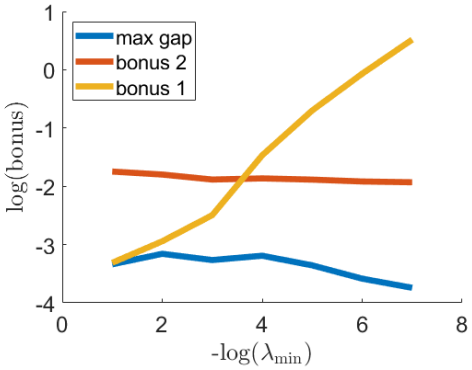


Figure 7.1: bonus size simulation

Note that our algorithm and theorem are adaptive to the unknown relative coverage κ, but is not adaptive to the level of contamination ε (i.e., algorithm requires knowing ε or a tight upper bound of ε). One may ask whether there exists an agnostic result, similar to Theorem 7.3.2, where an algorithm can be adaptive simultaneously to unknown values of ϵ and coverage parameter κ. Our last result shows that this is unfortunately not possible without full data coverage. In particular, we show that no algorithm can achieve a best-of-both-world guarantee in both clean and ϵ-corrupted environments. More specifically, in this setting, κ is still unknown to the learner, and the adversary either corrupt ε amount of tuples (ϵ is known) or does not corrupt at all—but the learner does not know which situation it is.

Theorem 7.3.4 (Agnostic learning is impossible without full coverage)**.** *Under assumption 7.2.1 and 7.3.2, for any algorithm* $L : (\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{A})^N \to \Pi$ *that achieves diminishing suboptimality in clean environment, i.e., for any clean dataset* $\tilde{\mathcal{D}}$ *it achieves* $\mathit{SubOpt}(L(\tilde{\mathcal{D}})) = g(N)$ *for some positive function* g *such that* $\lim_{N\to\infty} g(N) = 0$*, we have that for any* $\epsilon \in (0, 1/2]$*, there exists an MDP* $\mathcal{M}^\dagger$ *such that with probability at least* $1/4$*,*

$$\max_{f_c} \mathit{SubOpt}(\hat{\pi}, \tilde{\pi}) \geq 1/2 \tag{7.20}$$

Intuitively, the logic behind this result is that in order to achieve vanishing errors in the clean environment, the learner has no choice but to *trust* all data as clean. However, it is also possible that the same dataset could be generated under some adversarial corruption from another MDP with a very different optimal policy—thus the learner cannot be robust to corruption under that MDP.

Specifically, consider a 2-arm bandit problem. The learner observes a dataset of N data points of arm-reward pairs, of which p fraction is arm a_1 and $(1-p)$ fraction is arm a_2. For simplicity, we assume that N large enough such that the empirical distribution converges to the underlying sampling distribution. Assume further that the average reward observed for a_1 is $\hat{r}_1 = \frac{1}{2} + \frac{\epsilon}{2p}$, for some $\epsilon \leq p$, and the average reward observed for a_2 is $\frac{1}{2}$. Given such a dataset, two data generating

processes can generate such a dataset with equal likelihood and thus indistinguishable based only on the data:

1. There is no contamination. The MDP has reward setting where a_1 indeed has reward $r_1 =$ Bernoulli($\frac{1}{2} + \frac{\epsilon}{2p}$) and a_2 has $r_2 =$ Bernoulli($\frac{1}{2}$). Since there is no corruption, $\kappa = 1/p$ in this MDP.

2. The data is ϵ-corrupted. In particular, in this MDP, the actual reward of a_1 is $r_1 =$ Bernoulli($\frac{1}{2} - \frac{\epsilon}{2p}$), and the adversary is able to increase empirical mean by ϵ/p via changing ϵN number of data points from $(a_1, 0)$ to $(a_1, 1)$. One can show that this can be achieved by the adversary with probability at least $1/2$ (which is where the probability $1/2$ in the theorem statement comes from). In this MDP, we have $\kappa = 1/(1-p)$.

Now, since the algorithm achieves a diminishing suboptimal gap in all clean environments, it must return a_1 with high probability given such a dataset, due to the possibility of the learner facing the data generation process 1. However, committing to action a_1 will incur $\epsilon/2p$ suboptimal gap in the second MDP with the data generation process 2. On the other hand, note that the relative condition number in the second MDP is bounded, i.e. $\frac{1}{1-p} \leq 2$ for $\epsilon \leq p \leq 1/2$. Therefore, for any $\epsilon \in (0, 1/2]$, one can construct such an instance with $p = \epsilon$, such that the relative condition number for the second MDP is $\frac{1}{1-p} \leq 2$ and the relative condition number for the first MDP is $\frac{1}{\epsilon} < \infty$, while the learner would always suffer $\epsilon/2p = 1/2$ suboptimality gap in the second MDP if she had to commit to a_1 under the first MDP where data is clean.

Remark 7.3.9 (Offline vs. Online RL: Agnostic Learning)**.** Theorem 7.3.4 shows that no algorithm can simultaneously achieve good performance in both clean and corrupted environments without knowing which one it is currently experiencing. This is in sharp contrast to the recent result in [187], which shows that in the online RL setting, natural policy gradient (NPG) algorithm can find an $O(\sqrt{\kappa\epsilon})$-optimal policy for any unknown contamination level ϵ with the help of an exploration policy with finite relative condition number. Without such a helper policy, however, robust RL is much harder, and the best-known result [118] can only handle $\epsilon \leq O(1/\sqrt{T})$ corruption, but still does not require the knowledge of ϵ. Intuitively, such adaptivity is lost in the offline setting, because the learner is no longer able to evaluate the current policy by collecting on-policy data. In the online setting, the construction in Theorem 7.3.4 will not work. Our construction heavily relies on the fact that ν has ε probability of sampling a_1, which allows adversary in the second MDP to concentrate its corruption budget all on a_1. In the online setting, one can simply uniform randomly try a_1 and a_2 to significantly increase the probability of sampling a_1 which in turn makes the estimation of r_1 accurate (up to $O(\varepsilon)$ in the corrupted data generation process).

7.4 Discussions

In this chapter, we studied corruption-robust RL in the offline setting. We provided an information-theoretical lower-bound and two near-matching upper-bounds for cases with or without full data coverage, respectively. When specialized to the uncorrupted setting, our algorithm and analysis also obtained tighter error bounds while under weaker assumptions. Many future works remain:

1. Our upper-bounds do not yet match with the lower-bound in terms of their dependency on H, and the $\sqrt{\epsilon}$ rather than ϵ dependency on ϵ in the partial coverage setting. Tightening the dependency on ϵ likely requires designing a new and tighter bonus term, as our simulation shows that neither of the currently known bonus is tight.

2. Unlike the online counter-part, in the offline setting we do not yet have an empirically robust algorithm that incorporates more flexible function approximators, such as neural networks.

3. While we show that dimension-scaling is unavoidable (Theorem 7.3.1) in the worst case in the Optimal Policy Identification (OPI) task, it remains unknown whether it can be achieved in the less challenging tasks, such as offline policy evaluation (OPE) and imitation learning (IL).

Proof of modified Proposition 2.2.1.

In this version, we assume $(\mathbf{w}, \mathbf{x}, y)$ is a trajectory of (2.1) rather than being a trajectory of (2.8).

All we need to show is that for any pair of $(\mathbf{x}, y)$, there exist another pair $(\tilde{\mathbf{x}}, R_y)$, such that they give the same update. In particular, we set $\tilde{\mathbf{x}} = a\mathbf{x}$ and show that there always exists an $a \in [-1, 1]$ such that

$$(y - \mathbf{w}^\mathsf{T}\mathbf{x})\mathbf{x} = (R_y - \mathbf{w}^\mathsf{T} a\mathbf{x})a\mathbf{x}.$$

This simplifies to

$$g(a) := (\mathbf{w}^\mathsf{T}\mathbf{x})a^2 - R_y a + (y - \mathbf{w}^\mathsf{T}\mathbf{x}) = 0. \tag{A.1}$$

The discriminant of the quadratic (A.1) is

$$\begin{aligned} R_y^2 - 4\mathbf{w}^\mathsf{T}\mathbf{x}(y - \mathbf{w}^\mathsf{T}\mathbf{x}) &\geq R_y^2 - 4|\mathbf{w}^\mathsf{T}\mathbf{x}|\big(R_y + |\mathbf{w}^\mathsf{T}\mathbf{x}|\big) \\ &= \big(R_y - 2|\mathbf{w}^\mathsf{T}\mathbf{x}|\big)^2 \geq 0 \end{aligned}$$

So there always exists a solution $a \in \mathbb{R}$. Moreover, $g(-1) = R_y + y \geq 0$ and $g(1) = -R_y + y \leq 0$, so there must be a real root in $[-1, 1]$. ∎

Proof of Theorem 2.2.2.

We showed in Section 2.2 that Regime V trajectories are 2D. We also argued that solutions that reach $\mathbf{w}_\star$ via Regime III–IV are not unique and need not be 2D. We will now show that it's always possible to construct a 2D solution.

We begin by characterizing the set of $\mathbf{w}_\star$ reachable via Regime III–IV. Recall from Section 2.2 that the transition between III and IV occurs when $\|\mathbf{w}\| = R := \dfrac{R_y}{2R_x}$. If t_0 is the time at which this transition occurs, then for $0 \leq t \leq t_0$, the solution is $\mathbf{x} = \frac{R_x}{\|\mathbf{w}\|}\mathbf{w}$, which leads to a straight-line trajectory from $\mathbf{w}_0$ to $\mathbf{w}(t_0)$.

Now consider the part of the trajectory in Regime IV, where $t_0 \leq t \leq t_f$. As derived in Section 2.2, Regime IV trajectories satisfy $\dot{\mathbf{w}} = \mathbf{w}^\mathsf{T}\mathbf{x} = \dfrac{R_y}{2}$. These lead to $\frac{\mathrm{d}\|\mathbf{w}\|^2}{\mathrm{d}t} = \frac{R_y^2}{2}$, which means that $\|\mathbf{w}\|$ grows at the same rate regardless of $\mathbf{x}$. If our trajectory reaches $\mathbf{w}(t_f) = \mathbf{w}_\star$, then we can deduce via integration that

$$\|\mathbf{w}_\star\|^2 - \|\mathbf{w}(t_0)\|^2 = \frac{R_y^2}{2}(t_f - t_0), \tag{A.2}$$

Suppose $(\mathbf{w}(t), \mathbf{x}(t))$ for $t_0 \leq t \leq t_f$ is a trajectory that reaches $\mathbf{w}_\star$. Refer to Figure A.1. The reachable set at time t_f is a spherical sector whose boundary requires a trajectory that maximizes curvature.

We will now derive this fact.

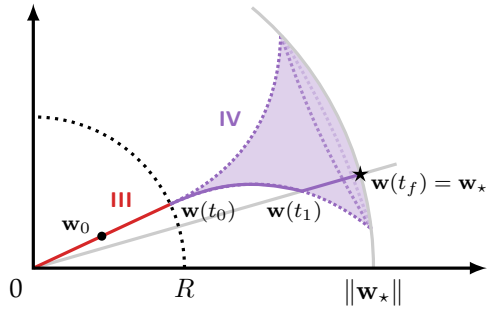


Figure A.1: If a reachable $\mathbf{w}_\star$ is contained in the concave funnel shape, which is the reachable set in Regime IV, it can be reached by some trajectory $(\mathbf{w}(t), \mathbf{x}(t))$ lying entirely in the 2D subspace defined by $\mathrm{span}\{\mathbf{w}_0, \mathbf{w}_\star\}$: follow the max-curvature solution until t_1 and then transition to a radial solution until t_f.

Let $\theta_{\max}$ be the largest possible angle between $\mathbf{w}(t_0)$ and any reachable $\mathbf{w}(t_f) = \mathbf{w}_\star$, where we have fixed t_f. Define $\theta(t)$ to be the angle between $\mathbf{w}(t)$ and $\mathbf{w}(t_f)$.

$$\theta(t_0) \;=\; \int_{t_0}^{t_f} \dot{\theta}\,\mathrm{d}t \;\le\; \int_{t_0}^{t_f} |\dot{\theta}|\,\mathrm{d}t$$

An alternative expression for this rate of change is the projection of $\dot{\mathbf{w}}$ onto the orthogonal complement of $\mathbf{w}$:

$$|\dot{\theta}| = \frac{\|\dot{\mathbf{w}} - (\dot{\mathbf{w}}^\mathsf{T}\frac{\mathbf{w}}{\|\mathbf{w}\|})\frac{\mathbf{w}}{\|\mathbf{w}\|}\|}{\|\mathbf{w}\|} = \frac{R_y\|\mathbf{x} - \frac{R_y}{2\|\mathbf{w}\|^2}\mathbf{w}\|}{2\|\mathbf{w}\|}$$

Where we used the fact that $\dot{\mathbf{w}} = \mathbf{w}^\mathsf{T}\mathbf{x} = \dfrac{R_y}{2}$ in Regime IV. Now,

$$\begin{aligned}
\theta_{\max} &= \max_{\substack{\mathbf{x}:\,\mathbf{w}^\mathsf{T}\mathbf{x}=R_y/2 \\ \|\mathbf{x}\|\le R_x}} \theta(t_0) \\
&\le \max_{\substack{\mathbf{x}:\,\mathbf{w}^\mathsf{T}\mathbf{x}=R_y/2 \\ \|\mathbf{x}\|\le R_x}} \int_{t_0}^{t_f} \frac{R_y\|\mathbf{x} - \frac{R_y}{2\|\mathbf{w}\|^2}\mathbf{w}\|}{2\|\mathbf{w}\|}\,\mathrm{d}t \\
&\le \int_{t_0}^{t_f} \frac{\sqrt{R_x^2 - (\frac{R_y}{2\|\mathbf{w}\|})^2}}{\|\mathbf{w}\|}\,\mathrm{d}t \qquad \text{(A.3)}
\end{aligned}$$

In the final step, we maximized over $\mathbf{x}$. Notice that the integrand (A.3) is an upper bound that only depends on t_0 and $\|\mathbf{w}_\star\|$ but not on $\mathbf{x}$. One can also verify that this upper bound is achieved by the

choice

$$\mathbf{x} = \frac{R_y}{2\|\mathbf{w}\|}\hat{\mathbf{w}} + \sqrt{R_x^2 - \left(\frac{R_y}{2\|\mathbf{w}\|}\right)^2}\,\frac{\mathbf{w}_\star - (\hat{\mathbf{w}}^\mathsf{T}\mathbf{w}_\star)\hat{\mathbf{w}}}{\|\mathbf{w}_\star - (\hat{\mathbf{w}}^\mathsf{T}\mathbf{w}_\star)\hat{\mathbf{w}}\|}.$$

where $\hat{\mathbf{w}} := \mathbf{w}/\|\mathbf{w}\|$ and $\mathbf{w}_\star$ is any vector that satisfies (A.2) with angle $\theta_{\max}$ with $\mathbf{w}(t_0)$. Any $\mathbf{w}_\star$ with this norm but angle $\theta_f < \theta_{\max}$ can also be reached by using the max-curvature control until time t_1, where t_1 is chosen such that $\theta_f = \int_{t_0}^{t_1} \frac{\sqrt{R_x^2 - \left(\frac{R_y}{2\|\mathbf{w}\|}\right)^2}}{\|\mathbf{w}\|}\,\mathrm{d}t$, and then using $\mathbf{x} = \frac{R_y}{2\|\mathbf{w}\|^2}\mathbf{w}$ for $t_1 \le t \le t_f$. This piecewise path is illustrated in Figure A.1.

Our constructed optimal trajectory lies in the 2D span of $\mathbf{w}_\star$ and $\mathbf{w}_0$. This shows that all reachable $\mathbf{w}_\star$ can be reached via a 2D trajectory. ■

Proof of Theorem 3.4.1

Proof. For any policy ϕ and state $s \in S$, we have

$$
\begin{aligned}
& |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| && \text{(B.1)}\\
\overset{\text{Bellman}}{=}\ & |g(s,\phi(s)) + \gamma \mathbb{E}_{\hat{T}(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - g(s,\phi(s)) - \gamma \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\mathcal{M}}(s')| \\
=\ & \gamma |\mathbb{E}_{\hat{T}(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\mathcal{M}}(s')| \\
=\ & \gamma |\mathbb{E}_{\hat{T}(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') + \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\mathcal{M}}(s')| \\
\overset{\text{tri.}}{\leq}\ & \gamma |\mathbb{E}_{\hat{T}(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s')| + \gamma |\mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\mathcal{M}}(s')| \\
\overset{\text{extremal}}{\leq}\ & \gamma |\mathbb{E}_{\hat{T}(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s') - \mathbb{E}_{T(s'|s,\phi(s))} V^{\phi}_{\hat{\mathcal{M}}}(s')| + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
=\ & \gamma \left| \left\langle \hat{T}(\cdot|s,\phi(s)) - T(\cdot|s,\phi(s)), V^{\phi}_{\hat{\mathcal{M}}}(\cdot) \right\rangle \right| + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
\overset{\text{const.}}{=}\ & \gamma \left| \left\langle \hat{T}(\cdot|s,\phi(s)) - T(\cdot|s,\phi(s)), V^{\phi}_{\hat{\mathcal{M}}}(\cdot) - \frac{C_{\max}}{2(1-\gamma)} \right\rangle \right| + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
\overset{\text{Hölder}}{\leq}\ & \gamma \|\hat{T}(\cdot|s,\phi(s)) - T(\cdot|s,\phi(s))\|_1 \sup_{s \in S} \left| V^{\phi}_{\hat{\mathcal{M}}}(s) - \frac{C_{\max}}{2(1-\gamma)} \right| + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
\overset{\text{range}}{\leq}\ & \gamma \|\hat{T}(\cdot|s,\phi(s)) - T(\cdot|s,\phi(s))\|_1 \frac{C_{\max}}{2(1-\gamma)} + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
=\ & \gamma \|\hat{P} - P\|_1 \frac{C_{\max}}{2(1-\gamma)} + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \\
\leq\ & \frac{\gamma C_{\max} \epsilon}{2(1-\gamma)} + \gamma \sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)|.
\end{aligned}
$$

Since this holds for all $s \in S$, we can also take the supremum on the LHS, which yields

$$
\sup_{s \in S} |V^{\phi}_{\hat{\mathcal{M}}}(s) - V^{\phi}_{\mathcal{M}}(s)| \leq \frac{\gamma C_{\max} \epsilon}{2(1-\gamma)^2}. \qquad \text{(B.2)}
$$

Now, for any $s \in S$,

$$V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s) = V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - V_{\hat{\mathcal{M}}}^{\phi^\star_{\mathcal{M}}}(s) + V_{\hat{\mathcal{M}}}^{\phi^\star_{\mathcal{M}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s) \tag{B.3}$$

$$\overset{\text{opt.}}{\leq} V_{\mathcal{M}}^{\phi^\star_{\hat{\mathcal{M}}}}(s) - \left(V_{\hat{\mathcal{M}}}^{\phi^\star_{\hat{\mathcal{M}}}}(s)\right) + V_{\hat{\mathcal{M}}}^{\phi^\star_{\mathcal{M}}}(s) - V_{\mathcal{M}}^{\phi^\star_{\mathcal{M}}}(s) \tag{B.4}$$

$$\overset{\text{(B.2)}}{\leq} \frac{\gamma C_{\max}\epsilon}{2(1-\gamma)^2} + \frac{\gamma C_{\max}\epsilon}{2(1-\gamma)^2} \tag{B.5}$$

$$= \frac{\gamma C_{\max}\epsilon}{(1-\gamma)^2}. \tag{B.6}$$

This completes the proof. ■

Proof of Theorem 3.4.2

Proof. We first want to establish an ℓ_1 concentration bound for multinomial distribution. Observe that for any vector $v \in \mathbb{R}^N$,

$$\|v\|_1 = \max_{u \in \{-1,1\}^N} u^\mathsf{T} v. \tag{B.7}$$

The plan is to prove concentration for each $u^\mathsf{T} v$ first, and then union bound over all u to obtain the ℓ_1 error bound. Observe that $u^\mathsf{T}\hat{P}$ is the average of n i.i.d. random variables $u^\mathsf{T} e_{x_i}$ with range $[-1, 1]$. Then, by Hoeffding's Inequality, with probability at least $1 - \delta/2^N$, we have

$$u^\mathsf{T}(\hat{P} - P) \leq 2\sqrt{\frac{1}{2n} \ln \frac{2^{N+1}}{\delta}}. \tag{B.8}$$

Then, we can apply union bound across all $u \in \{-1, 1\}^N$and get that, with probability at least $1 - \delta$,

$$\|\hat{P} - P\|_1 = \max_u u^\mathsf{T}(\hat{P} - P) \leq 2\sqrt{\frac{1}{2n} \ln \frac{2^{N+1}}{\delta}} \tag{B.9}$$

Substituting this quantity into Lemma 3.4.1 yields the desired result. ■

Proof of Proposition 4.3.2

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.0.1. *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

$$H_R(Q)(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{C.1}$$

Since $\gamma < 1$, $H_R(Q)$ is a contraction mapping, i.e., $\|H_R(Q_1) - H_R(Q_2)\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty, \forall Q_1, Q_2 \in \mathcal{Q}$. 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

$$R(s, a) = Q(s, a) - \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{C.2}$$

Thus the mapping is one-to-one. ■

Proof of Theorem 4.3.2. For any target policy $\pi^\dagger : \mathcal{S} \mapsto \mathcal{A}$, we construct the following Q:

$$Q(s, a) = \begin{cases} \epsilon & \forall s \in \mathcal{S}, a = \pi^\dagger(s), \\ 0, & \text{otherwise.} \end{cases} \tag{C.3}$$

The Q values in (C.3) satisfy the constraint (4.15). Note that we construct the Q values so that for all $s \in \mathcal{S}$, $\max_a Q(s, a) = \epsilon$. By proposition C.0.1, the corresponding reward function induced by Bellman optimality equation is

$$\hat{R}(s, a) = \begin{cases} (1 - \gamma)\epsilon & \forall s \in \mathcal{S}, a = \pi^\dagger(s), \\ -\gamma\epsilon, & \text{otherwise.} \end{cases} \tag{C.4}$$

Then one can let $r_t = \hat{R}(s_t, a_t)$ so that $\mathbf{r} = (r_0, ..., r_{T-1})$, $\hat{R}$ in (C.4), together with Q in (C.3) is a feasible solution to (4.12)-(4.15). ■

Proof of Theorem 4.3.1

The proof of Theorem 4.3.1 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 $\alpha \geq 1$. This result is used later to prove Lemma C.0.2.

Lemma C.0.1. *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*

$$i). \quad x_i = \frac{1}{|\mathcal{I}|}\sum_{j\in\mathcal{I}} y_j, \forall i \in \mathcal{I}, \qquad ii). \quad x_i = y_i, \forall i \neq \mathcal{I}. \tag{C.5}$$

Then for any $\alpha \geq 1$, we have $\|x\|_\alpha \leq \|y\|_\alpha$.

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

$$\|x\|_\alpha - \|y\|_\alpha = \sum_i |x_i| - \sum_j |y_j| = \sum_{i\in\mathcal{I}} |x_i| - \sum_{j\in\mathcal{I}} |y_j| = |\sum_{j\in\mathcal{I}} y_j| - \sum_{j\in\mathcal{I}} |y_j| \leq 0. \tag{C.6}$$

Case 2: $1 < \alpha < \infty$. We show $\|x\|_\alpha^\alpha \leq \|y\|_\alpha^\alpha$. Note that

$$\begin{aligned} \|x\|_\alpha^\alpha - \|y\|_\alpha^\alpha &= \sum_i |x_i|^\alpha - \sum_j |y_j|^\alpha = \sum_{i\in\mathcal{I}} |x_i|^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha \\ &= \frac{1}{|\mathcal{I}|^{\alpha-1}} |\sum_{j\in\mathcal{I}} y_j|^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha \leq \frac{1}{|\mathcal{I}|^{\alpha-1}} (\sum_{j\in\mathcal{I}} |y_j|)^\alpha - \sum_{j\in\mathcal{I}} |y_j|^\alpha. \end{aligned} \tag{C.7}$$

Let $\beta = \frac{\alpha}{\alpha-1}$. By Holder's inequality, we have

$$\sum_{j\in\mathcal{I}} |y_j| \leq (\sum_{j\in\mathcal{I}} |y_j|^\alpha)^{\frac{1}{\alpha}} (\sum_{j\in\mathcal{I}} 1^\beta)^{\frac{1}{\beta}} = (\sum_{j\in\mathcal{I}} |y_j|^\alpha)^{\frac{1}{\alpha}} |\mathcal{I}|^{1-\frac{1}{\alpha}}. \tag{C.8}$$

Plugging (C.8) into (C.7), we have

$$\|x\|_\alpha^\alpha - \|y\|_\alpha^\alpha \leq \frac{1}{|\mathcal{I}|^{\alpha-1}} (\sum_{j\in\mathcal{I}} |y_j|^\alpha) |\mathcal{I}|^{\alpha-1} - \sum_{j\in\mathcal{I}} |y_j|^\alpha = 0. \tag{C.9}$$

Case 3: $\alpha = \infty$. We have

$$\begin{aligned} \|x\|_\alpha &= \max_i |x_i| = \max\{\frac{1}{|\mathcal{I}|} |\sum_{j\in\mathcal{I}} y_j|, \max_{i\notin\mathcal{I}} |x_i|\} \leq \max\{\frac{1}{|\mathcal{I}|} \sum_{j\in\mathcal{I}} |y_j|, \max_{i\notin\mathcal{I}} |x_i|\} \\ &\leq \max\{\max_{j\in\mathcal{I}} |y_j|, \max_{i\notin\mathcal{I}} |x_i|\} = \max\{\max_{j\in\mathcal{I}} |y_j|, \max_{j\notin\mathcal{I}} |y_j|\} = \max_j |y_j| = \|y\|_\alpha. \end{aligned} \tag{C.10}$$

Therefore $\forall \alpha \geq 1$, we have $\|x\|_\alpha \leq \|y\|_\alpha$. ■

Next we prove Lemma C.0.2, which shows that one possible optimal attack solution to (4.12)-(4.15) 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.0.2. *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* (4.12)-(4.15).

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.0.2.

Proof. Let $\mathbf{r}^* = (r_0^*, ..., r_{T-1}^*)$, $\hat{R}^*$ and Q^* be any optimal solution to (4.12)-(4.15). Fix a particular state-action pair (s,a), we have

$$\hat{R}^*(s,a) = \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t^*. \tag{C.11}$$

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

$$r_t' = \begin{cases} r_t^0 + \hat{R}^*(s,a) - \hat{R}^0(s,a), & t \in T_{s,a}, \\ r_t^*, & t \notin T_{s,a}. \end{cases} \tag{C.12}$$

Now we show $\mathbf{r}'$, $\hat{R}^*$ and Q^* is another optimal solution to (4.12)-(4.15). We first verify that $\mathbf{r}'$, $\hat{R}^*$, and Q^* satisfy constraints (4.13)-(4.15). To verify (4.13), 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

$$\begin{aligned} \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} r_t' &= \frac{1}{|T_{s,a}|} \sum_{t \in T_{s,a}} \left(r_t^0 + \hat{R}^*(s,a) - \hat{R}^0(s,a) \right) \\ &= \hat{R}^0(s,a) + \hat{R}^*(s,a) - \hat{R}^0(s,a) = \hat{R}^*(s,a), \end{aligned} \tag{C.13}$$

Thus $\mathbf{r}'$ and $\hat{R}^*$ satisfy constraint (4.13). $\hat{R}^*$ and Q^* obviously satisfy constraints (4.14) and (4.15) 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.0.1 with $\mathcal{I} = T_{s,a}$. Therefore by Lemma C.0.1, we have

$$\|\mathbf{r}' - \mathbf{r}^0\|_\alpha = \|\delta'\|_\alpha \leq \|\delta^*\|_\alpha = \|\mathbf{r}^* - \mathbf{r}^0\|_\alpha. \tag{C.14}$$

But note that by our assumption, $\mathbf{r}^*$ is an optimal solution, thus $\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \leq \|\mathbf{r}' - \mathbf{r}^0\|_\alpha$, 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

$$r'_t = r^0_t + \hat{R}^*(s, a) - \hat{R}^0(s, a), \forall t \in T_{s,a}, \forall s, a, \tag{C.15}$$

together with $\hat{R}^*$ and Q^* is an optimal solution to (4.12)-(4.15). Let $\psi(s, a) = \hat{R}^*(s, a) - \hat{R}^0(s, a)$ conclude the proof. ■

Finally, Lemma C.0.3 provides a sensitive analysis on the value function Q as the reward function changes.

Lemma C.0.3. *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*

$$(1 - \gamma)\|Q' - Q^0\|_\infty \leq \|\hat{R} - \hat{R}^0\|_\infty \leq (1 + \gamma)\|Q' - Q^0\|_\infty. \tag{C.16}$$

Proof. Define the Bellman operator as

$$H_{\hat{R}}(Q)(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{P}(s' \mid s, a) \max_{a'} Q(s', a'). \tag{C.17}$$

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

$$\begin{aligned}
\|Q' - Q^0\|_\infty &= \|H_{\hat{R}'}(Q') - H_{\hat{R}^0}(Q^0)\|_\infty \\
&= \|H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0) + H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \\
&\leq \|H_{\hat{R}'}(Q') - H_{\hat{R}'}(Q^0)\|_\infty + \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \\
&\leq \gamma\|Q' - Q^0\|_\infty + \|H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0)\|_\infty \text{ (by contraction of } H_{\hat{R}'}(\cdot)) \\
&= \gamma\|Q' - Q^0\|_\infty + \|\hat{R}' - \hat{R}^0\|_\infty \text{ (by } H_{\hat{R}'}(Q^0) - H_{\hat{R}^0}(Q^0) = \hat{R}' - \hat{R}^0)
\end{aligned} \tag{C.18}$$

Rearranging we have $(1-\gamma)\|Q'-Q^0\|_\infty \leq \|\hat{R}'-\hat{R}^0\|_\infty$. Similarly we have

$$\begin{aligned}\|Q'-Q^0\|_\infty &= \|H_{\hat{R}'}(Q')-H_{\hat{R}^0}(Q^0)\|_\infty \\ &= \|H_{\hat{R}'}(Q^0)-H_{\hat{R}^0}(Q^0)+H_{\hat{R}'}(Q')-H_{\hat{R}'}(Q^0)\|_\infty \\ &\geq \|H_{\hat{R}'}(Q^0)-H_{\hat{R}^0}(Q^0)\|_\infty - \|H_{\hat{R}'}(Q')-H_{\hat{R}'}(Q^0)\|_\infty \\ &\geq \|H_{\hat{R}'}(Q^0)-H_{\hat{R}^0}(Q^0)\|_\infty - \gamma\|Q'-Q^0\|_\infty \\ &= \|\hat{R}'-\hat{R}^0\|_\infty - \gamma\|Q'-Q^0\|_\infty\end{aligned} \tag{C.19}$$

Rearranging we have $\|\hat{R}'-\hat{R}^0\|_\infty \leq (1+\gamma)\|Q'-Q^0\|_\infty$, concluding the proof. ■

Now we are ready to prove our main result.

Proof of Theorem 4.3.1. We construct the following value function Q'.

$$Q'(s,a) = \begin{cases} Q^0(s,a)+\dfrac{\Delta(\epsilon)}{2}, & \forall s\in\mathcal{S}, a=\pi^\dagger(s), \\ Q^0(s,a)-\dfrac{\Delta(\epsilon)}{2}, & \forall s\in\mathcal{S}, \forall a\neq\pi^\dagger(s). \end{cases} \tag{C.20}$$

Note that $\forall s\in\mathcal{S}$ and $\forall a\neq\pi^\dagger(s)$, we have

$$\begin{aligned}\Delta(\epsilon) &= \max_{s'\in\mathcal{S}}[\max_{a'\neq\pi^\dagger(s')} Q^0(s',a')-Q^0(s',\pi^\dagger(s'))+\epsilon]_+ \\ &\geq \max_{a'\neq\pi^\dagger(s)} Q^0(s,a')-Q^0(s,\pi^\dagger(s))+\epsilon \geq Q^0(s,a)-Q^0(s,\pi^\dagger(s))+\epsilon,\end{aligned} \tag{C.21}$$

which leads to

$$Q^0(s,a)-Q^0(s,\pi^\dagger(s))-\Delta(\epsilon)\leq -\epsilon, \tag{C.22}$$

thus we have $\forall s\in\mathcal{S}$ and $\forall a\neq\pi^\dagger(s)$,

$$\begin{aligned}Q'(s,\pi^\dagger(s)) &= Q^0(s,\pi^\dagger(s))+\frac{\Delta(\epsilon)}{2} \\ &= Q^0(s,a)-[Q^0(s,a)-Q^0(s,\pi^\dagger(s))-\Delta(\epsilon)]-\frac{\Delta(\epsilon)}{2} \\ &\geq Q^0(s,a)+\epsilon-\frac{\Delta(\epsilon)}{2} = Q'(s,a)+\epsilon.\end{aligned} \tag{C.23}$$

Therefore Q' satisfies the constraint (4.15). By proposition C.0.1, there exists a unique function R' such that Q' satisfies the Bellman optimality equation of MDP $\hat{M}'=(\mathcal{S},\mathcal{A},\hat{P},R',\gamma)$. 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

$$\begin{aligned}\hat{R}'(s,a) &= \frac{1}{|T_{s,a}|}\sum_{t\in T_{s,a}} r'_t = \frac{1}{|T_{s,a}|}\sum_{t\in T_{s,a}}\Big(r^0_t + R'(s,a) - \hat{R}^0(s,a)\Big) \\ &= \hat{R}^0(s,a) + R'(s,a) - \hat{R}^0(s,a) = R'(s,a).\end{aligned} \tag{C.24}$$

Thus $\mathbf{r}'$, $\hat{R}'$ and Q' is a feasible solution to (4.12)-(4.15). 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

$$\|Q' - Q^0\|_\infty = \frac{\Delta(\epsilon)}{2}, \tag{C.25}$$

thus by Lemma C.0.3, we have $\forall t$,

$$\begin{aligned}|r'_t - r^0_t| &= |\hat{R}'(s_t,a_t) - \hat{R}^0(s_t,a_t)| \le \max_{s,a}|\hat{R}'(s,a) - \hat{R}^0(s,a)| = \|\hat{R}' - \hat{R}^0\|_\infty \\ &\le (1+\gamma)\|Q' - Q^0\|_\infty = \frac{1}{2}(1+\gamma)\Delta(\epsilon).\end{aligned} \tag{C.26}$$

Therefore, we have

$$\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha \le \|\mathbf{r}' - \mathbf{r}^0\|_\alpha = (\sum_{t=0}^{T-1}|r'_t - r^0_t|^\alpha)^{\frac{1}{\alpha}} \le \frac{1}{2}(1+\gamma)\Delta(\epsilon)T^{\frac{1}{\alpha}}. \tag{C.27}$$

Now we prove the lower bound. We consider two cases separately.

Case 1: $\Delta(\epsilon) = 0$. We must have $Q^0(s,\pi^\dagger(s)) \ge Q^0(s,a) + \epsilon, \forall s \in \mathcal{S}, \forall a \ne \pi^\dagger(s)$. 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' $(a' \ne \pi^\dagger(s'))$ be a state-action pair such that

$$\Delta(\epsilon) = Q^0(s',a') - Q^0(s',\pi^\dagger(s')) + \epsilon. \tag{C.28}$$

Let $\mathbf{r}^*$, $\hat{R}^*$ and Q^* be an optimal solution to (4.12)-(4.15) that takes the form in Lemma C.0.2, i.e.,

$$r^*_t = r^0_t + \hat{R}^*(s,a) - \hat{R}^0(s,a), \forall t \in T_{s,a}, \forall s,a. \tag{C.29}$$

Constraint (4.15) ensures that $Q^*(s',\pi^\dagger(s')) \ge Q^*(s',a') + \epsilon$, in which case either one of the following two conditions must hold:

$$i).\quad Q^*(s',\pi^\dagger(s')) - Q^0(s',\pi^\dagger(s')) \ge \frac{\Delta(\epsilon)}{2}, \qquad ii).\quad Q^0(s',a') - Q^*(s',a') \ge \frac{\Delta(\epsilon)}{2}, \tag{C.30}$$

since otherwise we have

$$
\begin{aligned}
&Q^*(s', \pi^\dagger(s')) < Q^0(s', \pi^\dagger(s')) + \frac{\Delta(\epsilon)}{2} = Q^0(s', \pi^\dagger(s')) + \frac{1}{2}[Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon] \\
&= \frac{1}{2}Q^0(s', a') + \frac{1}{2}Q^0(s', \pi^\dagger(s')) + \frac{\epsilon}{2} = Q^0(s', a') - \frac{1}{2}[Q^0(s', a') - Q^0(s', \pi^\dagger(s')) + \epsilon] + \epsilon \qquad \text{(C.31)} \\
&= Q^0(s', a') - \frac{\Delta(\epsilon)}{2} + \epsilon < Q^*(s', a') + \epsilon.
\end{aligned}
$$

Next note that if either $i)$ or $ii)$ holds, we have $\|Q^* - Q^0\|_\infty \geq \frac{\Delta(\epsilon)}{2}$. By Lemma C.0.3, we have

$$
\max_{s,a} |\hat{R}^*(s, a) - \hat{R}^0(s, a)| = \|\hat{R}^* - \hat{R}^0\|_\infty \geq (1 - \gamma)\|Q^* - Q^0\|_\infty \geq \frac{1}{2}(1 - \gamma)\Delta(\epsilon). \qquad \text{(C.32)}
$$

Let $s^*, a^* \in \arg\max_{s,a} |\hat{R}^*(s, a) - \hat{R}^0(s, a)|$, then we have

$$
|\hat{R}^*(s^*, a^*) - \hat{R}^0(s^*, a^*)| \geq \frac{1}{2}(1 - \gamma)\Delta(\epsilon). \qquad \text{(C.33)}
$$

Therefore, we have

$$
\begin{aligned}
\|\mathbf{r}^* - \mathbf{r}^0\|_\alpha^\alpha &= \sum_{t=0}^{T-1} |r_t^* - r_t^0|^\alpha = \sum_{s,a} \sum_{t \in T_{s,a}} |r_t^* - r_t^0|^\alpha \geq \sum_{t \in T_{s^*,a^*}} |r_t^* - r_t^0|^\alpha \\
&= \sum_{t \in T_{s^*,a^*}} |\hat{R}^*(s^*, a^*) - \hat{R}^0(s^*, a^*)|^\alpha \geq \left(\frac{1}{2}(1 - \gamma)\Delta(\epsilon)\right)^\alpha |T_{s^*,a^*}| \qquad \text{(C.34)} \\
&\geq \left(\frac{1}{2}(1 - \gamma)\Delta(\epsilon)\right)^\alpha \min_{s,a} |T_{s,a}|.
\end{aligned}
$$

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

We finally point out that while an optimal solution $\mathbf{r}^*$ may not necessarily take the form in Lemma C.0.2, 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. ■

Convex Surrogate for LQR Attack Optimization

By pulling the positive semi-definite constraints on Q and R out of the lower level optimization (4.32), one can turn the original attack optimization (4.27)-(4.33) into the following surrogate optimization:

$$\min_{\mathbf{r},\hat{Q},\hat{R},\hat{q},\hat{c},X,x} \quad \|\mathbf{r}-\mathbf{r}_0\|_\alpha \tag{C.35}$$

$$\text{s.t.} \quad -\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}=K^\dagger, \tag{C.36}$$

$$-\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top x=k^\dagger, \tag{C.37}$$

$$X=\gamma\hat{A}^\top X\hat{A}-\gamma^2\hat{A}^\top X\hat{B}\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}+\hat{Q} \tag{C.38}$$

$$x=\hat{q}+\gamma(\hat{A}+\hat{B}K^\dagger)^\top x \tag{C.39}$$

$$(\hat{Q},\hat{R},\hat{q},\hat{c})=\arg\min\sum_{t=0}^{T-1}\left\|\frac{1}{2}s_t^\top Qs_t+q^\top s_t+a_t^\top Ra_t+c+r_t\right\|_2^2 \tag{C.40}$$

$$\hat{Q}\succeq 0,\hat{R}\succeq\epsilon I,X\succeq 0. \tag{C.41}$$

The feasible set of (C.35)-(C.41) 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 (C.42)-(C.43). The feasible set for $\hat{a}$ is a singleton set $\{0\}$, and the optimal objective value is 0.

$$\min_{\hat{a}} \quad 0 \tag{C.42}$$

$$\text{s.t.} \quad \hat{a}=\arg\min_{a\geq 0}(a+3)^2, \tag{C.43}$$

Once we pull the constraint out of the lower-level optimization (C.43), we end up with a surrogate optimization (C.44)-(C.46). Note that (C.45) requires $\hat{a}=-3$, which does not satisfy (C.46). Therefore the feasible set of the surrogate optimization is $\emptyset$, meaning it is more stringent than (C.42)-(C.43).

$$\min_{\hat{a}} \quad 0 \tag{C.44}$$

$$\text{s.t.} \quad \hat{a}=\arg\min(a+3)^2, \tag{C.45}$$

$$\hat{a}\geq 0 \tag{C.46}$$

Back to our attack optimization (C.35)-(C.41), this surrogate attack optimization comes with the advantage of being convex, thus can be solved to global optimality.

Proposition C.0.2. *The surrogate attack optimization* (C.35)-(C.41) *is convex.*

Proof. First note that the sub-level optimization (C.40) is itself a convex problem, thus is equivalent to the corresponding KKT condition. We write out the KKT condition of (C.40) to derive an explicit form of our attack formulation as below:

$$\min_{\mathbf{r},\hat{Q},\hat{R},\hat{q},\hat{c},X,x} \quad \|\mathbf{r}-\mathbf{r}_0\|_\alpha \tag{C.47}$$

$$\text{s.t.} \quad -\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}=K^\dagger, \tag{C.48}$$

$$-\gamma\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top x=k^\dagger, \tag{C.49}$$

$$X=\gamma\hat{A}^\top X\hat{A}-\gamma^2\hat{A}^\top X\hat{B}\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)^{-1}\hat{B}^\top X\hat{A}+\hat{Q} \tag{C.50}$$

$$x=\hat{q}+\gamma(\hat{A}+\hat{B}K^\dagger)^\top x \tag{C.51}$$

$$\sum_{t=0}^{T-1}(\frac{1}{2}s_t^\top\hat{Q}s_t+\hat{q}^\top s_t+a_t^\top\hat{R}a_t+\hat{c}+r_t)s_ts_t^\top=0, \tag{C.52}$$

$$\sum_{t=0}^{T-1}(\frac{1}{2}s_t^\top\hat{Q}s_t+\hat{q}^\top s_t+a_t^\top\hat{R}a_t+\hat{c}+r_t)a_ta_t^\top=0, \tag{C.53}$$

$$\sum_{t=0}^{T-1}(\frac{1}{2}s_t^\top\hat{Q}s_t+\hat{q}^\top s_t+a_t^\top\hat{R}a_t+\hat{c}+r_t)s_t=0, \tag{C.54}$$

$$\sum_{t=0}^{T-1}(\frac{1}{2}s_t^\top\hat{Q}s_t+\hat{q}^\top s_t+a_t^\top\hat{R}a_t+\hat{c}+r_t)=0, \tag{C.55}$$

$$\hat{Q}\succeq 0,\hat{R}\succeq\epsilon I,X\succeq 0. \tag{C.56}$$

The objective is obviously convex. (C.48)-(C.50) are equivalent to

$$-\gamma\hat{B}^\top X\hat{A}=\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)K^\dagger. \tag{C.57}$$

$$-\gamma\hat{B}^\top x=\left(\hat{R}+\gamma\hat{B}^\top X\hat{B}\right)k^\dagger. \tag{C.58}$$

$$X=\gamma\hat{A}^\top X(\hat{A}+\hat{B}K^\dagger)+\hat{Q}, \tag{C.59}$$

Note that these three equality constraints are all linear in X, $\hat{R}$, x, and $\hat{Q}$. (C.51) is linear in x and $\hat{q}$. (C.52)-(C.55) are also linear in $\hat{Q}$, $\hat{R}$, $\hat{q}$, $\hat{c}$ and $\mathbf{r}$. Finally, (C.56) 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.0.3. *Let $\hat{A}$, $\hat{B}$ be the learner's estimated transition kernel. Let*

$$L^\dagger(s,a) = \frac{1}{2}s^\top Q^\dagger s + (q^\dagger)^\top s + a^\top R^\dagger a + c^\dagger \tag{C.60}$$

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 (C.35)-(C.41) *is feasible. Furthermore, the optimal solution can be achieved.*

Proof. To prove feasibility, it suffices to construct a feasible solution to optimization (C.35)-(C.41). Let

$$r_t = \frac{1}{2}s_t^\top Q^\dagger s_t + {q^\dagger}^\top s_t + a_t^\top R^\dagger a_t + c^\dagger \tag{C.61}$$

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 (C.36)-(C.39) must be satisfied with some X and x. The poisoned reward vector $\mathbf{r}$ obviously satisfies (C.40) since it is constructed exactly as the minimizer. By our assumption, $R^\dagger \succeq \epsilon I$, thus (C.41) 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 (C.35)-(C.41) is feasible. Furthermore, since the feasible set is closed, the optimal solution can be achieved. ■

Conditions for The LQR Learner to Have Unique Estimate

The LQR learner estimates the cost function by

$$(\hat{Q},\hat{R},\hat{q},\hat{c}) = \underset{(Q\succeq 0, R\succeq \epsilon I, q, c)}{\arg\min} \frac{1}{2}\sum_{t=0}^{T-1}\left\|\frac{1}{2}s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c + r_t\right\|_2^2. \tag{C.62}$$

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

$$\psi_t = \frac{1}{2}s_t^\top Q s_t + q^\top s_t + a_t^\top R a_t + c, 0 \le t \le T-1. \tag{C.63}$$

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 (C.63) 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.0.4. *$\forall D$, $\Psi(D) = \{\psi(D,Q,R,q,c) \mid Q \succeq 0, R \succeq \epsilon I, q, c\}$ 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

$$\psi_{1,t} = \frac{1}{2} s_t^\top Q_1 s_t + q_1^\top s_t + a_t^\top R_1 a_t + c_1 \tag{C.64}$$

for some $Q_1 \succeq 0$, $R_1 \succeq \epsilon I$, q_1 and c_1, and

$$\psi_{2,t} = \frac{1}{2} s_t^\top Q_2 s_t + q_2^\top s_t + a_t^\top R_2 a_t + c_2 \tag{C.65}$$

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

$$\begin{aligned} \psi_{3,t} =& \frac{1}{2} s_t^\top [kQ_1 + (1-k)Q_2] s_t + [kq_1 + (1-k)q_2]^\top s_t \\ &+ a_t^\top [kR_1 + (1-k)R_2] a_t + kc_1 + (1-k)c_2 \end{aligned} \tag{C.66}$$

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 (C.62) is intrinsically a least-squares problem with positive semi-definite constraints on Q and R, and is equivalent to solving the following linear equation:

$$\frac{1}{2} s_t^\top \hat{Q} s_t + \hat{q}^\top s_t + a_t^\top \hat{R} a_t + \hat{c} = \psi_t^*, \forall t, \tag{C.67}$$

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 (C.67) is unique if and only if the following two conditions both hold

i). The projection ψ^* is unique.

ii). (C.67) 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. (C.67) 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

$$\mathbf{A} = \left[\begin{array}{cccccc|ccccc|c|c} \frac{s_{0,1}^2}{2} & \dots & \frac{s_{0,i}s_{0,j}}{2} & \dots & \frac{s_{0,n}^2}{2} & a_{0,1}^2 & \dots & a_{0,i}a_{0,j} & \dots & a_{0,m}^2 & s_0^\top & 1 \\ \frac{s_{1,1}^2}{2} & \dots & \frac{s_{1,i}s_{1,j}}{2} & \dots & \frac{s_{1,n}^2}{2} & a_{1,1}^2 & \dots & a_{1,i}a_{2,j} & \dots & a_{1,m}^2 & s_1^\top & 1 \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \frac{s_{t,1}^2}{2} & \dots & \frac{s_{t,i}s_{t,j}}{2} & \dots & \frac{s_{t,n}^2}{2} & a_{t,1}^2 & \dots & a_{t,i}a_{t,j} & \dots & a_{t,m}^2 & s_t^\top & 1 \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \frac{s_{T-1,1}^2}{2} & \dots & \frac{s_{T-1,i}s_{T-1,j}}{2} & \dots & \frac{s_{T-1,n}^2}{2} & a_{T-1,1}^2 & \dots & a_{T-1,i}a_{T-1,j} & \dots & a_{T-1,m}^2 & s_{T-1}^\top & 1 \end{array}\right],$$

$$\mathbf{x}^\top = \left[\begin{array}{ccccc|ccccc|cccc|c} \hat{Q}_{11} & \dots & \hat{Q}_{ij} & \dots & \hat{Q}_{nn} & \hat{R}_{11} & \dots & \hat{R}_{ij} & \dots & \hat{R}_{mm} & \hat{q}_1 & \dots & \hat{q}_i & \dots & \hat{q}_n & \hat{c} \end{array}\right],$$

then (C.67) 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.

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 C.1. 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.

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 C.2, 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 C.3, 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 4.3. However, the poisoning is highly sparse: only 10 out of 400 rewards are changed.

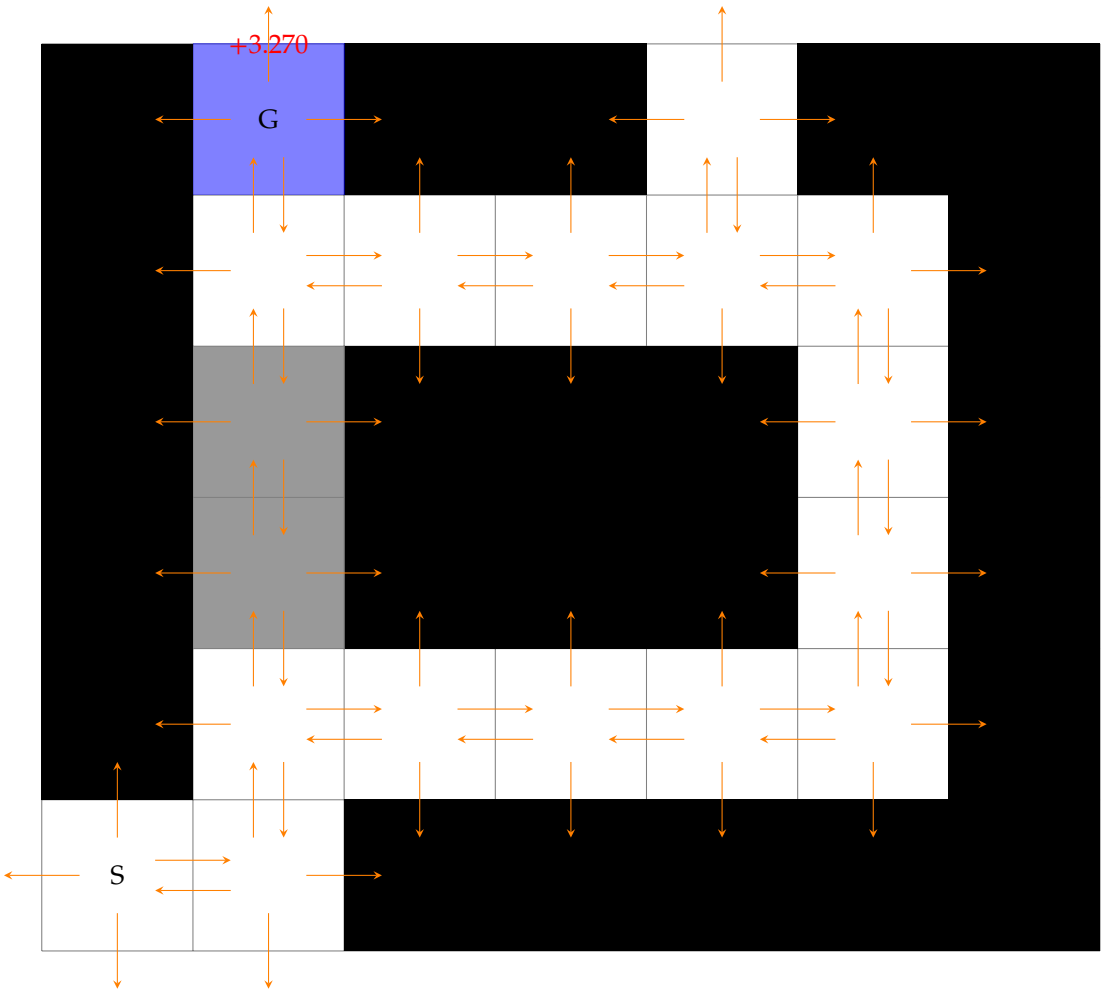


Figure C.1: Sparse reward modification for MDP experiment 2.

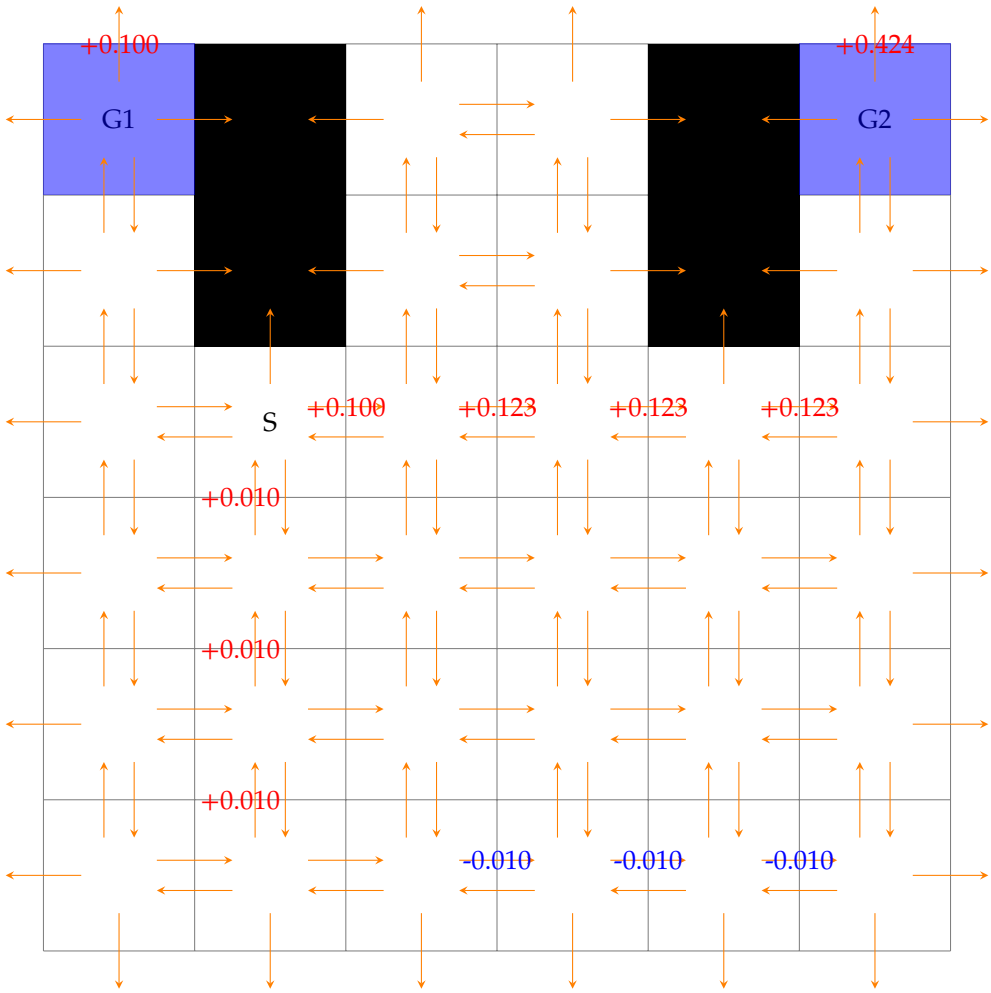
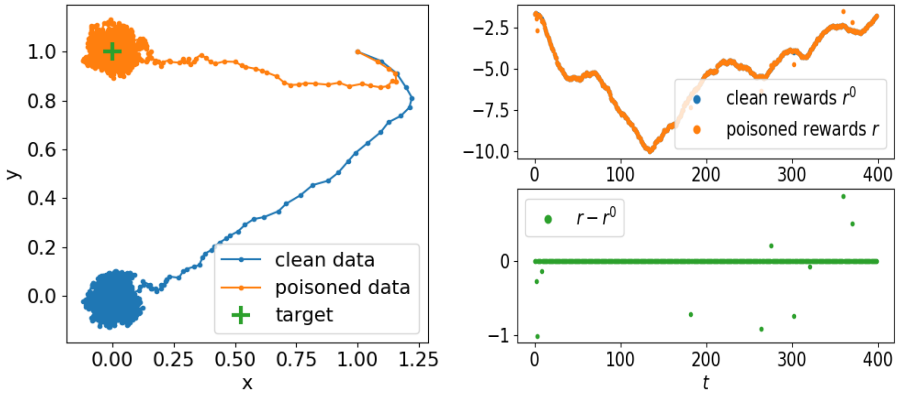


Figure C.2: Sparse reward modification for MDP experiment 3.



(a) Clean and poisoned vehicle trajectory. (b) Clean and poisoned rewards.

Figure C.3: Sparse-poisoning a vehicle running LQR in 4D state space.

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:

$$L(s,a) = \frac{1}{2}s^\top Qs + q^\top s + c + a^\top Ra. \tag{C.68}$$

When $q = 0, c = 0$, we recover the classic LQR setting. Assume the general value function takes the form $V(s) = \frac{1}{2}s^\top Xs + s^\top x + v$. 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:

$$\begin{aligned}
Q(s,a) &= \frac{1}{2}s^\top Qs + q^\top s + c + a^\top Ra + \gamma V(As + Ba) \\
&= \frac{1}{2}s^\top Qs + q^\top s + c + a^\top Ra + \gamma\left(\frac{1}{2}(As+Ba)^\top X(As+Ba) + (As+Ba)^\top x + v\right) \\
&= \frac{1}{2}s^\top(Q + \gamma A^\top XA)s + \frac{1}{2}a^\top(R + \gamma B^\top XB)a + s^\top(\gamma A^\top XB)a \\
&\quad + s^\top(q + \gamma A^\top x) + a^\top(\gamma B^\top x) + (c + \gamma v).
\end{aligned} \tag{C.69}$$

We minimize a above:

$$\begin{aligned}
&(R + \gamma B^\top XB)a + \gamma B^\top XAs + \gamma B^\top x = 0 \\
&\Rightarrow a = -\gamma(R + \gamma B^\top XB)^{-1}B^\top XAs - \gamma(R + \gamma B^\top XB)^{-1}B^\top x \triangleq Ks + k.
\end{aligned} \tag{C.70}$$

Now we substitute it back to $Q(s,a)$ and regroup terms, we get:

$$\begin{aligned} V(s) =& \frac{1}{2} s^\top (Q + \gamma A^\top X A + K^\top (R + \gamma B^\top X B) K + 2\gamma A^\top X B K) s \\ &+ s^\top (K^\top (R + \gamma B^\top X B) k + \gamma A^\top X B k + q + \gamma A^\top x + \gamma K^\top B^\top x) + C \end{aligned} \tag{C.71}$$

for some constant C, which gives us the following recursion:

$$\begin{aligned} X &= \gamma A^\top X A - \gamma^2 A^\top X B (R + \gamma B^\top X B)^{-1} B^\top X A + Q, \\ x &= q + \gamma (A + BK)^\top x. \end{aligned} \tag{C.72}$$

Proof of Theorem 5.3.1

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, $Q_{t,-\Delta} \to Q_{-\Delta}$ [124].

Next, we want to show that $Q_t(s, a) \leq Q_{t,+\Delta}(s, a)$ 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 $Q_k(s, a) \leq Q_{k,+\Delta}(s, a)$. We have

$$
\begin{aligned}
& Q_{k+1,+\Delta}(s_{k+1}, a_{k+1}) && \text{(D.1)} \\
= \; & (1 - \alpha_{k+1}) Q_{k,+\Delta}(s_{k+1}, a_{k+1}) + \alpha_{k+1} \left(r_{k+1} + \Delta + \gamma \max_{a' \in A} Q_{k,+\Delta}(s'_{k+1}, a') \right) && \text{(D.2)} \\
\geq \; & (1 - \alpha_{k+1}) Q_k(s_{k+1}, a_{k+1}) + \alpha_{k+1} \left(r_{k+1} + \delta_{k+1} + \gamma \max_{a' \in A} Q_k(s'_{k+1}, a') \right) && \text{(D.3)} \\
= \; & Q_{k+1}(s_{k+1}, a_{k+1}), && \text{(D.4)}
\end{aligned}
$$

which established the induction. Similarly, we have $Q_t(s, a) \geq Q_{t,-\Delta}(s, a)$. Since $Q_{t,+\Delta} \to Q_{+\Delta}$, $Q_{t,-\Delta} \to Q_{-\Delta}$, we have that for large enough t,

$$
Q_{-\Delta}(s, a) \leq Q_t(s, a) \leq Q_{+\Delta}, \forall s \in S, a \in A. \tag{D.5}
$$

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. ■

Proof of Theorem 5.3.5

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':

$$Q'(s,a) = \begin{cases} Q^*(s,a) + \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \in \pi^\dagger(s), \\ Q^*(s,a) - \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \notin \pi^\dagger(s), \\ Q^*(s,a), \forall s \notin S^\dagger, a, \end{cases} \tag{D.6}$$

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

$$\begin{aligned} Q'(s,a^\dagger) &= Q^*(s,a^\dagger) + \frac{\Delta}{(1+\gamma)} && \text{(D.7)} \\ &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q^*(s,a') - \frac{\Delta}{(1+\gamma)} && \text{(D.8)} \\ &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a'), && \text{(D.9)} \end{aligned}$$

Next note that

$$\begin{aligned} \Delta > \Delta_3 &\geq \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - \max_{a \in \pi^\dagger(s)} Q^*(s,a)] && \text{(D.10)} \\ &= \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - Q^*(s,a^\dagger)] && \text{(D.11)} \\ &\geq \frac{1+\gamma}{2}[Q^*(s,a') - Q^*(s,a^\dagger)], && \text{(D.12)} \end{aligned}$$

which is equivalent to

$$Q^*(s,a^\dagger) - Q^*(s,a') > -\frac{2\Delta}{1+\gamma}, \tag{D.13}$$

thus we have

$$\begin{aligned} Q'(s,a^\dagger) &= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a') && \text{(D.14)} \\ &> 0 + Q'(s,a') = Q'(s,a'). && \text{(D.15)} \end{aligned}$$

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 [120], the Bellman optimality equation

induces a unique reward function $R'(s,a)$ corresponding to Q':

$$R'(s,a) = Q'(s,a) - \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q'(s',a'). \tag{D.16}$$

We then construct our attack policy $\phi^{sas}_{\Delta_3}$ as:

$$\phi^{sas}_{\Delta_3}(s,a) = R'(s,a) - R(s,a), \forall s,a. \tag{D.17}$$

The $\phi^{sas}_{\Delta_3}(s,a)$ 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 $\phi^{sas}_{\Delta_3}(s,a)$, the objective value (5.5) is finite, thus the attack is feasible. To prove feasibility, we consider adapting Theorem 4 in [61], re-stated as below.

Lemma D.0.1 (Even-Dar & Mansour)**.** *Assume the attack is $\phi^{sas}_{\Delta_3}(s,a)$ 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 $\|Q_T - Q'\|_{\infty} \leq \tau$ with*

$$T = \Omega\left(L^{3+\frac{1}{\omega}} \frac{1}{\tau^2} (\ln \frac{1}{\delta\tau})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}} \ln \frac{1}{\tau} \right), \tag{D.18}$$

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 $\|Q_T - Q'\|_{\infty} \leq \tau_0$ implies $Q_T \in Q^{\dagger}$. From now on we fix this τ_0, thus the bound in the above theorem becomes

$$T = \Omega\left(L^{3+\frac{1}{\omega}} (\ln \frac{1}{\delta})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}} \right). \tag{D.19}$$

As the authors pointed out in [61], 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

$$T = \Omega\left(L^5 (\ln \frac{1}{\delta})^2 \right). \tag{D.20}$$

Now we represent δ as a function of T to obtain that $\forall T > 0$,

$$P[\|Q_T - Q'\|_{\infty} > \tau_0] \leq C \exp(-L^{-\frac{5}{2}} T^{\frac{1}{2}}). \tag{D.21}$$

Let $e_t = \mathbb{1}\left[\|Q_t - Q'\|_\infty > \tau_0\right]$, then we have

$$\begin{aligned}
\mathbb{E}_{\phi^{sas}_{\Delta_3}}\left[\sum_{t=1}^{\infty}\mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]\right] &\leq \mathbb{E}_{\phi^{sas}_{\Delta_3}}\left[\sum_{t=1}^{\infty} e_t\right] && \text{(D.22)}\\
&= \sum_{t=1}^{\infty} P[\|Q_T - Q'\|_\infty > \tau_0] \leq \sum_{t=1}^{\infty} C\exp(-L^{-\frac{5}{2}}t^{\frac{1}{2}}) && \text{(D.23)}\\
&\leq \int_{t=0}^{\infty} C\exp(-L^{-\frac{5}{2}}t^{\frac{1}{2}})dt = 2CL^5, && \text{(D.24)}
\end{aligned}$$

which is finite. Therefore the attack is feasible.

It remains to validate that $\phi^{sas}_{\Delta_3}$ is a legitimate attack, i.e., $|\delta_t| \leq \Delta$ under attack policy $\phi^{sas}_{\Delta_3}$. By Lemma 7 in [120], we have

$$\begin{aligned}
|\delta_t| &= |R'(s_t, a_t) - R(s_t, a_t)| && \text{(D.25)}\\
&\leq \max_{s,a}[R'(s,a) - R(s,a)] = \|R' - R\|_\infty && \text{(D.26)}\\
&\leq (1+\gamma)\|Q' - Q^*\| = (1+\gamma)\frac{\Delta}{(1+\gamma)} = \Delta. && \text{(D.27)}
\end{aligned}$$

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:

leftmirgin=*, nolistsep [74] produces the non-adaptive attack that is feasible with the smallest Δ. In particular, it solves for the following optimization problem:

$$\begin{aligned}
&\min_{\delta, Q\in\mathbb{R}^{S\times A}} \|\delta\|_\infty && \text{(D.28)}\\
&\quad \text{s.t. } Q(s,a) = \delta(s,a) + \mathbb{E}_{P(s'|s,a)}\left[R(s,a,s) + \gamma \max_{a'\in A} Q(s',a')\right] && \text{(D.29)}\\
&\quad Q \in \mathcal{Q}^\dagger && \text{(D.30)}
\end{aligned}$$

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.

leftmiirgiin=*, noliistsep $\phi^{sas}_{\Delta_3}$ 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.

leftmiiirgiiin=*, noliiistsep 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 [74]. It's not clear between $\phi^{sas}_{\Delta_3}$ and [74] 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 [74] does not. But there probably exist counterexamples on which [74] is faster than $\phi^{sas}_{\Delta_3}$.

D.1 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:

$$V_{n-1} = 0 \tag{D.31}$$

$$V_k = 1 + (1 - \frac{\epsilon}{2})V_{k-1} + \frac{\epsilon}{2}V_{k+1}, \text{for } k = 1, ..., n-2 \tag{D.32}$$

$$V_0 = 1 + (1 - \frac{\epsilon}{2})V_0 + \frac{\epsilon}{2}V_1 \tag{D.33}$$

Solving the recursive gives

$$V_0 = \frac{p(1 + p(1-2p))}{(1-2p)^2}\left[(\frac{1-p}{p})^{n-1} - 1\right] \tag{D.34}$$

where $p = \frac{\epsilon}{2} < \frac{1}{2}$ and thus $V_0 = O(\exp(n))$. ■

Proof of Theorem 5.3.8

Lemma D.1.1. *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:

$$V_1 = 0 \tag{D.35}$$

$$V_t = 1 + (1-\epsilon)V_{t-1}\epsilon \left[\frac{t-1}{|A|} V_{t-1} + \frac{1}{A} V_1 + \frac{|A|-t}{|A|} V_t \right] \tag{D.36}$$

Equation (D.36) can be simplified to

$$V_t = \frac{1 - \epsilon + \epsilon \frac{t-1}{|A|}}{1 - \epsilon \frac{|A|-t}{|A|}} V_{t-1} + \frac{1}{1 - \epsilon \frac{|A|-t}{|A|}} \tag{D.37}$$

$$\leq V_{t-1} + \frac{1}{1-\epsilon} \tag{D.38}$$

Thus, we have

$$V_t \leq \frac{t-1}{1-\epsilon} \leq \frac{|A|}{1-\epsilon} \tag{D.39}$$

as needed. ■

Now, we prove Theorem 5.3.8.

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

$$\pi_i^\dagger = \underset{\pi \in \Pi, \pi(s_j) \in \pi^\dagger(s_j), \forall j \leq i-1}{\arg\min} \mathbb{E}_{s_0 \sim \mu_0} \left[d_\pi(s_0, s_i) \right]. \tag{D.40}$$

First, we define another policy

$$\hat{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_1, ..., s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{D.41}$$

Clearly $\mathbb{E}_{s_0 \sim \mu_0}\left[d_{\pi_i^\dagger}(s_0, s_i)\right] \leq \mathbb{E}_{s_0 \sim \mu_0}\left[d_{\hat{\pi}_i^\dagger}(s_0, s_i)\right]$ for all i.

We now prove by induction that $d_{\hat{\pi}_i^\dagger}(s, s_i) \leq \left[\frac{|A|}{\epsilon}\right]^{i-1} D$ for all i and $s \in S$.

First, let $i = 1$, $\hat{\pi}_i^\dagger = \pi_{s_1}$, and thus $d_{\hat{\pi}_i^\dagger}(s, s_i) \leq D$.

Next, we assume that when $i = k$, $d_{\hat{\pi}_i^\dagger}(s, s_i) \leq D_k$, and would like to show that when $i = k + 1$, $d_{\tilde{\pi}_i^\dagger}(s, s_i) \leq \left[\frac{|A|}{\epsilon}\right] D_k$. Define another policy

$$\tilde{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_2, ..., s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{D.42}$$

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 $d_{\tilde{\pi}_i^\dagger}(s, s_i) \leq D_k$. 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

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s_1, s_k) &= (1-\epsilon)Q(s_1, \pi^\dagger(s_1)) + \epsilon \bar{Q}(s_1, a) && \text{(D.43)} \\ &\leq \max_{a \in A} Q(s_1, a) && \text{(D.44)} \\ d_{\tilde{\pi}_i^\dagger}(s_1, s_k) &= (1-\epsilon)Q(s_1, \pi_{s_1}(s_1)) + \epsilon \bar{Q}(s_1, a) && \text{(D.45)} \\ &\geq \frac{\epsilon}{|A|} \max_{a \in A} Q(s_1, a) && \text{(D.46)} \\ & && \text{(D.47)} \end{aligned}$$

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,

$$d_{\hat{\pi}_i^\dagger}(s, s_k) \leq \frac{|A|}{\epsilon} d_{\tilde{\pi}_i^\dagger}(s_1, s_k) \tag{D.48}$$

With this, we can perform the following decomposition:

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s, s_k) &= \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right]\left(d_{\hat{\pi}_i^\dagger}(s, s_1) + d_{\hat{\pi}_i^\dagger}(s_1, s_k)\right) + \mathbb{P}\left[\text{not visit } s_1\right]\left(d_{\hat{\pi}_i^\dagger}(s, s_1)|\text{not visit } s_1\right) \\ &\leq \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right]\left(d_{\tilde{\pi}_i^\dagger}(s, s_1) + \frac{|A|}{\epsilon} d_{\tilde{\pi}_i^\dagger}(s_1, s_k)\right) + \mathbb{P}\left[\text{not visit } s_1\right]\left(d_{\tilde{\pi}_i^\dagger}(s, s_k)|\text{not visit } s_1\right) \\ &= d_{\tilde{\pi}_i^\dagger}(s, s_k) + \left(\frac{|A|}{\epsilon} - 1\right) d_{\tilde{\pi}_i^\dagger}(s_1, s_k) \\ &\leq D_k + \left(\frac{|A|}{\epsilon} - 1\right) D_k = \frac{|A|}{\epsilon} D_k. \end{aligned}$$

This completes the induction. Thus, we have

$$d_{\hat{\pi}_i^\dagger}(s, s_i) \leq \left(\frac{|A|}{\epsilon}\right)^{i-1} D, \tag{D.49}$$

and the total number of iterations taken to arrive at all target states sequentially sums up to

$$\sum_{i=1}^{n} d_{\hat{\pi}_i^\dagger}(s, s_i) \leq \left(\frac{|A|}{\epsilon}\right)^{n} D. \tag{D.50}$$

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. ■

D.2 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 D.1.

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 paper. In section D.1, 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.

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 added to the actor network optimization objective.
buffer size	10^7	Replay buffer size, larger than total number of iterations.
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 D.1: Hyperparameters for TD3.

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 5.3.8 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.

D.3 Experiment Setting and Hyperparameters for TD3

Throughout the experiments, we use the following set of hyperparameters for TD3, described in Table D.1. 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.

D.4 Additional Experiments

Additional Plot for the rate comparison experiment

See Figure D.1.

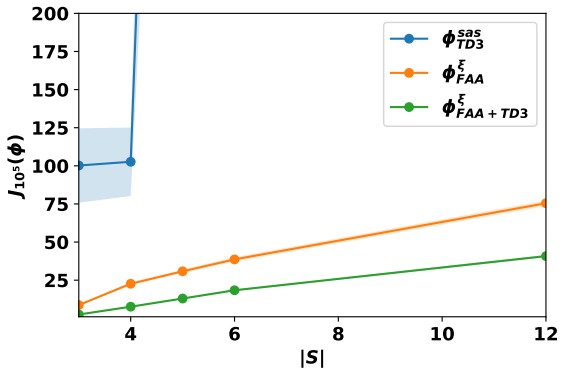


Figure D.1: Attack performances on the chain MDP of different length in the normal scale. As can be seen in the plot, both ϕ^{ξ}_{FAA} + $\phi^{\xi}_{TD3+FAA}$ achieve linear rate.

Additional Experiments: Attacking DQN

Throughout the main paper, 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. 6. 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 agent is a Deep Q-Network (DQN), the internal state will consist of the

Algorithm 6 Reward Poisoning against general RL agent

Parameters: MDP (S, A, R, P, μ_0), RL agent hyperparameters.

for $t = 0, 1, ...$ **do**
  agent at state s_t, has internal state θ_0.
  agent acts according to a behavior policy:
$a_t \leftarrow \pi_{\theta_t}(s_t)$
  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.
  attacker perturbs the reward to $r_t + \delta_t$
  agent receives $(s_{t+1}, r_t + \delta_t, EOE)$, performs one-step of internal state update:

$$\theta_{t+1} = f(\theta_t, s_t, a_t, s_{t+1}, r_t + \delta_t, EOE) \tag{D.51}$$

  environment resets if $EOE = 1$: $s_{t+1} \sim \mu_0$.

Q-network 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

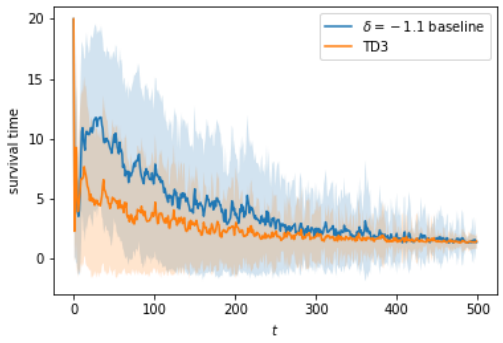


Figure D.2: Result for attacking DQN on the Cartpole environment. The left figure plots the cumulative attack cost $J_T(\phi)$ as a function of T. The right figure plot the performance of the DQN agent $J(\theta_t)$ under the two attacks.

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. D.2.

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.

E.1 Proof for the lower-bound result

Theorem E.1.1 (Theorem 6.2.1)**.** *For any algorithm, there exists an MDP such that the algorithm fails to find an* $\left(\frac{\epsilon}{2(1-\gamma)}\right)$*-optimal policy under the* ϵ*-contamination model with a probability of at least* $1/4$*.*

Proof of Theorem E.1.1. Consider two MDPs M_1, M_2, both with 3 states and 2 actions, defined as

$$P_1(s_2|s_1,a_1) = \frac{1-\epsilon}{2}, P_1(s_3|s_1,a_1) = \frac{1+\epsilon}{2}, P_1(s_3|s_1,a_2) = P_1(s_3|s_1,a_2) = \frac{1}{2} \tag{E.1}$$

$$P_2(s_2|s_1,a_1) = \frac{1+\epsilon}{2}, P_2(s_3|s_1,a_1) = \frac{1-\epsilon}{2}, P_2(s_3|s_1,a_2) = P_2(s_3|s_1,a_2) = \frac{1}{2} \tag{E.2}$$

and for both MDPs s_2, s_3 are absorbing states with constant reward 1 and 0, respectively. So for M_1, the optimal policy is $\pi_1^*(s_1) = a_2$, and for M_2, the optimal policy is $\pi_2^*(s_1) = a_1$. In both cases, choosing the alternative action in s_1 will incur a suboptimality gap of $\frac{\epsilon}{2(1-\gamma)}$.

Let $N(\cdot)$ be the probability function of Bernoulli distribution on $\{s_2, s_3\}$: $N(x) = \begin{cases} 1 & \text{if } x = s_2 \\ 0 & \text{if } x = s_3 \end{cases}$. First of all, notice that an 2ϵ*-oblivious adversary* can make the two MDPs M_1, M_2 indistinguishable by changing $P_1(\cdot \mid s_1, a_1)$ to be $(1 - \frac{2\epsilon}{1+\epsilon})P_1(\cdot \mid s_1, a_1) + \frac{2\epsilon}{1+\epsilon}N(\cdot)$, which is exactly $P_2(\cdot \mid s_1, a_1)$. Note that $\frac{2\epsilon}{1+\epsilon} \leq 2\epsilon$ and thus can be achieved by a 2ϵ-oblivious adversary.

When the two MDPs are indistinguishable, any rollout has the same probability under both MDP, and thus conditioned on any roll-out, the learner can at best obtain an $\frac{\epsilon}{2(1-\gamma)}$-optimal policy with probability $1/2$ on both MDP.

What remains to be shown is that with high probability, the ϵ-contamination adversary can simulate the oblivious adversary.

Let X_i, Y_i be Bernoulli random variables s.t. $X_i = \begin{cases} s_2 & U \leq \frac{1-\epsilon}{2} \\ s_3 & \text{o.w.} \end{cases}$, $Y_i = \begin{cases} s_2 & U \leq \frac{1+\epsilon}{2} \\ s_3 & \text{o.w.} \end{cases}$, where U is picked uniformly random in $[0, 1]$. Then (X_i, Y_i) is a coupling with law: $P((X_i, Y_i) = (s_2, s_2)) = \frac{1-\epsilon}{2}$, $P((X_i, Y_i) = (s_2, s_3)) = 0$, $P((X_i, Y_i) = (s_3, s_2)) = \epsilon$, $P((X_i, Y_i) = (s_3, s_3)) = \frac{1-\epsilon}{2}$, X_i and Y_i can be thought as the outcome of $P_1(\cdot \mid s_1, a_1)$, $P_2(\cdot \mid s_1, a_1)$ respectively. The ϵ-contamination adversary can simulate the oblivious adversary by changing X_i to Y_i when $X_1 \neq Y_i$, which has probability ϵ. This is possible when there are at most ϵ fraction of index i s.t. $X_i \neq Y_i$. Suppose there are T episodes, then

$$P\left(\sum_{i=1}^{T} \mathbb{1}_{\{a_1 \text{ is taken at } s_1\}} \mathbb{1}_{\{X_i \neq Y_i\}} \geq \epsilon T\right) \leq P(\sum_{i=1}^{T} \mathbb{1}_{\{X_i \neq Y_i\}} \geq T\epsilon) \leq \frac{1}{2} \tag{E.3}$$

because the median of Binomial(n, p) is at most $\lceil np \rceil$. Therefore, the probability that the adaptive adversary can simulate the oblivious adversary throughout T episodes is at least $1/2$. Assuming that when the adversary fails to simulate, the learner automatically succeed in finding the optimal policy, then we've established that the learner will still fail to find an $\left(\frac{\epsilon}{2(1-\gamma)}\right)$-optimal policy with probability $1/4$ on both MDPs. ■

E.2 Property of $\hat{Q}(s, a)$ sampled from Algorithm 4

To prepare for the analysis that follows, we first show that the $\hat{Q}(s, a)$ sampled from Algorithm 4 is unbiased and has bounded variance.

Lemma E.2.1. $\mathbb{E}\left[\hat{Q}^\pi(s, a)\right] = Q^\pi(s, a)$, *Var*$(\hat{Q}^\pi(s, a)) \leq \frac{\gamma}{(1-\gamma)^2} + \frac{\sigma^2}{1-\gamma}$. *The bound for variance is tight.*

Proof of Lemma E.2.1. In the following, we treat (s_0, a_0) as deterministic.

$$\begin{aligned}
\mathbb{E}\left[\hat{Q}^\pi(s_0, a_0)\right] =& \sum_{k=0}^{\infty} \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t) \middle| T = k\right] P(T = k) \quad \text{(by law of total expectation)} \\
=& \sum_{k=0}^{\infty} \mathbb{E}\left[\sum_{t=0}^{k} r(s_t, a_t)\right] (1-\gamma)\gamma^k \quad \text{(each } r(s, a) \text{ is independent of } T\text{)} \\
=& (1-\gamma) \sum_{k=0}^{\infty} \frac{\gamma^k}{1-\gamma} \mathbb{E}\left[r(a_k, s_k)\right] \\
=& Q^\pi(s_0, a_0)
\end{aligned}$$

Now, we upperbound the variance. Let $\bar{r}(s, a) := r(s, a) - e(s, a)$ be the expected reward over the zero-mean noise. Because the zero-mean noise is independent of state transition, we observe that:

$$\begin{aligned}
\mathbb{E}\left[r(s, a)\right] =& \mathbb{E}\left[\bar{r}(s, a)\right] \\
\mathbb{E}\left[r(s, a)^2\right] =& \mathbb{E}\left[(\bar{r}(s, a) + e(s, a))^2\right] = \mathbb{E}\left[\bar{r}(s, a)^2\right] + \mathbb{E}\left[e(s, a)^2\right] \leq \mathbb{E}\left[\bar{r}(s, a)^2\right] + \sigma^2 \\
\mathbb{E}\left[r(s_i, a_i) r(s_j, a_j)\right] =& \mathbb{E}\left[(\bar{r}(s_i, a_i) + e(s_i, a_i))(\bar{r}(s_j, a_j) + e(s_j, a_j))\right] = \mathbb{E}\left[\bar{r}(s_i, a_i)\bar{r}(s_j, a_j)\right],
\end{aligned}$$

for $i \neq j$.

Given the above observations, we can bound the variance as follows

$$
\begin{aligned}
& \mathrm{Var}(\hat{Q}^\pi(s_0,a_0)) \\
\leq\ & \sigma^2 + \mathbb{E}\left[(\hat{Q}^\pi(s_0,a_0) - \bar{r}(s_0,a_0))^2\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \quad \text{(separate the variance of } r(s_0,a_0)) \\
=\ & \sigma^2 + \sum_{k=1}^{\infty}(1-\gamma)\gamma^k \mathbb{E}\left[\left(\sum_{t=1}^{k} r(s_t,a_t)\right)^2\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
=\ & \sigma^2 + \sum_{k=1}^{\infty}(1-\gamma)\gamma^k \left(\sum_{t=1}^{k}\mathbb{E}\left[r(s_t,a_t)^2\right] + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k}\mathbb{E}\left[r(s_i,a_i)r(s_j,a_j)\right]\right) - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
=\ & \sigma^2 + \sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[r(s_t,a_t)^2\right] + 2\sum_{i=1}^{\infty}\sum_{j=i+1}^{\infty}\gamma^j\mathbb{E}\left[r(s_i,a_i)r(s_j,a_j)\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
\leq\ & \frac{\sigma^2}{1-\gamma} + \sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)^2\right] + 2\sum_{i=1}^{\infty}\sum_{j=i+1}^{\infty}\gamma^j\mathbb{E}\left[\bar{r}(s_i,a_i)\bar{r}(s_j,a_j)\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
\leq\ & \frac{\sigma^2}{1-\gamma} + \sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right] + 2\sum_{i=1}^{\infty}\sum_{j=i+1}^{\infty}\gamma^j\mathbb{E}\left[\bar{r}(s_i,a_i)\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
=\ & \frac{\sigma^2}{1-\gamma} + \sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right] + 2\sum_{i=1}^{\infty}\frac{\gamma^{i+1}}{1-\gamma}\mathbb{E}\left[\bar{r}(s_i,a_i)\right] - \left(\mathbb{E}\left[\hat{Q}^\pi(s_0,a_0)\right] - \bar{r}(s_0,a_0)\right)^2 \\
=\ & \frac{\sigma^2}{1-\gamma} + \frac{1+\gamma}{1-\gamma}\sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right] - \left(\sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right]\right)^2 \\
=\ & -\left(\sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right] - \frac{1+\gamma}{2(1-\gamma)}\right)^2 + \frac{(1+\gamma)^2}{4(1-\gamma)^2} + \frac{\sigma^2}{1-\gamma} \\
\leq\ & -\left(\sum_{t=1}^{\infty}\gamma^t - \frac{1+\gamma}{2(1-\gamma)}\right)^2 + \frac{(1+\gamma)^2}{4(1-\gamma)^2} + \frac{\sigma^2}{1-\gamma} = \frac{\gamma}{(1-\gamma)^2} + \frac{\sigma^2}{1-\gamma}
\end{aligned}
$$

The last line is because:

$$\sum_{t=1}^{\infty}\gamma^t\mathbb{E}\left[\bar{r}(s_t,a_t)\right] \leq \sum_{t=1}^{\infty}\gamma^t = \frac{\gamma}{1-\gamma} \leq \frac{1+\gamma}{2(1-\gamma)}.$$

The equality can be reached by the following reward setting: let $P(1 = \bar{r}(s_1,a_1) = \cdots = \bar{r}(s_t,a_t) = \cdots) = 1$ and therefore is tight. ■

E.3 Proofs for Section 6.3.

Lemma E.3.1 (Lemma 6.3.2)**.** *Suppose the adversarial rewards are bounded in* $[0,1]$*, and in a particular iteration* t*, the adversary contaminates* $\epsilon^{(t)}$ *fraction of the episodes, then given* M *episodes, it is guaranteed that with probability at least* $1-\delta$*,*

$$\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a)-\phi(s,a)^\top w^{(t)}\right)^2\right]\leq 4\left(W^2+WH\right)\left(\epsilon^{(t)}+\sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right).$$

where $H=(\log\delta-\log M)/\log\gamma$ *is the effective horizon.*

Proof of Lemma E.3.1. First of all, observe that since the adversarial reward is bounded in $[0,1]$, with probability $1-\delta$, the $\hat{Q}(s,a)$ estimates collected in the adversarial episodes are bounded by $H:=(\log\delta-\log M)/\log\gamma$.

Conditioned on the above event, consider three loss functions $\hat{f}$, $f^\dagger$ and f, representing the loss w.r.t. clean data, corrupted data and underlying distribution respectively, i.e.

$$\hat{f} = \frac{1}{M}\sum_{i=1}^{M}(y_i-x_i^\top w)^2 \tag{E.4}$$

$$f^\dagger = \frac{1}{M}\left[\sum_{i\in C}(y_i^\dagger-x_i^{\dagger\top}w)^2+\sum_{i\notin C}(y_i-x_i^\top w)^2\right] \tag{E.5}$$

$$f = \mathbb{E}(y_i-x_i^\top w)^2 \tag{E.6}$$

Then, for all w, we can make the following decomposition

$$||\nabla_w f^\dagger-\nabla_w f||\leq||\nabla_w f^\dagger-\nabla_w\hat{f}||+||\nabla_w\hat{f}-\nabla_w f||. \tag{E.7}$$

We next bound each of the two terms in equation E.7. For the first term,

$$\|\nabla_w f^\dagger-\nabla_w\hat{f}\| \tag{E.8}$$

$$= \left\|\frac{2}{M}\sum_{i\in C}\left[(x_i^\dagger x_i^{\dagger\top}-x_i x_i^\top)w+(y_i^\dagger x_i^\dagger-y_i x_i)\right]\right\| \tag{E.9}$$

$$\leq 4\left(W+H\right)\epsilon^{(t)} \tag{E.10}$$

where the last step uses the fact that $|C|/M\leq\epsilon^{(t)}$, and $\|x\|\leq 1$, $|y^\dagger|\leq H$ and $\|w\|\leq W$. For the

second term

$$
\begin{aligned}
& ||\nabla_w \hat{f} - \nabla_w f|| && \text{(E.11)} \\
\leq\ & 2\left\|\left(\mathbb{E}[xx^\top] - \frac{1}{M}\sum_{i=1}^{M} x_i x_i^\top\right) w - \left(\mathbb{E}[yx] - \frac{1}{M}\sum_{i=1}^{M} y_i x_i\right)\right\| && \text{(E.12)} \\
\leq\ & 2\left(\frac{2}{3M}\log\frac{4d}{\delta} + \sqrt{\frac{2}{M}\log\frac{4d}{\delta}}\right) W + 2\sqrt{\frac{2}{M}\log\frac{4d}{\delta}}\cdot 2H && \text{(E.13)} \\
\leq\ & 4\sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\,(W+H)\,, \text{for } M \geq 2\log\frac{4d}{\delta}. && \text{(E.14)}
\end{aligned}
$$

where in step (E.13) we apply Matrix Bernstein inequality [160] on the first term and vector Hoeffding's inequality [80] on the second term. The constant in Corollary 7 of [80] is instantiated to be $c = 1$, because boundedness means we always have condition 2 in Lemma 2 of [80]. This condition is all we need throughout the proof for the vector Hoeffding.

Now, let M be sufficiently large, and instantiate w to be w^t, i.e. the constrained linear regression solution w.r.t $f^\dagger$, then our result above implies that for any vector v such that $||w + v|| \leq W$, we have $\nabla_w f^\dagger (w^t)^\top v/||v|| \geq 0$, and thus

$$
\nabla_w f(w^t)^\top v/||v|| \geq -4\,(W+H)\left(\epsilon^{(t)} + \sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right) \tag{E.15}
$$

which by Lemma B.8 of [50] implies that

$$
\epsilon^{(t)}_{stat} \leq 4\left(W^2 + HW\right)\left(\epsilon^{(t)} + \sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right), \text{w.p. } 1 - 2\delta. \tag{E.16}
$$

■

Theorem E.3.1 (Theorem 6.3.1)**.** *Under assumptions 7.2.1 (linear Q function) and 6.2.2 (reset distribution with small κ), given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2 guarantees with a poly$(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination with adversarial rewards bounded in $[0, 1]$, we have*

$$
\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \leq \tilde{O}\left(\max\left[\alpha, W\sqrt{\frac{|\mathcal{A}|\kappa\epsilon}{(1-\gamma)^3}}\right]\right)
$$

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

Proof of Theorem E.3.1. First note that $\epsilon_{stat} = \mathbb{E}_{s,a\sim d^{(t)}}[\left(\phi(s,a)^\top(w^{(t)} - w^*)\right)^2] \leq 4W^2$, because

$\|\phi(s,a)\| \leq 1$ and $\|w^{(t)}\|, \|w^*\| \leq W$. As a result, the high probability bound in Lemma 6.3.2 can be ready translate into an expected bound:

$$\mathbb{E}\left[\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a) - \phi(s,a)^\top w^{(t)}\right)^2\right]\right] \leq 4\left(W^2 + HW\right)\left(\epsilon^{(t)} + \sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right) + 8\delta W^2 \tag{E.17}$$

where δ becomes a free parameter. Plugging this into Lemma 6.3.1, we get

$$\begin{aligned}
& \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\{V^*(\mu_0) - V^{(t)}(\mu_0)\}\right] \\
\leq\ & \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{4|\mathcal{A}|\kappa\epsilon^{(t)}_{stat}}{(1-\gamma)^3}} \\
\leq\ & \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{16|\mathcal{A}|\kappa\left(\left(W^2+HW\right)\left(\epsilon^{(t)} + \sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right) + 2\delta W^2\right)}{(1-\gamma)^3}} \\
\leq\ & \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{16|\mathcal{A}|\kappa\left(\left(W^2+HW\right)\sqrt{\frac{8}{M}\log\frac{4d}{\delta}} + 2\delta W^2\right)}{(1-\gamma)^3}} \\
& \quad + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{16|\mathcal{A}|\kappa\left(W^2+HW\right)\epsilon^{(t)}}{(1-\gamma)^3}} \\
\leq\ & \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \sqrt{\frac{16|\mathcal{A}|\kappa\left(\left(W^2+HW\right)\sqrt{\frac{8}{M}\log\frac{4d}{\delta}} + 2\delta W^2\right)}{(1-\gamma)^3}} + \sqrt{\frac{16|\mathcal{A}|\kappa\left(W^2+HW\right)\epsilon}{(1-\gamma)^3}}
\end{aligned}$$

where the last step is by Cauchy Schwarz and the fact that the attacker only has ϵ budget to distribute, which implies that $\sum_{t=1}^{T}\epsilon^{(t)} = T\epsilon$. Setting

$$T = \frac{2W^2\log|\mathcal{A}|}{\alpha^2(1-\gamma)^2} \tag{E.18}$$

$$\delta = \frac{\alpha^2(1-\gamma)^3}{32W^2|\mathcal{A}|\kappa} \tag{E.19}$$

$$M = \frac{512|\mathcal{A}|^2W^2(W+H)^2\kappa^2}{\alpha^4(1-\gamma)^6}\log\frac{4d}{\delta}, \tag{E.20}$$

we get

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\{V^*(\mu_0) - V^{(t)}(\mu_0)\}\right] \leq 3\alpha + \sqrt{\frac{16|\mathcal{A}|\kappa\left(W^2+HW\right)\epsilon}{(1-\gamma)^3}}. \tag{E.21}$$

with sample complexity

$$TM = \frac{1024|\mathcal{A}|^2 \log|\mathcal{A}| W^4 (W+H)^2 \kappa^2}{\alpha^6 (1-\gamma)^8} \log \frac{128 W^2 |\mathcal{A}| \kappa d}{\alpha^2 (1-\gamma)^3}. \tag{E.22}$$

■

Next, we prove this tighter version of Theorem 6.3.1 in the special case of tabular MDPs.

Corollary E.3.1 (Corollary 6.3.1)**.** *Given a tabular MDP and assumption 6.2.2, given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2 guarantees with a $poly(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination with adversarial rewards bounded in $[0,1]$, we have*

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \leq \tilde{O}\left(\max\left[\alpha, \sqrt{\frac{|\mathcal{A}|\kappa\epsilon}{(1-\gamma)^5}}\right]\right) \tag{E.23}$$

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

The proof follows the exact same structure as the proof of Theorem E.3.1, but with a tighter robustness bound of linear regression.

Lemma E.3.2. *Assume a tabular MDP and the adversarial rewards are bounded in $[0,1]$, and in a particular iteration t, the adversary contaminates $\epsilon^{(t)}$ fraction of the episodes, then given M episodes, it is guaranteed that with probability at least $1-\delta$,*

$$\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a) - \phi(s,a)^\top w^{(t)}\right)^2\right] \leq H^2\epsilon^{(t)} + 3\left(W^2 + WH\right)\sqrt{\frac{\log 1/\delta}{M}}. \tag{E.24}$$

where $H = (\log\delta - \log M)/\log\gamma$ is the effective horizon.

Proof of Lemma E.3.2. The proof is largely based on Lemma G.1 of [2]. We assumed that the constrained linear regression problem is solved using Projected Online Gradient Descent [198] on the sequence of loss functions $(w^\top\phi_i - \hat{Q}_i)^2$, i.e.

$$w_{i+1} = \text{Proj}_{\|w\|\leq W}\left(w_i - \eta_i(w_i^\top\phi_i - \hat{Q}_i)\phi_i\right), \text{ for all } i \in [M], \tag{E.25}$$

where $\eta_i = W^2/((W+H)\sqrt{N})$ and we set $w^{(t)} = \frac{1}{M}\sum_{i=1}^M w_i$.

Using the projected online gradient descent regret guarantee, we have that:

$$\sum_{i\in C}(w_i^\top \phi_i^\dagger - \hat{Q}_i^\dagger)^2 + \sum_{i\notin C}(w_i^\top \phi_i - \hat{Q}_i)^2 \le \sum_{i\in C}(w^{\star\top} \phi_i^\dagger - \hat{Q}_i^\dagger)^2 + \sum_{i\notin C}(w^{\star\top} \phi_i - \hat{Q}_i)^2 + \underbrace{W(W+H)}_{:=Q}\sqrt{M}. \tag{E.26}$$

which implies

$$\sum_{i\in[M]}(w_i^\top \phi_i - \hat{Q}_i)^2 - \sum_{i\in[M]}(w^{\star\top} \phi_i - \hat{Q}_i)^2 \tag{E.27}$$

$$\le \sum_{i\in C}\left[(w^{\star\top} \phi_i^\dagger - \hat{Q}_i^\dagger)^2 - (w^{\star\top} \phi_i - \hat{Q}_i)^2\right] - \sum_{i\in C}\left[(w_i^\top \phi_i^\dagger - \hat{Q}_i^\dagger)^2 - (w_i^\top \phi_i - \hat{Q}_i)^2\right] + Q\sqrt{M}. \tag{E.28}$$

We now want to show by induction that $w_i^\top \phi \in [0, H]$ for any i and ϕ. $w_0 = 0$ which satisfies $w_0^\top \phi \in [0, H]$. Now, assume that $w_i^\top \phi \in [0, H]$, we want to show $w_{i+1}^\top \phi \in [0, H]$. In a tabular MDP, ϕ is an one-hot vector, and thus for $\phi \neq \phi_i$, $w_{i+1}^\top \phi = w_i^\top \phi \in [0, H]$. If $\phi = \phi_i$, then

$$w_{i+1}^\top \phi = \left(w_i - \eta_i(w_i^\top \phi_i - \hat{Q}_i)\phi_i\right)^\top \phi_i \le (1-\eta_i)w_i^\top \phi_i + \eta\hat{Q}_i \in [0, H] \tag{E.29}$$

because both $w_i \top \phi_i$ (by induction hypothesis) and $\hat{Q}_i$ (by assumption on bounded attack) are in $[0, H]$. Therefore, we have shown that $w_i^\top \phi \in [0, H]$ for any i and ϕ. Then, (E.28) implies that

$$\sum_{i\in[M]}(w_i^\top \phi_i - \hat{Q}_i)^2 \le \sum_{i\in[M]}(w^{\star\top} \phi_i - \hat{Q}_i)^2 + 2H^2\epsilon^{(t)}M + Q\sqrt{M}. \tag{E.30}$$

Denote random variable $z_i = (\theta_i \cdot x_i - y_i)^2 - (\theta^\star \cdot x_i - y_i)^2$. Denote $\mathbb{E}_i$ as the expectation taken over the randomness at step i conditioned on all history $t = 1$ to $i - 1$. Note that for $\mathbb{E}_i[z_i]$, we have:

$$\mathbb{E}_i\left[(\theta_i \cdot x - y)^2 - (\theta^\star \cdot x - y)^2\right] \tag{E.31}$$

$$= \mathbb{E}_i\left[(\theta_i \cdot x - \mathbb{E}[y|x])^2\right] \tag{E.32}$$

$$- \mathbb{E}_i\left[2(\theta_i \cdot x - \mathbb{E}[y|x])(\mathbb{E}[y|x] - y) - (\theta^\star \cdot x - \mathbb{E}[y|x])^2 + 2(\theta^\star \cdot x - \mathbb{E}[y|x])(\mathbb{E}[y|x] - y))\right] \tag{E.33}$$

$$= \mathbb{E}_i\left[(\theta_i \cdot x - \mathbb{E}[y|x])^2 - (\theta^\star \cdot x - \mathbb{E}[y|x])^2\right], \tag{E.34}$$

where we use $\mathbb{E}[\mathbb{E}[y|x] - y] = 0$. Also for $|z_i|$, we can show that for $|z_i|$ we have:

$$|z_i| = |(\theta_i \cdot x_i - \theta^\star \cdot x_i)(\theta_i \cdot x_i + \theta^\star \cdot x_i - 2y_i)| \le W(2W + 2H) = 2W(W + H). \tag{E.35}$$

Note that z_i forms a Martingale difference sequence. Using Azuma-Hoeffding's inequality, we have

that with probability at least $1-\delta$:

$$\left|\sum_{i=1}^{M} z_i - \sum_{i=1}^{M} \mathbb{E}_i\left[(\theta_i \cdot x - \mathbb{E}[y|x])^2 - (\theta^\star \cdot x - \mathbb{E}[y|x])^2\right]\right| \leq 2W(W+H)\sqrt{\ln(1/\delta)M}, \tag{E.36}$$

which implies that:

$$\sum_{i=1}^{M} \mathbb{E}_i\left[(\theta_i \cdot x - \mathbb{E}[y|x])^2 - (\theta^\star \cdot x - \mathbb{E}[y|x])^2\right] \leq \sum_{i=1}^{M} z_i + 2W(W+H)\sqrt{\ln(1/\delta)M} \tag{E.37}$$

$$\leq 2W(W+H)\sqrt{\ln(1/\delta)M} + 2H^2 M\epsilon^{(t)} + Q\sqrt{M}. \tag{E.38}$$

Apply Jensen's inequality on the LHS of the above inequality, we have that:

$$\mathbb{E}\left(\hat{\theta} \cdot x - \mathbb{E}[y|x]\right)^2 \leq \mathbb{E}\left(\theta^\star \cdot x - \mathbb{E}[y|x]\right)^2 + 2H^2\epsilon^{(t)} + (Q + 2W(W+H))\sqrt{\frac{\ln(1/\delta)}{M}}. \tag{E.39}$$

■

E.4 A modified analysis for SEVER

In this section, we will derive an expected error bound for SEVER [50] when applied to a linear regression problem. The high level idea is to use the results of [53] to show the existence of a stable set and change the probabilistic argument in [50] to an expectation argument. We note that the original result in [50] works only with probability $9/10$, and there is no direct way of translating it into either a high-probability argument or an expectation argument.

In the following, we consider a robust linear regression problem. We observe pairs $(X_i, Y_i) \in \mathbb{R}^d \times \mathbb{R}$ for $i \in [n]$, where X_i's are drawn i.i.d. from a distribution D_x and $Y_i = w^{*\top} X_i + e_i$ for some unknown $w^* \in \mathbb{R}^d$. e_i's are i.i.d, noise from some distribution $D_{e|x}$. Note that here e_i and X_i may not be independent. We let D_{xy} be the joint distribution of (X, Y). Let $f_i(w) = (Y_i - w^\top X_i)^2$. Given a multiset of observations $\{(X_i, Y_i)\}_{i=1}^n$, our goal is to minimize the objective function

$$\bar{f}(w) = \mathbb{E}_{(X,Y)\sim D_{xy}}[(Y - w^\top X)^2] \tag{E.40}$$

on a convex feasible set $\mathcal{H}$. Let $r := \max_{w\in\mathcal{H}}\|w\|$ be the ℓ_2-radius of $\mathcal{H}$. In the following, we use $\|\cdot\|$ to denote the spectral norm of a matrix and the 2-norm of a vector. We use Cov to denote the covariance matrix of a random vector: $\mathrm{Cov}[X] = \mathbb{E}\left[(X - \mathbb{E}X)(X - \mathbb{E}X)^\top\right]$. When S is a set, we use $\mathbb{E}_S$ and Cov_S to denote the expectation and covariance over the empirical distribution on S. We allow for an ϵ-fraction of the observations to be arbitrary outliers. The ϵ-corruption model is defined in more detail in the Appendix A of [50].

Due to our application, we make assumptions on the linear regression model that is slight different from Assumption E.1 in [50]:

Assumption E.4.1. *Given the model for linear regression described above, assume the following conditions for $D_{e|x}$ and D_x:*

- $\mathbb{E}[e|X] = 0$;
- $\mathbb{E}\left[e^2\middle|X\right] \leq \xi$;
- $\mathbb{E}_{X\sim D_x}[XX^\top] \preceq s^2 I$ *for some* $s > 0$;
- *There is a constant* $C > 0$, *such that for all unit vectors* v, $\mathbb{E}_{X\sim D_x}[\langle v, X\rangle^4] \leq Cs^4$.

In [50], the noise term e and X are independent. We weaken the assumption on e and bound its first and second moments conditional on X.

Stability with subgaussian rate

We first note that the gradient of f_i, $\nabla f_i(w)$ has bounded covariance matrix. We will show this by following the proof of Lemma E.3 in [50], but make minor changes as we do not assume e and X are independent:

Lemma E.4.1 (A variant of Lemma E.3 in [50])**.** *Suppose D_{xy} satisfies the conditions of Assumption E.4.1. Then for all unit vectors $v \in \mathbb{R}^d$, we have*

$$v^\top \operatorname{Cov}_{(X_i,Y_i)\sim D_{xy}}[\nabla f_i(w)]v \leq 4s^2\xi + 4Cs^4\|w^* - w\|^2. \tag{E.41}$$

Proof of Lemma E.4.1. We first note that $f_i(w) = (Y_i - w^\top X_i)^2$ and $\nabla f_i(w) = -2((w^* - w)^\top X_i + e_i)X_i$. By the property of conditional expectation, for any function $g(\cdot), h(\cdot)$, we have $\mathbb{E}[g(X)h(e)] = \mathbb{E}_X\left[\mathbb{E}_{h(e)|X}[g(X)h(e)|X]\right] = \mathbb{E}_X\left[g(X)\mathbb{E}_{h(e)|X}[h(e)|X]\right]$. Then

$$\begin{aligned}
&\mathbb{E}\left[\nabla f_i(w)\nabla f_i(w)^\top\right] = 4\mathbb{E}\left[((w^* - w)^\top X_i + e_i)^2 X_i X_i^\top\right] && \text{(E.42)}\\
=\ &4\mathbb{E}\left[((w^* - w)^\top X_i)^2 X_i X_i^\top\right] + 4\mathbb{E}\left[e_i^2 X_i X_i^\top\right] + 4\mathbb{E}\left[2(w^* - w)^\top X_i e_i X_i X_i^\top\right] && \text{(E.43)}\\
=\ &4\mathbb{E}\left[((w^* - w)^\top X_i)^2 X_i X_i^\top\right] + 4\mathbb{E}\left[X_i X_i^\top \mathbb{E}\left[e_i^2\middle|X_i\right]\right] && \text{(E.44)}
\end{aligned}$$

By Assumption E.4.1, for all unit vectors $v \in \mathbb{R}^d$, we have

$$\begin{aligned}
v^\top\mathbb{E}\left[((w^* - w)^\top X_i)^2 X_i X_i^\top\right] v \ &= \ \mathbb{E}\left[((w^* - w)^\top X_i)^2 (v^\top X_i)^2\right] && \text{(E.45)}\\
&\leq \ \sqrt{\mathbb{E}\left[((w^* - w)^\top X_i)^4\right]\mathbb{E}\left[(v^\top X_i)^4\right]} && \text{(E.46)}\\
&\leq \ Cs^4\|w^* - w\|^2 && \text{(E.47)}
\end{aligned}$$

and

$$v^\top \mathbb{E}\left[X_i X_i^\top \mathbb{E}\left[e_i^2 \middle| X_i\right]\right] v \le \xi v^\top \mathbb{E}\left[X_i X_i^\top\right] v \le s^2 \xi \tag{E.48}$$

Thus for all unit vectors $v \in \mathbb{R}^d$, we have

$$v^\top \operatorname{Cov}_{(X_i,Y_i)\sim D_{xy}}[\nabla f_i(w)] v \le v^\top \mathbb{E}\left[\nabla f_i(w) \nabla f_i(w)^\top\right] v \le 4s^2\xi + 4Cs^4 \|w^* - w\|^2. \tag{E.49}$$

■

We then use the following Theorem E.4.1 to show that the observations $f_1, \ldots, f_n$ satisfies the Assumption E.4.2 with high probability:

Theorem E.4.1 (Theorem 1.4 in [53]). *Fix any $0 < \tau < 1$. Let S be a multiset of n i.i.d. samples from a distribution on $\mathbb{R}^d$ with mean μ and covariance Σ. Let $\epsilon' = \tilde{C}\left(\log(1/\tau)/n + \epsilon\right) = O(1)$, for some constant $\tilde{C} > 0$. Then, with probability at least $1 - \tau$, there exists a subset $S' \subseteq S$ such that $|S'| \ge (1-\epsilon')n$ and for every $S'' \subseteq S'$ with $|S''| \ge (1-2\epsilon')|S'|$, the following conditions hold: (i) $\|\mu_{S''} - \mu\| \le \sqrt{\|\Sigma\|}\delta$, and (ii) $\|\overline{\Sigma}_{S''} - \|\Sigma\| I\| \le \|\Sigma\| \delta^2/(2\epsilon')$, for $\delta = O\left(\sqrt{(d\log d)/n} + \sqrt{\epsilon} + \sqrt{\log(1/\tau)/n}\right)$.*

where $\mu_{S''} = \frac{1}{|S''|}\sum_{x\in S''} x$ and $\overline{\Sigma}_{S''} = \frac{1}{|S''|}\sum_{x\in S''}(x-\mu)(x-\mu)^\top$.

We use a notion of stability similar to that in [50] but allow the parameter to depend on the confidence level and sample size:

Assumption E.4.2 (A variant of Assumption B.1 in [50]). *Fix $0 < \epsilon < 1/2$. With probability at least $1 - \tau$, there exists an unknown set $I_{good} \subseteq [n]$ with $|I_{good}| \ge (1-\epsilon)n$ of "good" functions $\{f_i\}_{i\in I_{good}}$ and parameters σ, $\alpha(\epsilon, n, \tau), \beta(\epsilon, n, \tau) \in \mathbb{R}_+$ such that for all $w \in \mathcal{H}$:*

$$\left\| \frac{1}{|I_{good}|} \sum_{i\in I_{good}} \nabla f_i(w) - \nabla \bar{f}(w) \right\| \le \sigma \alpha(\epsilon, n, \tau) \tag{E.50}$$

and

$$\left\| \frac{1}{|I_{good}|} (\nabla f_i(w) - \nabla \bar{f}(w))(\nabla f_i(w) - \nabla \bar{f}(w))^\top \right\| \le \sigma^2 \beta(\epsilon, n, \tau) \tag{E.51}$$

We can then equivalently write Theorem E.4.1 as the following Proposition:

Proposition E.4.1. Given a linear regression model $f_i(w) = (Y_i - w^\top X_i)^2$ satisfying Assumption E.4.1, $X_i \sim D_x$, $D_e \sim D_e$, with probability at least $1-\tau$, $\{f_i\}_{i\in[n]}$ satisfies Assumption E.4.2 with $\sigma = 2s\sqrt{\xi} + 2\sqrt{C}s^2\|w^* - w\|$, $\alpha(\epsilon, n, \tau) = O\left(\sqrt{(d\log d)/n} + \sqrt{\epsilon} + \sqrt{\log(1/\tau)/n}\right)$ and $\beta(\epsilon, n, \tau) = \left(\frac{d\log d}{\log(1/\tau) + n\epsilon} + 1\right)$.

Proof of Proposition E.4.1. By Theorem E.4.1 and Lemma E.4.1, with probability at least $1-\tau$, there exist an unknown set $I_{good} \subseteq [n]$ with $|I_{good}| \geq (1-\epsilon')n$, s.t.

$$\begin{aligned}
& \left\| \frac{1}{|I_{good}|} (\nabla f_i(w) - \nabla \bar{f}(w))(\nabla f_i(w) - \nabla \bar{f}(w))^\top \right\| && \text{(E.52)} \\
\leq & \left\| \frac{1}{|I_{good}|} (\nabla f_i(w) - \nabla \bar{f}(w))(\nabla f_i(w) - \nabla \bar{f}(w))^\top - \|\mathrm{Cov}_{f \in p^*}[\nabla f]\| \, I \right\| + \|\mathrm{Cov}_{f \in p^*}[\nabla f]\| && \text{(E.53)} \\
\leq & \left(4s^2\xi + 4Cs^4\|w^* - w\|^2\right) O\left(\frac{d \log d}{\log(1/\tau) + n\epsilon} + 1\right) && \text{(E.54)} \\
\leq & \left(2s\sqrt{\xi} + 2\sqrt{C}s^2\|w^* - w\|\right)^2 O\left(\frac{d \log d}{\log(1/\tau) + n\epsilon} + 1\right) =: \sigma^2 \beta(\epsilon, n, \tau). && \text{(E.55)}
\end{aligned}$$

$$\|\nabla \hat{f}(w) - \nabla \bar{f}(w)\| \leq \sigma O\left(\sqrt{(d \log d)/n} + \sqrt{\epsilon} + \sqrt{\log(1/\tau)/n}\right) =: \sigma \alpha(\epsilon, n, \tau). \quad \text{(E.56)}$$

■

The expected optimality gap

In order to prove the expected optimality gap, we first state a slightly modified version of the main theorem in [50] by specifying the probability of success;

Theorem E.4.2 (Theorem B.2 in [50])**.** *Let the corruption level $\epsilon \in [0, c]$, for some small enough $c > 0$. Suppose that the functions $f_1, \ldots, f_n, \bar{f} : \mathcal{H} \to \mathbb{R}$ are bounded below, and that Assumption E.4.2 is satisfied. Then SEVER applied to $f_1, \ldots, f_n$ returns a point $w \in \mathcal{H}$ that, fix $p \geq \sqrt{\epsilon}$, with probability at least $1-p$, is a $O\left(\sigma\left(\alpha(\epsilon, n, \tau) + \sqrt{\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)}\sqrt{\epsilon/p}\right)\right)$-approximate critical point of $\bar{f}$, i.e. for all unit vectors v where $w + \lambda v \in \mathcal{H}$ for arbitrarily small positive λ, we have that $v \cdot \nabla f(w) \geq -O\left(\sigma\left(\alpha(\epsilon, n, \tau) + \sqrt{\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)}\sqrt{\epsilon/p}\right)\right)$.*

if $\bar{f}$ is convex, we have the following optimality gap. Recall r is the radius of the convex set $\mathcal{H}$ where w^* belongs.

Corollary E.4.1 (Corollary B.3 in [50])**.** *Let the corruption level $\epsilon \in [0, c]$, for some small enough $c > 0$. For functions $f_1, \ldots, f_n : \mathcal{H} \to \mathbb{R}$, suppose that Assumption E.4.2 holds and that $\mathcal{H}$ is convex. Then, fix $p \geq \sqrt{\epsilon}$, with probability at least $1-p$, the output of SEVER satisfies the following: if $\bar{f}$ is convex, the algorithm finds a $w \in \mathcal{H}$ such that $\bar{f}(w) - \bar{f}(w^*) = O\left(r\sigma\left(\alpha(\epsilon, n, \tau) + \sqrt{\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)}\sqrt{\epsilon/p}\right)\right)$*

Given Theorem E.4.1, we can prove the following expected optimality gap:

Theorem E.4.3 (expected optimality gap)**.** *Let the corruption level $\epsilon \in [0, c]$, for some small enough $c > 0$. Let $\mathcal{H}$ be a convex set. Given n samples from a linear regression model $f(w) = (Y - w^\top X)^2$ satisfying*

Assumption E.4.1, where $X \sim D_x$, $e \sim D_e$, $Y = w^{*\top} X + e$ for some unknown $w^* \in \mathcal{H}$, SEVER will find a $w \in \mathcal{H}$, such that

$$\mathbb{E}\left[\bar{f}(w) - \bar{f}(w^*)\right] = O\left(\left(sr\sqrt{\xi} + s^2 r^2\right)\left(\tau + \sqrt{(d\log d)/n} + \sqrt{\epsilon} + \sqrt{\log(1/\tau)/n}\right)\right). \tag{E.57}$$

where the expectation above is over both the randomness of SEVER and (X_i, Y_i) pairs.

Proof of Theorem E.4.3. In the following, we use α and β as shorthands of $\alpha(\epsilon, n, \tau)$ and $\beta(\epsilon, n, \tau)$. We first show that $\bar{f}(w) - \bar{f}(w^*)$ is upper bounded:

$$\begin{aligned}
\bar{f}(w) - \bar{f}(w^*) &= \mathbb{E}_{X,Y}\left[(Y - w^\top X)^2 - (Y - w^{*\top} X)^2\right] && \text{(E.58)}\\
&= \mathbb{E}_{X,e}\left[(w^* - w)^\top X + e)^2 - e^2\right] && \text{(E.59)}\\
&= (w^* - w)^\top \mathbb{E}_X[XX^\top](w^* - w) \le s^2 (w - w^*)^2 \le 4s^2 r^2. && \text{(E.60)}
\end{aligned}$$

For some constant $M > 0$, define $x_1 := Mr\sigma\left(\alpha/\sqrt{\epsilon} + \sqrt{\alpha^2 + \beta}\right)\sqrt{\epsilon}$. Let A_1 be the event of $\{$Assumption E.4.2 holds$\}$. Let A_2 be the event of $\{$SEVER removes less than $(1 + 1/\sqrt{\epsilon})\epsilon n$ points$\}$. Let $A_3(p)$ be the event of $\left\{\bar{f}(w) - \bar{f}(w^*) > Mr\sigma\left(\alpha + \sqrt{\alpha^2 + \beta}\sqrt{\epsilon/p}\right)\right\}$. Then, $\forall 0 \le p < \sqrt{\epsilon}$

$$P(A_2, A_3(p) \mid A_1) = 0. \tag{E.61}$$

By Corollary E.4.1, $\forall \sqrt{\epsilon} \le p \le 1$

$$P(A_2, A_3(p) \mid A_1) \le p. \tag{E.62}$$

By Proposition E.4.1,

$$P(A_1) \ge 1 - \tau. \tag{E.63}$$

By Lemma E.4.3,

$$P(A_2 \mid A_1) \ge 1 - \sqrt{\epsilon}, \tag{E.64}$$

and thus

$$1 - P(A_1, A_2) = 1 - P(A_2 \mid A_1)P(A_1) \le \tau + \sqrt{\epsilon}. \tag{E.65}$$

Then, we have:

$$\begin{aligned}
&P\left(\bar{f}(w) - \bar{f}(w^*) > x_1/\sqrt{p} \mid A_1, A_2\right) && \text{(E.66)}\\
\le &P\left(A_3(p) \mid A_1, A_2\right) = P(A_2, A_3(p) \mid A_1)/P(A_2 \mid A_1) && \text{(E.67)}\\
\le &\begin{cases} 0 & 0 \le p < \sqrt{\epsilon} \\ \frac{p}{1 - \sqrt{\epsilon}} & \sqrt{\epsilon} \le p \le 1 \end{cases}. && \text{(E.68)}
\end{aligned}$$

Let $x = x_1/\sqrt{p}$, we have:

$$P\left(\bar{f}(w) - \bar{f}(w^*) > x \middle| A_1, A_2\right) \leq \begin{cases} 0 & x \geq x_1\epsilon^{-1/4} \\ \frac{1}{1-\sqrt{\epsilon}}\frac{x_1^2}{x^2} & x_1 \leq x < x_1\epsilon^{-1/4} \\ 1 & 0 \leq x < x_1 \end{cases}. \tag{E.69}$$

By Proposition E.4.1 and law of total expectation, we can bound the expected optimality gap by:

$$\begin{aligned}
\mathbb{E}\left[\bar{f}(w) - \bar{f}(w^*)\right] &\leq \mathbb{E}\left[\bar{f}(w) - \bar{f}(w^*) \middle| A_1, A_2\right] P(A_1, A_2) + 4s^2r^2(1 - P(A_1, A_2)) && \text{(E.70)} \\
&\leq \int_0^\infty P\left(\bar{f}(w) - \bar{f}(w^*) > x \middle| A_1, A_2\right) dx + 4s^2r^2(\tau + \sqrt{\epsilon}) && \text{(E.71)} \\
&= \int_0^{x_1} 1dx + \frac{1}{1-\sqrt{\epsilon}} \int_{x_1}^{x_1\epsilon^{-1/4}} \frac{x_1^2}{x^2} dx + 4s^2r^2(\tau + \sqrt{\epsilon}) && \text{(E.72)} \\
&\leq 2x_1 + 4s^2r^2(\tau + \sqrt{\epsilon}) && \text{(E.73)} \\
&= 2Mr\sigma\left(\alpha/\sqrt{\epsilon} + \sqrt{\alpha^2 + \beta}\right)\sqrt{\epsilon} + 4s^2r^2(\tau + \sqrt{\epsilon}) && \text{(E.74)} \\
&= O\left(\left(sr\sqrt{\xi} + s^2r^2\right)\left(\tau + \sqrt{(d\log d)/n} + \sqrt{\epsilon} + \sqrt{\log(1/\tau)/n}\right)\right) && \text{(E.75)}
\end{aligned}$$

Note that the expectation above is over both the randomness of SEVER and (X_i, Y_i) pairs. ■

Proof of Theorem E.4.2

In this proof, we mainly follow the steps in [50] but use our notion of stability in Assumption E.4.2. We also allow the success probability to vary, so that we can give an expected error bound later on.

We first restate the SEVER algorithm in Algorithm 7 and Algorithm 8. Throughout this proof we let I_{good} be as in Assumption E.4.2. We require the following three lemmas. Roughly speaking, the first states that with high probability, we will not remove too many points throughtout the process, the second states that on average, we remove more corrupted points than uncorrupted points, and the third states that at termination, and if we have not removed too many points, then we have reached a point at which the empirical gradient is close to the true gradient. Formally:

Lemma E.4.2. *If the samples satisfy Assumption E.4.2, $|S| \geq c_1 n$, and the filtering threshold is at least*

$$\frac{2(1-\epsilon)\sigma^2}{c_1 - 2\epsilon}\left(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)\right) \tag{E.76}$$

then if S' is the output of FILTER(S, τ, σ)*, we have that*

$$\mathbb{E}[|I_{\text{good}} \cap (S \backslash S')|] \leq \mathbb{E}[|([n] \backslash I_{\text{good}}) \cap (S \backslash S')|]. \tag{E.77}$$

Algorithm 7 SEVER$(f_{1:n}, \mathcal{L}, \sigma)$

Input: Sample functions $f_1, \ldots, f_n : \mathcal{H} \to \mathbb{R}$, bounded below on a closed domain $\mathcal{H}$, γ-approximate learner $\mathcal{L}$, and parameter $\sigma \in \mathbb{R}_+$.
Initialize $S \leftarrow \{1, \ldots, n\}$.
repeat
  $w \leftarrow \mathcal{L}(\{f_i\}_{i \in S})$. ▷ Run approximate learner on points in S.
  Let $\widehat{\nabla} = \frac{1}{|S|} \sum_{i \in S} \nabla f_i(w)$.
  Let $G = [\nabla f_i(w) - \widehat{\nabla}]_{i \in S}$ be the $|S| \times d$ matrix of centered gradients.
  Let v be the top right singular vector of G.
  Compute the vector τ of *outlier scores* defined via $\tau_i = \left((\nabla f_i(w) - \widehat{\nabla}) \cdot v \right)^2$.
  $S' \leftarrow S$
  $S \leftarrow$ FILTER(S', τ, σ) ▷ Remove some i's with the largest scores τ_i from S; see Algorithm 8.
until $S = S'$.
Return w.

Algorithm 8 FILTER(S, τ, σ)

Input: Set $S \subseteq [n]$, vector τ of outlier scores, and parameter $\sigma \in \mathbb{R}_+$.
If $\frac{1}{|S|} \sum_{i \in S} \tau_i \leq c_0 \cdot \sigma^2$, for some constant $c_0 > 1$, return S ▷ We only filter out points if the variance is larger than an appropriately chosen threshold.
Draw T from the uniform distribution on $[0, \max_i \tau_i]$.
Return $\{i \in S : \tau_i < T\}$.

Lemma E.4.3 (Revised version of Lemma 6 in [50])**.** *Assume filtering threshold is* $4(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau))\sigma^2$, $\epsilon \leq 1/16$, *then we have that for any given* $p \geq \sqrt{\epsilon}$, *with probability at least* $1 - p$, $n - |S| \leq (1 + 1/p)\epsilon n$ *when the filtering algorithm terminates.*

Lemma E.4.4. *If the samples satisfy Assumption E.4.2,* FILTER$(S, \tau, \sigma) = S$, *and* $n - |S| \leq (1 + 1/p)\epsilon n$, *for* $p \geq \sqrt{\epsilon}$, *then*

$$\left\| \nabla \bar{f}(w) - \frac{1}{|I_{\text{good}}|} \sum_{i \in S} \nabla f_i(w) \right\|_2 \leq O\left(\sigma \left(\alpha(\epsilon, n, \tau) + \sqrt{\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)} \sqrt{\epsilon / p} \right) \right) \tag{E.78}$$

Before we prove these lemmata, we show how together they imply Theorem E.4.2.

Proof of Theorem E.4.2 assuming Lemma E.4.3 and Lemma E.4.4. First, we note that the algorithm must terminate in at most n iterations. This is easy to see as each iteration of the main loop except for the last must decrease the size of S by at least 1.

It thus suffices to prove correctness. Note that Lemma E.4.3 says that with probability at least $1 - p$, SEVER will not remove too many points, this will allow us to apply Lemma E.4.4 to complete the proof, using the fact that w is a critical point of $\frac{1}{|I_{\text{good}}|} \sum_{i \in S} \nabla f_i(w)$. ■

Thus it suffices to prove these three lemmata.

Proof of Lemma E.4.2. Let $S_{\text{good}} = S \cap I_{\text{good}}$ and $S_{\text{bad}} = S \backslash I_{\text{good}}$. We wish to show that the expected number of elements thrown out of S_{bad} is at least the expected number thrown out of S_{good}. We note that our result holds trivially if $\textsc{Filter}(S, \tau, \sigma) = S$. Thus, we can assume that $\mathbb{E}_{i \in S}[\tau_i] \geq \frac{2(1-\epsilon)\sigma^2}{c_1 - 2\epsilon} \left(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)\right)$.

It is easy to see that the expected number of elements thrown out of S_{bad} is proportional to $\sum_{i \in S_{\text{bad}}} \tau_i$, while the number removed from S_{good} is proportional to $\sum_{i \in S_{\text{good}}} \tau_i$ (with the same proportionality). Hence, it suffices to show that $\sum_{i \in S_{\text{bad}}} \tau_i \geq \sum_{i \in S_{\text{good}}} \tau_i$.

We first note that since $\text{Cov}_{i \in I_{\text{good}}}[\nabla f_i(w)] \preceq \sigma^2 I$, we have that

$$\text{Cov}_{i \in S_{\text{good}}}[v \cdot \nabla f_i(w)] \leq \frac{1-\epsilon}{c_1 - \epsilon} \text{Cov}_{i \in I_{\text{good}}}[v \cdot \nabla f_i(w)] \quad (\text{since } |S_{\text{good}}| \geq \tfrac{c_1 - \epsilon}{1-\epsilon} |I_{\text{good}}|) \tag{E.79}$$

$$= \frac{1-\epsilon}{c_1 - \epsilon} \left(\frac{1}{|I_{\text{good}}|} \sum_{i \in I_{\text{good}}} (v \cdot (\nabla f_i(w) - \bar{f}(w)))^2 - (\bar{f}(w) - \mathbb{E}_{i \in I_{\text{good}}}[v \cdot \nabla f_i(w)])^2 \right) \tag{E.80}$$

$$\leq \frac{(1-\epsilon)\sigma^2}{c_1 - \epsilon} \left(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)\right) \quad (\text{By Assumption E.4.2}), \tag{E.81}$$

Let $\mu_{\text{good}} = \mathbb{E}_{i \in S_{\text{good}}}[v \cdot \nabla f_i(w)]$ and $\mu = \mathbb{E}_{i \in S}[v \cdot \nabla f_i(w)]$. Note that

$$\mathbb{E}_{i \in S_{\text{good}}}[\tau_i] = \text{Cov}_{i \in S_{\text{good}}}[v \cdot \nabla f_i(w)] + (\mu - \mu_{\text{good}})^2 \leq \frac{(1-\epsilon)\sigma^2}{c_1 - \epsilon} \left(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)\right) + (\mu - \mu_{\text{good}})^2 . \tag{E.82}$$

We now split into two cases.

Firstly, if

$$(\mu - \mu_{\text{good}})^2 \geq \frac{\epsilon}{c_1 - 2\epsilon} \frac{(1-\epsilon)\sigma^2}{c_1 - \epsilon} \left(\alpha(\epsilon, n, \tau)^2 + \beta(\epsilon, n, \tau)\right), \tag{E.83}$$

we let $\mu_{\text{bad}} = \mathbb{E}_{i \in S_{\text{bad}}}[v \cdot \nabla f_i(w)]$, and note that $|\mu - \mu_{\text{bad}}||S_{\text{bad}}| = |\mu - \mu_{\text{good}}||S_{\text{good}}|$. We then have

that

$$\mathbb{E}_{i\in S_{\rm bad}}[\tau_i] = \mathrm{Cov}_{i\in S_{\rm bad}}[v\cdot\nabla f_i(w)] + (\mu-\mu_{\rm bad})^2 \geq (\mu-\mu_{\rm bad})^2 \tag{E.84}$$

$$= (\mu-\mu_{\rm good})^2\left(\frac{|S_{\rm good}|}{|S_{\rm bad}|}\right)^2 \tag{E.85}$$

$$\geq \frac{|S_{\rm good}|}{|S_{\rm bad}|}\frac{c_1-\epsilon}{\epsilon}(\mu-\mu_{\rm good})^2 \quad \text{(because } |S_{\rm good}|\geq (c_1-\epsilon)n \text{ and } |S_{\rm bad}|\leq \epsilon n) \tag{E.86}$$

$$= \frac{|S_{\rm good}|}{|S_{\rm bad}|}\left(\frac{c_1-2\epsilon}{\epsilon}(\mu-\mu_{\rm good})^2 + (\mu-\mu_{\rm good})^2\right) \tag{E.87}$$

$$\geq \frac{|S_{\rm good}|}{|S_{\rm bad}|}\left(\frac{(1-\epsilon)\sigma^2}{c_1-\epsilon}\left(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)\right) + (\mu-\mu_{\rm good})^2\right) \quad \text{(by (E.83))} \tag{E.88}$$

$$\geq \frac{|S_{\rm good}|}{|S_{\rm bad}|}\mathbb{E}_{i\in S_{\rm good}}[\tau_i] \quad \text{(by (E.82))}. \tag{E.89}$$

Hence, $\sum_{i\in S_{\rm bad}}\tau_i \geq \sum_{i\in S_{\rm good}}\tau_i$.

On the other hand, if $(\mu-\mu_{\rm good})^2 \leq \frac{\epsilon}{c_1-2\epsilon}\frac{(1-\epsilon)\sigma^2}{c_1-\epsilon}\left(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)\right)$, then $\mathbb{E}_{i\in S_{\rm good}}[\tau_i] \leq \left(1+\frac{\epsilon}{c-2\epsilon}\right)\frac{(1-\epsilon)\sigma^2}{c_1-\epsilon}\left(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)\right) \leq \mathbb{E}_{i\in S}[\tau_i]/2$. Therefore $\sum_{i\in S_{\rm bad}}\tau_i \geq \sum_{i\in S_{\rm good}}\tau_i$ once again. This completes our proof. ■

Proof of Lemma E.4.3. Define the event

$$A = \{n-|S|\leq (1+1/p)\epsilon n\}, \tag{E.90}$$

and we want to lower-bound $P(A)$. Given that $\epsilon \leq 1/16$, the threshold is $4(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau))\sigma^2$ and $p\geq\sqrt{\epsilon}$, and conditioned on the event A, it can be verified that the asusumption of Lemma E.4.2 is satisfied. In particular, simple calculation shows that given $c_1 = 1-(1+1/p)\epsilon$, $\epsilon\leq 1/16$, $p\geq\sqrt{\epsilon}$, we have

$$4\sigma^2 \geq \frac{2(1-\epsilon)\sigma^2}{c_1-2\epsilon} \tag{E.91}$$

And Lemma E.4.2 implies that $|([n]\backslash I_{\rm good})\cap S| + |I_{\rm good}\backslash S|$ is a supermartingale. Since its initial size is at most ϵn, with probability at least $1-p$, it never exceeds $\epsilon n/p$, and therefore at the end of the algorithm, we must have that $n-|S|\leq \epsilon n + |I_{\rm good}\backslash S| \leq (1+1/p)\epsilon n$. ■

We now prove Lemma E.4.4.

Proof of Lemma E.4.4. We note that

$$\left\|\sum_{i\in S}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2 \tag{E.92}$$

$$\leq\left\|\sum_{i\in I_{\text{good}}}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2+\left\|\sum_{i\in(I_{\text{good}}\setminus S)}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2+\left\|\sum_{i\in(S\setminus I_{\text{good}})}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2 \tag{E.93}$$

$$\leq\left\|\sum_{i\in(I_{\text{good}}\setminus S)}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2+\left\|\sum_{i\in(S\setminus I_{\text{good}})}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2+n\sigma\alpha(\epsilon,n,\tau). \tag{E.94}$$

First we analyze

$$\left\|\sum_{i\in(I_{\text{good}}\setminus S)}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2. \tag{E.95}$$

This is the supremum over unit vectors v of

$$\sum_{i\in(I_{\text{good}}\setminus S)}v\cdot(\nabla f_i(w)-\nabla\bar{f}(w)). \tag{E.96}$$

However, we note that

$$\sum_{i\in I_{\text{good}}}(v\cdot(\nabla f_i(w)-\nabla\bar{f}(w)))^2\leq n\sigma^2\beta(\epsilon,n,\tau). \tag{E.97}$$

Since $|I_{\text{good}}\setminus S|\leq(1+1/p)\epsilon n$, we have by Cauchy-Schwarz that

$$\sum_{i\in(I_{\text{good}}\setminus S)}v\cdot(\nabla f_i(w)-\nabla\bar{f}(w))=\sqrt{(n\sigma^2\beta(\epsilon,n,\tau))((1+1/p)\epsilon n)}=n\sigma\sqrt{\beta(\epsilon,n,\tau)(1+1/p)\epsilon}, \tag{E.98}$$

as desired.

Let

$$\Delta:=\left\|\sum_{i\in S}(\nabla f_i(w)-\nabla\bar{f}(w))\right\|_2. \tag{E.99}$$

Because our Filter algorithm terminates with $n-|S|\leq(1+1/p)\epsilon n$, and the stopping condition is set as $\|\frac{1}{|S|}\sum_{i\in S}(\nabla f_i(w)-\nabla\hat{f}(w))(\nabla f_i(w)-\nabla\hat{f}(w))^\top\|\leq 4(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau))\sigma^2$, we note that since for any such v that

$$\sum_{i\in S}(v\cdot(\nabla f_i(w)-\nabla\bar{f}(w)))^2=\sum_{i\in S}(v\cdot(\nabla f_i(w)-\nabla\hat{f}(w)))^2+|S|(v\cdot(\nabla\hat{f}(w)-\nabla\bar{f}(w)))^2 \tag{E.100}$$

$$\leq \sum_{i\in S}(v\cdot(\nabla f_i(w)-\nabla\hat{f}(w)))^2+\Delta^2/|S|\leq n4(\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau))\sigma^2+\Delta^2/((1-(1+1/p)\epsilon)n) \tag{E.101}$$

and since $|S\backslash I_{\text{good}}|\leq(1+1/p)\epsilon n$, and so we have similarly that

$$\left\|\sum_{i\in(S\backslash I_{\text{good}})}\nabla f_i(w)-\nabla\bar{f}(w)\right\|_2\leq 2n\sigma\sqrt{\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)}\sqrt{(1+1/p)\epsilon}+\Delta\sqrt{\frac{(1+1/p)\epsilon}{1-(1+1/p)\epsilon}} \tag{E.102}$$

Combining with the above we have that

$$\frac{\Delta}{n}\leq\sigma\alpha(\epsilon,n,\tau)+\sigma\sqrt{\beta(\epsilon,n,\tau)(1+1/p)\epsilon}+2\sigma\sqrt{\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)}\sqrt{(1+1/p)\epsilon}+\frac{\Delta}{n}\sqrt{\frac{(1+1/p)\epsilon}{1-(1+1/p)\epsilon}}, \tag{E.103}$$

Thus

$$\frac{\Delta}{n}\leq\frac{1}{1-\sqrt{\frac{(1+1/p)\epsilon}{1-(1+1/p)\epsilon}}}\left(\sigma\alpha(\epsilon,n,\tau)+6\sigma\sqrt{\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)}\sqrt{\epsilon/p}\right) \tag{E.104}$$

and therefore, $\frac{\Delta}{n}=O\left(\sigma\left(\alpha(\epsilon,n,\tau)+\sqrt{\alpha(\epsilon,n,\tau)^2+\beta(\epsilon,n,\tau)}\sqrt{\epsilon/p}\right)\right)$ as desired. ■

E.5 Proofs for Section 6.4

Lemma E.5.1 (Lemma 6.4.1)**.** *Suppose the adversarial rewards are unbounded, and in a particular iteration t, the adversarial contaminate $\epsilon^{(t)}$ fraction of the episodes, then given M episodes, it is guaranteed that if $\epsilon^{(t)}\leq c$, for some absolute constant c, and any constant $\tau\in[0,1]$, we have*

$$\mathbb{E}\left[\mathbb{E}_{s,a\sim d^{(t)}}\left[\left(Q^{\pi^{(t)}}(s,a)-\phi(s,a)^\top w^{(t)}\right)^2\right]\right] \tag{E.105}$$
$$\leq O\left(\left(W^2+\frac{\sigma W}{1-\gamma}\right)\left(\sqrt{\epsilon^{(t)}}+f(d,\tau)M^{-\frac{1}{2}}+\tau\right)\right).$$

where $f(d,\tau)=\sqrt{d\log d}+\sqrt{\log(1/\tau)}$.

Proof of Lemma E.5.1. The proof of Lemma 6.4.1 follows by instantiating Theorem E.4.3 to our specific linear regression problem instance. To specify the constants in Theorem E.4.3, we make the following observations

leftmirgin=* By Lemma E.2.1, we have that $\xi=\frac{1}{(1-\gamma)^2}+\frac{\sigma^2}{1-\gamma}$.

leftmiirgiin=* Since $\|X\| \leq 1$, $\mathbb{E}_{X\sim D_x}\left[XX^\top\right] \leq I$, and thus $s = 1$.

leftmiiirgiin=* $\max_{\|v\|=1}\mathbb{E}\left[(v^\top X)^4\right] \leq \mathbb{E}\left[\|v\|^4\|X\|^4\right] \leq 1$, thus $C = 1$.

Plugging in the above instantiation to Theorem E.4.3 concludes the proof. ■

Theorem E.5.1 (Theorem 6.4.1)**.** *Under assumptions 7.2.1 and 6.2.2, given a desired optimality gap α, there exists a set of hyperparameters agnostic to the contamination level ϵ, such that Algorithm 2, using Algorithm 3 as the linear regression solver, guarantees with a $poly(1/\alpha, 1/(1-\gamma), |\mathcal{A}|, W, \sigma, \kappa)$ sample complexity that under ϵ-contamination, we have*

$$\mathbb{E}\left[V^*(\mu_0) - V^{\hat{\pi}}(\mu_0)\right] \leq \tilde{O}\left(\max\left[\alpha, \sqrt{\frac{|\mathcal{A}|\kappa\left(W^2 + \sigma W\right)}{(1-\gamma)^4}}\epsilon^{1/4}\right]\right). \tag{E.106}$$

where $\hat{\pi}$ is the uniform mixture of $\pi^{(1)}$ through $\pi^{(T)}$.

Proof of Theorem E.5.1. Denote $z := 2W$ and again $\epsilon_{stat} \leq (2W)^2 = z^2$. Denote $\left(W^2 + \frac{\sigma W}{1-\gamma}\right) = b$. Notice that Lemma 6.4.1 only holds when $\epsilon^{(t)} \leq c$ for some absolute constant c, and there are at most $\epsilon T/c$ iterations in which $\epsilon^{(t)} > c$, which incurs at most $\epsilon_{stat} \leq z^2$ error. Given this observation we can now plugging Lemma 6.4.1 into Lemma 6.3.1, and we get

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\{V^*(\mu_0) - V^{(t)}(\mu_0)\}\right] \tag{E.107}$$

$$\leq \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{4|\mathcal{A}|\kappa\epsilon_{stat}^{(t)}}{(1-\gamma)^3}} \tag{E.108}$$

$$\leq \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{z^2}{c}\epsilon + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{4|\mathcal{A}|\kappa b\left(\sqrt{\epsilon^{(t)}} + \sqrt{(d\log d)/M} + \sqrt{\log(1/\tau)/M} + \tau\right)}{(1-\gamma)^3}} \tag{E.109}$$

$$\leq \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{z^2}{c}\epsilon + \sqrt{\frac{4|\mathcal{A}|\kappa b\left(\sqrt{(d\log d)/M} + \sqrt{\log(1/\tau)/M} + \tau\right)}{(1-\gamma)^3}} + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{4|\mathcal{A}|\kappa b\sqrt{\epsilon^{(t)}}}{(1-\gamma)^3}} \tag{E.110}$$

$$\leq \frac{W}{1-\gamma}\sqrt{\frac{2\log|\mathcal{A}|}{T}} + \frac{z^2}{c}\epsilon + \sqrt{\frac{4|\mathcal{A}|\kappa b\left(\sqrt{(d\log d)/M} + \sqrt{\log(1/\tau)/M} + \tau\right)}{(1-\gamma)^3}} + \sqrt{\frac{4|\mathcal{A}|\kappa b}{(1-\gamma)^3}}\epsilon^{1/4} \tag{E.111}$$

where the last two steps are by Cauchy Schwarz and the fact that the attacker only has ϵ budget to distribute, which implies that $\sum_{t=1}^{T} \epsilon^{(t)} = T\epsilon$. Setting

$$T = \frac{2W^2 \log|\mathcal{A}|}{\alpha^2(1-\gamma)^2} \tag{E.112}$$

$$\tau = \frac{\alpha^2(1-\gamma)^3}{4|\mathcal{A}|b\kappa} \tag{E.113}$$

$$M = \frac{16|\mathcal{A}|^2 b^2 \kappa^2}{\alpha^4(1-\gamma)^6} \max\left[d\log d, \log(1/\tau)\right] \tag{E.114}$$

we get

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\{V^*(\mu_0) - V^{(t)}(\mu_0)\}\right] \leq O\left(\alpha + \sqrt{\frac{|\mathcal{A}|\kappa b}{(1-\gamma)^3}}\epsilon^{1/4}\right). \tag{E.115}$$

with sample complexity

$$TM = \frac{32W^2|\mathcal{A}|^2 \log|\mathcal{A}| b^2\kappa^2}{\alpha^6(1-\gamma)^8} \max\left[d\log d, \log(1/\tau)\right]. \tag{E.116}$$

■

E.6 Proof of Theorem 6.5.1

In PC-PG, aside from the robust linear regression step in Algorithm 9, in step 4 of Algorithm 4, we also needs to robustly estimate the covariance matrix under ϵ-contamination. Luckily, by assumption, $\phi(s,a)$ is bounded, and thus the current empirical mean estimation is already robust to adversarial contamination:

Lemma E.6.1 (Robust variant of Lemma G.3 in [2])**.** *Given $\nu \in \Delta(S \times A)$ and K ϵ-contaminated samples from ν. Denote $\Sigma = \mathbb{E}_{(s,a)\sim\nu}\left[\phi(s,a)\phi(s,a)^\top\right]$. Then, with probability at least $1-\delta$, we have that under ϵ-corruption*

$$\max_{\|x\|\leq 1}\left|x^\top\left(\sum_{i=1}^{K}\phi(s_i,a_i)\phi(s_i,a_i)^\top/K - \Sigma\right)x\right| \leq \sqrt{\frac{8\log(8d/\delta)}{K}} + 2\epsilon. \tag{E.118}$$

Proof. Without contamination, Lemma G.3 in [2] shows that

$$\max_{\|x\|\leq 1}\left|x^\top\left(\sum_{i=1}^{K}\phi(s_i,a_i)\phi(s_i,a_i)^\top/K - \Sigma\right)x\right| \leq \frac{2\log(8d/\delta)}{3K} + \sqrt{\frac{2\log(8d/\delta)}{K}}. \tag{E.119}$$

Algorithm 9 Robust NPG Update

1: **Input** ρ^n_{cov}, b^n, learning rate η, sample size M for critic fitting, iterations T
2: Define $\mathcal{K}^n = \{s : \forall a \in \mathcal{A}, b^n(s,a) = 0\}$
3: Initialize policy $\pi^0 : \mathcal{S} \to \Delta(\mathcal{A})$, such that

$$\pi^0(\cdot|s) = \begin{cases} \text{Uniform}(\mathcal{A}) & s \in \mathcal{K}^n \\ \text{Uniform}(\{a \in \mathcal{A} : b^n(s,a) > 0\}) & s \notin \mathcal{K}^n. \end{cases}$$

4: **for** $t = 0 \to T-1$ **do**
5: Draw M i.i.d samples $\left\{s_i, a_i, \widehat{Q}^{\pi^t}(s_i, a_i; r + b^n)\right\}_{i=1}^M$ with $s_i, a_i \sim \rho^n_{\text{cov}}$ (see Alg 4)
6: **Critic** fit: Call Algorithm 3 to solve for the robust linear regression problem

$$\theta^t = \underset{\|\theta\| \le W}{\arg\min} \sum_{i=1}^M \left(\theta \cdot \phi(s_i, a_i) - \left(\widehat{Q}^{\pi^t}(s_i, a_i; r + b^n) - b^n(s_i, a_i)\right)\right)^2$$

7: **Actor** update

$$\pi^{t+1}(\cdot|s) \propto \pi^t(\cdot|s) \exp\left(\eta \left(b^n(s,\cdot) + \theta^t \cdot \phi(s,\cdot)\right) \mathbf{1}\{s \in \mathcal{K}^n\}\right) \tag{E.117}$$

8: **return** $\pi^n := \text{Uniform}\{\pi^0, ..., \pi^{T-1}\}$.

Since both x and $\phi(s,a)$ has norm bounded by 1, the ϵ fraction of contaminated samples can only bias the estimate by at most 2ϵ, i.e. with ϵ-contamination

$$\max_{\|x\| \le 1} \left| x^\top \left(\sum_{i=1}^K \phi(s_i, a_i)\phi(s_i, a_i)^\top / K - \Sigma \right) x \right| \le \sqrt{\frac{8 \log(8d/\delta)}{K}} + 2\epsilon. \tag{E.120}$$

■

Lemma E.6.2 (Lemma G.4 in [2]). *Denote $\eta(K) = \sqrt{\frac{8\log(8d/\delta)}{K}} + 2\epsilon$. Then, under ϵ-contamination, $\phi(s,a)^\top (\Sigma^n_{cov})^{-1} \phi(s,a) \le \beta$ is guaranteed with probability $1-\delta$, if $N\eta(K) \le \lambda/2$.*

Lemma E.6.3 (variant of Lemma C.2 in [2]). *Assuming that for all iterations n but m of them, we have $\phi(s,a)^\top (\Sigma^n_{cov})^{-1} \phi(s,a) \le \beta$ for $(s,a) \in \mathcal{K}^n$, then*

$$V^* - V^{\hat{\pi}} \le \frac{1}{1-\gamma} \left(2W \sqrt{\frac{\log A}{T}} + 2\sqrt{\beta \lambda W^2} + \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} 2\sqrt{\beta N \epsilon^{(n,t)}_{stat}} + \frac{2 I_N(\lambda)}{\beta N} + 2Hm \right) \tag{E.121}$$

Proof. The proof follows exactly the proof of Lemma C.2 in [2], except that we note that for iterations

in which the assumption is not satisfied, the worst-case loss is bounded:

$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left(A^t_{b_n}(s,a)-\hat{A}^t_{b_n}(s,a)\right)\mathbf{1}\{s\in\mathcal{K}^n\}\right)\leq 2H \tag{E.122}$$

■

Proof of Theorem 6.5.1. First of all, we need to upper-bound m. The condition in Lemma E.6.2 is satisfied as long as $2\epsilon^{(n)}\leq\frac{\lambda}{4N}$ and $K\geq\frac{128N^2\log(8\tilde{d}/\delta)}{\lambda^2}$. Also note that $\sum_{n=0}^{N-1}\epsilon^{(n)}\leq N\epsilon$, and thus m is at most $\frac{8N^2\epsilon}{\lambda}$.

Also, by Lemma E.3.1,

$$\varepsilon^{(n,t)}_{stat}\leq 4\left(W^2+WH\right)\left(\epsilon^{(n,t)}+\sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right). \tag{E.123}$$

Plugging both into Lemma E.6.3, we get

$$V^*-V^{\hat{\pi}}\leq\frac{1}{1-\gamma}\left(2W\sqrt{\frac{\log A}{T}}+2\sqrt{\beta\lambda W^2}+2\sqrt{4\left(W^2+WH\right)\beta N\left(\epsilon+\sqrt{\frac{8}{M}\log\frac{4d}{\delta}}\right)}+\frac{2I_N(\lambda)}{\beta N}+\frac{16HN^2\epsilon}{\lambda}\right) \tag{E.124}$$

Let

$$T = \frac{4W^2\log A}{(1-\gamma)^2\alpha^2},\qquad \lambda=1,\qquad \beta=\frac{\alpha^2(1-\gamma)^2}{4W^2},\qquad N=\frac{4W^2d\log(N+1)}{\alpha^3(1-\gamma)^3} \tag{E.125}$$

$$M = \frac{2d^2\log^2(N+1)(W^2+WH)^2\log(\frac{4d}{\delta})}{\alpha^6(1-\gamma)^6},\qquad K=128N^2\log(8\tilde{d}/\delta) \tag{E.126}$$

Then, (E.124) gives

$$V^*-V^{\hat{\pi}}\leq 4\alpha+\sqrt{\frac{16(W^2WH)d\log(N+1)}{\alpha(1-\gamma)^3}\epsilon}+\frac{256HW^4d^2\log^2(N+1)}{\alpha^6(1-\gamma)^6}\epsilon \tag{E.127}$$

Let $\alpha=\epsilon^{1/7}$, then

$$V^*-V^{\hat{\pi}} \leq 4\epsilon^{1/7}+\sqrt{\frac{16(W^2WH)d\log(N+1)}{(1-\gamma)^3}}\epsilon^{3/7}+\frac{256HW^4d^2\log^2(N+1)}{(1-\gamma)^6}\epsilon^{1/7} \tag{E.128}$$

$$\leq \tilde{O}(d^2\epsilon^{1/7}) \tag{E.129}$$

Algorithm 10 `FPG-TRPO`

Input: initial policy parameter θ_0; initial value function parameter ϕ_0.
Hyperparameters: KL-divergence limit δ; backtracking coefficient α; maximum number of backtracking steps K; upper-bound of corruption level ϵ; episode length H; batch size M.
for $k = 0, 1, \ldots$ **do**
  Collect set of M trajectories $D_k = \{\tau_i\}_{1:M}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_{t,i} = \sum_{h=t}^{H} \gamma^{h-t} r_{h,i}$.
  Using GAE to compute advantage estimate $\hat{A}_{t,i}$ based on the current value function V_{ϕ_k}.
  Compute and save $\hat{g}_{t,i} = \nabla_\theta \log \pi_\theta(a_{t,i}, s_{t,i})|_{\theta_k}$ for all $t = 1 : H$ and $i = 1 : M$.
  Call the filtered conjugate gradient algorithm in Alg. 11 to get $S_k \subset [M] \times [H], \hat{x}_k = FCG(\hat{g}_{t,i}, \hat{A}_{t,i})$.
  Compute policy gradient estimate $\hat{g}_k = \frac{1}{|S_k|} \sum_{(t,i)\in S_k} \hat{g}_{t,i} \hat{A}_{t,i}$.
  Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k \hat{g}_k}} \hat{x}_k \tag{E.130}$$

  where $j \in \{0, 1, 2, ..., K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.
  Fit the value function by regression on mean-squared error on the filtered trajectories S_k:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|S_k|} \sum_{(t,i)\in S_k} \left(V_\phi(s_{t,i}) - \hat{R}_{t,i} \right)^2 \tag{E.131}$$

  In practice, one often only take a few gradient steps in each iteration k, instead of optimizing to convergence.

This concludes the proof. ■

E.7 Implementation Details of `FPG-TRPO`

In the experiment, we use a TRPO variant of FPG implementation, which differs from Alg. 2 in several ways:

1. Most existing TRPO implementation uses the conjugate gradient (CG) method instead of linear regression to solve for the matrix inverse vector product problem. We follow this convention and design FPG-TRPO to use a filtered conjugate gradient (FCG) subroutine to replace the standard CG produce. The FPG procedure is detailed in Alg. 11. At a high level FCG performs a filtering algorithm (a.k.a. outlier removal) on the residues of CG with respect to each data point.

Algorithm 11 Filtered Conjugate Gradient (FCG)

1: **Input:** $\hat{g}_{t,i}, \hat{A}_{t,i}$
2: **Hyperparameters:** Number of iterations r (default $r = 4$), fraction of data filtered in each iteration p (default $p = \epsilon/2$, i.e. filter out 2ϵ data in total).
3: Initialize $S = \{1, 2, \ldots, M\}$.
4: **for** $k = 1, \ldots, r$ **do**
5: Call standard CG to solve for $\hat{x} = \hat{F}^{-1}\hat{g}$, where $\hat{F} = \frac{1}{S}\sum_{(t,i)\in S} \hat{g}_{t,i}\hat{g}_{t,i}^\top$ and $\hat{g} = \frac{1}{S}\sum_{(t,i)\in S} \hat{g}_{t,i}\hat{A}_{t,i}$.
6: Compute the residues $r_{t,i} = \hat{g}_{t,i}\hat{g}_{t,i}^\top\hat{x} - \hat{g}_{t,i}\hat{A}_{t,i}$ for $(t,i) \in S$ and save in a matrix G of size $d \times |S|$.
7: Let v be the top right singular vector of G.
8: Compute the vector τ of *outlier scores* defined via $\tau_{t,i} = \left(r_{t,i}^\top v\right)^2$.
9: Remove (HMp) number of (t,i) pair with the largest outlier scores from S.
10: Call standard CG one more time and return $(S, \hat{x})$.

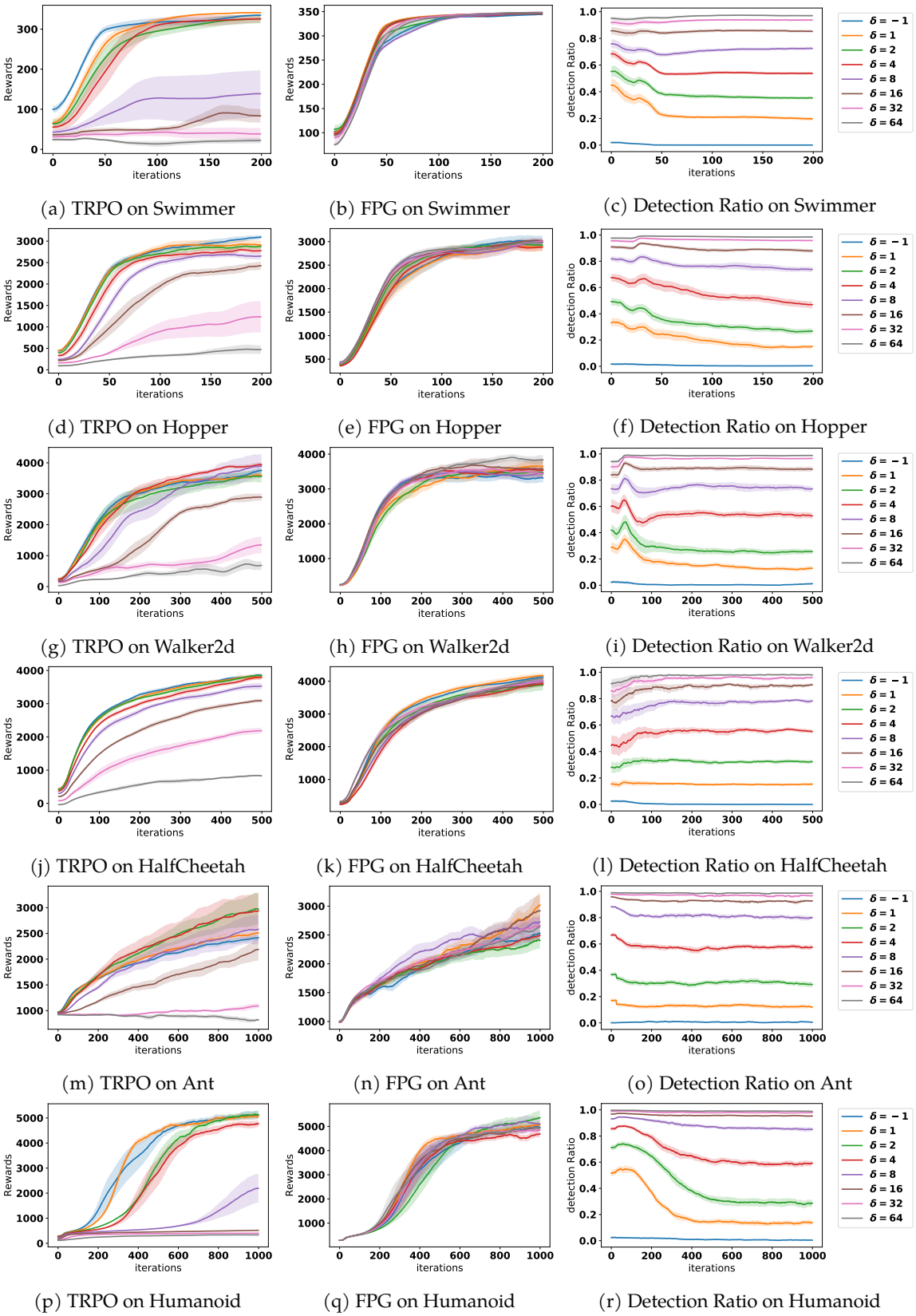
2. Again following existing TRPO implementations, FPG-TRPO builds another network to estimate the value function for the purpose of variance reduction, effectively resulting in an actor-critic algorithm. Instead of performing robust learning procedure on both policy and value function learning, we perform the main filtering algorithm on the policy learning procedure (the CG step discussed above), which also returns a filtered subset of data as a by-product. We then use this filtered subset of data to perform the rest of the learning procedure, including value function update and the sample loss estimation in backtracking line search. This allows us to perform the robust learning procedure only once per PG iteration.

3. FPG-TRPO uses a deterministic variant of the filtering algorithm suggested in [50], which empirically performs better and is simpler to implement than the stochastic variant used for theoretical analysis. Specifically, the filtering algorithm will simply remove a fixed fraction of points with the largest deviation along the top singular value direction (step 9 of Alg. 11).

The pseudo-code of `FPG-TRPO` can be found in Alg. 10. Similar to the NPG variant of FPG, the only difference between Alg. 10 and a standard TRPO implementation is the replacement of the CG subroutine with the FCG subroutine. This modular implementation allows one to easily replace Alg. 11 with any state-of-the-art robust CG procedure in the future. Table E.1 lists all the hyperparameters we used in our experiments, which are taken from open-source implementations of TRPO tuned for the MuJoCo environments. Our code to reproduce the experiment result is included in the supplementary material and will be open-sourced. Finally, Figure E.1 presents the detailed results on all experiments, completing the partial results shown in Figure 6.3.

Parameters	Values	Description
γ	0.995	discounting factor.
λ	0.97	GAE parameter [149].
l2-reg	0.001	L2 regularization weight in value loss.
δ	0.01	KL constraint in TRPO.
damping	0.1	damping factor in conjugate gradient.
batch-size	25000	number of time steps per policy gradient iteration.
α	0.5	backtracking coefficient.
K	10	maximum number of backtracking steps.

Table E.1: Hyperparameters for FPG-TRPO.



(a) TRPO on Swimmer (b) FPG on Swimmer (c) Detection Ratio on Swimmer

(d) TRPO on Hopper (e) FPG on Hopper (f) Detection Ratio on Hopper

(g) TRPO on Walker2d (h) FPG on Walker2d (i) Detection Ratio on Walker2d

(j) TRPO on HalfCheetah (k) FPG on HalfCheetah (l) Detection Ratio on HalfCheetah

(m) TRPO on Ant (n) FPG on Ant (o) Detection Ratio on Ant

(p) TRPO on Humanoid (q) FPG on Humanoid (r) Detection Ratio on Humanoid

Figure E.1: Detailed Results on the MuJoCo benchmarks.

F.1 Basics

Lemma F.1.1. *$\|w_h^*\| \leq H\sqrt{d}$ for all h.*

Proof. By definition, we have

$$w_h^* = \theta + \int_{\mathcal{S}} \hat{V}_{h+1}(s')\mu_h(s')ds' \tag{F.1}$$

and thus

$$\|w_h^*\| \leq \|\theta\| + \|\int_{\mathcal{S}} \hat{V}_{h+1}(s')\mu_h(s')ds'\| \tag{F.2}$$

$$\leq \|\theta\| + \int_{\mathcal{S}} \|\hat{V}_{h+1}(s')\mu_h(s')\|ds' \tag{F.3}$$

$$\leq \sqrt{d} + (H - h + 1)\sqrt{d} \tag{F.4}$$

$$\leq H\sqrt{d}. \tag{F.5}$$

■

Lemma F.1.2. *Note that $\mathbb{E}[[(r(s,a) + \hat{V}(s')) - (\mathbb{B}_h\hat{V})(s,a)]^2|s,a] \leq \gamma^2 = (\sigma + H/2)^2$*

Proof.

$$Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y) \leq Var(X) + Var(Y) + 2\sqrt{Var(X)Var(Y)}$$

Because $0 \leq \hat{V}(s') \leq H$,

$$\mathbb{E}[(\hat{V}(s') - \mathbb{E}[\hat{V}(s')|s,a])^2|s,a] = \mathbb{E}[\hat{V}(s')^2|s,a] - \mathbb{E}[\hat{V}(s')|s,a]^2 \tag{F.6}$$

$$\leq H\mathbb{E}[\hat{V}(s')|s,a] - \mathbb{E}[\hat{V}(s')|s,a]^2 \leq \frac{H^2}{4}. \tag{F.7}$$

$$\mathbb{E}[[(r(s,a)+\hat{V}(s'))-(\mathbb{B}_h\hat{V})(s,a)]^2|s,a] \tag{F.8}$$
$$=\mathbb{E}[[(r(s,a)+\hat{V}(s'))-\mathbb{E}[r(s,a)+\hat{V}(s')|s,a]]^2|s,a] \tag{F.9}$$
$$=\mathbb{E}[(r(s,a)-\mathbb{E}[r(s,a)|s,a])^2|s,a]+\mathbb{E}[(\hat{V}(s')-\mathbb{E}[\hat{V}(s')|s,a])^2|s,a] \tag{F.10}$$
$$+2\mathbb{E}[(r(s,a)-\mathbb{E}[r(s,a)|s,a])(\hat{V}(s')-\mathbb{E}[\hat{V}(s')|s,a])|s,a] \tag{F.11}$$
$$\leq\mathbb{E}[(r(s,a)-\mathbb{E}[r(s,a)|s,a])^2|s,a]+\mathbb{E}[(\hat{V}(s')-\mathbb{E}[\hat{V}(s')|s,a])^2|s,a] \tag{F.12}$$
$$+2\sqrt{\mathbb{E}[(r(s,a)-\mathbb{E}[r(s,a)|s,a])^2|s,a]\mathbb{E}[(\hat{V}(s')-\mathbb{E}[\hat{V}(s')|s,a])^2|s,a]} \quad \text{(By Cauchy's Ineq)} \tag{F.13}$$
$$=Var(r(s,a)\mid(s,a))+Var(\hat{V}(s')\mid(s,a))+2\sqrt{Var(r(s,a)\mid(s,a))Var(\hat{V}(s')\mid(s,a))} \tag{F.14}$$
$$=\left(\sqrt{Var(r(s,a)\mid(s,a))}+\sqrt{Var(\hat{V}(s')\mid(s,a))}\right)^2\leq(\sigma+H/2)^2 \tag{F.15}$$

]■

F.2 Proof of the Minimax Lower-bound

Proof of Theorem 7.3.1. Given any dimension d, time horizon H, consider a tabular MDP with action space size $A>2$ and state space size $S\leq\left(\frac{A}{2}\right)^{H/2}$ s.t. $SA=d$. Consider a "tree" with self-loops, which has S nodes and depth $\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil$. There is 1 node at the first level, $\frac{A}{2}$ nodes at the second level, $\left(\frac{A}{2}\right)^2$ nodes at the third level, …, $\left(\frac{A}{2}\right)^{\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil-2}$ nodes at the second to last level. The rest nodes are all at the last level. Define the MDP induced by this graph, where each state corresponds to a node, and each action corresponds to an edge. The agent always starts from the first level. For each state at the first $\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil-2$ levels, there are $A/2$ actions that leads to child nodes, and the rest leads back to that state, i.e. self-loops. The leaf states are absorbing state, i.e. all actions lead to self-loops. Denote this transition structure as P. Let's consider two MDPs with the same transition structure and different reward function, i.e. $M=(P,R)$, $M'=(P,R')$.

For MDP M, define $R(s^*,a^*)=$ Bernoulli$(SA\epsilon/2)$ on one particular (s^*,a^*) pair, where s^* is a leaf state at the last level, a^* is a self-loop action. Every other (s,a) pair receive reward 0. Let $(s',a')=\arg\min_{(s,a)}\nu(s,a)$ be the state-action pair appears least often in the data collecting distribution. For MCP M', define $R'(s^*,a^*)=$ Bernoulli$(SA\epsilon/2)$, $R'(s',a')=$ Bernoulli$(SA\epsilon)$ and 0 everywhere else. Then, it can be easily verified that: on M, the expected cumulative reward of the optimal policy is $\left(H-\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil\right)SA\epsilon/2$; on M', the expected cumulative reward of the optimal policy is at least $\left(H-\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil\right)SA\epsilon$; no policy can be simultaneously better than $\left(H-\lceil\log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\rceil\right)SA\epsilon/4$-optimal on both M and M'. Note that because

$S \le \left(\frac{A}{2}\right)^{H/2}$,

$$\left(H - \left\lceil \log_{A/2}\left(S\left(\frac{A}{2}-1\right)+1\right)\right\rceil\right) SA\epsilon/4 = \Omega(HSA\epsilon). \tag{F.16}$$

With probability at least $1/2$, we have $N(s',a') \le T\nu(s',a') \le T/SA$ by the pigeonhole principle. Conditioning on $N(s',a') \le T/SA$, with probability at least $1/2$, the amount of positive reward $r(s',a')$ will not exceed $SA\epsilon N(s',a') \le \epsilon T$, and thus an ϵ-contamination adversary can perturb all the positive rewards on (s',a') to 0. In other words, with probability $1/4$, the learner will observe a dataset whose likelihood under M and ($M' + \epsilon$-contamination) are exactly the same, and thus the learner must suffer at least $\Omega(HSA\epsilon)$ regret on one of the MDPs. ■

F.3 Proof of Upper-bounds

Proof of Lemma 7.3.2. Applying Lemma F.6.2 with $\pi = \hat{\pi}$, $\pi' = \tilde{\pi}$, and $\{\hat{Q}_h\}_{h=1}^H$ being the Q-functions constructed by the meta-algorithm, we have

$$\begin{aligned}\hat{V}_1(s) - V_1^{\tilde{\pi}}(s) = &\sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}\left[\langle \hat{Q}_h(s_h,\cdot), \hat{\pi}_h(\cdot|s_h) - \tilde{\pi}_h(\cdot|s_h)\rangle_{\mathcal{A}} | s_1 = s\right] \\ &+ \sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}\left[\hat{Q}_h(s_h,a_h) - (\mathbb{B}_h\hat{V}_{h+1})(s_h,a_h) | s_1 = s\right]\end{aligned} \tag{F.17}$$

Similarly, applying Lemma F.6.2 with $\pi = \pi' = \hat{\pi}$, we have

$$\hat{V}_1(s) - V_1^{\hat{\pi}}(s) = \sum_{h=1}^H \mathbb{E}_{\hat{\pi}}\left[\hat{Q}_h(s_h,a_h) - (\mathbb{B}_h\hat{V}_{h+1})(s_h,a_h) | s_1 = s\right] \tag{F.18}$$

Then, we have

$$\text{SubOpt}(\hat{\pi},\tilde{\pi}) = \left(V_1^{\tilde{\pi}}(\mu) - \hat{V}_1(\mu)\right) + \left(\hat{V}_1(\mu) - V_1^{\hat{\pi}}(\mu)\right) \tag{F.19}$$

$$= -\sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}\left[(\mathbb{B}_h\hat{V}_{h+1}) - \hat{Q}_h\right] + \sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}\left[(\mathbb{B}_h\hat{V}_{h+1}) - \hat{Q}_h\right] \tag{F.20}$$

$$+ \sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}\left[\langle \hat{Q}_h(s_h,\cdot), \tilde{\pi}_h(\cdot|s_h) - \hat{\pi}_h(\cdot|s_h)\rangle_{\mathcal{A}}\right] \tag{F.21}$$

$$\le 0 + 2\sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}[\Gamma_h(s,a)] + 0 \tag{F.22}$$

$$= 2\sum_{h=1}^H \mathbb{E}_{\tilde{\pi}}[\Gamma_h(s,a)] \tag{F.23}$$

as needed. ■

Proof of Theorem 7.3.3. To simplify notation, below we use N for the number of data points per time step, i.e. $N := N/H$. We first show that

$$|\hat{Q}_h(s,a) - (\mathbb{B}_h \hat{V}_{h+1})(s,a)| \leq \Gamma(s,a). \tag{F.24}$$

The robust least-square oracle guarantees

$$\mathbb{E}_\nu \left(\|x^\top (\hat{w} - w^*)\|_2^2 \right) \leq c_2(\delta) \cdot \left(\frac{\gamma^2 \text{poly}(d)}{N} + \gamma^2 \epsilon \right) \tag{F.25}$$

$$\implies \|\hat{w}_h - w_h^*\|_\Sigma^2 \leq c_2(\delta) \cdot \left(\frac{\gamma^2 \text{poly}(d)}{N} + \gamma^2 \epsilon \right) \tag{F.26}$$

$$\implies \|\hat{w}_h - w_h^*\|_{\Sigma + (2\epsilon+\lambda)I}^2 \leq c_2(\delta) \cdot \left(\frac{\gamma^2 \text{poly}(d)}{N} + \gamma^2 \epsilon + (2\epsilon + \lambda) H^2 d \right) \tag{F.27}$$

Then,

$$|\hat{Q}_h(s,a) - (\mathbb{B}_h \hat{V}_{h+1})(s,a)| = |\phi(s,a)(\hat{w}_h - w_h^*)| \tag{F.28}$$

$$\leq \|\hat{w}_h - w_h^*\|_{(\Sigma + (2\epsilon+\lambda)I)} \|\phi(s,a)\|_{(\Sigma + (2\epsilon+\lambda)I)^{-1}} \tag{F.29}$$

$$\leq \sqrt{c_2(\delta) \cdot \left(\frac{\gamma^2 \text{poly}(d)}{N} + \gamma^2 \epsilon + (2\epsilon + \lambda) H^2 d \right)} \|\phi(s,a)\|_{(\Sigma + (2\epsilon+\lambda)I)^{-1}} \tag{F.30}$$

$$\leq \sqrt{c_2(\delta)} \cdot \left(\frac{\gamma \text{poly}(d)}{\sqrt{N}} + (\gamma + 2H\sqrt{d})\sqrt{\epsilon} + H\sqrt{d\lambda}) \right) \|\phi(s,a)\|_{\Lambda^{-1}} \tag{F.31}$$

where the last step are due to $W \leq H\sqrt{d}$ and

$$\Lambda = \frac{3}{5} \left(\frac{1}{N} \sum_{i=1}^N \phi_i \phi_i^\top + (\epsilon + \lambda) \cdot I \right) \tag{F.32}$$

$$\preceq \frac{3}{5} \left(\frac{1}{N} \sum_{i=1}^N \tilde{\phi}_i \tilde{\phi}_i^\top + (2\epsilon + \lambda) \cdot I \right) \tag{F.33}$$

$$\preceq (\Sigma + (2\epsilon + \lambda) \cdot I) \tag{F.34}$$

where the last step applies Lemma F.6.3 because $N(2\epsilon + \lambda) \geq \Omega(d \log(N/\delta))$ due to the definition of λ and $\epsilon \geq 0$.

Next, we show that Algorithm 5 achieves the desired optimality gap. By Lemma 7.3.2, we have

$$\begin{aligned}
\text{SubOpt}(\hat{\pi}) &\leq 2H\mathbb{E}_{\pi^*}[\Gamma(s,a)] && \text{(F.35)}\\
&\leq \sqrt{c_2(\delta)} \cdot \left(\frac{\gamma H \text{poly}(d)}{\sqrt{N}} + (H\gamma + 2H^2\sqrt{d})\sqrt{\epsilon} + H^2\sqrt{d\lambda} \right) \mathbb{E}_{\pi^*}\left[\|\phi(s,a)\|_{\Lambda^{-1}} \right] && \text{(F.36)}
\end{aligned}$$

Focusing on the last term, applying Lemma F.6.3 again, we have

$$\begin{aligned}
\mathbb{E}_{d^*}[\|\phi(s,a)\|_{\Lambda^{-1}}] \leq & \mathbb{E}_{d^*}[\|\phi(s,a)\|_{(\frac{1}{5}(\Sigma+\lambda I))^{-1}}] && \text{(F.37)}\\
= & \mathbb{E}_{d^*}\left[\sqrt{\phi^\top (\frac{1}{5}(\Sigma+\lambda I))^{-1}\phi} \right] && \text{(F.38)}\\
\leq & \sqrt{\mathbb{E}_{d^*}[\phi^\top (\frac{1}{5}(\Sigma+\lambda I))^{-1}\phi]} && \text{(F.39)}\\
\leq & \sqrt{tr\left(\Sigma_* (\frac{1}{5}(\Sigma+\lambda I))^{-1} \right)} && \text{(F.40)}\\
\leq & \sqrt{\kappa tr\left(\Sigma (\frac{1}{5}(\Sigma+\lambda I))^{-1} \right)} && \text{(F.41)}\\
\leq & \sqrt{5\kappa \sum_{i=1}^{d} \frac{\sigma_i}{\sigma_i+\lambda}} && \text{(F.42)}\\
\leq & \sqrt{5d\kappa} && \text{(F.43)}
\end{aligned}$$

Combining the two terms give the desired results. ■

F.4 Proof of uncorrupted learning results

In this section, we prove the conclusion in Corollary 7.3.1 and 7.3.2. The proof follows closely the classic analysis of Least Squared Value Iteration (LSVI) methods with the only difference being the data splitting which allows us to ditch the covering argument and obtain a tighter bound. Such a trick is only possible in the offline setting where the data are assumed to be i.i.d. For completeness, we specify the uncorrupted algorithm in Alg. 12.

Algorithm 12 Uncorrupted Least-Square Value Iteration (R-LSVI)

Input: Dataset $D = \{(s_i, a_i, r_i, s_i')\}_{1:N}$; pessimism bonus $\Gamma_h(s,a) \geq 0$, $\lambda > 0$.
Split the dataset randomly into H subset: $D_h = \{(s_i^h, a_i^h, r_i^h, s_i'^h)\}_{1:(N/H)}$, for $h \in [H]$.
Initialization: Set $\hat{V}_{H+1}(s) \leftarrow 0$.
for step $h = H, H-1, \ldots, 1$ **do**
    Set $\Lambda_h \leftarrow \frac{H}{N}\sum_{i=1}^{N/H} \phi(s_i^h, a_i^h)\phi(s_i^h, a_i^h)^\top + \lambda \cdot I$.
    Set $\hat{w}_h \leftarrow \Lambda_h^{-1}(\frac{H}{N}\sum_{i=1}^{N/H} \phi(s_i^h, a_i^h) \cdot (r_i^h + \hat{V}_{h+1}(s_i^{h+1})))$.
    Set $\hat{Q}_h(s,a) \leftarrow \phi(s,a)^\top \hat{w}_h - \Gamma_h(s,a)$, clipped within $[0, H-h+1]$.
    Set $\hat{\pi}_h(a|s) \leftarrow \arg\max_a \hat{Q}_h(s,a)$ and $\hat{V}_h(s) \leftarrow \max_a \hat{Q}_h(s,a)$.
Output: $\{\hat{\pi}_h\}_{h=1}^H$.

We first prove the following lemma:

Lemma F.4.1 (Bound on the Bellman Error)**.** *Under assumption 7.2.1, given a dataset of size N, Algorithm 5 achieves*

$$|(\mathbb{B}_h\hat{V}_{h+1})(s,a) - \hat{Q}_h(s,a)| \leq H\left(\sqrt{d\cdot\lambda} + \sqrt{\frac{Hd\log(N/\delta\lambda)}{N}}\right)\cdot\sqrt{\phi(x,a)^\top\Lambda_h^{-1}\phi(x,a)}$$

for all $(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[H]$, with probability at least $1-\delta$.

Proof. We start by applying the following decomposition

$$\begin{aligned}
&(\mathbb{B}_h\hat{V}_{h+1})(s,a) - \hat{Q}_h(s,a) && \text{(F.44)}\\
=&(\mathbb{B}_h\hat{V}_{h+1})(s,a) - (\hat{\mathbb{B}}_h\hat{V}_{h+1})(s,a) && \text{(F.45)}\\
=&\underbrace{\phi(s,a)^\top w_h - \phi(s,a)^\top\Lambda_h^{-1}\left(\frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i,a_i)\cdot(\mathbb{B}_h\hat{V}_{h+1})(s_i,a_i)\right)}_{\text{(i)}} - && \text{(F.46)}\\
&\underbrace{\phi(s,a)^\top\Lambda_h^{-1}\left(\frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i,a_i)\cdot\left(r_i + \hat{V}_{h+1}(s_i') - (\mathbb{B}_h\hat{V}_{h+1})(s_i,a_i)\right)\right)}_{\text{(ii)}} && \text{(F.47)}
\end{aligned}$$

Therefore, by triangle inequality we have

$$|(\mathbb{B}_h\hat{V}_{h+1})(s,a) - \hat{Q}_h(s,a)| \leq |\text{(i)}| + |\text{(ii)}| \qquad \text{(F.48)}$$

Then, we bound the two terms separately:

$$\begin{aligned}|(\mathrm{i})| &= \left|\phi(s,a)^\top w_h - \phi(s,a)^\top \Lambda_h^{-1}\left(\frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i,a_i)\cdot\phi(s_i,a_i)^\top w_h\right)\right| \\ &= \left|\phi(s,a)^\top w_h - \phi(s,a)^\top\Lambda_h^{-1}(\Lambda_h - \lambda\cdot I)w_h\right| = \lambda\cdot\left|\phi(s,a)^\top\Lambda_h^{-1}w_h\right| \\ &\leq \lambda\cdot\|w_h\|_{\Lambda_h^{-1}}\cdot\|\phi(s,a)\|_{\Lambda_h^{-1}} \leq H\sqrt{d\cdot\lambda}\cdot\sqrt{\phi(s,a)^\top\Lambda_h^{-1}\phi(s,a)}.\end{aligned}$$

For the second term, define

$$\epsilon_i^h(V) = r_i^h + V(s_i^{h\prime}) - (\mathbb{B}_h V)(s_i^h, a_i^h) \tag{F.49}$$

Then, we have

$$\begin{aligned}|(\mathrm{ii})| &= \left|\phi(s,a)^\top\Lambda_h^{-1}\left(\frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i,a_i)\cdot\epsilon_i^h(\hat{V}_{h+1})\right)\right| \\ &\leq \underbrace{\left\|\frac{H}{N}\sum_{i=1}^{N/H}\phi(s_i,a_i)\cdot\epsilon_i^h(\hat{V}_{h+1})\right\|_{\Lambda_h^{-1}}}_{(\mathrm{iii})}\cdot\sqrt{\phi(x,a)^\top\Lambda_h^{-1}\phi(x,a)}.\end{aligned} \tag{F.50}$$

From here, because of our data splitting, $\hat{V}_{h+1}$ is independent from D_h, and thus we can bypass the covering argument and directly apply matrix concentrations. In particular, by applying Lemma F.6.1, we have that with probability at least $1-\delta$

$$(\mathrm{iii}) \leq H\sqrt{\frac{Hd\log(1+N/H\lambda) + 2H\log(1/\delta)}{N}} \tag{F.51}$$

Combining the two terms gives

$$|(\mathbb{B}_h\hat{V}_{h+1})(s,a) - \hat{Q}_h(s,a)| \leq H\left(\sqrt{d\cdot\lambda} + \sqrt{\frac{Hd\log(N/\delta\lambda)}{N}}\right)\cdot\sqrt{\phi(x,a)^\top\Lambda_h^{-1}\phi(x,a)} \tag{F.52}$$

■

Now, given Lemma F.4.1, applying Lemma 7.3.2, we have

$$\mathrm{SubOpt}(\hat{\pi},\tilde{\pi}) \leq 2\sum_{h=1}^{H}\mathbb{E}_{d^{\tilde{\pi}}}[\Gamma_h(s,a)] \leq 2H^2\left(\sqrt{d\cdot\lambda} + \sqrt{\frac{Hd\log(N/\delta\lambda)}{N}}\right)\cdot\mathbb{E}_{d^{\tilde{\pi}}}[\sqrt{\phi(x,a)^\top\Lambda_h^{-1}\phi(x,a)}] \tag{F.53}$$

The last step would be to bound $\mathbb{E}_{d^{\tilde{\pi}}}[\sqrt{\phi(x,a)^\top \Lambda_h^{-1}\phi(x,a)}]$, similar to the last section. In particular, applying Lemma F.6.3, we have

$$\begin{aligned}
\mathbb{E}_{d^{\tilde{\pi}}}\left[\sqrt{\phi(x,a)^\top \Lambda_h^{-1}\phi(x,a)}\right] \leq & \mathbb{E}_{d^{\tilde{\pi}}}\left[\sqrt{3\phi(x,a)^\top(\Sigma+\lambda I)\phi(x,a)}\right] && \text{(F.54)}\\
\leq & \sqrt{3\mathbb{E}_{d^{\tilde{\pi}}}\left[\phi(x,a)^\top(\Sigma+\lambda\cdot I)\phi(x,a)\right]} && \text{(F.55)}\\
\leq & \sqrt{3d\kappa} && \text{(F.56)}
\end{aligned}$$

where step F.54 requires $\lambda \geq H\Omega(d\log(N/\delta))/N$. Thus,

$$\begin{aligned}
\text{SubOpt}(\hat{\pi},\tilde{\pi}) \leq & 2H^2\left(\sqrt{d\cdot\lambda}+\sqrt{Hd\log(N/\delta\lambda)}\right)\sqrt{\frac{3d\kappa}{N}} && \text{(F.57)}\\
\leq & \tilde{O}\left(H^2\left(d\sqrt{\log(N/\delta)}+\sqrt{Hd\log(N/(d\delta))}\right)\sqrt{\frac{3d\kappa}{N}}\right) && \text{(F.58)}
\end{aligned}$$

F.5 Lower-bound on best-of-both-world results

Proof of Theorem 7.3.4. Consider two instances of the offline RL problem, with two MDPs, M and M', both of which are actually simple two-arm bandit problems, along with their data generating distribution ν and ν', defined below.

1. Instance 1: Bandit M has $r_1 = \text{Bernoulli}(\frac{1}{2}+\frac{\epsilon}{2p})$ and $r_2 = \text{Bernoulli}(\frac{1}{2})$. The data generating distribution is $\nu(a_1) = p$ and $\nu(a_2) = 1-p$. The relative condition number is $1/p$.
2. Instance 2: Bandit M has $r_1 = \text{Bernoulli}(\frac{1}{2}-\frac{\epsilon}{2p})$ and $r_2 = \text{Bernoulli}(\frac{1}{2})$. The data generating distribution is $\nu(a_1) = p$ and $\nu(a_2) = 1-p$, same as instance 1. The relative condition number is $1/(1-p)$.

Let D and D' be i.i.d. dataset of size N generated by instance 1 and 2 respectively, generated with the following *coupling* process. First, the actions are sampled from ν and shared across instances, e.g. $N_D(a_1) = N_{D'}(a_1)$ and $N_D(a_2) = N_{D'}(a_2)$. Then, the rewards of a_2 are sampled from $\text{Bernoulli}(\frac{1}{2})$ and shared across tasks, e.g. $N_D(a_2,0) = N_{D'}(a_2,0)$ and $N_D(a_2,1) = N_{D'}(a_2,1)$.

Finally, let X_i, Y_i be Bernoulli random variables s.t. $X_i = \begin{cases} 0 & U \leq \frac{1}{2}-\frac{\epsilon}{2p} \\ 1 & \text{o.w.} \end{cases}$, $Y_i = \begin{cases} 0 & U \leq \frac{1}{2}+\frac{\epsilon}{2p} \\ 1 & \text{o.w.} \end{cases}$, where U is picked uniformly random in $[0,1]$. Then (X_i, Y_i) is a coupling with law: $P((X_i,Y_i) = (0,0)) = \frac{1}{2}-\frac{\epsilon}{2p}$, $P((X_i,Y_i)=(1,0)) = 0$, $P((X_i,Y_i)=(0,1)) = \frac{\epsilon}{2p}$, $P((X_i,Y_i)=(s_3,s_3)) = \frac{1}{2}-\frac{\epsilon}{2p}$, X_i and Y_i can be thought as the outcome of $\text{Bernoulli}(\frac{1}{2}+\frac{\epsilon}{2p})$, $\text{Bernoulli}(\frac{1}{2}+\frac{\epsilon}{2p})$ respectively. Then,

let the rewards of a_1 of the two instances be generated by Y_i and X_i respectively. We then have

$$P(\sum_{i=1}^{N(a_1)} \mathbb{1}\left[X_i \neq Y_i\right]) \geq P(N(a_1) \leq pN) \cdot P(\sum_{i=1}^{pN} \mathbb{1}\left[X_i \neq Y_i\right]) \geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \tag{F.59}$$

In other word, with probability at least $\frac{1}{4}$, instance 1 and 2 are indistinguishable under ϵ-contamination, in particular the adversary can replace at most ϵN of $(a_1, 0)$ with $(a_1, 1)$ in D' to replicate D. Therefore, instance 1 and (instance 2 + ϵ-contamination) are with probability at least $1/4$ indistinguishable. Now, if an algorithm wants to achieve best of both world guarantee, it must return a_1 as the optimal arm with high probability when observing a dataset generated as above, in which case it will suffer a suboptimality of $\frac{\epsilon}{2p}$ if the data is generated by (instance 2 + ϵ-contamination). As $p \geq \epsilon \geq 0$ goes to 0, this gap blows up, while the relative condition number $1/(1-p)$ remains bounded, thus contradiction.

■

F.6 Technical Lemmas

Lemma F.6.1 (Concentration of Self-Normalized Processes [1]). *Let $\{\epsilon_t\}_{t=1}^{\infty}$ be a real-valued stochastic process that is adaptive to a filtration $\{\mathcal{F}_t\}_{t=0}^{\infty}$. That is, ϵ_t is $\mathcal{F}_t$-measurable for all $t \geq 1$. Moreover, we assume that, for any $t \geq 1$, conditioning on $\mathcal{F}_{t-1}$, ϵ_t is a zero-mean and σ-subGaussian random variable such that*

$$\mathbb{E}[\epsilon_t | \mathcal{F}_{t-1}] = 0 \qquad \textit{and} \qquad \mathbb{E}[\exp(\lambda \epsilon_t) | \mathcal{F}_{t-1}] \leq \exp(\lambda^2 \sigma^2 / 2), \quad \forall \lambda \in \mathbb{R}. \tag{F.60}$$

Besides, let $\{\phi_t\}_{t=1}^{\infty}$ be an $\mathbb{R}^d$-valued stochastic process such that ϕ_t is $\mathcal{F}_{t-1}$-measurable for all $t \geq 1$. Let $M_0 \in \mathcal{R}^{d \times d}$ be a deterministic and positive-definite matrix, and we define $M_t = M_0 + \sum_{s=1}^{t} \phi_s \phi_s^\top$ for all $t \geq 1$. Then for any $\delta > 0$, with probability at least $1 - \delta$, we have for all $t \geq 1$ that

$$\Big\| \sum_{s=1}^{t} \phi_s \cdot \epsilon_s \Big\|_{M_t^{-1}}^2 \leq 2\sigma^2 \cdot \log\left(\frac{\det(M_t)^{1/2} \det(M_0)^{-1/2}}{\delta} \right).$$

Lemma F.6.2 (Extended Value Difference [35]). *Let $\pi = \{\pi_h\}_{h=1}^{H}$ and $\pi' = \{\pi'_h\}_{h=1}^{H}$ be two arbitrary policies and let $\{\hat{Q}_h\}_{h=1}^{H}$ be any given Q-functions. For any $h \in [H]$, we define a value function $\hat{V}_h \colon \mathcal{S} \to \mathbb{R}$*

by letting $\hat{V}_h(x) = \langle \hat{Q}_h(x,\cdot), \pi_h(\cdot|x)\rangle_{\mathcal{A}}$ *for all* $s \in \mathcal{S}$*. Then for all* $s \in \mathcal{S}$*, we have*

$$\hat{V}_1(s) - V_1^{\pi'}(s) = \sum_{h=1}^{H} \mathbb{E}_{\pi'}\left[\langle \hat{Q}_h(s_h,\cdot), \pi_h(\cdot|s_h) - \pi'_h(\cdot|s_h)\rangle_{\mathcal{A}} | s_1 = s\right] \tag{F.61}$$

$$+ \sum_{h=1}^{H} \mathbb{E}_{\pi'}\left[\hat{Q}_h(s_h,a_h) - (\mathbb{B}_h \hat{V}_{h+1})(s_h,a_h) | s_1 = s\right], \tag{F.62}$$

where the expectation $\mathbb{E}_{\pi'}$ *is taken with respect to the trajectory generated by* π'*, and* $\mathbb{B}_h$ *is the Bellman operator.*

Lemma F.6.3 (Concentration of Covariances [181])**.** *Let* $\{\phi_i\}_{1:N} \subset \mathbb{R}^d$ *be i.i.d. samples from an underlying bounded distribution* ν*, with* $\|\phi_i\|_i \leq 1$ *and covariance* Σ*. Define*

$$\Lambda = \sum_{i=1}^{N} \phi_i \phi_i^\top + \lambda \cdot I \tag{F.63}$$

for some $\lambda \geq \Omega(d \log(N/\delta))$*. Then, we have that with probability at least* $(1-\delta)$*,*

$$\frac{1}{3}(N\Sigma + \lambda I) \preceq \Lambda \preceq \frac{5}{3}(N\Sigma + \lambda I) \tag{F.64}$$

Proof. See [181] Lemma 32 for detailed proof. ■

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, volume 11, pages 2312–2320, 2011.

[2] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020.

[3] Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in Neural Information Processing Systems*, 33, 2020.

[4] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *arXiv preprint arXiv:1908.00261*, 2019.

[5] Naman Agarwal, Brian Bullins, Elad Hazan, Sham M Kakade, and Karan Singh. Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*, 2019.

[6] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[7] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[8] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

[9] Jason Altschuler, Victor-Emmanuel Brunel, and Alan Malek. Best arm identification for contaminated bandits. *Journal of Machine Learning Research*, 20(91):1–39, 2019.

[10] John Asmuth, Michael L Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 2*, pages 604–609. AAAI Press, 2008.

[11] Karl J Åström. *Introduction to stochastic control theory*. Courier Corporation, 2012.

[12] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[13] Michael Athans and Peter L Falb. *Optimal control: An introduction to the theory and its applications*. Courier Corporation, 2013.

[14] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in neural information processing systems*, pages 89–96, 2009.

[15] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F Yang. Model-based reinforcement learning with value-targeted regression. *arXiv preprint arXiv:2006.01107*, 2020.

[16] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272, 2017.

[17] Ainesh Bakshi and Adarsh Prasad. Robust linear regression: Optimal rates in polynomial time. *arXiv preprint arXiv:2007.01394*, 2020.

[18] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[19] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[20] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[21] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[22] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. SIAM, 2010.

[23] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[24] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In John Langford and Joelle Pineau, editors, *29th Int'l Conf. on Machine Learning (ICML)*. Omnipress, Omnipress, 2012.

[25] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[26] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[27] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[28] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[29] Daniel S Brown and Scott Niekum. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7749–7758, 2019.

[30] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[31] Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

[32] Cody Burkard and Brent Lagesse. Analysis of causative attacks against svms learning from data streams. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, pages 31–36. ACM, 2017.

[33] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. Knitro: An integrated package for nonlinear optimization. In *Large Scale Nonlinear Optimization, 35–59, 2006*, pages 35–59. Springer Verlag, 2006.

[34] Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. *arXiv preprint arXiv:1912.05830*, 2019.

[35] Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, pages 1283–1294. PMLR, 2020.

[36] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[37] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60, 2017.

[38] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464. ACM, 2019.

[39] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[40] Yiding Chen and Xiaojin Zhu. Optimal adversarial attack on autoregressive models. *arXiv preprint arXiv:1902.00202*, 2019.

[41] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Non-stationary reinforcement learning: The blessing of (more) optimism. *Available at SSRN 3397818*, 2019.

[42] Alon Cohen, Avinatan Hassidim, Tomer Koren, Nevena Lazic, Yishay Mansour, and Kunal Talwar. Online linear quadratic control. *arXiv preprint arXiv:1806.07104*, 2018.

[43] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[44] Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5713–5723, 2017.

[45] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the sample complexity of the linear quadratic regulator. *arXiv preprint arXiv:1710.01688*, 2017.

[46] Esther Derman, Daniel Mankowitz, Timothy Mann, and Shie Mannor. A bayesian approach to robust reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 648–658. PMLR, 2020.

[47] Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. IFAAMAS, 2012.

[48] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[49] I Diakonikolas, G Kamath, DM Kane, J Li, A Moitra, and A Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664, 2016.

[50] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606, 2019.

[51] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. *arXiv preprint arXiv:1703.00893*, 2017.

[52] Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

[53] Ilias Diakonikolas, Daniel M Kane, and Ankit Pensia. Outlier robust mean estimation with subgaussian rates via stability. *Advances in Neural Information Processing Systems*, 33, 2020.

[54] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[55] Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. A kernel-based approach to non-stationary reinforcement learning in metric spaces. *arXiv preprint arXiv:2007.05078*, 2020.

[56] Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. Provably efficient q-learning with function approximation via distribution shift error checking oracle. In *Advances in Neural Information Processing Systems*, pages 8060–8070, 2019.

[57] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017.

[58] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[59] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[60] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

[61] Eyal Even-Dar and Yishay Mansour. Learning rates for q-learning. *Journal of machine learning Research*, 5(Dec):1–25, 2003.

[62] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[63] Huckleberry Febbo. Nloptcontrol.jl, 2017.

[64] Claude-Nicolas Fiechter. Pac adaptive control of linear systems. In *Annual Workshop on Computational Learning Theory: Proceedings of the tenth annual conference on Computational learning theory*, volume 6, pages 72–80. Citeseer, 1997.

[65] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

[66] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[67] S. Goldman and M. Kearns. On the complexity of teaching. *Journal of Computer and Systems Sciences*, 50(1):20–31, 1995.

[68] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

[69] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[70] Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. *arXiv preprint arXiv:1902.08647*, 2019.

[71] Lasse Holmström and Jussi Klemelä. Asymptotic bounds for the expected l1 error of a multivariate kernel density estimator. *Journal of multivariate analysis*, 42(2):245–266, 1992.

[72] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[73] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[74] Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. *arXiv preprint arXiv:1906.10571*, 2019.

[75] Peter J Huber et al. The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 221–233. University of California Press, 1967.

[76] Nan Jiang. Notes on tabular methods. 2020.

[77] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2017.

[78] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

[79] Chi Jin, Tiancheng Jin, Haipeng Luo, Suvrit Sra, and Tiancheng Yu. Learning adversarial markov decision processes with bandit feedback and unknown transition. In *International Conference on Machine Learning*, pages 4860–4869. PMLR, 2020.

[80] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint arXiv:1902.03736*, 2019.

[81] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[82] Tiancheng Jin and Haipeng Luo. Simultaneously learning stochastic and adversarial episodic mdps with known transition. *arXiv preprint arXiv:2006.05606*, 2020.

[83] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

[84] Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. *Adversarial Machine Learning*. Cambridge University Press, 2018.

[85] Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Xiaojin Zhu. Adversarial attacks on stochastic bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[86] Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. *Advances in Neural Information Processing Systems*, 33, 2020.

[87] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[88] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14:1531–1538, 2001.

[89] P Kamalaruban, R Devidze, V Cevher, and A Singla. Interactive teaching algorithms for inverse reinforcement learning. In *28th International Joint Conference on Artificial Intelligence*, pages 604–609, 2019.

[90] Chiu Yen Kao, Stanley Osher, and Jianliang Qian. Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations. *Journal of Computational Physics*, 196(1):367–391, 2004.

[91] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[92] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[93] Donald E Kirk. *Optimal control theory: An introduction*. Courier Corporation, 2012.

[94] Adam Klivans, Pravesh K Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. In *Conference On Learning Theory*, pages 1420–1430. PMLR, 2018.

[95] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[96] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[97] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[98] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[99] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[100] Panqanamala Ramana Kumar and Pravin Varaiya. *Stochastic systems: Estimation, identification, and adaptive control*, volume 75. SIAM, 2015.

[101] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.

[102] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[103] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

[104] Gregory F Lawler. Expected hitting times for a random walk on a connected graph. *Discrete mathematics*, 61(1):85–92, 1986.

[105] Chung-Wei Lee, Haipeng Luo, Chen-Yu Wei, and Mengxiao Zhang. Bias no more: high-probability data-dependent regret bounds for adversarial bandits and mdps. *Advances in Neural Information Processing Systems*, 33, 2020.

[106] Laurent Lessard, Xuezhou Zhang, and Xiaojin Zhu. An optimal control approach to sequential machine teaching. *arXiv preprint arXiv:1810.06175*, 2018.

[107] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[108] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, pages 1885–1893, 2016.

[109] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[110] Daniel Liberzon. *Calculus of variations and optimal control theory: A concise introduction*. Princeton University Press, 2011.

[111] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[112] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[113] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[114] Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pages 4042–4050, 2019.

[115] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B Smith, James M Rehg, and Le Song. Iterative machine teaching. In *International Conference on Machine Learning*, pages 2149–2158, 2017.

[116] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[117] Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–122, 2018.

[118] Thodoris Lykouris, Max Simchowitz, Aleksandrs Slivkins, and Wen Sun. Corruption robust exploration in episodic reinforcement learning. *arXiv preprint arXiv:1911.08689*, 2019.

[119] Yuzhe Ma, Kwang-Sung Jun, Lihong Li, and Xiaojin Zhu. Data poisoning attacks in contextual bandits. In *Conference on Decision and Game Theory for Security (GameSec)*, 2018.

[120] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. In *Advances in Neural Information Processing Systems*, pages 14570–14580, 2019.

[121] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[122] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[123] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[124] Francisco S Melo. Convergence of q-learning: A simple proof.

[125] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[126] Gina Neff and Peter Nagy. Automation, algorithms, and politics| talking to bots: Symbiotic agency and the case of tay. *International Journal of Communication*, 10:17, 2016.

[127] Gergely Neu, András György, and Csaba Szepesvári. The online loop-free stochastic shortest-path problem. In *COLT*, volume 2010, pages 231–243. Citeseer, 2010.

[128] Gergely Neu, Andras Gyorgy, and Csaba Szepesvári. The adversarial stochastic shortest path problem with unknown transition probabilities. In *Artificial Intelligence and Statistics*, pages 805–813, 2012.

[129] Gergely Neu and Julia Olkhovskaya. Online learning in mdps with linear function approximation and bandit feedback. *arXiv preprint arXiv:2007.01612*, 2020.

[130] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 83–93. ACM, 2014.

[131] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[132] Melkior Ornik and Ufuk Topcu. Learning and planning for time-varying mdps using maximum likelihood estimation. *arXiv preprint arXiv:1911.12976*, 2019.

[133] Ronald Ortner, Pratik Gajane, and Peter Auer. Variational regret bounds for reinforcement learning. In *UAI*, page 16, 2019.

[134] Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. *Advances in Neural Information Processing Systems*, 27:1466–1474, 2014.

[135] Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

[136] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[137] Kaustubh Patil, Xiaojin Zhu, Lukasz Kopec, and Bradley Love. Optimal teaching for limited-capacity human learners. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[138] Michael A Patterson and Anil V Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[139] Tomi Peltola, Mustafa Mert Çelikok, Pedram Daee, and Samuel Kaski. Machine teaching of active sequential learners. In *Advances in Neural Information Processing Systems*, pages 11202–11213, 2019.

[140] Ankit Pensia, Varun Jog, and Po-Ling Loh. Robust regression with covariate filtering: Heavy tails and adversarial contamination. *arXiv preprint arXiv:2009.12976*, 2020.

[141] Ian R Petersen, Valery A Ugrinovskii, and Andrey V Savkin. *Robust Control Design Using H-∞ Methods*. Springer Science & Business Media, 2012.

[142] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[143] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[144] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

[145] Aviv Rosenberg and Yishay Mansour. Online convex optimization in adversarial markov decision processes. *arXiv preprint arXiv:1905.07773*, 2019.

[146] Sosale Shankara Sastry and Alberto Isidori. Adaptive control of linearizable systems. *IEEE Transactions on Automatic Control*, 34(11):1123–1131, 1989.

[147] Jürgen Schmidhuber. 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*, pages 222–227, 1991.

[148] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[149] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[150] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[151] Ayon Sen, Scott Alfeld, Xuezhou Zhang, Ara Vartanian, Yuzhe Ma, and Xiaojin Zhu. Training set camouflage. In *Conference on Decision and Game Theory for Security (GameSec)*, 2018.

[152] Ayon Sen, Purav Patel, Martina A. Rau, Blake Mason, Robert Nowak, Timothy T. Rogers, and Xiaojin Zhu. Machine beats human at sequencing visuals for perceptual-fluency practice. In *Educational Data Mining*, 2018.

[153] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[154] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[155] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[156] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 2898–2933. PMLR, 2019.

[157] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, pages 1057–1063, 1999.

[158] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[159] Daniela Tonon, Maria Soledad Aronna, and Dante Kalise. *Optimal control: Novel directions and applications*, volume 2180. Springer, 2017.

[160] Joel A Tropp. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

[161] John N Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[162] John W. Tukey. A survey of sampling from contaminated distributions. STRG Technical report 33, 1959.

[163] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[164] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[165] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[166] Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

[167] Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

[168] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[169] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[170] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015.

[171] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[172] Yasin Abbasi Yadkori, Peter L Bartlett, Varun Kanade, Yevgeny Seldin, and Csaba Szepesvári. Online learning in markov decision processes with adversarially chosen transition probability distributions. In *Advances in neural information processing systems*, pages 2508–2516, 2013.

[173] Lin Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004. PMLR, 2019.

[174] Lin F Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. *arXiv preprint arXiv:1905.10389*, 2019.

[175] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. *arXiv preprint arXiv:2102.05815*, 2021.

[176] Ming Yin, Yu Bai, and Yu-Xiang Wang. Near optimal provable uniform convergence in off-policy evaluation for reinforcement learning. *arXiv preprint arXiv:2007.03760*, 2020.

[177] Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. *arXiv preprint arXiv:2102.01748*, 2021.

[178] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

[179] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

[180] Andrea Zanette, David Brandfonbrener, Emma Brunskill, Matteo Pirotta, and Alessandro Lazaric. Frequentist regret bounds for randomized least-squares value iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 1954–1964, 2020.

[181] Andrea Zanette, Ching-An Cheng, and Alekh Agarwal. Cautiously optimistic policy optimization and exploration with linear function approximation. *arXiv preprint arXiv:2103.12923*, 2021.

[182] Haoqi Zhang and David C Parkes. Value-based policy teaching with active indirect elicitation. 2008.

[183] Haoqi Zhang, David C Parkes, and Yiling Chen. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 295–304, 2009.

[184] Kaiqing Zhang, Bin Hu, and Tamer Basar. On the stability and convergence of robust adversarial reinforcement learning: A case study on linear quadratic systems. *Advances in Neural Information Processing Systems*, 33, 2020.

[185] Kaiqing Zhang, Bin Hu, and Tamer Basar. Policy optimization for h-2 linear control with h-∞ robustness guarantee: Implicit regularization and global convergence. In *Learning for Dynamics and Control*, pages 179–190. PMLR, 2020.

[186] Kaiqing Zhang, Xiangyuan Zhang, Bin Hu, and Tamer Başar. Derivative-free policy optimization for risk-sensitive and robust control design: Implicit regularization and sample complexity. *arXiv preprint arXiv:2101.01041*, 2021.

[187] Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Robust policy gradient against strong data corruption. *arXiv preprint arXiv:2102.05800*, 2021.

[188] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. *arXiv preprint arXiv:2003.12613*, 2020.

[189] Xuezhou Zhang and Xiaojin Zhu. Online data poisoning attack. *arXiv preprint arXiv:1903.01666*, 2019.

[190] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103. ACM, 2018.

[191] Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.

[192] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

[193] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

[194] Xiaojin Zhu. Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence* (*AAAI "Blue Sky" Senior Member Presentation Track*), 2015.

[195] Xiaojin Zhu. An optimal control view of adversarial machine learning. *arXiv preprint arXiv:1811.04422*, 2018.

[196] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. An Overview of Machine Teaching. *ArXiv e-prints*, January 2018. https://arxiv.org/abs/1801.05927.

[197] Alexander Zimin and Gergely Neu. Online learning in episodic markovian decision processes by relative entropy policy search. In *Advances in neural information processing systems*, pages 1583–1591, 2013.

[198] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning* (*icml-03*), pages 928–936, 2003.