Guaranteed Policy Performance in Reinforcement Learning

Thesis by
Cameron Voloshin

In Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

California Institute of Technology
Pasadena, California

2024
Defended June 13, 2023

Cameron Voloshin
ORCID: 0009-0007-7725-6660
cvoloshin.com

ACKNOWLEDGEMENTS

To my loving mother, Mila Voloshin.

I am deeply grateful to my immediate family, Oleg, Etana Voloshin, and Danielle Bronner. You have been an incredible support system, and your unwavering belief in me has pushed me to strive for excellence.

To all of the remarkable people that I have had the privilege of calling close friends: Thomas Anderson, Ada Campagna, Jessica Vautor, Joe Slote, Eitan Levin, Dima Burov, Terry Gdoutos, Jacob Gilman, Max Obolsky, Colin Miller, Jessica Wang, Melodie Kao, Rogelio Gomez, I want to express my heartfelt gratitude. You have been there for me through the highs and lows, and I have learned tremendously from each and every one of you. Your unique and special presence in my life have made this thesis possible.

Without any doubt, I want to extend my sincerest appreciation to my advisor, Yisong Yue. Your guidance, support, insights, and encouragement before and throughout my PhD adventure have been invaluable. You have made my dreams possible many times over. My gratitude also goes out to the remaining members of my thesis committee, Adam Wierman, Katie Bouman, and Swarat Chaudhuri, for their insightful feedback and suggestions.

I would like to thank my other collaborators, Hoang Le, Abhinav Verma, and Nan Jiang, as well as my Argo internship coworkers Andrew Hartnett and Matthew Hausknecht. Working with such amazing individuals has allowed me to learn and grow tremendously.

It has also been a privilege to engage in discussions with my current and former Caltech colleagues, whose names are too numerous to mention individually. You all have contributed to creating a supportive and collaborative research environment.

To all of you, thank you for being there for me and allowing me to lean on you. Your presence, guidance, and contributions have been instrumental in my journey, and I am truly grateful.

ABSTRACT

Decision-making is ubiquitous in everyday life. Increasingly, researchers are seeking answers on how to optimally solve sequential decision-making tasks. Thanks to recent availability of computation, advances in deep learning, and released open-sourced code, it has become easy to train a computational agent to make decisions in many domains. Nevertheless, in realistic scenarios where the consequences of failure are high, running a trained computational agent in the wild poses substantial risk.

The goal of this thesis is to develop and advance techniques that guarantee a learned agent does what we expect it to do. The thesis tackles two central questions:

1) Given an agent, how can we predict if it will perform desirably?

2) Can we structure the learning process to guarantee desirable post-learning performance?

On the former question, this thesis proposes multiple algorithms to evaluate such agents, finds factors that have high influence on the success of agent evaluation, and open-sources benchmarks for further development in the space.

On the latter question, this thesis formulates desirable agent behavior as a constrained optimization with varying types of constraints depending on the structure afforded to the practitioner. Constraining the search space over the learning process ensures post-learning behaviors will, by definition, perform as desired.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Cameron Voloshin, Abhinav Verma, and Yisong Yue. "Eventual Discounting Temporal Logic Counterfactual Experience Replay". In: *International Conference on Machine Learning (ICML)*. 2023.
CV participated in the conception of the project, formulated, implemented and analyzed the method, prepared the data, conducted the experiments, and led in the writing of the manuscript.

[2] Cameron Voloshin, Hoang Minh Le, Swarat Chaudhuri, and Yisong Yue. "Policy Optimization with Linear Temporal Logic Constraints". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=yZcPRIZEwOG.
CV participated in the conception of the project, formulated, implemented and analyzed the method, prepared the data, conducted the experiments, and led in the writing of the manuscript.

[3] Cameron Voloshin, Nan Jiang, and Yisong Yue. "Minimax Model Learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1612–1620.
CV participated in the conception of the project, formulated, implemented and analyzed the method, prepared the data, conducted the experiments, and led in the writing of the manuscript.

[4] Cameron Voloshin, Hoang Minh Le, Nan Jiang, and Yisong Yue. "Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. 2021. URL: https://openreview.net/forum?id=IsK8iKbL-I.
CV participated in the conception of the project, formulated, implemented and analyzed the methods, prepared the data, conducted the experiments, and led in the writing of the manuscript.

[5] Hoang M Le, Cameron Voloshin, and Yisong Yue. "Batch Policy Learning under Constraints". In: *International Conference on Machine Learning (ICML)*. 2019.
CV participated in the formulation, implementation and experimentation in the project, and in the writing of the manuscript.

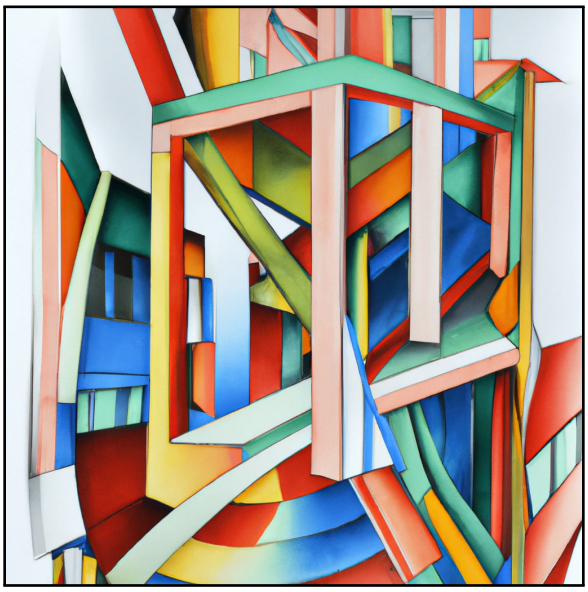
TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Part I

INTRODUCTION



Chapter 1

MOTIVATION AND GOAL

This chapter motivates and presents the goal and developments of this thesis.

Simply put, solving a sequential decision-making task involves finding a sequence of actions over time that achieve a desired outcome. It is easy to appreciate the wide range of tasks to which this concept applies, such as traffic routing, power system management, recommendation systems, financial trading, motion planning in autonomous vehicles and robotics, games like chess, and even simple tasks like making a sandwich. The list is abundant.

Beyond some basic systems, classical solutions have become impractical due to the extensive search space they require. Increasingly, researchers and practitioners have leaned on learning-based approaches, fueled by superhuman levels of computation, leveraging success in deep learning including computer vision [112, 49], and increasingly language models [213]. These powerful deep learning advances have contributed to dramatic successes in sequential decision domains, most notably in games [144, 185] and recommendation [127].

However, many decision-making domains involve substantial risks to human life, hardware, and resources. This raises a fundamental question when deploying "learned agents" capable of interacting with the world:

How can we be sure that the agent will succeed? (Performance Guarantee)

As of the writing of this thesis, providing a satisfying answer to this question remains elusive, limiting the applicability of learning-based approaches to systems where repeated failure is tolerable.

In the following sections, we present our attempt to address this fundamental question. Our approach consists of two key steps:

1) We will massage any existing data to extract meaningful signal on our agent's expected performance.

2) We will re-frame the learning process to proactively ensure satisfactory performance of the agent.

1.1 Guarantee from existing data: Off Policy Evaluation (OPE)

A key approach to providing a performance guarantee for a learned agent, particularly when the cost of deployment is high, involves leveraging existing data. This data may include human demonstrations, or limited deployments of classical or conservative agents.

The problem of estimating the performance of a new agent using pre-collected historical data generated by other agents is known as the Off-Policy Policy Evaluation (OPE) problem.

In practice, the dataset contains actions chosen by a different agent from the one being evaluated. Had we run the new agent in the world we may have observed different actions and outcomes than the ones available in the dataset. Accounting for this distribution shift poses the fundamental challenge in OPE. Ignoring this shift can result in poor estimation of agent performance.

Given its importance, the research community actively advances OPE techniques, both for the bandit [54, 24, 196, 219, 128, 138] and reinforcement learning settings [97, 54, 60, 133, 223, 209, 152, 225, 44]. These new developments reflect practical interests in deploying reinforcement learning to safety-critical situations [127, 220, 24, 17], and the increasing importance of off-policy learning and counterfactual reasoning [47, 203, 149, 123, 135, 155]. OPE is also similar to the dynamic treatment regime problem in the causal inference literature [151].

In Part II, our work focuses on OPE. Chapter 3 surveys and benchmarks the existing techniques for OPE, teasing out some of the key factors that have high influence on method performance. We summarize guidelines for method selection depending on the circumstance and open source an extensive package for new OPE experimentation and development. We also outline further directions remaining in OPE development. To the best of our knowledge, this is the first comprehensive study on OPE.

Chapter 4 restricts its attention to model-based OPE. The chapter argues that current model-based approaches for OPE have an incomplete learning objective and do not properly correct for distribution shift. We develop a novel method known as Minimax Model Learning (MML), provide theoretical guarantees

on its predictive performance, and demonstrate empirical improvement over previous methods. We also show that this idea can, additionally, be extended to do learning, which we will get to in the next section.

Similarly, Chapter 5 attends to model-free OPE. Here we develop Fitted Q Evaluation (FQE), a simple method for OPE and derive guarantees on its predictive performance. It is demonstrably more reliable than previous model-free methods and significantly easier to implement.

1.2 Guarantee from restructuring learning: Learning with Guarantees

A policy serves as the decision system that guides an agent's actions within the world. The interaction protocol follows a sequence of steps: the world reveals a context, the agent selects an action based on that context using its policy, the agent executes the action and experiences a cost, the world is influenced by the action, and the next context is revealed, and so on. Typically, a policy maps previous contexts to a distribution of possible actions. Common assumptions include the policy depending only on the last context (Markovian) and the environment providing all relevant information (fully observable). This process is formalized as a Markov Decision Process (MDP) [166] and is foundational to the reinforcement learning framework [190, 21].

At a high level, the primary objective of the interaction loop is to find sequences of actions that minimize the agent's accumulated cost, leading to optimal solutions. By achieving this, the agent is expected to demonstrate desired behaviors and successfully accomplish whatever task is implied within the costs. Indeed, the underlying premise, known as the "reward hypothesis" [194, 186], posits that a scalar cost function fully captures the task specification. However, designing a single appropriate scalar cost function can be excessively challenging, particularly in complex real-world systems like self-driving cars. These tasks often involve numerous constraints, like safety, reliability, and comfort, in addition to a main objective like reaching a destination. In the context of the reward hypothesis, these constraints need to be integrated into the scalar cost function. This blending of the main objective and real-world constraints introduces complexity in both guaranteeing and validating the success of the agent's behavior.

In fact, designing a cost (reward) function can be so hard that there is an entire effort to investigate learning a cost function [79, 63, 233, 188]. Finding

the correct cost function can be as hard as solving the problem itself [154]. Recent theoretical evidence suggests that certain tasks are simply not reducible to scalar costs [2]. Nevertheless, to date, almost all theoretical understanding of RL is focused on this scalar cost minimization setting (e.g., [207, 100, 101, 197, 160, 14, 72, 45, 15, 6, 7, 126, 167, 168]).

In practice, one circumvents these challenges using heuristics such as adding "breadcrumbs", known as cost-shaping [187], to guide the agent to qualitatively behave the way the practitioner intended. However, such heuristics can lead to catastrophic failures in which the learning agent ends up exploiting the cost function in an unanticipated way [171, 206, 93, 227, 154]. Additionally, the process of cost-shaping often requires iterative refinement, as practitioners are unlikely to achieve the desired behavior on their first attempt. Consequently, this iterative process is only viable when the cost of failure is relatively low. As far as guarantees are concerned, a cost-shaped function has lost precision in specifying the desired behavior.

To address these limitations and pitfalls, Part III of this thesis focuses on reconfiguring the learning problem without relying on the reward hypothesis. Instead, we explore two alternative approaches:

1) Value-Based Guarantees

2) Linear Temporal Logic (LTL)-Based Guarantees

Learning with Value Based Guarantees In this approach, the domain expert decomposes the task into multiple simultaneous constraints, each associated with a cost function and a corresponding value function (cumulative cost). The objective is to keep each value function below its respective threshold, which may be determined by prior domain knowledge or historical data.

Chapter 6 presents an approach that allows the imposition of value-based constraints in a counterfactual manner when historical data already exists, without the need for further agent-world interaction. We provide end-to-end finite-sample performance guarantees, leveraging Off-Policy Policy Evaluation (Part II) to reason about constraint satisfaction.

Learning with LTL-Based Guarantees

In this setting, the domain expert specifies the task using the language of Linear Temporal Logic (LTL). Unlike a scalar cost function, LTL enables precise description of a task and further enables transparent blending of any additional constraints into a single specification. When desired, a cost function can be defined to discern the "best" LTL-satisfying policy, based on factors like time, energy, or effort.

Chapter 7 introduces an approach to finding (cost-optimal) LTL-satisfying policies in discrete context systems with access to a generative model of the world. Strong guarantees are provided with minimal assumptions. We develop several theoretical tools for our analyses and empirically validate the strength of our method.

In Chapter 8, we address the scenario where a generative model is not available, and an LTL-satisfying policy needs to be learned through online interaction with the environment. We develop a proxy value function and a technique called LTL-guided counterfactual experience replay (LCER) to overcome the challenges of sparse learning signals, allowing us to take advantage of specific structure afforded to us by LTL over traditional cost-functions. Empirical evaluations demonstrate the performance gains of LCER with various reinforcement learning algorithms.

Chapter 2

FOUNDATIONS

This chapter introduces the reader to the preliminaries underlying this thesis.

2.1 Preliminaries to Policy Learning, and Markov Decision Processes

As discussed in the previous chapter, sequential decision making involves iterated interaction between agent and world. Formally, this is abstracted to discounted MDP framework specified by a tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the state (context) space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function of the world, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta([-\bar{R}, \bar{R}])$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. It may sometimes be convenient to consider $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \Delta([-\bar{C}, \bar{C}])$, a cost function instead of a reward function. Here, $\bar{C}, \bar{R} \in \mathbb{R}$. Let $\mathcal{X} \equiv \mathcal{S} \times \mathcal{A}$.

Given an MDP, a (stochastic) policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ and a starting state distribution $d_0 \in \Delta(\mathcal{S})$ together determine a distribution over trajectories of the form $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$, where $s_0 \sim d_0, a_t \sim \pi(s_t), r_t \sim \mathcal{R}(s_t, a_t)$, and $s_{t+1} \sim P(s_t, a_t)$ for $t \geq 0$.

The standard optimization objective for policy learning is to optimize the value function (or cumulative reward) defined as:

$$V^P(\pi) \equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_i \middle| \begin{smallmatrix} s_0 \sim d_0 \\ a_i \sim \pi(\cdot|s_i) \\ s_{i+1} \sim P(\cdot|s_i, a_i) \\ r_i \sim \mathcal{R}(\cdot|s_i, a_i) \end{smallmatrix}\right], \tag{2.1}$$

over all π in some policy class Π.

It will occasionally be convenient to keep track of the action-value function defined as:

$$Q_\pi(s, a) \equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_i \middle| \begin{smallmatrix} s_0 = s, a_0 = a \\ a_i \sim \pi(\cdot|s_i) \\ s_{i+1} \sim P(\cdot|s_i, a_i) \\ r_i \sim \mathcal{R}(\cdot|s_i, a_i) \end{smallmatrix}\right], \quad Q_\pi(s, \pi(s)) \equiv E_{a \sim \pi(s)}\left[Q_\pi(s, a)\right].$$

It is easy to see that $V^P(\pi) = E_{s \sim d_0}[Q_\pi(s, \pi(s))] = E_{s \sim d_0}\left[E_{a \sim \pi(s))}\left[Q_\pi(s, a)\right]\right]$.

2.2 Preliminaries to OPE

Since policies are selected by optimizing Eq (2.1), we can evaluate it using the same metric:

$$V^P(\pi) \equiv J(\pi, P) \equiv E_{s\sim d_0}[V_\pi^P(s)], \tag{2.2}$$

where we can recursively write V using the Bellman Equation,

$$V_\pi^P(s) \equiv E_{a\sim\pi(\cdot|s)}[E_{r\sim\mathcal{R}(\cdot|s,a)}[r] + \gamma E_{\tilde{s}\sim P(\cdot|s,a)}[V_\pi^P(\tilde{s})]]. \tag{2.3}$$

A useful equivalent measure of performance is:

$$J(\pi, P) = E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{r\sim\mathcal{R}(\cdot|s,a)}[r]], \tag{2.4}$$

where $d_{\pi,\gamma}^P(s,a) \equiv \sum_{t=0}^\infty \gamma^t d_{\pi,t}^P(s,a)$ is the (discounted) distribution of state-action pairs induced by running π in P and $d_{\pi,t}^P \in \Delta(\mathcal{X})$ is the distribution of (s_t, a_t) induced by running π under P. The first term in $d_{\pi,\gamma}^P$ is $d_{\pi,0}^P = d_0$. $d_{\pi,t}^P$ has a recursive definition that we use in Section 4.3:

$$d_{\pi,t}^P(s,a) = \int d_{\pi,t-1}^P(\tilde{s},\tilde{a}) P(s|\tilde{s},\tilde{a}) \pi(a|s) d\nu(\tilde{s},\tilde{a}), \tag{2.5}$$

where ν is the Lebesgue measure.

In the batch learning setting, we are given a dataset $D = \{(s_i, a_i, s_i')\}_{i=1}^n$, where $s_i \sim d_{\pi_b}(s)$, $a_i \sim \pi_b$, and $s_i' \sim P(\cdot|s_i, a_i)$, where π_b is some behavior policy that collects the data. For convenience, we write $(s, a, s') \sim D_{\pi_b} P$, where $D_{\pi_b}(s,a) = d_{\pi_b}(s)\pi_b(a|s)$.

The OPE objective is to estimate:

$$J(\pi, P^*) \equiv V^{P^*}(\pi) \equiv E\left[\sum_{i=0}^\infty \gamma^i r_i \middle| \begin{smallmatrix} s_0\sim d_0 \\ a_i\sim\pi(\cdot|s_i) \\ s_{i+1}\sim P^*(\cdot|s_i,a_i) \\ r_i\sim\mathcal{R}(\cdot|s_i,a_i) \end{smallmatrix}\right], \tag{2.6}$$

the performance of an evaluation policy π in the true environment P^*, using only logging data D with samples from $D_{\pi_b} P^*$. Solving this objective is difficult because the actions in our dataset were chosen with π_b rather than π. Thus, any $\pi \neq \pi_b$ potentially induces a "shifted" state-action distribution $D_\pi \neq D_{\pi_b}$, and ignoring this shift can lead to poor estimation.

OPE is sometimes considered in the episodic RL setting. In this situation, $D = \{\tau^i\}_{i=1}^N$, a set of N trajectories (or episodes), where i indexes over trajectories, and $\tau^i = (s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i)$ and T may be fixed or a random variable.

2.3 Preliminaries to Linear Temporal Logic

We give the necessary background and examples to understand Linear Temporal Logic. An *atomic proposition* is a variable that takes on a truth value. An

alphabet over a set of atomic propositions AP is given by $\Sigma = 2^{\text{AP}}$. For example, if AP $= \{x, y\}$ then $\Sigma = \{\{\}, \{x\}, \{y\}, \{x, y\}\}$. $\Delta(A)$ represents the set of probability distributions over a set A.

Running Example

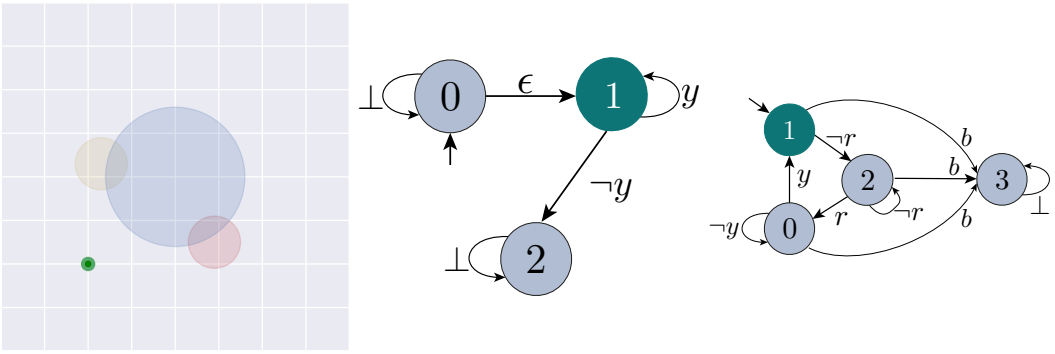


Figure 2.1: *Examples.* ***First:*** Illustration of the Flatworld environment. The agent is a green dot and there are 3 zones: yellow, blue and red. ***Second:*** LDBA $\mathcal{B}$ for "FGy". $\mathcal{S}^{\mathcal{B}^*} = \{1\}$, denoted by green circle. The initial state is $b_{-1} = 0$. ***Third:*** LDBA $\mathcal{B}$ for "$GF(y \,\&\, XFr) \,\&\, G\neg b$". $\mathcal{S}^{\mathcal{B}^*} = \{1\}$, denoted by green circle. The initial state is $b_{-1} = 1$.

We will be using the environment illustrated in Figure 2.1 (First) as a running example. The agent is given by a green dot and there are three circular regions in the environment colored yellow, blue, and red. The AP are given by $\{y, b, r\}$, referring to the respective colored zones. Elements of Σ indicate which zone(s) the agent is in.

MDPs with Labelled State Spaces

We extend an MDP from Section 2.1 to a labelled Markov Decision Process (MDP) framework given by the tuple $\mathcal{M} = (\mathcal{S}^{\mathcal{M}}, \mathcal{A}^{\mathcal{M}}, P^{\mathcal{M}}, d_0^{\mathcal{M}}, \gamma, L^{\mathcal{M}})$ consisting of a state space $\mathcal{S}^{\mathcal{M}}$, an action space $\mathcal{A}^{\mathcal{M}}$, an ***unknown*** transition function $P^{\mathcal{M}} : \mathcal{S}^{\mathcal{M}} \times \mathcal{A}^{\mathcal{M}} \to \Delta(\mathcal{S}^{\mathcal{M}})$, an initial state distribution $d_0^{\mathcal{M}} \in \Delta(\mathcal{S}^{\mathcal{M}})$, and a labelling function $L^{\mathcal{M}} : \mathcal{S}^{\mathcal{M}} \to \Sigma$. Let $A^{\mathcal{M}}(s)$ to be the set of available actions in state s.

Notice, unlike traditional MDPs, $\mathcal{M}$ has a labeling function $L^{\mathcal{M}}$ which returns the atomic propositions that are true in that state. For example, in Figure 2.1 (First), when the agent enters a state $s \in \mathcal{S}^{\mathcal{M}}$ such that it is both in the yellow and blue zone then $L^{\mathcal{M}}(s) = \{y, b\}$.

Linear Temporal Logic (LTL)

Here we give a basic introduction to LTL. For a more comprehensive overview, see Baier and Katoen [16].

Definition 2.3.1 (LTL Specification, φ)**.** An LTL specification φ is the entire description of the task, constructed from a composition of atomic propositions with logical connectives: not ($\neg$), and ($\&$), and implies ($\rightarrow$); and temporal operators: next (X), repeatedly/always/globally (G), eventually (F), and until (U).

Examples. For $AP = \{x, y\}$, some basic task specifications include safety ($G\neg x$), reachability (Fx), stability (FGx), response ($x \rightarrow Fy$), and progress ($x \ \& \ XFy$).

Consider again the environment in Figure 2.1 (First) where $AP = \{y, r, b\}$. If the task is to eventually reach the yellow zone and stay there (known as stabilization) then we write $\varphi = FGy$. Or, if we would like the agent to infinitely loop between the yellow and red zone while avoiding the blue zone then $\varphi = GF(y \ \& \ XFr) \ \& \ G\neg b$, a combination of safety, reachability, and progress.

LTL Satisfaction

LTL has recursive semantics defining the meaning for logical connective satisfaction. Without loss of generality, we will be using a specialized automaton, an LDBA $\mathcal{B}_\varphi$ [184], defined below to keep track of the progression of φ satisfaction. More details for constructing LDBAs are in [80, 16, 111]. We drop φ from $\mathcal{B}_\varphi$ for brevity.

Definition 2.3.2. (Limit Deterministic Büchi Automaton, LDBA [184]) An ***LDBA*** is a tuple $\mathcal{B} = (\mathcal{S}^\mathcal{B}, \Sigma \cup \mathcal{A}_\mathcal{B}, P^\mathcal{B}, \mathcal{S}^{\mathcal{B}^*}, b^\mathcal{B}_{-1})$ consisting of (i) a finite set of states $\mathcal{S}^\mathcal{B}$, (ii) a finite alphabet $\Sigma = 2^{\text{AP}}$, $\mathcal{A}_\mathcal{B}$ is a set of indexed jump transitions (iii) a transition function $P^\mathcal{B} : \mathcal{S}^\mathcal{B} \times (\Sigma \cup \mathcal{A}_\mathcal{B}) \rightarrow \mathcal{S}^\mathcal{B}$, (iv) accepting states $\mathcal{S}^{\mathcal{B}^*} \subseteq \mathcal{S}^\mathcal{B}$, and (v) initial state $b^\mathcal{B}_{-1}$. There exists a mutually exclusive partitioning of $\mathcal{S}^\mathcal{B} = \mathcal{S}^\mathcal{B}_D \cup \mathcal{S}^\mathcal{B}_N$ such that $\mathcal{S}^{\mathcal{B}^*} \subseteq \mathcal{S}^\mathcal{B}_D$, and for $b \in S^\mathcal{B}_D, a \in \Sigma$ then $P^\mathcal{B}(b, a) \subseteq \mathcal{S}^\mathcal{B}_D$, closed. $\mathcal{A}_\mathcal{B}(b)$ is only (possibly) non-empty for $b \in \mathcal{S}^\mathcal{B}_N$ and allows $\mathcal{B}$ to transition to $\mathcal{S}^\mathcal{B}_D$ without reading an AP. A *path* $\varrho = (b_0, b_1, \ldots)$ is a sequence of states in $\mathcal{B}$ reached through successive transitions under $P^\mathcal{B}$.

Definition 2.3.3. ($\mathcal{B}$ accepts) $\mathcal{B}$ **accepts** a path ϱ if there exists some state $b \in \mathcal{S}^{\mathcal{B}^*}$ in the path that is visited infinitely often.

Examples. Consider again the environment in Figure 2.1 (First) where $AP = \{y, r, b\}$. If we would like to make an LDBA for $\varphi = FGy$ (reach and stabilize at y) then we would get the state machine seen in Figure 2.1 (Second). In this state machine, the agent starts at state 0. The accepting set is given by $\mathcal{S}^{\mathcal{B}^*} = \{1\}$. The transition between state 0 and state 1 is what is formally referred to as a jump transition: $\mathcal{A}_{\mathcal{B}}(0) = \{\epsilon\}$ while $\mathcal{A}_{\mathcal{B}}(\cdot) = \varnothing$ otherwise. Whenever the agent is in state 0 of the LDBA, there is a choice of whether to stay at state 0 or transition immediately to state 1. This choice amounts to the agent believing that it has satisfied the "eventually" part of the LTL specification. When the agent takes this jump, then it must thereafter satisfy y to stay in state 1. The agent gets the decision of when it believes it is capable of satisfying y thereafter. When the agent takes the jump, if it fails to stay in y, it immediately transitions to the sink, denoted state 2. The LDBA accepts when the state 1 is reached infinitely often, meaning the agent satisfies "always y" eventually, as desired.

Another example, this time without jump transitions, would be for $\varphi = GF(y \;\&\; XFr) \;\&\; G\neg b$ (oscillate between y and r forever while avoiding b). The LDBA can be seen in Figure 2.1 (Third). In this state machine, the agent starts at state 1 and the accepting set is given by $\mathcal{S}^{\mathcal{B}^*} = \{1\}$. To make a loop back to state 1, the agent must visit both r and y. Doing so infinitely often satisfies the LDBA condition and therefore the specification. If at any point b is encountered then the agent transitions to the sink, denoted state 3.

We first introduce slightly more notation. Let $\mathcal{Z} = \mathcal{S}^{\mathcal{M}} \times \mathcal{S}^{\mathcal{B}}$. Let $\Pi : \mathcal{Z} \times \mathcal{A} \to \Delta([0, 1])$ be a (stochastic) policy class over the product space of the MDP and the LDBA (defined below), where $\mathcal{A}((s, b)) = \mathcal{A}^{\mathcal{M}}(s) \cup \mathcal{A}^{\mathcal{B}}(b)$, to account for jump transitions in $\mathcal{B}$.

Synchronizing the MDP with the LDBA. For any $(s, b) \in \mathcal{Z}$, a policy $\pi \in \Pi$ is able to select an action in $\mathcal{A}^{\mathcal{M}}(s)$ or an action in $\mathcal{A}^{\mathcal{B}}(b)$, if available. We can therefore generate a **trajectory** as the sequence $\tau = (s_0, b_0, a_0, s_1, b_0, a_1, \ldots)$

under a new probabilistic transition relation given by

$$P(s', b'|s, b, a) \quad = \quad \begin{cases} P^{\mathcal{M}}(s, a, s') & a \in A^{\mathcal{M}}(s), b' \in P^{\mathcal{B}}(b, L(s')) \\ 1, & a \in A^{\mathcal{B}}(b), b' \in P^{\mathcal{B}}(b, a), s = s' \\ 0, & \text{otherwise} \end{cases}. \quad (2.7)$$

Let the LDBA projection of τ be the subsequence $\tau_{\mathcal{B}} = (b_0, b_1, \ldots)$. Elements of $\tau_{\mathcal{B}}$ can be thought of as tracking an agent's LTL specification satisfaction:

Definition 2.3.4 (Run Satisfaction, $\tau \models \varphi$)**.** We say a trajectory satisfies φ if $\mathcal{B}$ accepts $\tau_{\mathcal{B}}$, which happens if $\exists b \in \tau_{\mathcal{B}}$ infinitely often with $b \in \mathcal{S}^{\mathcal{B}^*}$.

Let $\mathrm{T}_\pi^P = \mathbb{E}_{z \sim d_0^{\mathcal{M}} \times \{b_{-1}\}}[\mathrm{T}_\pi^P(z)]$ be the distribution over all possible trajectories starting from any initial state $z \in d_0^{\mathcal{M}} \times \{b_{-1}\}$ where $\mathrm{T}_\pi^P(z)$ is the (conditional) distribution over all possible trajectories starting from $z \in \mathcal{Z}$ generated by π under relation P (given in (2.7)). The probability of LTL satisfaction results from counting how many of the trajectories satisfy the LTL specification:

Definition 2.3.5 (State Satisfaction, $z \models \varphi$)**.** $\mathbb{P}_\pi[z \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P(z)}[\mathbf{1}_{\{\tau \models \varphi\}}] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\mathbf{1}_{\{\tau \models \varphi\}} | z_0 = z]$.

Definition 2.3.6 (Policy Satisfaction, $\pi \models \varphi$)**.** $\mathbb{P}[\pi \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\mathbf{1}_{\{\tau \models \varphi\}}]$ where $\mathbf{1}_X$ is the indicator for X.

2.4 Policy Learning With Guarantees

Optimizing the standard objective in Eq (2.1) requires that the whole task be encoded in the reward/cost function given by $\mathcal{R}$ or $\mathcal{C}$. Instead, we give a brief introduction to the problems we will be solving instead, which give more nuanced guarantee over policy performance.

Value-Based Guarantees

In the case there are real-world constraints which have their own individual cost functions to keep under control we may take an objective of the form:

$$\begin{aligned} &\arg\min_{\pi \in \Pi} \quad V(\pi) \\ &\text{s.t.} \quad G_1(\pi) \leq \tau_1, \ldots, G_k(\pi) \leq \tau_k, \end{aligned} \quad \text{(Value-Based Guarantees)}$$

where the k constraint value functions are $G_i^\pi \equiv \mathbb{E}_{s \sim d_0}\left[\sum_{t=0}^\infty \gamma^t g_i(s_t, a_t) | s_0 = s\right]$. Each $g_i : \mathcal{X} \to \mathbb{R}$ are known constraint cost functions and τ_i represents the allowable threshold. For examples of problems of this form, see Chapter 6.

LTL-Based Guarantees

Similarly, we can instead optimize over probability optimal policies in the form:

$$\arg\max_{\pi \in \Pi} \mathbb{P}\left[\pi \models \varphi\right] \qquad \text{(LTL-Based Guarantees)}$$

to find the policies that best satisfy our task, encoded in an LTL formula φ. For examples of problems of this form, see Chapter 8.

Further, we could write this as a constrained problem in the form:

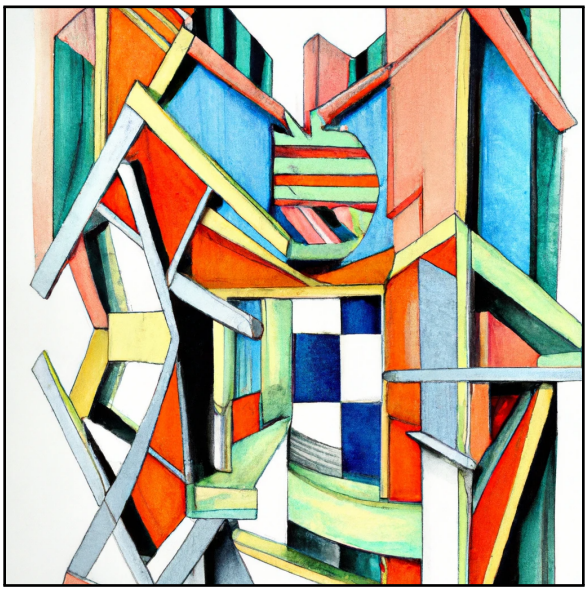
$$\arg\max_{\pi \in \Pi} V(\pi) \qquad \text{(LTL-Based Guarantees, with auxiliary cost)}$$
$$\text{s.t.} \quad \pi \in \{\arg\max_{\pi' \in \Pi} \mathbb{P}\left[\pi' \models \varphi\right]\}.$$

For examples of problems of this form, see Chapter 7.

Remark 2.4.1. *The policy class* Π *is different between the Value-Based Guarantees and LTL-Based Guarantees. In the former,* $\Pi : \mathcal{S} \to \Delta(\mathcal{A})$ *are functions from MDP states to MDP actions. In the latter,* $\Pi : (\mathcal{S}^{\mathcal{M}} \times \mathcal{S}^{\mathcal{B}}) \to \Delta(\mathcal{A}^{\mathcal{M}} \cup \mathcal{A}^{\mathcal{B}})$ *are functions from states to actions in the product MDP given by synchronizing a labelled MDP with an LDBA.*

Part II

OFF POLICY EVALUATION



Chapter 3

CALTECH OPE BENCHMARK AND EMPIRICAL STUDY

In this chapter, we develop an experimental benchmark and empirical study for off-policy policy evaluation (OPE) in reinforcement learning, which is a key problem in many safety critical applications. Given the increasing interest in deploying learning-based methods, there has been a flurry of recent proposals for OPE method, leading to a need for standardized empirical analyses. Our work takes a strong focus on diversity of experimental design to enable stress testing of OPE methods. We provide a comprehensive benchmarking suite to study the interplay of different attributes on method performance. We also distill the results into a summarized set of guidelines for OPE in practice. Our software package, the Caltech OPE Benchmarking Suite (COBS), is open-sourced and we invite interested researchers to further contribute to the benchmark.

We present the **Caltech OPE Benchmarking Suite (COBS)**, which benchmarks OPE techniques via experimental designs that give thorough considerations to factors that influence performance. The reality of method performance, as we will discuss, is nuanced and comparison among different estimators is tricky without pushing the experimental conditions along various dimensions. Our philosophy and contributions can be summarized as follows:

- We establish a benchmarking methodology that considers key factors that influence OPE performance, and design a set of domains and experiments to systematically expose these factors. The proposed experimental domains are complementary to continuous control domains from recent offline RL benchmarks [69, 68]. We differ from these recent benchmarks in two important ways:

  1) COBS allows researchers fine-grained control over experimental design, other than just access to a pre-collected dataset. The offline data can be generated "on-the-fly" based on experimental criteria, e.g., the divergence between behavior and target policies.

2) We offer significant diversity in experimental domains, covering a wide range of dimensionality and stochasticity. Together, the goal of this greater level of access is to enable a deeper look at when and why certain methods work well.

- As a case study, we select a representative set of established OPE baseline methods, and test them systematically. We further show how to distill the empirical findings into key insights to guide practitioners and inform researchers on directions for future exploration.
- COBS is an extensive software package that can interface with new environments and methods to run new OPE experiments at scale.[1] Given the fast-changing nature of this active area of research, our package is designed to accommodate the rapidly growing body of OPE estimators. COBS is already actively used by multiple research groups to benchmark new algorithms.

Prior Work. Empirical benchmarks have long contributed to the scientific understanding, advancement, and validation of machine learning techniques [37, 35, 36, 177, 53, 49]. Recently, many have called for careful examination of empirical findings of contemporary deep learning and deep reinforcement learning efforts [89, 136]. As OPE is central to real-world applications of reinforcement learning, proper benchmarking is critical to ensure in-depth understanding and accelerate progress. While many recent methods are built on sound mathematical principles, a notable gap in the current literature is a standard for benchmarking empirical studies, with perhaps a notable exception from the recent DOPE [69] and D4RL benchmarks [68].

Compared to prior complementary work on OPE evaluation for reinforcement learning [68, 69], our benchmark offers two main advantages. First, we focus on maximizing reproducibility and nuanced experimental control with minimal effort, covering data generation and fine-grained control over factors such as relative "distance" between the offline data distribution and the distribution induced by evaluation policies. Second, we study a diverse set of environments, spanning range of desiderata such as stochastic-vs-deterministic and different representations for the same underlying environment. Together, these attributes enable our benchmarking suite to conduct systematic analyses of the method

[1] https://github.com/clvoloshin/COBS

performance under different scenarios, and provide a holistic summary of the challenges one may encounter in different scenarios.

Background & Notation. Recall from the preliminaries of Section 2.2, we are given an evaluation policy π_e, the OPE problem is to estimate the value $V(\pi_e) = \mathbb{E}_{s\sim d_0}\left[\sum_{t=0}^{T-1}\gamma^t r_t | s_0 = s\right]$, with $a_t \sim \pi_e(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$, $r_t \sim \mathcal{R}(s_t, a_t)$, and $d_0 \subseteq \mathcal{S}$ is the initial state distribution. For notational convenience in this chapter, we assume a fixed episode length of T.

3.1 Benchmarking Design & Methodology

Design Philosophy

The design philosophy of the Caltech OPE Benchmarking Suite (COBS) starts with the most prominent decision factors that can make OPE difficult. These factors come from both the existing literature and our own experimental study, which we will further discuss. We then seek to design experimental conditions that cover a diverse range of these factors. As a sub-problem within the broader reinforcement learning problem class, OPE experiments in existing literature gravitate towards commonly used RL domains. Unsurprisingly, the most common experiments belong to the Mujoco group of deterministic continuous control tasks [204], or discrete domains that operate via OpenAI Gym interface [31]. For OPE, high-dimensional domains such as Atari [20] appear less often, but are also natural candidates for OPE testing. We selectively pick from these domains as well as design new domains, with the goal of establishing refined control over the decision factors.

Design Factors. We consider several domain characteristics that are often major factors in performance of OPE methods:

1) *Horizon length.* Long horizons can lead to catastrophic failure in some OPE methods due to an exponential blow-up in some of their components [128, 97, 133].
2) *Reward sparsity.* Sparse rewards represent a difficult credit assignment problem in RL. This factor is often not emphasized in OPE, and arguably goes hand-in-hand with horizon length.[2]
3) *Environment stochasticity.* Popular RL domains such as Mujoco and Atari are mostly deterministic. This is a fundamental limitation in many existing empirical studies since many theoretical challenges to RL only surface in

[2]Considered in isolation, long horizons may not be an issue if the reward signal is dense.

a stochastic setting. A concrete example is the famous double sampling problem [46], which is not applicable in many contemporary RL benchmarks.

4) *Unknown behavior policy.* This is related to the source of the collected data. The data may come from one or a more policies which may not be known. For example, existing dataset benchmarks, such as D4RL [68] can be considered to come from an unknown behavior policy. Some methods will require behavior policy estimation, thus introducing some bias.
5) *Policy and distribution mismatch.* The relative difference between the evaluation and behavior policy can play a critical role in the performance of many OPE methods. This difference induces a distribution mismatch between the dataset D and the dataset that would have been produced had we run the evaluation policy. Performing out-of-distribution estimation is a key challenge for robust OPE. We focus on providing a systematic way to stress test OPE methods under this mismatch, which we accomplish by offering a control knob for flexible data generation to induce various degrees of mismatch.
6) *Model misspecification.* Model misspecification refers to the insufficient representation power of the function class used to approximate different objects of interest, whether the transition dynamics, value functions, or state distribution density ratio. In realistic applications, it is reasonable to expect at least some degree of misspecification. We study the effect of misspecification via two controlled scenarios: (i) we start with designing simple domains to test OPE methods under tabular representation and (ii) we test the same OPE methods and same tabular data generation process, but the input representation for OPE methods is now modified to expose the impact of choosing a different function class for representation.

Domains

Ultimately, many of the aforementioned factors are intertwined and their usefulness in evaluating OPE performance cannot be considered in isolation. However, they serve as a valuable guide in our selection of benchmark environments. To that end, our benchmark suite includes eight environments. We use two standard RL benchmarks from OpenAI [29]. As many standard RL benchmarks are fixed and deterministic, we design six additional environments that allow control over different design factors. Figure 3.1 depicts one such design factor: the representation complexity.

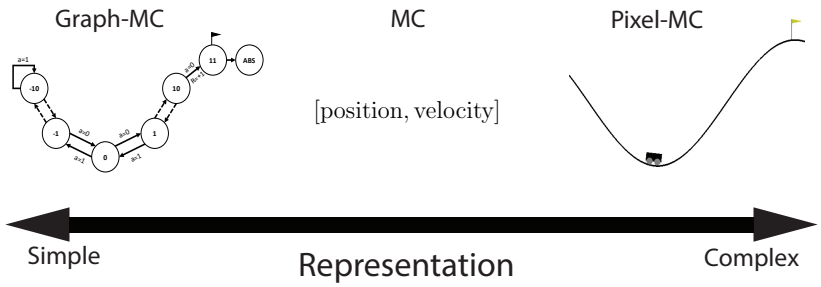


Figure 3.1: Depicting one of the dimensions which COBS provides control. For the Mountain Car environment, we can select either a tabular, standard coordinate-based, or pixel-based representation of the state while holding other factors fixed.

Graph: A flexible discrete environment that can vary in horizon, stochasticity, and sparsity.

Graph-POMDP: An extension of Graph to a POMDP setting, where selected information is omitted from the observations that form the behavior data. This enables controlled study of the effect of insufficient representation power relative to other settings in the Graph domain above.

Gridworld (GW): A gridworld design that offers larger state and action space than the Graph domains, longer horizon, and similarly flexible design choices for other environmental factors. Using some version of gridworld is standard across many RL experiments. Gridworld enables simple integration of various designs, and fast data collection.

Pixel-Gridworld (Pix-GW): A scaled-up domain from Gridworld which enables pixel-based representation of the state space. While such usage is not standard in existing literature, this design offers compelling advantage over many existing standard RL benchmarks. First, this domain enables simple controlled experiments to understand the impact of high-dimensional representation on OPE performance, where the ground truth of various quantities to be estimated is readily obtainable thanks to the access to underlying simpler grid representation. Second, this domain effectively simulates high-dimensional experiments with easily tuned experimental conditions, e.g., degree of stochasticity. This design freedom is not available with many currently standard RL benchmarks.

Mountain Car (MC): A standard control domain, which is known to have

challenging credit assignment due to sparsity of the reward. Our benchmark for this standard domains allows for function approximation to vary between a linear model and feed-forward neural network, in order to highlight the effects of model misspecification.

Pixel Mountain Car (Pix-MC): A modified version of Mountain Car where the state input is pixel-based, testing the methods' ability to work in high dimensional settings.

Tabular Mountain Car (Graph-MC): A simplified version of Mountain Car to a graph, allowing us to complete the test for model misspecification by considering the tabular case.

Atari (Enduro): A pixel-based Atari domain. Note that all Atari environments are deterministic and high-dimensional. Instead of choosing many different Atari domains to study, we instead opt to select Enduro as the representative Atari environment, due to its sparsity of reward (and commonly regarded as a highly challenging task). All Atari environments share similar interaction protocol, and can be seamlessly integrated into COBS, if desired.

All together, our benchmark consists of 8 environments with characteristics summarized in Table 3.1. Complete descriptions can be found in Appendix A.6. All environments have finite action spaces.

Table 3.1: Environment characteristics.

Environment	Graph	Graph-MC	MC	Pix-MC	Enduro	G-POMDP	GW	Pix-GW
Is MDP?	yes	yes	yes	yes	yes	no	yes	yes
State desc.	pos.	pos.	[pos, vel]	pixels	pixels	pos.	pos.	pixels
T	4 or 16	250	250	250	1000	2 or 8	25	25
Stoch Env?	variable	no	no	no	no	no	no	variable
Stoch Rew?	variable	no	no	no	no	no	no	no
Sparse Rew?	variable	terminal	terminal	terminal	dense	terminal	dense	dense
$\hat{Q}$ Class	tabular	tabular	linear/NN	NN	NN	tabular	tabular	NN
Initial state	0	0	variable	variable	gray img	0	variable	variable
Absorb. state	2T	22	[.5,0]	img([.5,0])	zero img	2T	64	zero img
Frame height	1	1	2	2	4	1	1	1
Frame skip	1	1	5	5	1	1	1	1

Experiment Protocol

Selection of Policies. We use two classes of policies. The first is state-independent with some probability of taking any available action. For example, in the Graph environment with two actions, $\pi(a = 0) = p, \pi(a = 1) = 1 - p$ where p is a parameter we can control. The second is a state-dependent

$\epsilon-$Greedy policy. We train a policy Q^* (using value iteration or DDQN [85]) and then vary the deviation away from the policy. Hence $\epsilon - Greedy(Q^*)$ implies we follow a mixed policy $\pi = \arg\max_a Q^*(s,a)$ with probability $1-\epsilon$ and uniform with probability ϵ. Here ϵ is a parameter we can control.

Most OPE methods explicitly require absolute continuity among the policies ($\pi_b > 0$ whenever $\pi_e > 0$). Thus, all policies will remain stochastic with this property maintained.

Data Generation & Metrics. Each experiment depends on specifying an environment and its properties, behavior policy π_b, evaluation policy π_e, and number of trajectories N from rolling-out π_b for historical data. The true on-policy value $V(\pi_e)$ is the Monte-Carlo estimate via $10,000$ rollouts of π_e. We repeat each experiment $m = 10$ times with different random seeds. We judge the quality of a method via two metrics:

- Relative mean squared error (*Relative MSE*): $\frac{1}{m}\sum_{i=1}^{m} \frac{(\widehat{V}(\pi_e)_i - \frac{1}{m}\sum_{j=1}^{m} V(\pi_e)_j)^2}{(\frac{1}{m}\sum_{j=1}^{m} V(\pi_e)_j)^2}$, which allows a fair comparison across different conditions.[3]
- *Near-top Frequency*: For each experimental condition, we include the number of times each method is within 10% of the best performing one to facilitate aggregate comparison across domains.

Implementation & Hyperparameters. COBS allows running experiments at scale and easy integration with new domains and techniques for future research. The package consists of many domains and reference implementations of OPE methods.

Hyperparameters are selected based on publication, code release or author consultation. We maintain a consistent set of hyperparameters for each estimator and each environment across experimental conditions (see hyperparameter choice in appendix Table A.7).[4]

Baselines

OPE methods were historically categorized into importance sampling methods, direct methods, or doubly robust methods. This demarcation was first intro-

[3]The metric used in prior OPE work is typically mean squared error: MSE= $\frac{1}{m}\sum_{i=1}^{m}(\widehat{V}(\pi_e)_i - V(\pi_e)_i)^2$.

[4]In practice, hyperparameter tuning is not practical for OPE due to a lack of validation signal.

duced for contextual bandits [54], and later extended to the RL setting [97]. Some recent methods have blurred the boundary of these categories. Examples include Retrace(λ) [149] that uses a product of importance weights of multiple time steps for off-policy Q correction, and MAGIC [201] that switches between importance weighting and direct methods. In this benchmark, we propose to group OPE into three similar classes of methods, but with an expanded definition for each category: Inverse Propensity Scoring, Direct Methods, and Hybrid Methods. For the current benchmark, we select representative established baselines from each category. Appendix A.5 contains a full description of all methods under consideration.

Inverse Propensity Scoring (IPS) We consider the main four variants: Importance Sampling (IS), Per-Decision Importance Sampling (PDIS), Weighted Importance Sampling (WIS), and Per-Decision WIS (PDWIS). IPS has a rich history in statistics [163, 81, 92], with successful crossover to RL [165]. The key idea is to reweight the rewards in the historical data by the importance sampling ratio between π_e and π_b, i.e., likelihood of reward under π_e versus π_b.

Direct Methods (DM) While some direct methods make use of importance weight adjustments, a key distinction of direct methods is the focus on regression-based techniques to (more) directly estimate the value functions of the evaluation policy (Q^{π_e} or V^{π_e}). This is an area of very active research with rapidly growing literature. We consider 8 different direct approaches, taken from the following respective families of direct estimators.

Model-based estimators Perhaps the most commonly used DM is *Model-based* (also called approximate model, denoted AM), where the transition dynamics, reward function and termination condition are directly estimated from historical data [97, 161]. The resulting learned MDP is then used to compute the value of π_e, e.g., by Monte-Carlo policy evaluation. There are also some recent variants of the model-based estimator, e.g., [228].

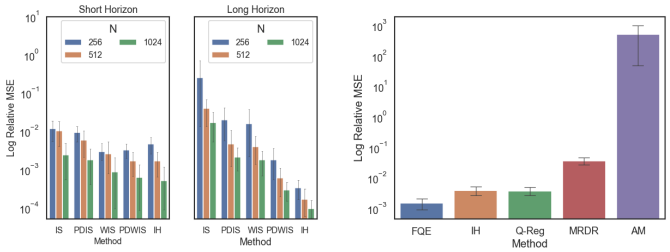
Value-based estimators *Fitted Q Evaluation (FQE)* is a model-free counterpart to AM, and is functionally a policy evaluation counterpart to batch Q learning [123]. *$Q^\pi(\lambda)$ & Retrace(λ) & Tree-Backup(λ)* Several model-free methods originated from off-policy learning settings, but are also natural for OPE. $Q^\pi(\lambda)$ [82] can be viewed as a generalization of FQE that looks to the horizon limit to incorporate the long-term value into the backup step. Retrace(λ) [149] and Tree-Backup(λ) [165] also use full trajectories, but additionally incorporate

varying levels of clipped importance weights adjustment. The λ-dependent term mitigates instability in the backup step, and is selected based on experimental findings of [149].

Regression-based estimators *Direct Q Regression (Q-Reg) & More Robust Doubly-Robust (MRDR)* [60] propose two direct methods that make use of cumulative importance weights in deriving the regression estimate for Q^{π_e}, solved through a quadratic program. MRDR changes the objective of the regression to that of directly minimizing the variance of the Doubly-Robust estimator.

Minimax-style estimators [133] recently proposed a method for the infinite horizon setting — we refer to this estimator as IH. While IH can be viewed as a Rao-Blackwellization of the IS estimator, we include it in the DM category because it solves the Bellman equation for state distributions and requires function approximation, which are more characteristic of DM. IH shifts the focus from importance sampling over action sequences to importance ratio between *state density distributions* induced by π_b and π_e. Starting with IH, this style of minimax estimator has recently attracted significant attention in OPE literature, including state-action extension of IH [209, 96] and DICE family of estimators [152, 229, 225, 44]. For our benchmarking purpose, we choose IH as the representative of this family.

Hybrid Methods (HM) Hybrid methods subsume doubly robust-like approaches, which combine aspects of both IPS and DM. Standard doubly robust OPE (denoted DR) [97] is an unbiased estimator that leverages DM to decrease the variance of the unbiased estimates produced by importance sampling techniques. Other HM include Weighted Doubly-Robust (WDR) and MAGIC. WDR replaces the importance weights with self-normalized importance weights (similar to WIS). MAGIC introduces adaptive switching between DR and DM; in particular, one can imagine using DR to estimate the value for part of a trajectory and then using DM for the remainder. Using this idea, MAGIC [201] finds an optimal linear combination among a set that varies the switch point between WDR and DM. Note that any DM that returns $\widehat{Q}^{\pi_e}(s, a; \theta)$ yields a set of corresponding DR, WDR, and MAGIC estimators. As a result, we consider 21 hybrid approaches in our experiments.



Method	Near-top Freq.
MAG. FQE	30.0%
DM FQE	23.7%
IH	19.0%
WDR FQE	17.8%
MAGIC $Q^\pi(\lambda)$	17.3%

Figure 3.2: Left: *(Graph domain) Comparing IPS (and IH) under short and long horizon. Mild policy mismatch setting. PDWIS is often best among IPS. But IH outperforms in long horizon.* Center: *(Pixel-MC) Comparing direct methods in high-dimensional, long horizon setting. Relatively large policy mismatch. FQE and IH tend to outperform. AM is significantly worse in complex domains. Retrace(λ), Q(λ) and Tree-Backup(λ) are very computationally expensive and thus excluded.* Right: *(Top Methods) The top 5 methods which perform the best across all conditions and domains.*

3.2 Empirical Evaluation

We evaluate 33 different OPE methods by running thousands of experiments across the 8 domains. Due to limited space, we show only the results from selected environmental conditions in the next section. The full detailed results, with highlighted best method in each class, are available in the appendix. The goal of the evaluation is to demonstrate the flexibility of the benchmark suite to systematically test the different factors of influence. We synthesize the results, and present further considerations and directions for research in Section 3.3.

What is the best method?

The first important takeaway is that *there is no clear-cut winner*: no single method or method class is consistently the best performer, as multiple environmental factors can influence the accuracy of each estimator. With that caveat in mind, based on the aggregate top performance metrics, we can recommend from our selected methods the following for each method class (See Figure 3.2 right, appendix Table A.13, and appendix Table A.1).

Inverse propensity scoring (IPS). In practice, weighted importance sampling, which is biased, tends to be more accurate and data-efficient than unbiased basic importance sampling methods. Among the four IPS-based estimators, *PDWIS tends to perform best* (Figure 3.2 left).

Direct methods (DM). Generally, FQE, $Q^\pi(\lambda)$, *and IH tend to perform the best among DM* (appendix Table A.1). FQE tends to be more data efficient and is the best method when data is limited (Figure 3.5). $Q^\pi(\lambda)$ generalizes

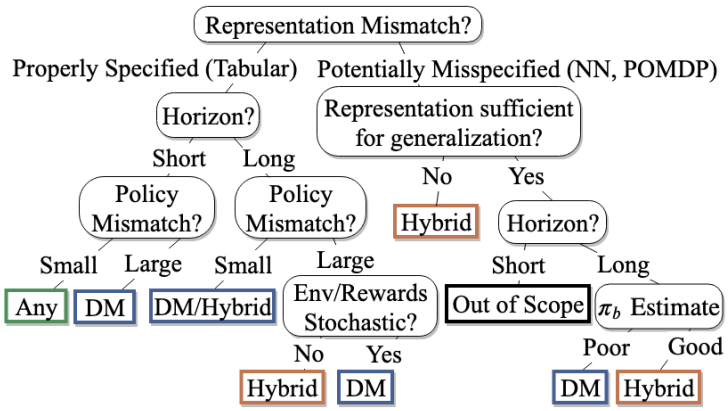


Figure 3.3: *General Guideline Decision Tree.*

FQE to multi-step backup, and works particularly well with more data, but is computationally expensive in complex domains. IH is highly competitive in long horizons and with high policy mismatch in a tabular setting (appendix Tables A.5, A.6). In pixel-based domains, however, choosing a good kernel function for IH is not straightforward, and IH can underperform other DM (appendix Table A.9). We provide a comparison among direct methods for tabular (appendix Figure A.13) and complex settings (Figure 3.2 center).

Hybrid methods (HM). With the exception of IH, each DM corresponds to three HM: standard doubly robust (DR), weighted doubly robust (WDR), and MAGIC. *For each DM, its WDR version often outperforms its DR version.* MAGIC can often outperform WDR and DR. However, MAGIC comes with additional hyperparameters, as one needs to specify the set of partial trajectory length to be considered. Unsurprisingly, their performance highly depends on the underlying DM. In our experiments, FQE and $Q^\pi(\lambda)$ are typically the most reliable: *MAGIC with FQE or MAGIC with $Q^\pi(\lambda)$ tend to be among the best hybrid methods* (see appendix Figures A.19 - A.23).

A recipe for method selection

Figure 3.3 summarizes our general guideline for navigating key factors that affect the accuracy of different estimators. To guide the readers through the process, we now dive further into our experimental design to test various factors, and discuss the resulting insights.

Do we potentially have representation mismatch? Representation mismatch comes from two sources: model misspecification and poor generalization. Model misspecification refers to the insufficient representation power of the

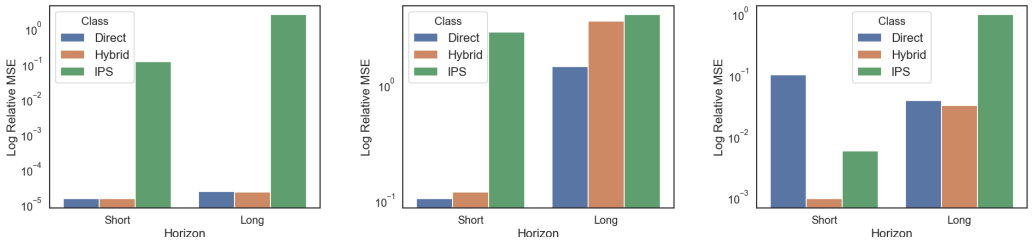


Figure 3.4: *Comparing IPS versus Direct methods versus Hybrid methods under short and long horizon, large policy mismatch and large data.* Left: *(Graph domain) Deterministic environment.* Center: *(Graph domain) Stochastic environment and rewards.* Right: *(Graph-POMDP) Model misspecification (POMDP). Minimum error per class is shown.*

function class used to approximate either the transition dynamics (AM), value function (other DM), or state distribution density ratio (in IH).

Having a tabular representation controls for representation mismatch by ensuring adequate function class capacity, as well as zero inherent Bellman error (left branch, Fig 3.3). In such cases, we may suffer from poor generalization without sufficient data coverage, which depends on other domain-specific factors.

The effect of representation mismatch (right branch, Fig 3.3) can be understood via two scenarios:

- *Misspecified and poor generalization:* We expose the impact of this severe mismatch scenario via the Graph POMDP construction, where selected information is omitted from an otherwise equivalent Graph MDP. Here, HM substantially outperforms DM (Figure 3.4 right versus left).
- *Misspecified but good generalization:* Function classes such as neural networks have powerful generalization ability, but may introduce bias and inherent Bellman error[5] [150, 38] (see linear vs. neural networks comparison for Mountain Car in appendix Fig A.10). Still, powerful function approximation makes (biased) DM very competitive with HM, especially under limited data and in complex domains (see pixel-Gridworld in appendix Fig A.24-A.26). However, function approximation bias may cause serious problems for high-dimensional and long horizon settings. In the extreme case of Enduro (very long horizon and sparse rewards), all DM fail to convincingly outperform a naïve average of behavior data (appendix Fig A.9).

[5]Inherent Bellman error is defined as $\sup_{g \in \mathrm{F}} \inf_{f \in \mathrm{F}} ||f - \mathbb{T}^{\pi} g||_{d_\pi}$, where F is function class chosen for approximation, and d_π is state distribution induced by evaluation policy π.

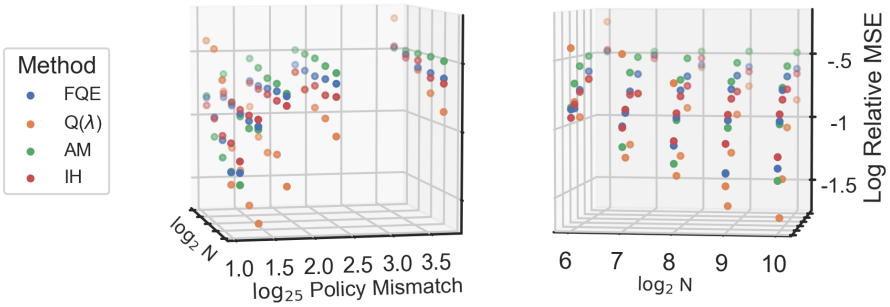


Figure 3.5: *(Gridworld domain) Errors are directly correlated with policy mismatch but inversely correlated with data size. We pick the best direct methods for illustration. The two plots represent the same figure from two different vantage points.*

Short horizon vs. Long horizon? It is well-known that IPS methods are sensitive to trajectory length [128]. Long horizon leads to an exponential blow-up of the importance sampling term, and is exacerbated by significant mismatch between π_b and π_e. This issue is inevitable for any unbiased estimator [97] (a.k.a., the curse of horizon [133]). Similar to IPS, DM relying on importance weights also suffer in long horizons (appendix Fig A.13), though to a lesser degree. IH aims to bypass the effect of cumulative weighting in long horizons, and indeed performs substantially better than IPS methods in very long horizon domains (Fig 3.2 left).

A frequently ignored aspect in previous OPE work is a proper distinction between fixed, finite horizon tasks (IPS focus), infinite horizon tasks (IH focus), and indefinite horizon tasks, where the trajectory length is finite but varies depending on the policy. Many applications should properly belong to the indefinite horizon category.[6] Applying HM in this setting requires proper padding of the rewards (without altering the value function in the infinite horizon limit) as DR correction typically assumes fixed length trajectories.

How different are behavior and target policies? Similar to IPS, the performance of DM is negatively correlated with the degree of policy mismatch. Figure 3.5 shows the interplay of increasing policy mismatch and historical data size, on the top DM in the deterministic gridworld. We use $(\sup_{a\in A, x\in X} \frac{\pi_e(a|s)}{\pi_b(a|s)})^T$ as an environment-independent metric of mismatch between the two policies. The performance of the top DM (FQE, $Q^\pi(\lambda)$, IH) tend to hold up better than

[6] Applying IH in the indefinite horizon case requires setting up an absorbing state that loops over itself with zero terminal reward.

IPS methods when the policy gap increases (appendix Figure A.15). FQE and IH are best in the small data regime, and $Q^\pi(\lambda)$ performs better as data size increases (Figure 3.5). Increased policy mismatch weakens the DM that use importance weights (Q-Reg, MRDR, Retrace(λ) and Tree-Backup(λ)).

Do we have a good estimate of the behavior policy? Often the behavior policy may not be known exactly and requires estimation, which can introduce bias and cause HM to underperform DM, especially in low data regime (e.g., pixel gridworld appendix Figure A.24-A.26). Similar phenomenon was observed in the statistics literature [104]. As the data size increases, HMs regain the advantage as the quality of the π_b estimate improves.

Is the environment stochastic or deterministic? While stochasticity affects all methods by straining the data requirement, HM are more negatively impacted than DM (Figure 3.4 center, Figure A.14). This can be justified by e.g., the variance analysis of DR, which shows that the variance of the value function with respect to stochastic transitions will be amplified by cumulative importance weights and then contribute to the overall variance of the estimator; see [97, Theorem 1] for further details. We empirically observe that DM frequently outperform their DR versions in the small data case (Figure A.14). In a stochastic environment and tabular setting, HM do not provide significant edge over DM, even in short horizon case. The gap closes as the data size increases (Figure A.14).

Challenging common wisdom

To illustrate the value of a flexible benchmarking tool, in this section we further synthesize the empirical findings and stress-test several commonly held beliefs about the high-level performance of OPE methods.

Is HM always better than DM? No. Overall, DM are surprisingly competitive with HM. Under high-dimensionality, long horizons, estimated behavior policies, or reward/environment stochasticity, HM can underperform simple DM, sometimes significantly (e.g., see appendix Figure A.14).

Concretely, HM can perform worse than DM in the following scenarios that we tested:

- Tabular with large policy mismatch, or stochastic environments (appendix Figure A.14, Table A.3, A.6).

- Complex domains with long horizon and unknown behavior policy (app. Figure A.24-A.26, Table A.8).

When data is sufficient, or model misspecification is severe, HM provides consistent gains over DM.

Is horizon length the most important factor? No. Despite conventional wisdom suggesting IPS methods are most sensitive to horizon length, we find that this is not always the case. Policy divergence $\sup_{a\in A,x\in X} \frac{\pi_e(a|s)}{\pi_b(a|s)}$ can be just as, if not more, meaningful. For comparison, we designed two scenarios with identical mismatch $(\sup_{a\in A,x\in X} \frac{\pi_e(a|s)}{\pi_b(a|s)})^T$ as defined in Section 3.2 (see appendix Tables A.11, A.12). Starting from a baseline scenario of short horizon and small policy divergence (appendix Table A.10), extending horizon length leads to $10\times$ degradation in accuracy, while a comparable increase in policy divergence causes a $100\times$ degradation.

How good is model-based direct method (AM)? AM can be among the worst performing direct methods (appendix Table A.1). While AM performs well in tabular setting in the large data case (appendix Figure A.13), it tends to perform poorly in high dimensional settings with function approximation (e.g., Figure 3.2 center). Fitting the transition model $P(s'|s,a)$ is often more prone to small errors than directly approximating $Q(s,a)$. Model fitting errors also compound with long horizons.

3.3 Discussion and Future Directions

Finally, we close with a brief discussion on some limitations common to recent OPE benchmarks and more generally OPE experimental studies, and point to areas of development for future studies.

Lack of short-horizon benchmark in high-dimensional settings. Evaluation of other complex RL tasks with short horizon is currently beyond the scope of our study, due to the lack of a natural benchmark. For contextual bandits, it has been shown that while DR is highly competitive, it is sometimes substantially outperformed by DM [219]. New benchmark tasks should have longer horizon than contextual bandits, but shorter than typical Atari games. We also currently lack natural stochastic environments in high-dimensional RL benchmarks. An example candidate for medium horizon, complex OPE domain is NLP tasks such as dialogue.

Other OPE settings. We outline practically relevant settings that can benefit from benchmarking:

- *Missing data coverage.* A common assumption in the analysis of OPE is a full support assumption: $\pi_e(a|s) > 0$ implies $\pi_b(a|s) > 0$, which often ensure unbiasedness of estimators [165, 133, 54]. This assumption is often not verifiable in practice. Practically, violation of this assumption requires regularization of unbiased estimators to avoid ill-conditioning [133, 60]. One avenue to investigate is to optimize the bias-variance trade-off when the full support is not applicable.
- *Confounding variables.* Existing OPE research often assumes that the behavior policy chooses actions solely based on the state. This assumption is often violated when the decisions in the historical data are made by humans instead of algorithms, who may base their decisions on variables not recorded in the data, causing confounding effects. Tackling this challenge, possibly using techniques from causal inference [200, 156], is an important future direction.
- *Strategic Environmental Behavior.* Most OPE methods have focused exclusively on single-agent scenarios under well-defined MDP. Realitic applications of offline RL may have to deal with nonstationary and partial observability induced by strategic behavior from multiple agents [230]. There is currently a lack of a compelling domain to study such a setting.

Evaluating new OPE estimators. For our empirical evaluation, we selected a representative set of established baseline approaches from multiple OPE method families. Currently this area of research is very active and as such, new OPE estimators have been and will continue to be proposed. We discuss several new minimax style estimators, notably the DICE family in section 3.1. A minimax-style estimator has also been recently proposed for the model-based regime [214]. Among the ideas that use marginalized state distribution [223] to improve over standard IPS, [102, 103] analyze double reinforcement learning estimator that makes use of both estimates for Q function and state density ratio. While we have not included all estimators in our current benchmark, our software implementation is highly modular and can easily accommodate new estimators and environments.

Algorithmic approach to method selection. Using COBS, we showed how to distill a general guideline for selecting OPE methods. However, it is often not easy to judge whether some decision criteria are satisfied (e.g., quantifying model misspecification, degree of stochasticity, or appropriate data size). As more OPE methods continue to be developed, an important missing piece is a systematic technique for model selection, given a relatively high degree of variability among existing techniques.

Chapter 4

ADVANCES IN MODEL BASED OPE

We present a novel off-policy loss function for learning a transition model in model-based reinforcement learning. Notably, our loss is derived from the off-policy policy evaluation objective with an emphasis on correcting distribution shift. Compared to previous model-based techniques, our approach allows for greater robustness under model mis-specification or distribution shift induced by learning/evaluating policies that are distinct from the data-generating policy. We provide a theoretical analysis and show empirical improvements over existing model-based off-policy evaluation methods. We provide further analysis showing our loss can be used for off-policy optimization (OPO) and demonstrate its integration with more recent improvements in OPO.

4.1 Introduction to Model-Based OPE and OPO

We study the problem of learning a transition model in a batch, off-policy reinforcement learning (RL) setting, i.e., of learning a function $P(s'|s,a)$ from a pre-collected dataset $D = \{(s_i, a_i, s'_i)\}_{i=1}^n$ without further access to the environment. Contemporary approaches to model learning focus primarily on improving the performance of models learned through maximum likelihood estimation (MLE) [193, 48, 113, 43, 42, 137]. The goal of MLE is to pick the model within some model class $\mathcal{P}$ that is most consistent with the observed data or, equivalently, most likely to have generated the data. This is done by minimizing negative log-loss (minimizing the KL divergence) summarized as follows:

$$\widehat{P}_{\text{MLE}} = \arg\min_{P \in \mathcal{P}} \frac{1}{n} \sum_{(s_i, a_i, s'_i) \in D} -\log(P(s'_i|s_i, a_i)). \tag{4.1}$$

A key limitation of MLE is that it focuses on picking a good model under the data distribution while ignoring how the model is actually used.

In an RL context, a model can be used to either learn a policy (policy learning/optimization) or evaluate some given policy (policy evaluation), without having to collect more data from the true environment. We call this actual

objective the "decision problem". Interacting with the environment to solve the decision problem can be difficult, expensive, and dangerous, whereas a model learned from batch data circumvents these issues. Since MLE (4.1) does not optimize over the distribution of states induced by the policy from the decision problem, it thus does not prioritize solving the decision problem. Notable previous works that incorporate the decision problem into the model learning objective are Value-Aware Model Learning (VAML) and its variants [59, 58, 1]. These methods, however, still define their losses w.r.t. the data distribution as in MLE, and ignore the *distribution shift* from the pre-collected data to the policy-induced distribution.

In contrast, we directly focus on requiring the model to perform well under unknown distributions instead of the data distribution. In other words, we are particularly interested in developing approaches that directly model the batch (offline) learning setting. As such, we ask: *"From only pre-collected data, is there a model learning approach that naturally controls the decision problem error?"*.

We present a new loss function for model learning that: (1) only relies on batch or offline data; (2) takes into account the distribution shift effects; and (3) directly relates to the performance metrics for off-policy evaluation and learning under certain realizability assumptions. The design of our loss is inspired by recent advances in model-free off-policy evaluation [e.g., 134, 208], which we build upon to develop our approach.

4.2 Preliminaries

Recall from Section 2.2, in the batch learning setting, we are given a dataset $D = \{(s_i, a_i, s_i')\}_{i=1}^n$, where $s_i \sim d_{\pi_b}(s)$, $a_i \sim \pi_b$, and $s_i' \sim P(\cdot|s_i, a_i)$, where π_b is some behavior policy that collects the data. For convenience, we write $(s, a, s') \sim D_{\pi_b} P$, where $D_{\pi_b}(s, a) = d_{\pi_b}(s)\pi_b(a|s)$.

Finally, we need three classes $\mathcal{W}, \mathcal{V}, \mathcal{P}$ of functions. $\mathcal{W} \subset (\mathcal{X} \to \mathbb{R})$ represents ratios between state-action occupancies, $\mathcal{V} \subset (\mathcal{S} \to \mathbb{R})$ represents value functions and $\mathcal{P} \subset (\mathcal{X} \to \Delta(\mathcal{S}))$ represents the class of models (or simulators) of the true environment.

Note. Any Lemmas or Theorems presented without proof have full proofs in the Appendix.

4.3 Minimax Model Learning (MML) for OPE

Natural Derivation

We start with the off-policy evaluation (OPE) learning objective and derive the MML loss (Def 4.3.1). In Section 4.4, we show the loss also bounds off-policy optimization (OPO) error through its connection with OPE.

OPE Decision Problem. Recall from Section 2.2, the OPE objective is to estimate:

$$J(\pi, P^*) \equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_i \middle| \begin{smallmatrix} s_0 \sim d_0 \\ a_i \sim \pi(\cdot|s_i) \\ s_{i+1} \sim P^*(\cdot|s_i, a_i) \\ r_i \sim \mathcal{R}(\cdot|s_i, a_i) \end{smallmatrix}\right], \tag{4.2}$$

the performance of an evaluation policy π in the true environment P^*, using only logging data D with samples from $D_{\pi_b} P^*$.

Model-Based OPE. Given a model class $\mathcal{P}$ and a desired evaluation policy π, we want to find a simulator $\widehat{P} \in \mathcal{P}$ using only logging data D such that:

$$\widehat{P} = \arg\min_{P \in \mathcal{P}} |J(\pi, P) - J(\pi, P^*)|. \tag{4.3}$$

Interpreting Eq. (4.3), we run π in P to compute $J(\pi, P)$ as a proxy to $J(\pi, P^*)$. If we find some $P \in \mathcal{P}$ such that $|\delta_\pi^{P,P^*}| = |J(\pi, P) - J(\pi, P^*)|$ is small, then P is a good simulator for P^*.

Derivation. Using (2.2) and (2.4), we have:

$$\begin{aligned} \delta_\pi^{P,P^*} &= J(\pi, P) - J(\pi, P^*) \\ &= E_{s\sim d_0}[V_\pi^P(s)] - E_{(s,a)\sim d_{\pi,\gamma}^{P^*}(\cdot,\cdot)}[E_{r\sim\mathcal{R}(\cdot|s,a)}[r]]. \end{aligned}$$

Adding and subtracting $E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s)]$, we have:

$$\delta_\pi^{P,P^*} = E_{s\sim d_0}[V_\pi^P(s)] - E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s)] \tag{4.4}$$

$$+ E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]]. \tag{4.5}$$

To simplify the above expression, we make the following observations. First, Eq. (4.5) can be simplified through the Bellman equation from Eq. (2.3). To see this, notice that $d_{\pi,\gamma}^{P^*}$ is equivalent to some $d(s)\pi(a|s)$ for an appropriate choice of $d(s)$. Thus,

$$\begin{aligned} &E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]] \\ &\quad = E_{s\sim d(\cdot)}[E_{a\sim\pi(\cdot|s)}[V_\pi^P(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]]] \\ &\quad = E_{s\sim d(\cdot)}[E_{a\sim\pi(\cdot|s)}[E_{s'\sim P(\cdot|s,a)}[\gamma V_\pi^P(s)]]] \\ &\quad = \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')]]. \end{aligned}$$

Second, we can manipulate Eq. (4.4) using the definition of $d_{\pi,\gamma}^P$ and recursive

property of $d^P_{\pi,t}$ from Eq. (2.5):

$$\begin{aligned}
&E_{s\sim d_0}[V^P_\pi(s)] - E_{(s,a)\sim d^{P^*}_{\pi,\gamma}}[V^P_\pi(s)] \\
&= -\sum_{t=1}^{\infty}\gamma^t \int d^{P^*}_{\pi,t}(s,a)V^P_\pi(s)d\nu(s,a) \\
&= -\gamma\sum_{t=0}^{\infty}\gamma^t \int d^{P^*}_{\pi,t+1}(s,a)V^P_\pi(s)d\nu(s,a) \\
&= -\gamma\sum_{t=0}^{\infty}\gamma^t \int d^{P^*}_{\pi,t}(\tilde{s},\tilde{a})P^*(s|\tilde{s},\tilde{a})\pi(a|s)V^P_\pi(s)d\nu(\tilde{s},\tilde{a},s,a) \\
&= -\gamma\sum_{t=0}^{\infty}\gamma^t \int d^{P^*}_{\pi,t}(s,a)P^*(s'|s,a)V^P_\pi(s')d\nu(s,a,s') \\
&= -\gamma E_{(s,a)\sim d^{P^*}_{\pi,\gamma}}[E_{s'\sim P^*(\cdot|s,a)}[V^P_\pi(s')]].
\end{aligned}$$

Combining the above allows us to succinctly express:

$$\begin{aligned}
\delta^{P,P^*}_\pi = \gamma E_{(s,a)\sim d^{P^*}_{\pi,\gamma}}&[E_{s'\sim P(\cdot|s,a)}[V^P_\pi(s')]] \\
&- \gamma E_{(s,a)\sim d^{P^*}_{\pi,\gamma}}[E_{s'\sim P^*(\cdot|s,a)}[V^P_\pi(s')]].
\end{aligned}$$

Since D contains samples from D_{π_b} and not $d^{P^*}_{\pi,\gamma}$, we use importance sampling to simplify the right-hand side of δ^{P,P^*}_π to:

$$\gamma \underset{(s,a,s')\sim D_{\pi_b}P^*}{E}\left[\frac{d^{P^*}_{\pi,\gamma}}{D_{\pi_b}}\left(\underset{\tilde{s}\sim P(\cdot|s,a)}{E}[V^P_\pi(\tilde{s})] - V^P_\pi(s')\right)\right]. \tag{4.6}$$

Define $w^P_\pi(s,a) \equiv \frac{d^P_{\pi,\gamma}(s,a)}{D_{\pi_b}(s,a)}$. If we knew $w^{P^*}_\pi(s,a)$ and V^P_π (for every $P \in \mathcal{P}$), then we can select a $P \in \mathcal{P}$ to directly control δ^{P,P^*}_π. We encode this intuition as:

Definition 4.3.1. [MML Loss] $\forall w \in \mathcal{W}, V \in \mathcal{V}, P \in \mathcal{P}$,

$$\begin{aligned}
\mathcal{L}_{MML}(w,V,P) =& E_{(s,a,s')\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)}[w(s,a)\cdot \\
&\Big(E_{\tilde{s}\sim P(\cdot|s,a)}[V(\tilde{s})] - V(s')\Big)].
\end{aligned}$$

When unambiguous, we will drop the MML subscript.

Here we have replaced $w^{P^*}_\pi(s,a)$ with w coming from function class $\mathcal{W}$ and V^P_π with V from class $\mathcal{V}$. The function class $\mathcal{W}$ represents the possible distribution shifts, while $\mathcal{V}$ represents the possible value functions.

With this intuition, we can formally guarantee that $J(\pi,P) \approx J(\pi,P^*)$ under the following *realizability conditions*:

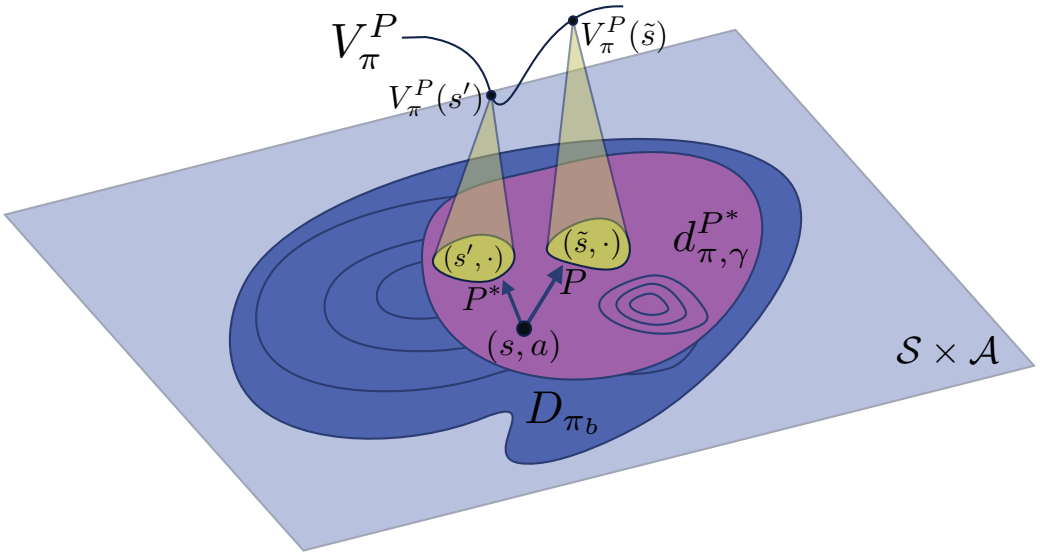


Figure 4.1: *Visual of Eq.* (4.6). *The error at every point* (s,a) *in* D_{π_b} *is the difference between* $V_\pi^P(\tilde{s})$ *(induced by following* P*) and* $V_\pi^P(s')$ *(induced by following* P^**). We re-weight the points* (s,a) *in* D_{π_b} *to mimic* $d_{\pi,\gamma}^{P^*}$*. Accumulating the errors exactly yields the OPE error of using* P *as a simulator. MLE, instead, finds a* P *"pointing" in the same direction as* P^* *for all points in* D_{π_b}*, ignoring the discrepancy with* $d_{\pi,\gamma}^{P^*}$.

Assumption 4.1 (Adequate Support)**.** $D_{\pi_b}(s,a) > 0$ whenever $d_{\pi,\gamma}^P(s,a) > 0$. Define $w_\pi^P(s,a) \equiv \frac{d_{\pi,\gamma}^P(s,a)}{D_{\pi_b}(s,a)}$.

Assumption 4.2 (OPE Realizability)**.** For a given π, $\mathcal{W}\times\mathcal{V}$ contains at least one of $(w_\pi^P, V_\pi^{P^*})$ or $(w_\pi^{P^*}, V_\pi^P)$ for every $P\in\mathcal{P}$.

Theorem 4.3.1 (MML & OPE)**.** *Under Assumption 4.2,*

$$|J(\pi,\widehat{P}) - J(\pi,P^*)| \le \gamma \min_{P\in\mathcal{P}} \max_{w\in\mathcal{W},V\in\mathcal{V}} |\mathcal{L}(w,V,P)|, \tag{4.7}$$

where $\widehat{P} = \arg\min_{P\in\mathcal{P}} \max_{w\in\mathcal{W},V\in\mathcal{V}} |\mathcal{L}(w,V,P)|$.

Remark 4.3.2. *We want to choose* $\mathcal{V},\mathcal{W},\mathcal{P}$ *carefully so that many* $P\in\mathcal{P}$ *satisfy* $\mathcal{L}(w,V,P)=0$ *and Assumption 4.2. By inspection,* $\mathcal{L}(w,V,P^*)=0$ *for any* $V\in\mathcal{V}, w\in\mathcal{W}$.

Remark 4.3.3. *While* $V_\pi^P\in\mathcal{V}\ \forall P\in\mathcal{P}$ *appears strong, it can be verified for every* $P\in\mathcal{P}$ *before accessing the data, as the condition does not depend on* P^*. *In principle, we may redesign* $\mathcal{V}$ *to guarantee this condition.*

Remark 4.3.4. *When* $\gamma=0$, J *does not depend on a transition function, so* $J(\pi,P)=J(\pi,P^*)\ \forall P\in\mathcal{P}$.

$\mathcal{L}(w, V, P^*) = 0$ and Theorem 4.3.1 implies that the following learning procedure will be robust to any distribution shift in $\mathcal{W}$ and any value function in $\mathcal{V}$:

Definition 4.3.2 (Minimax Model Learning (MML))**.**

$$\widehat{P} = \arg\min_{P \in \mathcal{P}} \max_{w \in \mathcal{W}, V \in \mathcal{V}} |\mathcal{L}_{MML}(w, V, P)|. \tag{4.8}$$

Interpretation and Verifiability

Figure 4.1 gives a visual illustration of Eq. (4.6) which leads to the MML Loss (Def 4.3.1). π_b has induced an "inbalanced" training dataset D_{π_b} and the importance sampling term acts to rebalance our data because our test dataset will be $d^{P^*}_{\pi,\gamma}$, induced by π. Because the objective is OPE, we don't mind that $\hat{P}$ is different than P^* so long as $E_{\hat{P}}[V^{\hat{P}}_\pi] \approx E_{P^*}[V^{\hat{P}}_\pi]$. In other words, the size of $V^{\hat{P}}_\pi$ tells us which state transitions are important to model correctly. We want to appropriately utilize the capacity of our model class $\mathcal{P}$ so that $\hat{P}$ models P^* when $V^{\hat{P}}_\pi$ is large. When it is small, it may be better off to ignore the error in favor of other states.

Theorem 4.3.1 quantifies the error incurred by evaluating π in $\widehat{P}$ instead of P^*, assuming Assumption 4.2 holds. For OPE, $\widehat{P}$ is a reasonable proxy for P. In this sense, MML is a principled method approach for model-based OPE. See Appendix B.1 for a complete proof of Thm 4.3.1 and Appendix B.1 for the sample complexity analysis.

If the exploratory state distribution d_{π_b} and π_b are known then D_{π_b} is known. In this case, we can also verify that $w^P_\pi \in \mathcal{W}$ for every $P \in \mathcal{P}$ a priori. Together with Remark 4.3.3, we may assume that both $w^P_\pi \in \mathcal{W}$ and $V^P_\pi \in \mathcal{V}$ for all $P \in \mathcal{P}$. Consequently, only one of $V^{P^*}_\pi \in \mathcal{V}$ or $w^{P^*}_\pi \in \mathcal{W}$ has to be realizable for Theorem 4.3.1 to hold.

Instead of checking for realizability apriori, we can perform post-verification that $w^{\widehat{P}}_\pi \in \mathcal{W}$ and $V^{\widehat{P}}_\pi \in \mathcal{V}$. Together with the terms depending on P^*, realizability of these are also sufficient for Theorem 4.3.1 to hold. This relaxes the strong "for all $P \in \mathcal{P}$" condition.

Comparison to Model-Free OPE

Recent model-free OPE literature [e.g., 134, 208] has similar realizability assumptions to Assumption 4.2.

As an example, the method MWL [208] takes the form of:

$$J(\pi, P^*) \approx E_{(s,a,r)\sim D_{\pi_b}}[\widehat{w}(s,a)r]$$
$$\text{where } \widehat{w} = \arg\min_{w\in\mathcal{W}} \max_{Q\in\mathcal{Q}} |\mathcal{L}_{MWL}(w,Q)|,$$

requiring $Q_\pi^{P^*}$ to be realized to be a valid upper bound. Here $\mathcal{Q}$ is analogous to our function class $\mathcal{V}$ where $E_{a\sim\pi(a|s)}[Q_\pi^{P^*}(s,a)] = V_\pi^{P^*}(s)$. The loss $\mathcal{L}_{MWL}$ has no dependence on P and is therefore model-free. MQL [208] has analogous realizability conditions to MWL.

Our loss, $\mathcal{L}_{MML}$, has the same realizability assumptions in addition to one related to $\mathcal{P}$ (and not $\mathcal{P}^*$). As discussed in Remark 4.3.3, these $\mathcal{P}$-related assumptions can be verified a priori and in principle, satisfied by redesigning the function classes. Therefore, they do not pose a substantial theoretical challenge. See Section 4.6 for a practical discussion.

An advantage of model-free approaches is that when both $w_\pi^{P^*}, Q_\pi^{P^*}$ are realized, they return an exact OPE point estimate. In contrast, MML additionally requires some $P \in \mathcal{P}$ that makes the loss zero for any $w \in \mathcal{W}, V \in \mathcal{V}$. The advantage of MML is the increased flexibility of a model, enabling OPO (Section 4.4) and visualization of results through simulation (leading to more transparency).

While recent model-free OPE and our method both take a minimax approach, the classes $\mathcal{W}, \mathcal{V}, \mathcal{P}$ play different roles. In the model-free case, minimization is w.r.t either $\mathcal{W}$ or $\mathcal{V}$ and maximization is w.r.t the other. In our case, $\mathcal{W}, \mathcal{V}$ are on the same (maximization) team, while minimization is over $\mathcal{P}$. This allows us to treat $\mathcal{W} \times \mathcal{V}$ as a single unit, and represents distribution-shifted value functions. A member of this class, $E_{\text{data}}[wV]$ $(= E_{(s,a)\sim D_{\pi_b}}[\frac{d_{\pi,\gamma}^{P^*}}{D_{\pi_b}} V_\pi^P(s)])$, ties together the OPE estimate.

Misspecification of $\mathcal{P}, \mathcal{V}, \mathcal{W}$

Suppose Assumption 4.2 does not hold and $P^* \notin \mathcal{P}$. Define a new function $h(s,a,s') \in \mathcal{H} = \{w(s,a)V(s') | (w,V) \in \mathcal{W} \times \mathcal{V}\}$. Then we redefine $\mathcal{L}$:

$$\mathcal{L}(h,P) = E_{(s,a,s')\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)}[$$
$$E_{x\sim P(\cdot|s,a)}[h(s,a,x)] - h(s,a,s')].$$

Proposition 4.3.5 (Misspecification discrepancy for OPE)**.** *Let $\mathcal{H} \subset (\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R})$ be a set of functions on (s,a,s'). Denote $(WV)^* = w_\pi^{P^*}(s,a)V_\pi^P(s')$*

(or, equivalently, $(WV)^* = w_\pi^P(s,a)V_\pi^{P^*}(s')$*).*

$$|J(\pi, \widehat{P}) - J(\pi, P^*)| \leq \gamma \min_P \max_{h \in \mathcal{H}} |\mathcal{L}(h, P)| + \gamma \epsilon_{\mathcal{H}}, \tag{4.9}$$

where $\epsilon_{\mathcal{H}} = \max_{P \in \mathcal{P}} \min_{h \in \mathcal{H}} |\mathcal{L}((WV)^* - h, P)|$.

$\mathcal{L}(WV^* - h, P)$ measures the difference between h and $(WV)^*$. Another interpretation of Prop 4.3.5 is if $\arg\max_{\mathcal{H} \cup \{(WV)^*\}} \mathcal{L}(h, P) = (WV)^*$ for some $P \in \mathcal{P}$ then MML returns a value $\gamma\epsilon_{\mathcal{H}}$ below the true upper bound, otherwise the output of MML remains the upperbound. This result illustrates that realizability is sufficient but not necessary for MML to be an upper-bound on the loss.

Application to the Online Setting

While the main focus of MML is batch OPE and OPO, we will make a few remarks relating to the online setting. In particular, if we assume we can engage in online data collection then $\mathcal{W} = \{\mathbf{1}\}$ (the constant function), representing no distribution shift since $\pi_b = \pi$. When VAML and MML share the same function class $\mathcal{V}$, we can show that $\min_{\mathcal{M}} \max_{\mathcal{W},\mathcal{V}} \mathcal{L}_{MML}(w, V, P)^2 \leq \min_P \mathcal{L}_{VAML}(\mathcal{V}, P)$ for any $\mathcal{V}, \mathcal{M}$. In other words, MML is a tighter decision-aware loss even in online data collection. In addition, MML enables greater flexibility in the choice of $\mathcal{V}$. See Appendix B.1 for further details.

4.4 Off-Policy Optimization (OPO)

Natural Derivation

In this section we examine how MML can be integrated into the policy learning/optimization objective. In this setting, the goal is to find a good policy with respect to the true environment P^* without interacting with P^*.

OPO Decision Problem. Given a policy class Π and access to only a logging dataset D with samples from $D_{\pi_b} P^*$, find a policy $\pi \in \Pi$ that is competitive with the unknown optimal policy $\pi^*_{P^*}$:

$$\widehat{\pi}^* = \arg\min_{\pi \in \Pi} |J(\pi, P^*) - J(\pi^*_{P^*}, P^*)|. \tag{4.10}$$

Note: No additional exploration is allowed.

Model-Based OPO. Given a model class $\mathcal{P}$, we want to find a simulator $\widehat{P} \in \mathcal{P}$ using only logging data D and subsequently learn $\pi^*_{\widehat{P}} \in \Pi$ in $\widehat{P}$ through any policy optimization algorithm which we call Planner$(\cdot)$.

Algorithm 1 Standard Model-Based OPO

Require: $D = D_{\pi_b} P^*$, Modeler, Planner
1: Learn $\widehat{P} \leftarrow \text{Modeler}(D)$
2: Learn $\widehat{\pi}^*_P \leftarrow \text{Planner}(\widehat{P})$
3: **return** $\widehat{\pi}^*_P$

In Algorithm 1, Modeler$(\cdot)$ refers to any (batch) model learning procedure. The hope for model-based OPO is that the ideal in-simulator policy $\pi^*_{\widehat{P}}$ and the actual best (true environment) policy $\pi^*_{P^*}$ perform competitively: $J(\pi^*_{\widehat{P}}, P^*) \approx J(\pi^*_{P^*}, P^*)$. Hence, instead of minimizing Eq (4.10) over all $\pi \in \Pi$, we can focus $\Pi = \{\pi^*_P\}_{P \in \mathcal{P}}$.

Derivation. Beginning with the objective, we add zero twice:

$$J(\pi^*_{P^*}, P^*) - J(\pi^*_P, P^*) = \underbrace{J(\pi^*_{P^*}, P^*) - J(\pi^*_{P^*}, P)}_{(a)}$$
$$+ \underbrace{J(\pi^*_{P^*}, P) - J(\pi^*_P, P)}_{(b)} + \underbrace{J(\pi^*_P, P) - J(\pi^*_P, P^*)}_{(c)}.$$

Term (b) is non-positive since π^*_P is optimal in P ($\pi^*_{P^*}$ is suboptimal), so we can drop it in an upper bound. Term (a) is the OPE estimate of $\pi^*_{P^*}$ and term (c) the OPE estimate of π^*_P, implying that we should use Theorem 4.3.1. With this intuition, we have:

Theorem 4.4.1 (MML & OPO)**.** *If* $w^{P^*}_{\pi^*_{P^*}}, w^{P^*}_{\pi^*_P} \in \mathcal{W}$ *and* $V^P_{\pi^*_{P^*}}, V^P_{\pi^*_P} \in \mathcal{V}$ *for every* $P \in \mathcal{P}$ *then:*

$$|J(\pi^*_{P^*}, P^*) - J(\pi^*_{\widehat{P}}, P^*)| \leq 2\gamma \min_P \max_{w,V} |\mathcal{L}(w, V, P)|.$$

The statement also holds if, instead, $w^P_{\pi^*_{P^*}}, w^P_{\pi^*_P} \in \mathcal{W}$ *and* $V^{P^*}_{\pi^*_{P^*}}, V^{P^*}_{\pi^*_P} \in \mathcal{V}$ *for every* $P \in \mathcal{P}$.

Interpretation and Verifiability

Theorem 4.4.1 compares two different policies in the same (true) environment, since $\pi^*_{\widehat{P}}$ will be run in P^* rather than $\widehat{P}$. In contrast, Theorem 4.3.1 compared the same policy in two different environments. The derivation of Theorem 4.4.1 (see Appendix B.2) shows that having a good bound on the OPE objective is sufficient for OPO. MML shows how to learn a model that exploits this relationship.

Furthermore, the realizability assumptions of Theorem 4.4.1 relax the requirements of an OPE oracle. Rather than requiring the OPE estimate for every

π, it is sufficient to have the OPE estimate of $\pi^*_{P^*}$ and π^*_P (for every $P \in \mathcal{P}$) when there is a $P \in \mathcal{P}$ such that $\mathcal{L}(w, V, P)$ is small for any $w \in \mathcal{W}, V \in \mathcal{V}$.

We could have instead examined the quantity $\min_\pi |J(\pi^*_{P^*}, P^*) - J(\pi, P^*)|$ directly from Eq (4.10). What we would find is that the upper bound is $2 \min_P \max_{w,V} |E_{d_0}[V] - \mathcal{L}(w, V, P)|$ and the realizability requirements would be that $V^P_\pi \in \mathcal{V}, w^{P^*}_\pi \in \mathcal{W}$ for every π in some policy class. This is a much stronger requirement than in Theorem 4.4.1.

For OPO, apriori verification of realizability is possible by enumerating over $P \in \mathcal{P}$. Whereas the target policy π was fixed in OPE, now π^*_P varies for each $P \in \mathcal{P}$. It may be more practical to, as in OPE, perform post-verification that $w^P_{\pi^*_{\widehat{P}}} \in \mathcal{W}$ and $V^P_{\pi^*_{\widehat{P}}} \in \mathcal{V}$. If they do not hold, then we can modify the function classes until they do. This relaxes the "for every $P \in \mathcal{P}$" condition and leaves only a few unverifiable quantities relating to P^*.

Sample complexity and function class misspecification results for OPO can be found in Appendix B.2, B.2.

Comparison to Model-Free OPO

For minimax model-free OPO, [39] have analyzed a minimax variant of Fitted Q Iteration (FQI) [55], inspired by Antos et al. [12]. FQI is a commonly used model-free OPO method. In addition to realizability assumptions, these methods also maintain a completeness assumption: the function class of interest is closed under bellman update. Increasing the function class size can only help realizability but may break completeness. It is unknown if the completeness assumption of FQI is removable [39]. MML only has realizability requirements.

4.5 Scenarios & Considerations

In this section we investigate a few scenarios where we can calculate the class $\mathcal{V}$ and $\mathcal{W}$ or modify the loss based on structural knowledge of $\mathcal{P}, \mathcal{W}$, and $\mathcal{V}$.

In examining the scenarios, we aim to verify that MML gives *sensible* results. For example, in scenarios where we know MLE to be optimal, MML should ideally coincide. Indeed, we show this to be the case for the tabular setting and Linear-Quadratic Regulators. Other scenarios include showing that MML is compatible with incorporating prior knowledge using either a nominal dynamics model or a kernel.

The proofs for any Lemmas in this section can be found in Appendix B.4.

Linear & Tabular Function Classes

When $\mathcal{W}, \mathcal{V}, \mathcal{P}$ are linear function classes then the entire minimax optimization has a closed form solution. In particular, $\mathcal{P}$ takes the form $P = \varphi(s,a,s')^T\alpha$ where $\varphi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}\times\mathcal{S}|}$ is some basis of features with $\alpha \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}\times\mathcal{S}|}$ its parameters and $(w(s,a), V(s')) \in \mathcal{WV} = \{\psi(s,a,s')^T\beta : \|\beta\|_\infty < +\infty\}$ where $\psi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}\times\mathcal{S}|}$.

Proposition 4.5.1 (Linear Function classes)**.** *Let $P = \varphi(s,a,s')^T\alpha$ where $\varphi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}\times\mathcal{S}|}$ is some basis of features with α its parameters. Let $(w(s,a), V(s')) \in \mathcal{WV} = \{\psi(s,a,s')^T\beta : \|\beta\|_\infty < +\infty\}$. Then,*

$$\widehat{\alpha} = E_n^{-T}\left[\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right] E_n[\psi(s,a,s')], \tag{4.11}$$

if $E_n\left[\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right]$ has full rank.

The tabular setting, when the state-action space is finite, is a common special case. We can choose:

$$\psi(s,a,s') = \varphi(s,a,s') = e_i \tag{4.12}$$

as the ith standard basis vector where $i = s|\mathcal{A}||\mathcal{S}| + a|\mathcal{S}| + s'$. There is no model misspecification in the tabular setting (i.e., $P^* \in \mathcal{P}$), therefore $\widehat{P} = P^*$ in the case of infinite data.

Proposition 4.5.2 (Tabular representation)**.** *Let $P = \varphi(s,a,s')^T\alpha$ with $\varphi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}\times\mathcal{S}|}$ as in Eq (4.12) and α its parameters. Let $(w(s,a), V(s')) \in \mathcal{WV} = \{\varphi(s,a,s')^T\beta : \|\beta\|_\infty < +\infty\}$. Assume we have at least one data point from every (s,a) pair. Then:*

$$\widehat{P}_n(s'|s,a) = \frac{\#\{(s,a,s') \in D\}}{\#\{(s,a,\cdot) \in D\}}. \tag{4.13}$$

Prop. 4.5.2 shows that MML and MLE coincide, even in the finite-data regime. Both models are simply the observed propensity of entering state s' from tuple (s,a).

Linear Quadratic Regulator (LQR)

The Linear Quadratic Regulator (LQR) is defined as linear transition dynamics $P^*(s'|s,a) = A^*s + B^*a + w^*$ where w^* is random noise and a quadratic reward function $\mathcal{R}(s,a) = s^TQs + a^TRa$ for $Q, R \geq 0$ symmetric positive semi-definite. For ease of exposition we assume that $w^* \sim N(0, \sigma^{*2}I)$. We assume that (A^*, B^*) is controllable. Exploiting the structure of this problem, we can

check that every $V \in \mathcal{V}$ takes the form $V(s) = s^T U s + q$ for some symmetric semi-positive definite U and constant q (Appendix Lemma B.4.1).

Furthermore, we know controllers of the form $\pi(a|s) = -Ks$ where $K \in \mathbb{R}^{k \times n}$ are optimal in LQR [22]. We consider determistic and therefore *misspecified* models of the form $P(s'|s,a) = As + Ba$. $\mathcal{W}$ is a Gaussian mixture and we can write $\mathcal{L}_{MML}$ as a function of U, K and (A, B) (Appendix Lemma B.4.2).

Proposition 4.5.3 (MML + MLE Coincide for LQR)**.** *Let $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times k}, K \in \mathbb{R}^{k \times n}$. Let $U \in \mathcal{S}^n$ be positive semi-definite. Set $k = 1$, a single input system. Then,*

$$\begin{aligned} \arg\min_{(A,B)} \max_{K,U} |\mathcal{L}_{MML}(K, U, (A, B))| &= (A^*, B^*) \\ &= \arg\min_{(A,B)} \mathcal{L}_{MLE}(A, B). \end{aligned}$$

Despite model misspecification, both MLE and MML give the correct parameters $(\widehat{A}, \widehat{B}) = (A^*, B^*)$. We leave showing that MML and MLE coincide in multi-input ($k > 1$) LQR systems for future work.

Residual Dynamics & Environment Shift

Suppose we already had some baseline model P_0 of P^*. Alternatively, we may view this as the real world starting with (approximately) known dynamics P_0 and drifting to P^*. We can modify MML to incorporate knowledge of P_0 to find the residual dynamics:

Definition 4.5.1. [Residual MML Loss] For $w \in \mathcal{W}, V \in \mathcal{V}, P \in \mathcal{P}$,

$$\begin{aligned} \mathcal{L}(w, V, P) = E_{(s,a,s') \sim D_{\pi_b}(\cdot,\cdot) P^*(\cdot|s,a)} [w(s,a) \cdot \\ \left(E_{x \sim P_0(\cdot|s,a)} [\frac{P_0(x|s,a) - P(x|s,a)}{P_0(x|s,a)} V(x)] - V(s') \right)]. \end{aligned}$$

This solution form matches the intuition that having prior knowledge in the form of P_0 focuses the learning objective on the difference between P^* and P_0.

Incorporating Kernels

Our approach is also compatible with incorporating kernels (which is a way of encoding domain knowledge such as smoothness) to learn in a Reproducing Kernel Hilbert Space (RKHS). For example, we may derive a closed form for $\max_{(w,V) \in \mathcal{W}\mathcal{V}} \mathcal{L}(w, V, P)^2$ when $\mathcal{W} \times \mathcal{V}$ corresponds to an RKHS and use standard gradient descent to find $\widehat{P} \in \mathcal{P}$, making the minimax problem

much more tractable. See Appendix B.4 for a detailed discussion on RKHS, computational issues relating to sampling from P and alternative approaches to solving the minimax problem.

4.6 Experiments

In our experiments, we seek to answer the following questions: (1) Does MML prefer models that minimize the OPE objective? (2) What can we expect when we have misspecification in $\mathcal{V}$? (3) How does MML perform against MLE and VAML in OPE? (4) Does our approach complement modern offline RL approaches? For this last question, we consider integrating MML with the recently proposed MOREL [106] approach for offline RL. See Appendix B.5 for details on MOREL.

Brief Environment Description/Setup

We perform our experiments in three different domains.

Linear-Quadratic Regulator (LQR). The LQR domain is a 1D environment with stochastic dynamics $P^*(s'|s,a)$. We use a finite class $\mathcal{P}$ consisting of deterministic policies. We ensure $V_\pi^P \in \mathcal{V}$ for all $P \in \mathcal{P}$ by solving the equations in Appendix Lemma B.4.1. We ensure $W_\pi^{P^*} \in \mathcal{W}$ using Appendix Equation (B.7).

Cartpole [30]. The reward function is modified to be a function of angle and location rather than 0/1 to make the OPE problem more challenging. Each $P \in \mathcal{P}$ is a parametrized NN that outputs a mean, and logvariance representing a normal distribution around the next state. We model the class $\mathcal{WV}$ as a RKHS as in Prop B.4.3 with an RBF kernel.

Inverted Pendulum (IP) [51]. This IP environment has a Runge-Kutta(4) integrator rather than Forward Euler (Runge-Kutta(1)) as in OpenAI [30], producing significantly more realistic data. Each $P \in \mathcal{P}$ is a deterministic model parametrized with a neural network. We model the class $\mathcal{WV}$ as a RKHS as in Prop B.4.3 with an RBF kernel.

Further Detail A thorough description of the environments, experimental details, setup, and hyperparameters can be found in Appendix B.5.

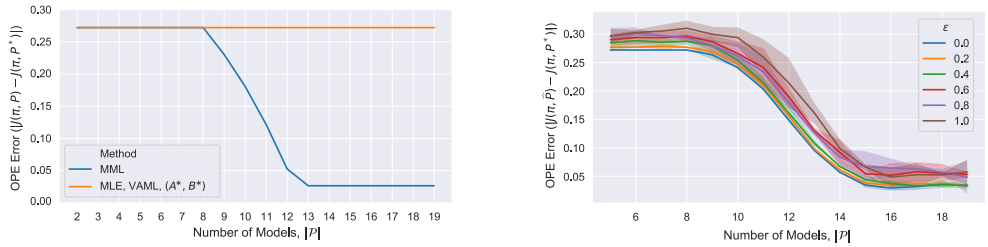


Figure 4.2: *LQR. (Left, OPE Error)* MML finds the $P \in \mathcal{P}$ with the lowest OPE error as $\mathcal{P}$ gets richer. Since calculations are done in expectation, no error bars are included. *(Right, Verifiability)* The OPE error (smoothed) increases with misspecification in $\mathcal{V}$ parametrized by ϵ, the expected MSE between the true $V_\pi^{P^*} \notin \mathcal{V}$ and the approximated $\widehat{V}_\pi^{P^*} \in \mathcal{V}$. Nevertheless, directionally they all follow the same trajectory as $\mathcal{P}$ gets richer.

Results

Does MML prefer models that minimize the OPE objective? We vary the size of the model class Figure 4.2 (left) testing to see if MML will pick up on the models which have better OPE performance. When the sizes of $|\mathcal{P}|$ are small, each method selects (A^*, B^*) (e.g. $P(s'|s,a) = A^*s + B^*a$), the deterministic version of the optimal model. However, increasing the richness of $\mathcal{P}$, MML begins to pick up on models that are able to better evaluate π.

Two remarks are in order. In LQR, policy optimization in (A^*, B^*) coincides with policy optimization in P^*. Therefore, if we tried to do policy optimization in our selected model then our policy would be suboptimal in P^*. Secondly, MML favors a model other than (A^*, B^*) because a good OPE estimate relies on appoximating the contribution from the stochastic part of P^*.

There is a trade-off between the OPE objective and the OPO objective. MML's preference is dependent on the capacities of $\mathcal{P}, \mathcal{W}, \mathcal{V}$. Figure 4.2 (left) illustrates OPE is preferred for $\mathcal{W}$ fixed. Appendix Figure B.1 explores the OPO objective and shows that if we increase $\mathcal{W}$ then OPO becomes favored. In some sense we are asking MML to be robust to many more OPE problems as $|\mathcal{W}| \uparrow$ and so the performance on any single one decreases, favoring OPO.

What can we expect when we have misspecification in $\mathcal{V}$? To check verifiability in practice, we would run π in a few $P \in \mathcal{P}$ and calculate V_π^P. We would check if $V_\pi^P \in \mathcal{V}$ by fitting $\widehat{V}_\pi^P$ and measuring the empirical gap $E[(\widehat{V}_\pi^P - V_\pi^P)^2] = \epsilon^2$.

Figure 4.2 (right) shows how MML performs when $V_\pi^P \notin \mathcal{V}$ but we do have

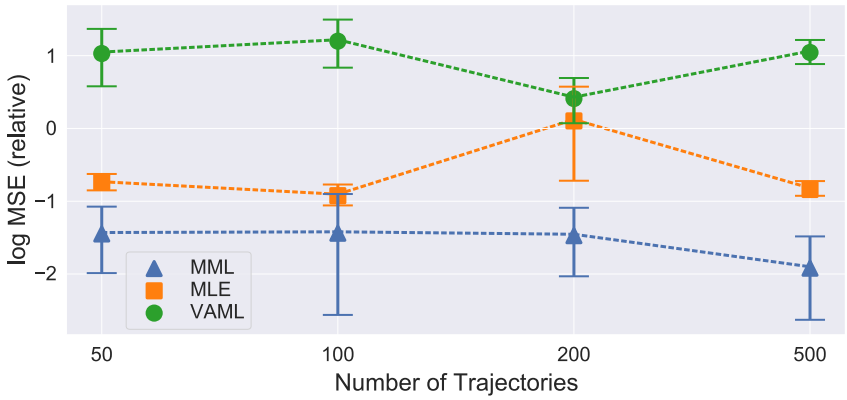


Figure 4.3: *(Cartpole, OPE Error) Comparison of model-based approaches for OPE. Lower is better. MML outperforms others. Not pictured: traditional model-free methods such as IS/PDIS have error of order 3-8.*

$\widehat{V}_\pi^P(s) = V_\pi^P(s) + \mathcal{N}(0, \epsilon) \in \mathcal{V}$. Since $E[(\widehat{V}_\pi^P - V_\pi^P)^2] = \epsilon^2$ then ϵ is the root-mean squared error between the two functions. Directionally all of the errors go down as $|\mathcal{P}| \uparrow$, however it is clear that ϵ has a noticeable effect. We speculate that if this error not distributed around zero and instead is dependent on the state then the effects can be worse.

How does MML perform against MLE and VAML in OPE? In addition to Figure 4.2 (left), Figure 4.3 also illustrates that our method outperforms the other model-learning approaches in OPE. The environment and reward function is challenging, requiring function approximation. Despite the added complexity of solving a minimax problem, doing so gives nearly an order of magnitude improvement over MLE and many orders over VAML. This validates that MML is a good choice for model-learning for OPE.

Algorithm 2 OPO Algorithm (based on MOREL [106])

Require: D, $\mathcal{L}$ among {MML, MLE, VAML}

1: Learn an ensemble of dynamics $P_1, \ldots, P_4 \in \mathcal{P}$ using $P_i = \arg\min_{P \in \mathcal{P}} \mathcal{L}(D)$

2: Construct a pessimistic MDP $\mathcal{M}$ (see Appendix B.5) with $P(s,a) = \frac{1}{4}\sum_{i=1}^4 P_i(s,a)$.

3: $\widehat{\pi} \leftarrow \text{PPO}(\mathcal{M})$ (Best of 3) [180]

Does our approach complement modern offline RL approaches? We integrate MML, VAML, and MLE with MOREL as in Algorithm 2. Consequently, Figure 4.4 shows that MML performs competitively with the other methods, achieving near-optimal performance as the number of trajectories

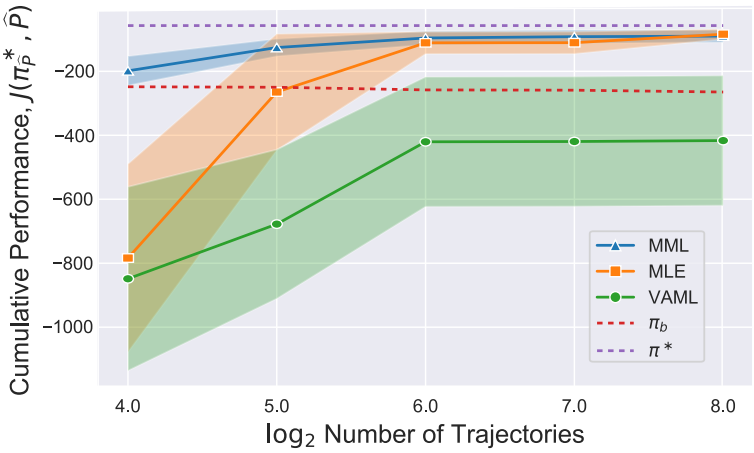


Figure 4.4: *(Invert. Pend., OPO Performance) Comparison of model-based approaches for OPO with function-approx using Algorithm 2. Higher is better. MML performs competitively even in low data regimes.*

increases. MML has good performance even in the low-data regime, whereas other methods perform worse than π_b. Performance in the low-data regime is of particular interest since sample efficiency is highly desirable.

Algorithm 2 forms a pessimistic MDP where a policy is penalized if it enters a state where there is disagreement between $P_1, \ldots, P_4$. Given that MML performs well in low-data, we can reason that MML produces models with support that stays within the dataset D or generalize well slightly outside this set. The other models poor performance is suggestive of incorrect over-confidence outside of D and PPO produces a policy taking advantage of this.

4.7 Other Related Work

Minimax and Model-Based RL. [170] introduce an iterative minimax approach to simultaneously find the optimal-policy and a model of the environment. Despite distribution-shift correction, online data collection is required and is not comparable to MML, where we focus on the batch setting.

Batch (Offline) Model-Based RL Recent improvements in batch model-based RL focus primarily on the issue of policies taking advantage of errors in the model [106, 48, 42, 95]. These improvements typically involve uncertainty quantification to keep the agent in highly certain states to avoid model exploitation. These improvements are independent of the loss function involved.

4.8 Discussion and Future Work

We have presented a novel approach to learning a model for batch, off-policy model-based reinforcement learning. Our approach follows naturally from the definitions of the OPE and OPO objectives and enjoys distributional robustness and decision-awareness. We examined different scenarios under which our method coincided with other methods as well as when closed form solutions were available. We provided sample complexity analysis and misspecification analysis. Finally, we empirically validated that our method was competitive with current model learning approaches.

A key component throughout this chapter has been the function class $\mathcal{W} \times \mathcal{V}$. Finding other interpretations for this term may prove to be useful outside of MML and is of interest in future work. Furthermore, MML remains part of a two-step OPO pipeline: first learn the model, then return the optimal policy in that model. Another direction of future research is to have a single-shot batch OPO objective that returns both a model and the optimal policy simultaneously, in effect combining MML with the minimax algorithm in [170]. Finally, it may be interesting to integrate MML with other forms of distributionally robust model learning, e.g., Liu et al. [132].

Chapter 5

ADVANCES IN MODEL FREE OPE

We propose Fitted Q Evaluation (FQE), a new and simple method for model-free off-policy policy evaluation (OPE) and derive PAC-style bounds. We show experimentally that our OPE method outperforms other popular OPE techniques on a standalone basis, especially in a high-dimensional setting.

5.1 Fitted Q Evaluation (FQE) for Off Policy Evaluation

We propose a new and simple model-free technique using function approximation, called Fitted Q Evaluation (FQE). FQE is based on an iterative reductions scheme similar to Fitted Q Iteration (FQI) [56], but for the problem of off-policy evaluation. Algorithm 3 lays out the steps. The key difference with FQI is that the *min* operator is replaced by $Q_{k-1}(s'_i, \pi(s'_i))$ (Line 3 of Algorithm 3). Each s'_i comes from the original D. Since we know $\pi(s'_i)$, each $\widetilde{\mathrm{D}}_k$ is well-defined. Note that FQE can be augmented with the doubly-robust techniques seen in Chapter 3.

Algorithm 3 Fitted Q Evaluation: $\texttt{FQE}(\pi, c)$

Require: Dataset $\mathrm{D} = \{s_i, a_i, s'_i, c_i\}_{i=1}^n \sim \pi_{\mathrm{D}}$. Function class F. Policy π to be evaluated

1: Initialize $Q_0 \in \mathrm{F}$ randomly
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Compute target $y_i = c_i + \gamma Q_{k-1}(s'_i, \pi(s'_i)) \ \forall i$
4: Build training set $\widetilde{\mathrm{D}}_k = \{(s_i, a_i), y_i\}_{i=1}^n$
5: Solve a supervised learning problem:
$$Q_k = \arg\min_{f \in \mathrm{F}} \frac{1}{n} \sum_{i=1}^n (f(s_i, a_i) - y_i)^2$$
6: **return** $Q_K(s, \pi(s))$ for any s

5.2 Generalization Guarantee of FQE

In this section, we provide sample complexity analysis for FQE. We use the notion of pseudo-dimension as capacity measure of non-linear function class F [65]. Pseudo-dimension $\texttt{dim}_{\mathrm{F}}$, which naturally extends VC dimension into the regression setting, is defined as the VC dimension of the function class induced

by the sub-level set of functions of F: $\texttt{dim}_{\text{F}} = \texttt{VC-dim}(\{(x, y) \mapsto \text{sign}(f(x) - y) : f \in \text{F}\})$.

Pseudo-dimension is finite for a large class of function approximators. For example, Bartlett et al. [18] bounded the pseudo-dimension of piece-wise linear deep neural networks (e.g., with ReLU activations) as $O(WL \log W)$, where W is the number of weights, and L is the number of layers.

FQE relies on reductions to supervised learning to update the value function. In off-policy evaluation, the evaluation policy induces a different state-action distribution compared to the data generating distribution μ. We use the notion of concentration coefficient for the worst case, proposed by [147], to measure the degree of distribution shift. The following standard assumption from analysis of related ADP algorithms limits the severity of distribution shift over future time steps:

Assumption 5.1 (Concentration coefficient of future state-action distribution)**.** [147, 148, 150, 11, 12, 119, 118, 57, 140]
Let P^π be the operator acting on $f : \mathcal{X} \to \mathbb{R}$ s.t.

$$(P^\pi f)(s, a) = \int_{\mathcal{S}} f(s', \pi(s')) P(ds'|s, a).$$

Given data generating distribution μ, initial state distribution χ, for $m \geq 0$ and an arbitrary sequence of stationary policies $\{\pi_m\}_{m \geq 1}$ define the concentration coeffient:

$$\beta_\mu(m) = \sup_{\pi_1, \ldots, \pi_m} \left\| \frac{d(d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m})}{d\mu} \right\|_\infty .$$

We assume $\beta_\mu = (1 - \gamma)^2 \sum_{m \geq 1} m \gamma^{m-1} \beta_\mu(m) < \infty$.

This assumption is valid for a fairly large class of MDPs [148]. For instance β_μ is finite for any finite MDP, or any infinite state-space MDP with bounded transition density.[1] Having a finite concentration coefficient is equivalent the top-Lyapunov exponent $\Gamma \leq 0$ [25], which means the underlying stochastic system is stable. We show below a simple sufficient condition for Assumption 5.1 (albeit stronger than necessary).

[1]This assumption ensures sufficient data diversity, even when the executing policy is deterministic. An example of how learning can fail without this assumption is based on the "combination lock" MDP [109]. In this deterministic MDP example, β_μ can grow arbitrarily large, and we need an exponential number of samples for both FQE and FQI. See Appendix C.1.

Example 5.2.1. Consider an MDP such that for any non-stationary distribution ρ, the marginals over states satisfy $\frac{\rho_s(s)}{\mu_s(s)} \leq L$ (i.e., transition dynamics are sufficiently stochastic), and $\exists M: \ \forall s, a: \ \mu(a|s) > \frac{1}{M}$ (i.e., the behavior policy is sufficiently exploratory). Then $\beta_\mu \leq LM$.

Recall that for a given policy π, the Bellman (evaluation) operator is defined as $(\mathbb{T}^\pi Q)(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} Q(s', \pi(s')) P(ds'|s,a)$. In general $\mathbb{T}^\pi f$ may not belong to F for $f \in \mathrm{F}$. For FQE (and FQI), the main operation in the algorithm is to iteratively project $\mathbb{T}^\pi Q_{k-1}$ back to F via $Q_k = \arg\min_{f \in \mathrm{F}} \|f - \mathbb{T}^\pi Q_{k-1}\|$. The performance of both FQE and FQI thus depend on how well the function class F approximates the Bellman operator. We measure the ability of function class F to approximate the Bellman evaluation operator via the worst-case Bellman error:

Definition 5.2.1 (inherent Bellman evaluation error)**.** Given a function class F and policy π, the *inherent Bellman evaluation error* of F is defined as $d_{\mathrm{F}}^\pi = \sup_{g \in \mathrm{F}} \inf_{f \in \mathrm{F}} \|f - \mathbb{T}^\pi g\|_\pi$ where $\|\cdot\|_\pi$ is the ℓ_2 norm weighted by the state-action distribution induced by π.

We are now ready to state the generalization bound for FQE:

Theorem 5.2.1 (Generalization error of FQE)**.** *Let the costs be bounded above by $\bar{C}$. Under Assumption 5.1, for $\epsilon > 0$ & $\delta \in (0,1)$, after K iterations of Fitted Q Evaluation (Algorithm 3), for $n = O\Big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \mathtt{dim}_{\mathrm{F}} \log\frac{\overline{C}^2}{\epsilon^2} + \log \mathtt{dim}_{\mathrm{F}})\Big)$, we have with probability $1-\delta$:*

$$|V(\pi) - V_K(\pi)| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\Big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}}^\pi + \epsilon) + \frac{2\gamma^{K/2}\overline{C}}{(1-\gamma)^{1/2}}\Big).$$

This result shows a dependency on ϵ of $\widetilde{O}(\frac{1}{\epsilon^2})$, compared to $\widetilde{O}(\frac{1}{\epsilon^4})$ from other related ADP algorithms [150, 12]. The price that we pay is a multiplicative constant 2 in front of the inherent error d_{F}^π. The error from second term on RHS decays exponentially with iterations K. The proof is in Appendix C.2.

5.3 Empirical Analysis

We invite the reader to see Chapter 3 for a full empirical study on the performance of FQE along with many state-of-the-art OPE methods. There will also be additional experiments in Chapter 6 when FQE will be used as a subroutine in policy learning as a means of guaranteeing post-learning performance.

Part III

POLICY LEARNING WITH GUARANTEES

Chapter 6

VALUE-BASED GUARANTEES

When learning policies for real-world domains, two important questions arise: (i) how to efficiently use pre-collected off-policy, non-optimal behavior data; and (ii) how to mediate among different competing objectives and constraints. We thus study the problem of batch policy learning under multiple constraints, and offer a systematic solution. We first propose a flexible meta-algorithm that admits any batch reinforcement learning and online learning procedure as subroutines. We then present a specific algorithmic instantiation and provide performance guarantees for the main objective and all constraints. Our algorithm achieves strong empirical results in different domains, including in a challenging problem of simulated car driving subject to multiple constraints such as lane keeping and smooth driving. We also show experimentally that our choice of OPE method (FQE) outperforms other popular OPE techniques on a standalone basis, especially in a high-dimensional setting.

6.1 Introduction

We study the problem of policy learning under multiple constraints. Contemporary approaches to learning sequential decision making policies have largely focused on optimizing some cost objective that is easily reducible to a scalar value function. However, in many real-world domains, choosing the right cost function to optimize is often not a straightforward task. Frequently, the agent designer faces multiple competing objectives. For instance, consider the aspirational task of designing autonomous vehicle controllers: one may care about minimizing the travel time while making sure the driving behavior is safe, comfortable, or fuel efficient. Indeed, many such real-world applications require the primary objective function be augmented with an appropriate set of constraints [9].

Contemporary policy learning research has largely focused on either online reinforcement learning (RL) with a focus on exploration, or imitation learning (IL) with a focus on learning from expert demonstrations. However, many real-

world settings already contain large amounts of pre-collected data generated by existing policies (e.g., existing driving behavior, power grid control policies, etc.). We thus study the complementary question: *can we leverage this abundant source of (non-optimal) behavior data in order to learn sequential decision making policies with provable guarantees on constraint satisfaction*?

We thus propose and study the problem of batch policy learning under multiple constraints. Historically, batch RL is regarded as a subfield of approximate dynamic programming (ADP) [116], where a set of transitions sampled from the existing system is given and fixed. From an interaction perspective, one can view many online RL methods (e.g., DDPG [129]) as running a growing batch RL subroutine per round of online RL. In that sense, batch policy learning is complementary to any exploration scheme. To the best of our knowledge, the study of constrained policy learning in the batch setting is novel.

We present an algorithmic framework for learning sequential decision making policies from off-policy data. We employ multiple learning reductions to online and supervised learning, and present an analysis that relates performance in the reduced procedures to the overall performance with respect to both the primary objective and constraint satisfaction.

Constrained optimization is a well studied problem in supervised machine learning and optimization. In fact, our approach is inspired by the work of Agarwal et al. [5] in the context of fair classification. In contrast to supervised learning for classification, batch policy learning for sequential decision making introduces multiple additional challenges. First, setting aside the constraints, batch policy learning itself presents a layer of difficulty, and the analysis is significantly more complicated. Second, verifying whether the constraints are satisfied is no longer as straightforward as passing the training data through the learned classifier. In sequential decision making, certifying constraint satisfaction amounts to an off-policy policy evaluation problem, which is a challenging problem and the subject of active research. In this chapter, we develop a systematic approach to address these challenges, provide a careful error analysis, and experimentally validate our proposed algorithms. In summary, our contributions are:

- We formulate the problem of batch policy learning under multiple constraints, and present the first approach of its kind to solve this problem. The definition of constraints is general and can subsume many objectives. Our meta-algorithm utilizes multi-level learning reductions, and we show

how to instantiate it using various batch RL and online learning subroutines. We show that guarantees from the subroutines provably lift to provide end-to-end guarantees on the original constrained batch policy learning problem.

- Leveraging techniques from batch RL as a subroutine, we provide a refined theoretical analysis for general non-linear function approximation that improves upon the previously known sample complexity bound [150] from $O(n^4)$ to $O(n^2)$.
- To evaluate and verify constraint satisfaction, we propose a simple new technique for off-policy policy evaluation, which is used as a subroutine in our main algorithm. We show that it is competitive to other off-policy policy evaluation methods.
- We validate our algorithm and analysis with two experimental settings. First, a simple navigation domain where we consider safety constraint. Second, we consider a high-dimensional racing car domain with smooth driving and lane keeping constraints.

6.2 Problem Formulation

We follow the notation introduced in Section 2.1. Let $\mathcal{S} \subset \mathbb{R}^d$ be a bounded and closed d-dimensional state space. Let $\mathcal{A}$ be a finite action space. Let $c : \mathcal{X} \mapsto [0, \overline{C}]$ be the primary objective cost function that is bounded by $\overline{C}$. Let there be m constraint cost functions, $g_i : \mathcal{X} \mapsto [0, \overline{G}]$, each bounded by $\overline{G}$. To simplify the notation, we view the set of constraints as a vector function $g : \mathcal{X} \mapsto [0, \overline{G}]^m$ where $g(s, a)$ is the column vector of individual $g_i(s, a)$. Let $P(\cdot|s, a)$ denote the (unknown) transition/dynamics model that maps state/action pairs to a distribution over the next state. Let $\gamma \in (0, 1)$ denote the discount factor. Let d_0 be the initial states distribution.

We consider the discounted infinite horizon setting. An MDP is defined using the tuple $(\mathcal{S}, \mathcal{A}, c, g, P, \gamma, d_0)$. A policy $\pi \in \Pi$ maps states to actions, i.e., $\pi(s) \in \mathcal{A}$. The value function $C^\pi : \mathcal{S} \mapsto \mathbb{R}$ corresponding to the primary cost function c is defined in the usual way: $C^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \mid s_0 = s\right]$, over the randomness of the policy π and transition dynamics P. We similarly define the vector-value function for the constraint costs $G^\pi : \mathcal{S} \mapsto \mathbb{R}^m$ as $G^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t, a_t) | s_0 = s\right]$. Define $C(\pi)$ and $G(\pi)$ as the expectation of $C^\pi(s)$ and $G^\pi(s)$, respectively, over the distribution d_0 of initial states.

Batch Policy Learning under Constraints

In batch policy learning, we have a pre-collected dataset,

$$\mathrm{D} = \{(s_i, a_i, s_i', c(s_i, a_i), g_{1:m}(s_i, a_i)\}_{i=1}^{n},$$

generated from (a set of) historical behavioral policies denoted jointly by π_{D}. The goal of batch policy learning under constraints is to learn a policy $\pi \in \Pi$ from D that minimizes the primary objective cost while satisfying m different constraints:

$$\begin{aligned} \min_{\pi \in \Pi} \quad & C(\pi) \\ \text{s.t.} \quad & G(\pi) \leq \tau, \end{aligned} \tag{OPT}$$

where $G(\cdot) = [g_1(\cdot), \ldots, g_m(\cdot)]^\top$ and $\tau \in \mathbb{R}^m$ is a vector of known constants. We assume that (OPT) is feasible. However, the dataset D might be generated from multiple policies that violate the constraints.

Examples of Policy Learning with Constraints

Counterfactual & Safe Policy Learning. In conventional online RL, the agent needs to "re-learn" from scratch when the cost function is modified. Our framework enables counterfactual policy learning assuming the ability to compute the new cost objective from the same historical data. A simple example is *safe* policy learning [70]. Define safety cost $G(s, a) = \varphi(s, a, c)$ as a new function of existing cost c and features associated with current state-action pair. The goal here is to counterfactually avoid undesirable behaviors observed from historical data. We experimentally study this safety problem in Section 6.5.

Other examples from the literature that belong to this safety perspective include planning under chance constraints [158, 23]. The constraint here is $G(\pi) = \mathbb{E}[\mathbb{I}(s \in \mathcal{S}_{error})] = \mathrm{P}(s \in \mathcal{S}_{error}) \leq \tau$.

Multi-objective Batch Learning. Traditional policy learning (RL or IL) presupposes that the agent optimizes a single cost function. In reality, we may want to satisfy multiple objectives that are not easily reducible to a scalar objective function. One example is learning fast driving policies under multiple behavioral constraints such as smooth driving and lane keeping consistency (see Section 6.5).

Equivalence between Constraint Satisfaction and Regularization

Our constrained policy learning framework subsumes several existing regularized policy learning settings. Regularization typically encodes prior knowledge, and has been used extensively in the RL and IL literature to improve learning performance. Many instances of regularized policy learning can be naturally cast into (OPT):

- *Entropy regularized RL* [78, 231] is equivalent to $G(\pi) = \mathbb{H}(\pi)$, where $\mathbb{H}(\pi)$ measures policy entropy.
- *Smooth imitation learning* [122] is equivalent to $G(\pi) = \min_{h \in \mathrm{H}} \Delta(h, \pi)$, where H is a class of provably smooth policies and Δ is a divergence metric.
- *Regularizing RL with expert demonstration* [91] is equivalent to $G(\pi) = \mathbb{E}[\ell(\pi(x), \pi^*(x))]$, where π^* is the expert policy.
- *Conservative policy improvement* [125, 179, 3] is equivalent to $G(\pi) = D_{KL}(\pi, \pi_k)$, where π_k is some "well-behaving" policy.

We provide a detailed equivalence derivation of the above examples in Appendix D.1. Of course, some problems are more naturally described using the regularization perspective, while others using constraint satisfaction.

More generally, one can establish the equivalence between regularized and constrained policy learning via a simple appeal to Lagrangian duality as shown in Proposition 6.2.1 below. This Lagrangian duality also has a game-theoretic interpretation (Section 5.4 of Boyd and Vandenberghe [26]), which serves as an inspiration for developing our approach.

Proposition 6.2.1. *Let Π be a convex set of policies. Let $C : \Pi \mapsto \mathbb{R}, C : \Pi \mapsto \mathbb{R}^K$ be value functions. Consider the two policy optimization tasks:*

$$\texttt{Regularization:} \quad \min_{\pi \in \Pi} \quad C(\pi) + \lambda^\top G(\pi)$$

$$\texttt{Constraint:} \quad \min_{\pi \in \Pi} \quad C(\pi) \quad \text{s.t. } G(\pi) \leq \tau.$$

Assume that the Slater's condition is satisfied in the `Constraint` *problem (i.e., $\exists \pi$ s.t. $G(\pi) < \tau$). Assume also that the constraint cannot be removed without changing the optimal solution, i.e., $\inf_{\pi \in \Pi} C(\pi) < \inf_{\pi \in \Pi : G(\pi) \leq \tau} C(\pi)$. Then $\forall$ $\lambda > 0$, $\exists$ τ, and vice versa, such that* `Regularization` *and* `Constraint` *share the same optimal solutions. (Proof in Appendix D.1.)*

Algorithm 4 Meta-algo for Batch Constrained Learning

for each round t **do**
  $\pi_t \leftarrow$ `Best-response`(λ_t)
  $\widehat{\pi}_t \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \pi_{t'}$, $\widehat{\lambda}_t \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \lambda_{t'}$
  $\mathrm{L}_{\max} = \max_{\lambda} L(\widehat{\pi}_t, \lambda)$
  $\mathrm{L}_{\min} = L($`Best-response`$(\widehat{\lambda}_t), \widehat{\lambda}_t)$
  if $\mathrm{L}_{\max} - \mathrm{L}_{\min} \leq \omega$ **then**
    Return $\widehat{\pi}_t$
  $\lambda_{t+1} \leftarrow$ `Online-algorithm`$(\pi_1, \ldots, \pi_{t-1}, \pi_t)$

6.3 Proposed Approach

To make use of strong duality, we first *convexify* the policy class Π by allowing stochastic combinations of policies, which effectively expands Π into its convex hull `Conv`(Π). Formally, `Conv`(Π) contains *randomized policies*,[1] which we denote $\pi = \sum_{t=1}^{T} \alpha_t \pi_t$ for $\pi_t \in \Pi$ and $\sum_{t=1}^{T} \alpha_t = 1$. Executing a mixed π consists of first sampling *one* policy π_t from $\pi_{1:T}$ according to distribution $\alpha_{1:T}$, and then executing π_t. Note that we still have $\mathbb{E}[\pi] = \sum_{t=1}^{T} \alpha_t \mathbb{E}[\pi_t]$ for any first-moment statistic of interest (e.g., state distribution, expected cost). It is easy to see that the augmented version of (OPT) over `Conv`(Π) has a solution at least as good as the original (OPT). As such, to lighten the notation, we will equate Π with its convex hull for the rest of the chapter.

Meta-Algorithm

The Lagrangian of (OPT) is $L(\pi, \lambda) = C(\pi) + \lambda^\top (G(\pi) - \tau)$ for $\lambda \in \mathbb{R}^m_+$. Clearly (OPT) is equivalent to the min-max problem: $\min_{\pi \in \Pi} \max_{\lambda \in \mathbb{R}^k_+} L(\pi, \lambda)$. We assume (OPT) is feasible and that Slater's condition holds (otherwise, we can simply increase the constraint τ by a tiny amount). Slater's condition and policy class convexification ensure that strong duality holds [26], and (OPT) is also equivalent to the max-min problem: $\max_{\lambda \in \mathbb{R}^k_+} \min_{\pi \in \Pi} L(\pi, \lambda)$.

Since $L(\pi, \lambda)$ is linear in both λ and π, strong duality is also a consequence of von Neumann's celebrated convex-concave minimax theorem for zero-sum games [217]. From a game-thoeretic perspective, the problem becomes finding the equilibrium of a two-player game between the $\pi-$player and the $\lambda-$player [64]. In this repeated game, the $\pi-$player minimizes $L(\pi, \lambda)$ given the current λ, and the $\lambda-$player maximizes it given the current (mixture over) π.

We first present a meta-algorithm (Algorithm 4) that uses any no-regret online

[1] This places no restrictions on the individual policies. Individual policies can be arbitrarily non-convex. Convexifiying a policy class amounts to allowing ensembles of learned policies.

learning algorithm (for λ) and batch policy optimization (for π). At each iteration, given λ_t, the π-player runs `Best-response` to get the best response:

$$\begin{aligned}\texttt{Best-response}(\lambda_t) &= \arg\min_{\pi\in\Pi} L(\pi, \lambda_t) \\ &= \arg\min_{\pi\in\Pi} C(\pi) + \lambda_t^\top (G(\pi) - \tau).\end{aligned}$$

This is equivalent to a standard batch reinforcement learning problem where we learn a policy that is optimal with respect to $c + \lambda_t^\top g$. The corresponding mixed strategy is the uniform distribution over all previous π_t. In response to the $\pi-$player, the $\lambda-$player employs `Online-algorithm`, which can be *any* no-regret algorithm that satisfies:

$$\sum_t L(\pi_t, \lambda_t) \geq \max_\lambda \sum_t L(\pi_t, \lambda) - o(T).$$

Finally, the algorithm terminates when the estimated primal-dual gap is below a threshold ω (Lines 7-8).

Leaving aside (for the moment) issues of generalization, Algorithm 4 is guaranteed to converge assuming: (i) `Best-response` gives the best single policy in the class, and (ii) $\mathrm{L}_{\max}$ and $\mathrm{L}_{\min}$ can be evaluated exactly.

Proposition 6.3.1. *Assuming (i) and (ii) above, Algorithm 4 is guaranteed to stop and the convergence depends on the regret of* `Online-algorithm`*. (Proof in Appendix D.2.)*

Our Main Algorithm

We now focus on a specific instantiation of Algorithm 4. Algorithm 5 is our main algorithm in this chapter.

Policy Learning. We instantiate `Best-response` with Fitted Q Iteration (FQI), a model-free off-policy learning approach [56]. FQI relies on a series of reductions to supervised learning. The key idea is to approximate the true action-value function Q^* by a sequence $\{Q_k \in \mathrm{F}\}_{k=0}^K$, where F is a chosen function class.

In Lines 3 & 9, $\texttt{FQI}(c+\lambda^\top g)$ is defined as follows. With Q_0 randomly initialized, for each $k = 1, \ldots, K$, we form a new training dataset $\widetilde{\mathrm{D}}_k = \{(s_i, a_i), y_i\}_{i=1}^n$ where:

$$\forall i: \quad y_i = (c_i + \lambda^\top g_i) + \gamma \min_a Q_{k-1}(s_i', a),$$

Algorithm 5 Constrained Batch Policy Learning

Require: Dataset $\mathrm{D} = \{s_i, a_i, s_i', c_i, g_i\}_{i=1}^n \sim \pi_\mathrm{D}$. Online algorithm parameters: ℓ_1 norm bound B, learning rate η

1: Initialize $\lambda_1 = (\frac{B}{m+1}, \ldots, \frac{B}{m+1}) \in \mathbb{R}^{m+1}$
2: **for** each round t **do**
3: Learn $\pi_t \leftarrow \texttt{FQI}(c + \lambda_t^\top g)$ // *FQI with cost* $c + \lambda_t^\top g$
4: Evaluate $\widehat{C}(\pi_t) \leftarrow \texttt{FQE}(\pi_t, c)$ // *Algo 3 with* π_t, *cost* c
5: Evaluate $\widehat{G}(\pi_t) \leftarrow \texttt{FQE}(\pi_t, g)$ // *Algo 3 with* π_t, cost g
6: $\widehat{\pi}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \pi_{t'}$
7: $\widehat{C}(\widehat{\pi}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{C}(\pi_{t'})$, $\widehat{G}(\widehat{\pi}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{G}(\pi_{t'})$
8: $\widehat{\lambda}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \lambda_{t'}$
9: Learn $\widetilde{\pi} \leftarrow \texttt{FQI}(c + \widehat{\lambda}_t^\top g)$ // *FQI with cost* $c + \widehat{\lambda}_t^\top g$
10: Evaluate $\widehat{C}(\widetilde{\pi}) \leftarrow \texttt{FQE}(\widetilde{\pi}, c), \widehat{G}(\widetilde{\pi}) \leftarrow \texttt{FQE}(\widetilde{\pi}, g)$
11: $\widehat{\mathrm{L}}_{\max} = \max_{\lambda, \|\lambda\|_1 = B} \left(\widehat{C}(\widehat{\pi}_t) + \lambda^\top \left[(\widehat{G}(\widehat{\pi}_t) - \tau)^\top, 0 \right]^\top \right)$
12: $\widehat{\mathrm{L}}_{\min} = \widehat{C}(\widetilde{\pi}) + \widehat{\lambda}_t^\top \left[(\widehat{G}(\widetilde{\pi}) - \tau)^\top, 0 \right]^\top$
13: **if** $\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min} \leq \omega$ **then**
14: Return $\widehat{\pi}_t$
15: Set $z_t = \left[(\widehat{G}(\pi_t) - \tau)^\top, 0 \right]^\top \in \mathbb{R}^{m+1}$
16: $\lambda_{t+1}[i] = B \frac{\lambda_t[i] e^{-\eta z_t[i]}}{\sum_j \lambda_t[j] e^{-\eta z_t[j]}} \forall i$ // $\lambda[i]$ *the* i^{th} *coordinate*

and $(s_i, a_i, s_i', c_i, g_i) \sim \mathrm{D}$ (original dataset). A supervised regression procedure is called to solve for:

$$Q_k = \arg\min_{f \in \mathrm{F}} \frac{1}{n} \sum_{i=1}^n (f(s_i, a_i) - y_i)^2.$$

The policy then: $\pi_K = \arg\min_a Q_K(\cdot, a)$.

FQI has been shown to work well with several empirical domains: spoken dialogue systems [162], physical robotic soccer [173], and cart-pole swing-up [172]. Another possible model-free subroutine is Least-Squares Policy Iteration (LSPI) [115]. One can also consider model-based alternatives [159].

Off-policy Policy Evaluation. A crucial difference between constrained policy learning and existing work on constrained supervised learning is the technical challenge of evaluating the objective and constraints. First, estimating $\widehat{L}(\pi, \lambda)$ (Lines 11-12) requires estimating $\widehat{C}(\pi)$ and $\widehat{G}(\pi)$. Second, any gradient-based approach to `Online-learning` requires passing in $\widehat{G}(\pi) - \tau$ as part of gradient estimate (line 15). This problem is known as the off-policy policy evaluation (OPE) problem: we need to evaluate $\widehat{C}(\pi)$ and $\widehat{G}(\pi)$ having only access to data $\mathrm{D} \sim \pi_\mathrm{D}$. We studied existing OPE techniques in Chapter 3, and proposed two methods in subsequent chapters. We will be using FQE, from

Chapter 5, to instantiate Algorithm 5

Online Learning Subroutine. As $L(\pi_t, \lambda)$ is linear in λ, many online convex optimization approaches can be used for `Online-algorithm`. Perhaps the simpliest choice is Online Gradient Descent (OGD) [234]. We include an instantiation using OGD in Appendix D.6.

For our main Algorithm 5, similar to [5], we use Exponentiated Gradient (EG) [108], which has a regret bound of $O(\sqrt{\log(m)T})$ instead of $O(\sqrt{mT})$ as in OGD. One can view EG as a variant of Online Mirror Descent [153] with a softmax link function, or of Follow-the-Regularized-Leader with entropy regularization [182]. Gradient-based algorithms generally require bounded λ. We thus force $\|\lambda\|_1 \leq B$ using hyperparameter B. Solving (OPT) exactly requires $B = \infty$. We will analyze Algorithm 5 with respect to finite B. With some abuse of notation, we augment λ into a $(m+1)-$dimensional vector by appending $B - \|\lambda\|_1$, and augment the constraint cost vector g by appending 0 (Lines 11, 12 & 15 of Algorithm 5).[2]

6.4 Theoretical Analysis

Convergence Guarantee

The convergence rate of Algorithm 5 depends on the radius B of the dual variables λ, the maximal constraint value $\overline{G}$, and the number of constraints m. In particular, we can show $O(\frac{B^2}{\omega^2})$ convergence for primal-dual gap ω.

Theorem 6.4.1 (Convergence of Algorithm 5)**.** *After T iterations, the empirical duality gap is bounded by*

$$\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min} \leq 2\frac{B\log(m+1)}{\eta T} + 2\eta B\overline{G}^2.$$

Consequently, to achieve the primal-dual gap of ω, setting $\eta = \frac{\omega}{4\overline{G}^2 B}$ will ensure that Algorithm 5 converges after $\frac{16B^2\overline{G}^2\log(m+1)}{\omega^2}$ iterations. (Proof in Appendix D.2.)

Convergence analysis of our main Algorithm 5 is an extension of the proof to Proposition 6.3.1, leveraging the no-regret property of the EG procedure [182].

[2]The $(m+1)^{th}$ coordinate of g is thus always satisfied. This augmentation is only necessary when executing EG.

Generalization Guarantee of FQI

We recall a few definitions from Chapter 5, including that for concentration of the state-action distribution and inherent bellman error.

Assumption 6.1 (Concentration coefficient of future state-action distribution)**.** [147, 148, 150, 11, 12, 119, 118, 57, 140]
Let P^π be the operator acting on $f : \mathcal{X} \mapsto \mathbb{R}$ s.t.

$$(P^\pi f)(s,a) = \int_{\mathcal{S}} f(x', \pi(x'))P(ds'|s,a).$$

Given data generating distribution μ, initial state distribution d_0, for $m \geq 0$ and an arbitrary sequence of stationary policies $\{\pi_m\}_{m\geq 1}$ define the concentration coefficient:

$$\beta_\mu(m) = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m})}{d\mu} \right\|_\infty .$$

We assume $\beta_\mu = (1-\gamma)^2 \sum\limits_{m\geq 1} m\gamma^{m-1}\beta_\mu(m) < \infty$.

Definition 6.4.1 (Inherent Bellman evaluation error)**.** Given a function class F and policy π, the *inherent Bellman evaluation error* of F is defined as $d_{\mathrm{F}}^\pi = \sup_{g\in \mathrm{F}} \inf_{f\in \mathrm{F}} \|f - \mathbb{T}^\pi g\|_\pi$ where $\|\cdot\|_\pi$ is the ℓ_2 norm weighted by the state-action distribution induced by π.

We can show, analogous to FQE, a generalization bound for FQI. While FQI has been widely used, to the best of our knowledge, a complete analysis of FQI for non-linear function approximation has not been previously reported.[3]

Definition 6.4.2 (Inherent Bellman optimality error)**.** [150] Recall that the Bellman optimality operator is defined as:

$$(\mathbb{T}Q)(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} \min_{a'\in\mathcal{A}} Q(s',a')P(ds'|s,a).$$

Given a function class F, the *inherent Bellman error* is defined as $d_{\mathrm{F}} = \sup_{g\in \mathrm{F}} \inf_{f\in \mathrm{F}} \|f - \mathbb{T}g\|_\mu$, where $\|\cdot\|_\mu$ is the ℓ_2 norm weighted by μ, the state-action distribution induced by π_{D}.

Theorem 6.4.2 (Generalization error of FQI)**.** *Under Assumption 6.1, for $\epsilon > 0$ & $\delta \in (0,1)$, after K iterations of Fitted Q Iteration, for $n = O\Big(\frac{\bar{C}^4}{\epsilon^2}(\log \frac{K}{\delta} +$*

[3] FQI for continuous action space from [11] is a variant of fitted policy iteration and not the version of FQI under consideration. The appendix of [120] contains a proof of FQI but for linear functions.

$\mathtt{dim}_{\mathrm{F}} \log \frac{\overline{C}^2}{\epsilon^2} + \log \mathtt{dim}_{\mathrm{F}})\Big)$, *we have with probability* $1-\delta$*:*

$$\Big|C^* - C(\pi_K)\Big| \leq \frac{2\gamma}{(1-\gamma)^3}\Big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}} + \epsilon) + 2\gamma^{K/2}\overline{C}\Big),$$

where π_K *is the policy acting greedy with respect to the returned function* Q_K*. (Proof in Appendix D.5.)*

End-to-End Generalization Guarantee

We are ultimately interested in the test-time performance and constraint satisfaction of the returned policy from Algorithm 5. We now connect the previous analyses from Theorems 6.4.1, 5.2.1 & 6.4.2 into an end-to-end error analysis.

Since Algorithm 5 uses FQI and FQE as subroutines, the inherent Bellman error terms d_{F} and d^{π}_{F} will enter our overall performance bound. Estimating the inherent Bellman error caused by function approximation is not possible in general (chapter 11 of Sutton and Barto [191]). Fortunately, a sufficiently expressive F can generally make d_{F} and d^{π}_{F} to arbitrarily small. To simplify our end-to-end analysis, we assume $d_{\mathrm{F}} = 0$ and $d^{\pi}_{\mathrm{F}} = 0$, i.e., the function class F is closed under applying the Bellman operator.[4]

Assumption 6.2. We consider function classes F sufficiently rich so that $\forall f:\ \mathbb{T}f \in \mathrm{F}\ \&\ \mathbb{T}^{\pi}f \in \mathrm{F}$ for the policies π returned by Algorithm 5.

With Assumptions 6.1 & 6.2, we have the following error bound:

Theorem 6.4.3 (Generalization guarantee of Algorithm 5)**.** *Let* π^* *be the optimal policy to* (OPT)*. Denote* $\overline{V} = \overline{C} + B\overline{G}$*. Let* K *be the number of iterations of FQE and FQI. Let* $\widehat{\pi}$ *be the policy returned by Algorithm 5, with termination threshold* ω*. For* $\epsilon > 0$ *&* $\delta \in (0,1)$*, when* $n = O\Big(\frac{\overline{V}^4}{\epsilon^2}(\log \frac{K(m+1)}{\delta} + \mathtt{dim}_{\mathrm{F}} \log \frac{\overline{V}^2}{\epsilon^2} + \log \mathtt{dim}_{\mathrm{F}})\Big)$*, we have with probability at least* $1-\delta$*:*

$$C(\widehat{\pi}) \leq C(\pi^*) + \omega + \frac{(4+B)\gamma}{(1-\gamma)^3}\Big(\sqrt{\beta_\mu}\epsilon + 2\gamma^{K/2}\overline{V}\Big),$$

and

$$G(\widehat{\pi}) \leq \tau + 2\frac{\overline{V}+\omega}{B} + \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\Big(\sqrt{\beta_\mu}\epsilon + \frac{2\gamma^{K/2}\overline{V}}{(1-\gamma)^{1/2}}\Big).$$

The proof is in Appendix D.3. This result guarantees that, upon termination of Algorithm 5, the true performance on the main objective can be arbitrarily

[4] A similar assumption was made in Cheng et al. [40] on near-realizability of learning the model dynamics.

close to that of the optimal policy. At the same time, each constraint will be approximately satisfied with high probability, assuming sufficiently large B & K, and sufficiently small ϵ.

6.5 Empirical Analysis

We perform experiments on two different domains: a grid-world domain (from OpenAI's FrozenLake) under a safety constraint, and a challenging high-dimensional car racing domain (from OpenAI's CarRacing) under multiple behavior constraints. We seek to answer the following questions in our experiments: (i) whether the empirical convergence behavior of Algorithm 5 is consistent with our theory, and (ii) how the returned policy performs with respect to the main objective and constraint satisfaction. Appendix D.7 includes a more detailed discussion of our experiments.

Frozen Lake.

Environment & Data Collection. The environment is an 8x8 grid. The agent has 4 actions N,S,E,W at each state. The main goal is to navigate from a starting position to the goal. Each episode terminates when the agent reaches the goal or falls into a hole. The main cost function is defined as $c = -1$ if goal is reached, otherwise $c = 0$ everywhere. We simulate a non-optimal data gathering policy π_{D} by adding random sub-optimal actions to the shortest path policy from any given state to goal. We run π_D for 5000 trajectories to collect the behavior dataset D (with constraint cost measurement specified below).

Counterfactual Safety Constraint. We augment the main objective c with safety constraint cost defined as $g = 1$ if the agent steps into a hole, and $g = 0$ otherwise. We set the constraint threshold $\tau = 0.1$, roughly 75% of the accumulated constraint cost of behavior policy π_{D}. The threshold can be interpreted as a counterfactually acceptable probability that we allow the learned policy to fail.

Results. The empirical primal dual gap $\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min}$ in Figure 6.1 (left) quickly decreases toward the optimal gap of zero. The convergence is fast and monotonic, supporting the predicted behavior from our theory. The test-time performance in Figure 6.1 (middle) shows the safety constraint is always satisfied, while the main objective cost also smoothly converges to the optimal value achieved by an online RL baseline (DQN) trained without regard to the constraint. The returned policy significantly outperformed the data gathering

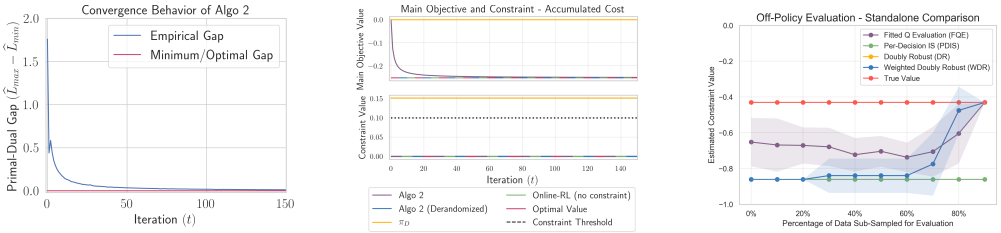


Figure 6.1: *FrozenLake Results. (Left)* Empirical duality gap of algorithm 5 vs. optimal gap. *(Middle)* Comparison of returned policy and others w.r.t. (top) main objective value and (bottom) safety constraint value. *(Right)* FQE vs. other OPE methods on a standalone basis.

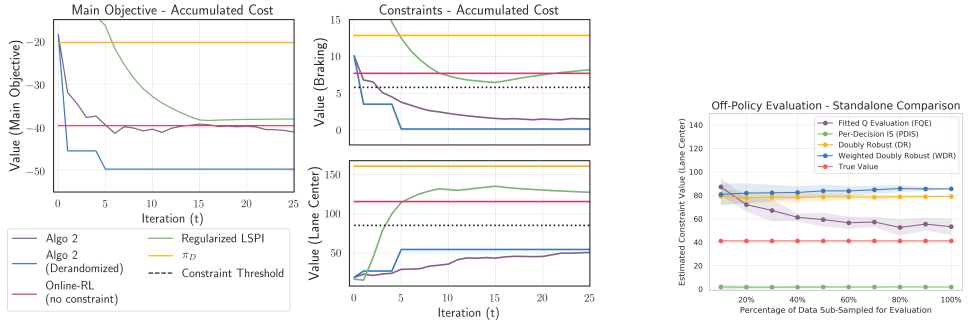


Figure 6.2: *CarRacing Results. (Left)* & *(Middle)* (Lower is better) Comparing our algorithm, regularized LSPI, online RL w/o constraints, behavior policy π_D w.r.t. main cost objectives and two constraints. *(Right)* FQE vs. other OPE methods on a standalone basis.

policy π_{D} on both main and safety cost.

Car Racing.

Environment & Data Collection. The car racing environment, seen in Figure D.1 (right), is a high-dimensional domain where the state is a $96 \times 96 \times 3$ tensor of raw pixels. The action space $\mathcal{A} = \{\text{steering} \times \text{gas} \times \text{brake}\}$ takes 12 discretized values. The goal in this episodic task is to traverse over 95% of the track, measured by a given number of "tiles" as a proxy for distance coverage. The agent receives a reward (negative cost) for each unique tile crossed and no reward if the agent is off-track. A small positive cost applies at every time step, with maximum horizon of 1000 for each episode. With these costs given by the environment, one can train online RL agent using DDQN [211]. We collect ≈ 1500 trajectories from DDQN's randomization, resulting in data set D with ≈ 94000 transition tuples.

Fast Driving under Behavioral Constraints. We study the problem of minimizing environment cost while subject to two behavioral constraints: smooth driving and lane centering. The first constraint G_0 approximates

smooth driving by $g_0(s, a) = 1$ if a contains braking action, and 0 otherwise. The second constraint cost g_1 measures the distance between the agent and center of lane at each time step. This is a highly challenging setup since three objectives and constraints are in direct conflict with one another, e.g., fast driving encourages the agent to cut corners and apply frequent brakes to make turns. Outside of this work, we are not aware of previous work in policy learning with 2 or more constraints in high-dimensional settings.

Baseline and Procedure. As a naïve baseline, DDQN achieves low cost, but exhibits "non-smooth" driving behavior (see our supplementary videos). We set the threshold for each constraint to 75% of the DDQN benchmark. We also compare against regularized batch RL algorithms [57], specifically regularized LSPI. We instantiate our subroutines, FQE and FQI, with multi-layered CNNs. We augment LSPI's linear policy with non-linear features derived from a well-performing FQI model.

Results. The returned mixture policy from our algorithm achieves low main objective cost, comparable with online RL policy trained without regard to constraints. After several initial iterations violating the braking constraint, the returned policy — corresponding to the appropriate λ trade-off — satisties both constraints, while improving the main objective. The improvement over data gathering policy is significant for both constraints and main objective.

Regularized policy learning is an alternative approach to (OPT) (section 6.2). We provide the regularized LSPI baseline the same set of λ found by our algorithm for one-shot regularized learning (Figures 6.2 (left & middle)). While regularized LSPI obtains good performance for the main objective, it does not achieve acceptable constraint satisfaction. By default, regularized policy learning requires parameter tuning heuristics. In principle, one can perform a grid-search over a range of parameters to find the right combination — we include such an example for both regularized LSPI and FQI in Appendix D.7. However, since our objective and constraints are in conflict, main objective and constraint satisfaction of policies returned by one-shot regularized learning are sensitive to step changes in λ. In constrast, our approach is systematic, and is able to avoid the curse-of-dimensionality of brute-force search that comes with multiple constraints.

In practice, one may wish to deterministically extract a single policy from the returned mixture for execution. A de-randomized policy can be obtained

naturally from our algorithm by selecting the best policy from the existing FQE's estimates of individual `Best-response` policies.

Off-Policy Evaluation

The off-policy evaluation by FQE is critical for updating policies in our algorithm, and is ultimately responsible for certifying constraint satisfaction. While other OPE methods can also be used in place of FQE, we find that the estimates from popular methods are not sufficiently accurate in a high-dimensional setting. Complementing the more extensive ablations study in Chapter 3, we select an individual policy and compare FQE against PDIS [164], DR [98] and WDR [202] with respect to the constraint cost evaluation. To compare both accuracy and data-efficiency, for each domain we randomly sample different subsets of dataset D (from 10% to 100% transitions, 30 trials each). Figure 6.1 (right) and 6.2 (right) illustrate the difference in quality. In the FrozenLake domain, FQE performs competitively with the top baseline method (DR and WDR), converging to the true value estimate as the data subsample grows close to 100%. In the high-dimensional car domain, FQE signficantly outperforms other methods.

6.6 Other Related Work

Constrained MDP (CMDP). The CMDP is a well-studied problem [9]. Among the most important techniques for solving CMDP are the Lagrangian approach and solving the dual LP program via occupation measure. However, these approaches only work when the MDP is completely specified, and the state dimension is small such that solving via an LP is tractable. More recently, the constrained policy optimization approach (CPO) by [3] learns a policy when the model is not initially known. The focus of CPO is on online safe exploration, and thus is not directly comparable to our setting.

Multi-objective Reinforcement Learning. Another related area is multi-objective reinforcement learning (MORL)[212, 174]. Generally, research in MORL has largely focused on approximating the Pareto frontier that trades-off competing objectives [212, 174]. The underlying approach to MORL frequently relies on linear or non-linear scalarization of rewards to heuristically turns the problem into a standard RL problem. Our proposed approach represents another systematic paradigm to solve MORL, whether in batch or online settings.

6.7 Discussion

We have presented a systematic approach for batch policy learning under multiple constraints. Our problem formulation can accommodate general definition of constraints, as partly illustrated by our experiments. We provide guarantees for our algorithm for both the main objective and constraint satisfaction. Our strong empirical results show a promise of making constrained batch policy learning applicable for real-world domains, where behavior data is abundant.

Our implementation complies with the steps laid out in Algorithm 5. In very large scale or high-dimensional problems, one could consider a noisy update version for both policy learning and evaluation. We leave the theorerical and practical exploration of this extension to future work.

Chapter 7

LTL-BASED GUARANTEES IN DISCRETE DOMAINS

We study the problem of policy optimization (PO) with linear temporal logic (LTL) constraints. The language of LTL allows flexible description of tasks that may be unnatural to encode as a scalar cost function. We consider LTL-constrained PO as a systematic framework, decoupling task specification from policy selection, and as an alternative to the standard of cost shaping. With access to a generative model, we develop a model-based approach that enjoys a sample complexity analysis for guaranteeing both task satisfaction and cost optimality (through a reduction to a reachability problem). Empirically, our algorithm can achieve strong performance even in low-sample regimes.

In this chapter, we propose a novel policy optimization framework for RL under LTL constraints. Our approach relies on two assumptions that are significantly less restrictive than those in prior work and circumvent the negative results on RL-modulo-LTL: the availability of a generative model of the environment and a lower bound on the transition probabilities in the underlying MDP. Under these assumptions, we derive a learning algorithm based on a reduction to a reachability problem. The reduction in our method can be instantiated with several planning procedures that handle unknown dynamics [21, 166]. We show that our algorithm offers strong constraint satisfaction guarantees and give a rigorous sample complexity analysis of the algorithm.

In summary, the contributions are:

1) We provide a novel approach to LTL-constrained RL that requires significantly fewer assumptions, and offers stronger guarantees, than previous work.

2) We develop several new theoretical tools for our analysis. These may be of independent interest.

3) We empirically validate using both infinite- and indefinite-horizon problems, and with composite specifications such as collecting items while avoiding

enemies. We find that our method enjoys strong performance, often requiring many fewer samples than our worst-case guarantees.

7.1 Motivating Examples

Figure 7.1: *(Left) Infinite Loop. The robot must perpetually walk between the coffee room and office. Without proper state-space augmentation, a markovian cost function cannot capture this task. (Right) Safe Delivery. A packet must be delivered without being interfered. Policy 2 should be chosen. One would need to penalize receiving the packet significantly over having it stolen: $R > S$.*

We examine two examples where standard cost engineering cannot capture the task (Figure 7.1). We consider the undiscounted setting here. See [131, 2] for difficult examples for the discounted setting.

Example 1 (Infinite Loop). A robot is given the task of perpetually walking between the coffee room and the office (Figure 7.1 (Left)). To achieve this behavior, both the policy and cost-function must be history-dependent. These can be made Markovian through proper state-space augmentation and has been studied in hierarchical reinforcement learning or learning with options [121, 189]. Options engineering is laborious and requires expertise. Nevertheless, without the appropriate augmentation, any cost-optimal policy of a Markovian cost function will fail at the task. We will see in Section 7.2 that any LTL expression comes with automatic state-space augmentation, requiring no expert input.

Example 2 (Safe Delivery). The goal is to maximize the probability of safely sending a packet from one computer to another (Figure 7.1 (Right)). Policy 1 leads to a hacker sniffing packets but passing them through, and is unsafe. Policy 2 leads to a hacker stealing packets with probability $p > 0$, and is safe with probability $1 - p$, and is the policy that satisfies the task. For cost engineering, let R and S be the recurring costs of the received and stolen states. For the two policies, the avg. costs are $g_1 = R$ and $g_2 = pS + (1 - p)R$. Strangely, we must set $R > S$ in order for $g_2 < g_1$. Fortunately, optimizing any cost function constrained to satisfying the LTL specification does not suffer

from this counter intuitive behavior as only policy 2 has any chance of satisfying the LTL expression.

7.2 Background and Problem Formulation

For an introduction to the background for LTL, see Section 2.3. From there, recall a few key definitions.

We construct a Product MDP from syncronizing a Labelled MDP $\mathcal{M}$ with a specification-specific automaton (LDBA) $\mathcal{B}$:

Definition 7.2.1. (Product MDP) $\mathcal{X}_{\mathcal{M},\mathcal{B}} = (\mathcal{S}, \mathcal{A}, P, \mathcal{C}, d_0, L, \mathcal{S}^*)$ is an MDP with $\mathcal{S} = \mathcal{S}^{\mathcal{M}} \times \mathcal{S}^{\mathcal{B}}$, $\mathcal{A} = \mathcal{A}^{\mathcal{M}} \cup \mathcal{A}^{\mathcal{B}}$, $\mathcal{C}((m,b),a) = \mathcal{C}^{\mathcal{M}}(m,a)$ if $a \in A^{\mathcal{M}}(m)$ otherwise 0, $d_0 = \{(m,b) | m \in d_0^{\mathcal{M}}, b \in P^{\mathcal{B}}(s_0^{\mathcal{B}}, L^{\mathcal{M}}(m))\}$, $L((m,b)) = L^{\mathcal{M}}(m)$, $S^* = \{(\cdot, b) \in \mathcal{S} | b \in \mathcal{S}^{\mathcal{B}*}\}$ accepting states, and $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ taking the form:

$$P((m,b),a,(m',b')) = \begin{cases} P^{\mathcal{M}}(m,a,m') & a \in A^{\mathcal{M}}(m), b' \in P^{\mathcal{B}}(b, L(m')) \\ 1, & a \in A^{\mathcal{B}}(b), b' \in P^{\mathcal{B}}(b,a), m = m' \\ 0, & \text{otherwise} \end{cases}.$$

A run $\tau = (s_0, s_1, \ldots) = ((m_0,b_0),(m_1,b_1),\ldots)$ in $\mathcal{X}$ is accepting (accepted) if $(b_0, b_1, \ldots)$, the projection onto $\mathcal{B}$, is accepted. Equivalently, some $s \in \mathcal{S}^*$ in $\mathcal{X}$ is visited infinitely often. This leads us to the following definition of LTL satisfaction:

Definition 7.2.2 (Satisfaction, $\tau \models \varphi$)**.** A run τ in $\mathcal{X}$ *satisfies* φ, denoted $\tau \models \varphi$, if it is accepted.

Definition 7.2.3. (Satisfaction, $\pi \models \varphi$) A policy *satisfies* φ with probability $\mathbb{P}[\pi \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\mathbf{1}_{\tau \models \varphi}]$. Here, $\mathbf{1}_X$ is an indicator variable which is 1 when X is true, otherwise 0. T_π^P is the set of trajectories induced by π in $\mathcal{X}$ with transition function P.

Problem Formulation

Our goal is to find a policy that simultaneously satisfies a given LTL specification φ with highest probability (probability-optimal) and is also optimal w.r.t. the cost function of the MDP. We consider (stochastic) Markovian policies Π, and define the set of all probability-optimal policies as $\Pi_{\max} = \{\arg\max_{\pi' \in \Pi} \mathbb{P}[\pi' \models \varphi]\}$. We first define the gain g (average-cost) and transient

cost J:

$$\begin{aligned} g_\pi^P \equiv & \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathcal{C}(s_t, \pi(s_t)) \middle| \tau \models \varphi\right], \\ J_\pi^P \equiv & \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}\left[\sum_{t=0}^{\kappa_\tau} \mathcal{C}(s_t, \pi(s_t)) \middle| \tau \models \varphi\right], \end{aligned} \tag{7.1}$$

where κ_τ is the first (hitting) time the trajectory τ leaves the transient states induced by π. When P is clear from context, we abbreviate g_π^P and J_π^P by g_π and J_π, respectively.

Gain optimality for infinite horizon problems has a long history in RL [21, 166]. Complementary to gain optimality, we consider a hybrid objective including the transient cost. For any $\lambda \geq 0$, we define the optimal policy as the probability-optimal policy with minimum combined cost:

$$\begin{aligned} \pi_\lambda^* &\equiv \arg\min_{\pi \in \Pi_{\max}} \; J_\pi + \lambda g_\pi \qquad \text{(LTL-OPT)} \\ &= \arg\min_{\pi \in \Pi_{\max}} \; (J_\pi + \lambda g_\pi)\mathbb{P}\left[\pi \models \varphi\right] \quad (\equiv V_{\pi,\lambda}^P). \end{aligned}$$

In other words, probability-optimal policies are those that satisfy the entirety of the task, both desired and required behaviors, where $V_{\pi,\lambda}^P \equiv (J_\pi + \lambda g_\pi)\mathbb{P}\left[\pi \models \varphi\right]$ is the normalized value function[1], corresponding to a notion of energy or effort required, with λ representing the tradeoff between gain and transient cost. We will often omit the dependence of V on P and λ for brevity.

Example. Consider the Safe Delivery example (Figure 7.1 (Right)). For policy 1, $\mathbb{P}\left[1 \models \varphi\right] = 0$ and so $1 \notin \Pi_{\max}$. Let policy 2 be a cost 1 timestep before stolen or receipt, then $g_2 = R$ is the (conditional) gain, $J_2 = 1$ is the (conditional) transient costs, $\mathbb{P}\left[2 \models \varphi\right] = 1 - p$, and $V_2 = (1 + \lambda R)(1 - p)$.

Problem 1 (Planning with Generative Model/Simulator). Suppose access to a generative model of the true dynamics P from which we can sample transitions $s' \sim P(s, a)$ for any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$.[2] With probability $1 - \delta$, for some errors $\epsilon_\varphi, \epsilon_V > 0$, find a policy $\pi \in \Pi$ that simultaneously has the following properties: (i) $|\mathbb{P}\left[\pi \models \varphi\right] - \mathbb{P}\left[\pi^* \models \varphi\right]| < \epsilon_\varphi$ (ii) $|V_\pi - V_{\pi^*}| < \epsilon_V$.

[1] Normalized objectives are not unusual in RL, e.g. in discounted settings, multiplication by $(1 - \gamma)$

[2] The use of a generative model is increasingly common in RL [72, 126, 6, 198], and is applicable in many settings where such a generative model is readily available as a simulator (e.g., [52]).

7.3 Approach

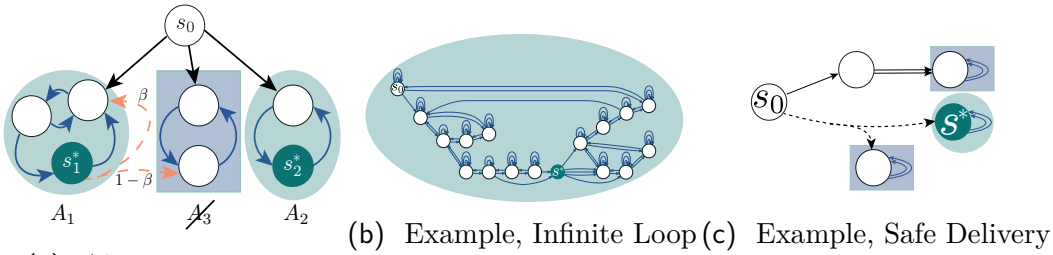
End Components & Accepting Maximal End Components

Our analysis relies on the idea of an end component: a recurrent, inescapable set of states when restricted to a certain action set. It is a sub-MDP of a larger MDP that is probabilistically closed.

Definition 7.3.1. (End Component, EC/MEC/AMEC [16]) Consider MDP $(\mathcal{S}, \mathcal{A}, P, \mathcal{C}, d_0, L, \mathcal{S}^*)$. An end component $(E, \mathcal{A}_E)$ is a set of states $E \subseteq \mathcal{S}$ and acceptable actions $\mathcal{A}_E(s) \subseteq \mathcal{A}(s)$ (where $s \in E$) such that $\forall (s,a) \in E \times \mathcal{A}_E$ then $Post(s,a) = \{s' | P(s,a,s') > 0\} \subseteq E$. Furthermore, $(E, \mathcal{A}_E)$ is strongly connected: any two states in E is reachable from one another by means of actions in $\mathcal{A}_E$. We say an end component $(E, \mathcal{A}_E)$ is *maximal* (MEC) if it is not contained within a larger end component $(E', \mathcal{A}_{E'})$, i.e., $\nexists (E', \mathcal{A}_{E'})$ EC where $E \subseteq E', \mathcal{A}_E(s) \subseteq \mathcal{A}_{E'}(s)$ for each $s \in A$. A MEC $(E, \mathcal{A}_E)$ is an *accepting* MEC (AMEC) if it contains an accepting state, $\exists s \in E$ s.t. $s \in \mathcal{S}^*$.

High-Level Intuition

The description of our approach, LTL Constrained Planning (LCP), in Section 7.3 is rather technical in order to yield theoretical guarantees. We thus first summarize the high-level intuitions.



(a) Abstract Diagram (b) Example, Infinite Loop (c) Example, Safe Delivery

Figure 7.2: Product MDP diagrams. (Left) The goal of LTL Constrained Policy Optimization can be reduced to a reachability problem. We want to reach A_1 or A_2 from s_0 and then follow the blue arrows with some distribution. A_3 with the blue arrows is a rejecting end component because it does not contain an accepting state s^*. For $\beta < 1$, the yellow action is not in the allowable action set of A_1 because there is a risk of entering A_3, strictly decreasing our probability of LTL satisfaction. (Center) Example for Infinite Loop, Figure 7.1 Left. (Right) Example for Safe Delivery, Figure 7.1 Right.

Solution Decomposition. Consider the accepting states s_1^*, s_2^* in Figure 7.2 (Left), which are the states we need to visit infinitely often to satisfy the

specification. First, let us identify the accepting maximal end components (AMECs) of s_1^* and s_2^*: the state sets A_1 and A_2 (resp.) and their corresponding action sets $\mathcal{A}_{A_1}$ and $\mathcal{A}_{A_2}$ (the blue arrows in A_1 and A_2). Note that these AMECs do not include the yellow action in Figure 7.2 (Left), which has a chance of leaving A_1 and getting stuck in A_3.

Our solution first runs a *transient* policy until reaching A_1 or A_2, and then switches to a (probability-optimal) *recurrent* policy that stays within A_1 or A_2 (resp.) while visiting s_1^* or s_2^* (resp.) infinitely often. A probability-optimal *recurrent* policy will select actions in $\mathcal{A}_{A_1}$ and $\mathcal{A}_{A_2}$ to visit s_1^*, s_2^* infinitely often (e.g., the uniform policies with the AMECs $(A_1, \mathcal{A}_{A_1})$ and $(A_2, \mathcal{A}_{A_2})$). Finding a *transient* policy from s_0 to A_1, A_2 can be viewed as a reachability problem, which we can solve via a Stochastic Shortest Path (SSP) problem and leverage recent literature [198, 110].

Cost Optimality. As stated in (LTL-OPT), the goal is to find a cost-optimal policy within the set of probability-optimal policies. For instance, the uniform policy over $\mathcal{A}_{A_1}$ and $\mathcal{A}_{A_2}$ (the blue arrows in Figure 7.2 (Left) is probability optimal, but may not be cost optimal. Similarly, the unconstrained cost-optimal policy may not be probability optimal. Consider just A_1 for the moment. Suppose the cost of the arrows between the white nodes is 4 while the other costs are 7. Then the uniform (probability-optimal) policy in A_1 over $\mathcal{A}_{A_1}$ has cost $\frac{1}{2}\left(\frac{4+4}{2}\right)+\frac{1}{2}\left(\frac{7+7+4}{3}\right) = 5$. The gain-optimal policy that deterministically selects the actions between the white nodes $\tilde{\pi}$ has cost $\left(\frac{4+4}{2}\right) = 4$, but is not probability optimal. If we perturb $\tilde{\pi}$ to make it even slightly stochastic (but still mostly deterministic, i.e η-greedy with $\eta \approx 0$), then it will be arbitrarily close to gain optimality and also recover probability optimality. This is a preferable probability-optimal policy over the uniform policy.

Overall Procedure. The high-level procedure is: (i) identify the AMECs (e.g. $(A_1, \mathcal{A}_{A_1}), (A_2, \mathcal{A}_2)$) by filtering out bad actions like the yellow arrow; (ii) find a cost-optimal (optimal gain cost) recurrent policy in each AMEC that visits some s^* infinitely often; (iii) instantiate an SSP problem that finds a cost-optimal (optimal transient cost) transient policy from s_0 to $A_1 \cup A_2$ and avoids A_3; (iv) return a policy that stitches together the policies from (*ii*) and (*iii*). See Section 7.3 for the algorithmic details. We show in Section 7.4 that this solution gives the optimal solution to LTL-OPT.

Additional Assumptions and Definitions

Perhaps surprisingly, when planning with a simulator (i.e., generative model), even infinite data is insufficient to verify an LTL formula without having a known lower-bound on the lowest nonzero probability of the transition function P [131]. Without this assumption, LTL constrained policy learning is not learnable [224]. We thus begin by assuming a known lower bound on entries in P.[3]

Assumption 7.1 (Lower Bound)**.** We assume we have access to a lower bound $\beta > 0$ on the lowest non-zero probability of the transition function P:

$$0 < \beta \leq \min_{s,a,s' \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}} \{P(s,a,s') | P(s,a,s') > 0\}. \tag{7.2}$$

We assume that all the costs are strictly positive, avoiding zero-cost (or negative-cost) cycles that trap a policy. Leveraging cost-perturbations and prior work [198] can remove the assumption.

Assumption 7.2 (Bounds on cost function)**.** The minimum cost $c_{\min} > 0$ is strictly positive.

Let $D = \{(s,a,s')\}$ be all the collected samples (s,a,s') while running the algorithm. At any point, $\widehat{P}(s,a,s') = \frac{|\{(s,a,s') \in D\}|}{|\{(s,a) \in D\}|}$ is the empirical frequency of visiting s' from (s,a). We introduce an event $\mathcal{E}$ and error $\psi(n)$ to quantify uncertainty on $\widehat{P}(s,a,s')$ based on current data: $n(s,a) = |\{(s,a) \in D\}|$. $\mathcal{E}$ is based on empirical Bernstein bounds [141], and holds w.p. $1-\delta$ (Lemma E.2.1).

Definition 7.3.2 (High Probability Event)**.** A high probability event $\mathcal{E}$:

$$\mathcal{E} = \{\forall s,a,s' \in S \times A \times S, \forall n(s,a) > 1 : |(P(s,a,s') - \widehat{P}(s,a,s'))| \leq \psi_{sas'}(n) \leq \psi(n)\},$$

where $\psi_{sas'}(n) \equiv \sqrt{2\hat{P}(s,a,s')(1-\hat{P}(s,a,s'))\xi(n)} + \frac{7}{3}\xi(n)$, $\psi(n) \equiv \sqrt{\frac{1}{2}\xi(n)} + \frac{7}{3}\xi(n)$, and $\xi(n) \equiv \log(\frac{4n^2|\mathcal{S}|^2|\mathcal{A}|}{\delta})/(n-1)$.

Remark 7.3.1. *For some $\rho > 0$, if we require $|P(s,a,s') - \widehat{P}(s,a,s')| \leq \rho$ then we need $n(s,a) = \psi^{-1}(\rho)$ samples for state-action pair (s,a). See Lemma E.2.2 for the quantity $\psi^{-1}(\rho)$.*

[3] Our assumptions are consistent with the minimal requirements studied by [131].

Definition 7.3.3 (Plausible Transition Function)**.** The set of plausible transition functions is given by

$$\mathcal{P} = \{\tilde{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S}) | \tag{7.3}$$

$$\begin{cases} \tilde{P}(s,a,s') = \widehat{P}(s,a,s'), & \widehat{P}(s,a,s') \in \{0,1\} \\ \tilde{P}(s,a,s') \in \widehat{P}(s,a,s') \pm \psi_{sas'} \cap [\beta, 1-\beta], & \text{otherwise} \end{cases} \} \tag{7.4}$$

Let $\mathcal{P}(s,a) \equiv \{P(s,a,\cdot) | P \in \mathcal{P}\}$ be the possible transition distributions for state-action pair (s,a). We denote $P_\pi(s,s') = \mathbb{E}_{a\sim\pi}[P(s,a,s')]$ as the Markov chain given dynamics P with policy π, and can be thought of as a $|\mathcal{S}| \times |\mathcal{S}|$ matrix $P_\pi = \{p_{ij}\}_{i,j=1}^{|\mathcal{S}|}$.

Main Algorithm: LTL Constrained Planning (LCP)

Algorithm 6 LTL Constrained Planning (LCP)

Require: Error $\epsilon_V > 0$, Error $\epsilon_\varphi > 0$, Tolerance $\delta > 0$, Lower bound $\beta > 0$ (see Assumption 7.1)

1: Globally, track $\widehat{P}(s,a,s') = \frac{|\{(s,a,s') \in D\}|}{|\{(s,a) \in D\}|}$ {Empirical estimate of P}
2: $((A_1, \mathcal{A}_{A_1}), \ldots, (A_m, \mathcal{A}_{A_k})) \leftarrow$ `FindAMEC`$((\mathcal{S}, \mathcal{A}, \widehat{P}))$
3: **for** $i = 1, \ldots, k$ **do**
4: Set $\pi_i, g_i \leftarrow$ `PlanRecurrent`$((A_i, \mathcal{A}_{A_i}), \frac{\epsilon_V}{7\lambda})$ {Plan gain-optimal policy π_i for A_i}
5: Set $\pi_0 \leftarrow$ `PlanTransient`$(((A_1, g_1), \ldots, (A_k, g_k)), \frac{2\epsilon_V}{9})$ {Plan shortest paths policy π_0 to $\cup_{i=1}^k A_i$}
6: **return** $\pi = \cup_{i=0}^k \pi_i$

Our approach, LTL Constrained Planning (LCP), has three components, as shown in Algorithm 6 and described below. Recall from Problem 1 that the policy optimization problem LTL-OPT is instantiated over a product MDP (Def. 7.2.1), and that we are given a generative model of the true dynamics P from which we can sample transitions $s' \sim P(s,a)$ for any state/action pair.

Finding AMECs (`FindAMEC`). Sampling each state-action pair $\varphi_{\texttt{FindAMEC}} = O(\frac{1}{\beta})$ times (see Prop. E.2.4), by Assumption 7.1, verifies the support of P. We can compute all of the MECs using Algorithm 47 from [16]. Among these MECs, we keep the AMECs, which amounts to checking if the MEC $(A_i, \mathcal{A}_{A_i})$ contains an accepting state $s^* \in \mathcal{S}^*$ from the given product MDP.

PlanRecurrent (`PR`). To plan in each AMEC $(A, \mathcal{A}_A)$ (i.e., find the optimal recurrent policy), we use Alg. 7 with (extended) relative value iteration (`VI`, Alg. 17 in appendix) using the optimistic Bellman operator $\mathcal{L}^\alpha_{\texttt{PR}}$ (see Table 7.1, we discuss α in next paragraph). Let π_v denote the greedy policy w.r.t. the fixed point $v = \mathcal{L}^\alpha_{\texttt{PR}} v$ (v is the optimistic value estimate). Using the η-greedy policy, $\pi \equiv (1-\eta)\pi_v + \eta\texttt{Unif}(\mathcal{A}_A)$ (Alg. 7, Line 7), together with P_π, makes A

Algorithm 7 `PlanRecurrent` (PR)

Require: AMEC $(A, \mathcal{A}_A)$, error $\epsilon_{\texttt{PR}} > 0$

1: Set $\rho \leftarrow 2\psi(\varphi_{\texttt{FindAMEC}}(\beta))$ $\{\rho \sim \|P - \tilde{P}\|_1^{-1}\}$
2: **repeat**
3: Set $\rho \leftarrow \frac{\rho}{2}$
4: Sample $\psi^{-1}(\rho)$ times $\forall (s,a) \in A \times \mathcal{A}_A$
5: $v', v, \tilde{P} \leftarrow \texttt{VI}(\mathcal{L}^{\alpha}_{\texttt{PR}}, d_{\texttt{PR}}, \epsilon^{\mathcal{L}}_{\texttt{PR}})$ $\{v' = \mathcal{L}^{\alpha}_{\texttt{PR}} v\}$
6: **until** $\rho > \frac{\epsilon_{\texttt{PR}}(1-\Delta(\tilde{P}))}{3|A|c_{\max}}$ $\{\|P - \tilde{P}\|_1 \ small\}$
7: Set policy $\pi \leftarrow \eta$-greedy policy w.r.t. v'
8: Set gain $g_\pi \leftarrow \frac{1}{2}(\max(v' - v) + \min(v' - v))$
9: **return** π, g_π

recurrent: $s^* \in A$ is visited infinitely often and $\mathbb{P}[\pi \models \varphi | s_0 \in A] = 1$. Since η can be arbitrarily small (Lemma E.2.7), then $g_\pi \approx g_{\pi_v}$ and π is both cost and probability optimal. As intuited in Section 7.3, π has full support over $\mathcal{A}_A$ but is nearly deterministic.[4]

`VI` in Line 5 of Alg. 7 is an iterative procedure (Alg. 17 in appendix), and terminates via $d_{\texttt{PR}} < \epsilon^{\mathcal{L}}_{\texttt{PR}}$ (Table 7.1). Convergence of extended `VI` is guaranteed [166, 94, 66], so long as the dynamics, $\tilde{P} = \arg\min_{p \in \mathcal{P}(s,a)} p^T v$, achieving the inner minimization of $\mathcal{L}^{\alpha}_{\texttt{PR}}$ are aperiodic — hence the aperiodicity transform $\alpha \in (0,1)$ in $\mathcal{L}^{\alpha}_{\texttt{PR}}$ [166]. Computing $\tilde{P}$ can be done efficiently [94] (Alg. 18 in appendix). For stability, we shift each entry of v_n by the value of the first entry $v_n(0)$ [21].

Alg. 7 returns the average gain cost g_π of policy π when we have enough samples for each state-action pair in $(A, \mathcal{A}_A)$ to verify that $n > \psi^{-1}\left(\frac{\epsilon_{\texttt{PR}}(1-\Delta(\tilde{P}_\pi))}{3|A|c_{\max}}\right)$ where $\Delta(\tilde{P}_\pi) = \frac{1}{2}\max_{ij}\sum_k |\tilde{p}_{ik} - \tilde{p}_{jk}|$. Here, $\Delta(\tilde{P}_\pi)$ is an easily computable measure on the ergodicity of the Markov chain $\tilde{P}_\pi$ [41]. We track $\psi(n)$ (recall Def. 7.3.2) via a variable ρ and sample $\psi^{-1}(\rho) \approx \frac{1}{\rho^2}$ (see Lemma E.2.2) samples from each state-action pair in $(A, \mathcal{A}_A)$ (Alg. 7, Line 4). We halve ρ each iteration (Alg. 7, Line 3) and convergence is guaranteed because ρ will never fall below some unknown constant $\frac{\epsilon_{\texttt{PR}}(1-\bar{\Delta}_A)}{6|A|c_{\max}}$ (see Lemma E.2.8); the halving trick is required because $\bar{\Delta}_A$ is unknown a priori.

Proposition 7.3.2 (PR Convergence & Correctness, Informal)**.** *Let π_A be the gain-optimal policy in AMEC $(A, \mathcal{A})$. Algorithm 7 terminates after at most*

[4] Typically, RL settings admit a fully deterministic optimal policy, but for LTL constrained policy optimization the optimal policy may not be deterministic (although can be very nearly so). See Cost Optimality in Section 7.3.

Table 7.1: Subroutine Operators and Parameters for Value Iteration.

Op/Param	Description
$\mathcal{L}^{\alpha}_{\texttt{PR}} v(s)$	$\min_{a\in\mathcal{A}_A(s)}\left(\mathcal{C}(s,a)+\alpha\min_{p\in\mathcal{P}(s,a)} p^T v\right)+(1-\alpha)v(s) \quad \forall s\in A$
$d_{\texttt{PR}}(v_{n+1},v_n)<\epsilon^{\mathcal{L}}_{\texttt{PR}}$	$\max_{s\in A}(v_{n+1}(s)-v_n(s))-\min_{s\in A}(v_{n+1}(s)-v_n(s))<\frac{2\epsilon_{\texttt{PR}}}{3}$
$\mathcal{L}_{\texttt{PT}} v(s)$	$\begin{cases}\min\left\{\min_{a\in\mathcal{A}_A(s)}\left(\mathcal{C}(s,a)+\min_{p\in\mathcal{P}(s,a)} p^T v\right), \bar{V}/\epsilon_\varphi\right\}, & s\in\mathcal{S}\setminus\cup_{i=1}^k A_i\\ \lambda g_i, & s\in A_i\end{cases}$
$d_{\texttt{PT}}(v_{n+1},v_n)<\epsilon^{\mathcal{L}}_{\texttt{PT}}$	$\|v_{n+1}-v_n\|_1<c_{\min}\epsilon_{\texttt{PT}}\epsilon_\varphi/(4\bar{V})$

Algorithm 8 `PlanTransient` (`PT`)

Require: States & gains: $\{(A_i,g_i)\}_{i=1}^k$, err. $\epsilon_{\texttt{PT}}>0$
1: Set $V_T(s)=\lambda g_i$ for $s\in A_i$ {*Terminal costs*}
2: Sample $\varphi_{\texttt{PT}}$ times $\forall(s,a)\in(\mathcal{S}\setminus\cup A_i)\times\mathcal{A}$
3: $v', v, \tilde{P}\leftarrow \texttt{VI}(\mathcal{L}_{\texttt{PT}}, d_{\texttt{PT}}, \epsilon^{\mathcal{L}}_{\texttt{PT}}, V_T)$ {$v'=\mathcal{L}_{\texttt{PT}} v$}
4: Set $\pi\leftarrow$greedy policy w.r.t v'
5: **return** π

$\log_2\left(\frac{6|A|c_{\max}}{\epsilon_{PR}(1-\bar{\Delta}_A)}\right)$ *repeats, and collects at most* $n=\tilde{\mathcal{O}}(\frac{|A|^2c^2_{\max}}{\epsilon^2_{PR}(1-\bar{\Delta}_A)^2})$ *samples for each* $(s,a)\in(A,\mathcal{A}_A)$. *The* η*-greedy policy* π *w.r.t.* v' *(Alg. 7, Line 5) is gain optimal and probability optimal:* $|g_\pi-g_{\pi_A}|<\epsilon_{PR}$, $\mathbb{P}\left[\pi\models\varphi|s_0\in A\right]=1$.

PlanTransient (`PT`). This is the stochastic shortest path (SSP) reduction step that finds a policy from the initial state s_0 to the AMECs (Alg. 8). The main algorithmic tool used by `PlanTransient` is similar to that of `PlanRecurrent`: it also uses extended value iteration (`VI`, Alg. 17 in appendix) but with a different optimistic Bellman operator $\mathcal{L}_{\texttt{PT}}$ (Table 7.1), and then returns a (fully deterministic) greedy policy w.r.t. the resulting optimistic value v (Alg. 8, Line 4). $\mathcal{L}_{\texttt{PT}}$ is used to calculate the highest probability, lowest cost path to the AMECs (Alg. 8, Line 3).

Since rejecting end components might exist (see A_3 from Figure 7.2 (Left)), a trajectory may end up stuck and accumulate cost indefinitely, and so we must bound $\|v\|_\infty<\bar{V}/\epsilon_\varphi$ to prevent blow up. In Prop. E.2.13, we show how to select $\bar{V}$ such that π will reach the target states (in this case, the AMECs), first with high prob and then with lowest cost. The existence of such a bound on $\|v\|_\infty$ was shown to exist, without construction, in [110]. In practice, choosing a large $\bar{V}$ is enough.

The terminal costs V_T (Alg. 8, Line 1) together with Bellman equation $\mathcal{L}_{\texttt{PT}}$ has value function $\tilde{V}_\pi\approx p(J_\pi+\frac{1}{p}\sum_{i=1}^k p_i g_{\pi_i})+(1-p)\bar{V}/\epsilon_\varphi\approx V_\pi$, relating to V_π (LTL-OPT); see Section E.1. Here, $p_i=\mathbb{P}\left[\pi \text{ reaches } A_i\right]\equiv\mathbb{E}_{\tau\sim \mathrm{T}^P_\pi}[1_{\exists s\in\tau \text{ s.t } s\in A_i}]$

and $p = \sum_{i=1}^{k} p_i$. `VI` converges when $d_{\texttt{PT}} < \epsilon_{\texttt{PT}}$ (see Table 7.1). Convergence of extended VI for SSP is guaranteed [198, 110]. The number of samples required for each state-action pair $(s,a) \in (\mathcal{S} \setminus \cup A_i) \times \mathcal{A}$ is $\varphi_{\texttt{PT}} = \psi^{-1}\left(\frac{c_{\min}\epsilon_{\texttt{PT}}\epsilon_{\varphi}^2}{14|\mathcal{S}\setminus\cup_{i=1}^{k}A_i|\bar{V}^2}\right)$.

Proposition 7.3.3 (`PlanTransient` Convergence & Correctness, Informal)**.** *Denote the cost- and prob-optimal policy as π'. After collecting at most $n = \tilde{\mathcal{O}}(\frac{|\mathcal{S}\setminus\cup_{i=1}^{k}A_i|^2\bar{V}^4}{c_{\min}^2\epsilon_{PT}^2\epsilon_{\varphi}^4})$ samples for each $(s,a) \in (\mathcal{S} \setminus \cup_{i=1}^{k} A_i) \times \mathcal{A}$, the greedy policy π w.r.t. v' (Alg. 8, Line 3) is both cost and probability optimal:*

$$\|\tilde{V}_{\pi} - \tilde{V}_{\pi'}\| < \epsilon_{PT}, \quad |\mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^{k} A_i\right] - \mathbb{P}\left[\pi' \text{ reaches } \cup_{i=1}^{k} A_i\right]| \leq \epsilon_{\varphi}.$$

7.4 End-To-End Guarantees

The number of samples necessary to guarantee an $(\epsilon_V, \epsilon_\varphi, \delta)$-PAC approximation to the cost-optimal and probability-optimal policy relies factors: β (lower bound on the min. non-zero transition probability of P), $\{c_{\min}, c_{\max}\}$ (bounds on the cost function $\mathcal{C}$), $\bar{\Delta}_{A_i}$ (worst-case coefficient of ergodicity for EC $(A_i, \mathcal{A}_{A_i})$), $\bar{V}$ (upper bound on the value function), and λ (tradeoff factor).

Theorem 7.4.1 (Sample Complexity)**.** *Under the event $\mathcal{E}$, Assumption 7.1 and 7.2, after*

$$n = \tilde{\mathcal{O}}\left(\frac{1}{\beta} + \frac{1}{\epsilon_V^2}\left(\frac{|\mathcal{S}|^2\bar{V}^4}{c_{\min}^2\epsilon_{\varphi}^4} + \lambda^2 \sum_{i=1}^{k} \frac{|A_i|^2 c_{\max}^2}{(1-\bar{\Delta}_{A_i})^2}\right)\right)$$

samples[5] *are collected from each state-action pair, the policy π returned by Algorithm 6 is, with probability $1-\delta$, simultaneously ϵ_V-cost optimal and ϵ_φ-probability optimal, satisfying:*

$$(i)\ \ |\mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^* \models \varphi]| \leq \epsilon_{\varphi} \quad (ii)\ \ \|V_{\pi} - V_{\pi^*}\|_{\infty} < \epsilon_V. \tag{7.5}$$

With a sufficiently large λ (which may not be verifiable in practice), π is also gain optimal.

Corollary 7.4.2 (Gain (Average Cost) Optimality)**.** *There exists $\lambda^* > 0$ s.t. for $\lambda > \lambda^*$, the policy π returned by Alg. 6 satisfies* (7.5)*, $g_\pi = \arg\min_{\pi' \in \Pi_{\max}} g_{\pi'}$, and is probability and gain optimal.*

The high-level structure of our analysis follows the algorithm structure in Section 7.3, via composing the constituent guarantees. To complete the analysis, we

[5]The lower bound relating to β from [131] is $\Omega(\frac{\log(2\delta)}{\log(1-\beta)})$ whereas ours is $\tilde{O}(\frac{1}{\beta})$. We conjecture that $\tilde{\Omega}(\frac{1}{\beta})$ samples are required. See Appendix Section E.3.

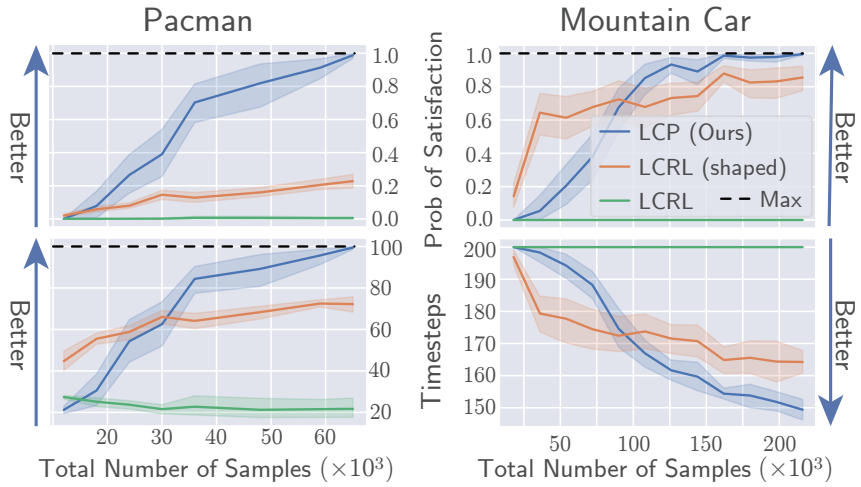


Figure 7.3: *Results. (Left Column) Pacman. φ is to eventually collect food and always avoid the ghost. We let the system run for a maximum of 100 timesteps. (Right Column) Discretized Mountain Car (MC). φ is to eventually reach the flag.*

develop some technical tools which may be of independent interest, including a gain simulation Lemma E.2.8 and an η-greedy optimality Lemma E.2.7. For ease of exposition, we also ignore paths between AMECs (see Appendix E.4).

7.5 Empirical Analysis

We perform experiments in two domains: (1) Pacman domain where an agent finds food and indefinitely avoids a ghost and (2) discretized version of mountain car (MC) [29], where the agent must reach the flag. Our goal is to understand whether: (i) our LCP approach (Alg.6) produces competitive polices; (ii) LCP can work in continuous state spaces through discretization; (iii) LCP can enjoy efficient sample complexity in practice. For a baseline, we use Logically Constrained RL (LCRL, [83]), which is a Q-learning approach to LTL-constrained PO in unknown MDPs. We also do heavy cost shaping to LCRL as another baseline. See App E.5 for more details and results.

Results

spec in Figure 7.3 (Left) approaches 1 much faster than the two baselines. The returned policy collects the food quickly and then stays close, but avoids, the ghost. Any policy that avoids the ghost is equally good, as we have not incentivized it to stay far away. LCRL redefines cost as 1 if the LTL is solved and 0 otherwise, which is too sparse and learning suffers. Indeed, shaped LCRL performs better than straight LCRL.

Performance in continuous state space? Similarly, the probability of satisfying the LTL spec in Figure 7.3 (Right) goes up to 1. However, here the LCRL (shaped) baseline performs relatively well as it is being given "breadcrumbs" for how to solve the task. Our algorithm performs well without needing any cost shaping. Standard LCRL fails to learn. This experiment demonstrates that LCP can be used even in discretized continuous settings.

Sample Complexity? Our theory is quite conservative w.r.t. empirical performance. In Pacman (Figure 7.3, Left), Thm. 7.4.1 suggests ≈ 350 samples per (s, a) pair just to calculate the AMECs. Empirically, LCP finds a good policy after 11 samples per (s, a) pair ($\sim 66k/6k$ samples/pair).

Other Considerations. One of the strengths and potential drawbacks of LTL is its specificity. If a φ, for a truly infinite horizon problem, is to "eventually" do something, then accomplishing the task quickly is not required. As a finite horizon problem, in MC (Fig. 7.3, Right) SSP finds the fastest path to the goal. In contrast, since any stochastic policy with full support will "eventually" work, the policy returned by LCP for Fig 7.1 (Left) (Fig. 7.2 Center, & App Fig. E.4) may take exponential time to complete a single loop. Two straightforward ways to address this issue are (a) including explicit time constraints in φ, and (b) cost shaping to prefer policies reaching some s^* quickly and repeatedly. Unlike standard cost-shaping, φ satisfaction is still guaranteed since the cost is decoupled from φ.

7.6 Related Work

Constrained Policy Optimization. One attempt at simplifying cost functions is to split the desired behaviors from the required behaviors. The desired behaviors remain as part of the cost function while the required behaviors are treated as constraints. Recent interest in constrained policy optimization within the RL community has been related to the constrained Markov Decision Process (CMDP) framework [10, 123, 4, 142]. This framework enables clean methods and guarantees, but enforces expected constraint violations rather than absolute constraint violations. Setting and interpreting constraint thresholds can be very challenging, and inappropriate in safety-critical problems [121].

LTL + RL. Recently, LTL-constrained policy optimization has been developed as an alternative to CMDPs [131]. Unlike CMPDs, the entire task is encoded

into an LTL expression and is treated as the constraint. Q-learning variants when dynamics are unknown and Linear Programming methods when dynamics are known are common solution concepts [176, 83, 27, 33, 50]. The Q-learning approaches rely on proper, unknowable tuning of discount factor for their guarantees. Theoretically oriented works include [67, 221]. While providing PAC-style guarantees, the assumptions made in these works rely on unknowable policy-environment interaction properties. We make no such assumptions here.

Another solution technique is employing reward machines [205, 34, 210] or high-level specifications that can be translated into reward machines [99]. These works are generally empirical and handle finite or repeated finite problems (episodic problems at test time); they can only handle a smaller set of LTL expressions, specifically regular expressions. Our work handles ω-regular expressions, subsuming regular expressions and requires a nontrivial leap, algorithmically and theoretically, to access the broader set of allowable expressions. Many problems are ω-regular problems, but not regular, such as liveness (something good will happen eventually) and safety (nothing bad will happen forever). The works that attempt to handle full LTL expressibility redefine reward as 1 if the LTL is solved and 0 otherwise; the cost function of the MDP is entirely ignored.

Verification and Planning. As an alternative to our approach, one might consider LTL satisfaction verification and extend it to an optimization technique by checking every policy (which will naively take an exponential amount of samples to verify a single policy [28, 13]). Many verification approaches exist [114, 16, 8, 226, 117, 90] and among the ones that do not assume known dynamics, the verification guarantees rely on quantities as difficult to calculate as the original verification problem itself [13].

7.7 Discussion

We have presented a novel algorithm, LCP, for policy optimization under LTL constraints in an unknown environment. We formally guarantee that the policy returned by LCP simultaneously has minimal cost with respect to the MDP cost function and maximal probability of LTL satisfaction. Our experiments verify that our policies are competitive and our sample estimates conservative.

The assumptions we make are strong, but to the best of our knowledge, are the most relaxed amongst tractable model-based algorithms proposed for this

space. Model-free algorithms (Q-learning) have less stringent assumptions but do not come with the kind of guarantees that our work has and largely ignore the cost function, solving only part of the problem. An interesting future direction would be to extend our work to continuous state and action spaces and settings with function approximation.

Chapter 8

LTL-BASED GUARANTEES IN CONTINUOUS DOMAINS

Linear temporal logic (LTL) offers a simplified way of specifying tasks for policy optimization that may otherwise be difficult to describe with scalar reward functions. However, the standard RL framework can be too myopic to find maximally LTL satisfying policies. This chapter makes two contributions. First, we develop a new value-function based proxy, using a technique we call eventual discounting, under which one can find policies that satisfy the LTL specification with highest achievable probability. Second, we develop a new experience replay method for generating off-policy data from on-policy rollouts via counterfactual reasoning on different ways of satisfying the LTL specification. Our experiments, conducted in both discrete and continuous state-action spaces, confirm the effectiveness of our counterfactual experience replay approach.

We focus on model-free policy learning of an LTL specified objective from online interaction with the environment. We make two technical contributions. First, we reformulate the RL problem with a modified value-function proxy using a technique we call *eventual discounting*. The key idea is to account for the fact that optimally satisfying the LTL specification may not depend on the length of time it takes to satisfy it (e.g., "eventually always reach the goal"). We prove in Section 8.2 that the optimal policy under eventual discounting maximizes the probability of satisfying the LTL specification.

Second, we develop an experience replay method to address the reward sparsity issue. Namely, any LTL formula can be converted to a fully known specialized finite state automaton from which we can generate multiple counterfactual trajectories from a single on-policy trajectory. We call this method *LTL-guided counterfactual experience replay*. We empirically validate the performance gains of our counterfactual experience replay approach using both finite state/action spaces as well as continuous state/action spaces using both Q-learning and Policy Gradient approaches.

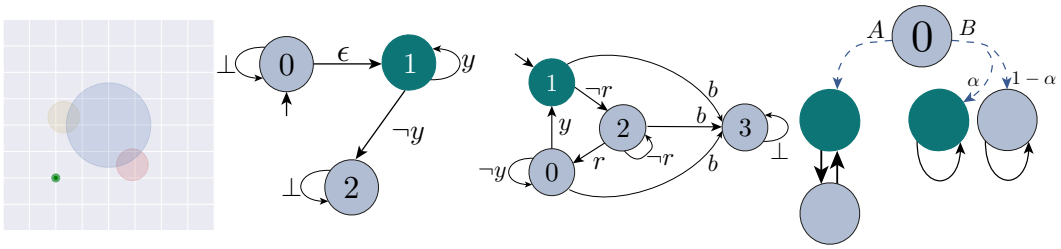


Figure 8.1: *Examples.* ***First:*** Illustration of the Flatworld environment. The agent is a green dot and there are 3 zones: yellow, blue and red. ***Second:*** LDBA $\mathcal{B}$ for "FGy". $\mathcal{S}^{\mathcal{B}^*} = \{1\}$, denoted by green circle. The initial state is $b_{-1} = 0$. ***Third:*** LDBA $\mathcal{B}$ for "$GF(y \,\&\, XFr) \,\&\, G\neg b$". $\mathcal{S}^{\mathcal{B}^*} = \{1\}$, denoted by green circle. The initial state is $b_{-1} = 1$. ***Fourth:*** For this example, an agent starting in state 0 and solving $\arg\max_{\pi\in\Pi} \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}[\sum_{i=0}^{\infty} \gamma^i \mathbf{1}_{\{b_i \in \mathcal{S}^{\mathcal{B}^*}\}}]$ where $\mathcal{S}^{\mathcal{B}^*}$ is illustrated as the green circles would choose to take action B with probability 1 if $\alpha \in (1/2, 1)$ for any $\gamma \in [0, 1]$. Such a policy has $\mathbb{P}[\pi \models \varphi] = \alpha < 1$. However the probability optimal policy deterministically takes action A, with $\mathbb{P}[\pi^* \models \varphi] = 1$. This illustrates catastrophic myopic behavior.

8.1 Problem Formulation

For an introduction to the background for LTL, see Section 2.3. From there, recall a few key definitions. Let $\mathcal{Z} = \mathcal{S}^{\mathcal{M}} \times \mathcal{S}^{\mathcal{B}}$. Let $\Pi : \mathcal{Z} \times \mathcal{A} \to \Delta([0, 1])$ be a (stochastic) policy class over the product space of the MDP and the LDBA (defined below), where $\mathcal{A}((s, b)) = \mathcal{A}^{\mathcal{M}}(s) \cup \mathcal{A}^{\mathcal{B}}(b)$, to account for jump transitions in $\mathcal{B}$.

Synchronizing the MDP with the LDBA. For any $(s, b) \in \mathcal{Z}$, a policy $\pi \in \Pi$ is able to select an action in $\mathcal{A}^{\mathcal{M}}(s)$ or an action in $\mathcal{A}^{\mathcal{B}}(b)$, if available. We can therefore generate a **trajectory** as the sequence $\tau = (s_0, b_0, a_0, s_1, b_0, a_1, \ldots)$ under a new probabilistic transition relation given by

$$P(s', b'|s, b, a) \quad = \quad \begin{cases} P^{\mathcal{M}}(s, a, s') & a \in A^{\mathcal{M}}(s), b' \in P^{\mathcal{B}}(b, L(s')) \\ 1, & a \in A^{\mathcal{B}}(b), b' \in P^{\mathcal{B}}(b, a), s = s' \\ 0, & \text{otherwise} \end{cases}. \quad (8.1)$$

Let the LDBA projection of τ be the subsequence $\tau_{\mathcal{B}} = (b_0, b_1, \ldots)$. Elements of $\tau_{\mathcal{B}}$ can be thought of as tracking an agent's LTL specification satisfaction:

Definition 8.1.1 (Run Satisfaction, $\tau \models \varphi$)**.** We say a trajectory satisfies φ if $\mathcal{B}$ accepts $\tau_{\mathcal{B}}$, which happens if $\exists b \in \tau_{\mathcal{B}}$ infinitely often with $b \in \mathcal{S}^{\mathcal{B}^*}$.

Let $\mathrm{T}_\pi^P = \mathbb{E}_{z\sim d_0^{\mathcal{M}} \times \{b_{-1}\}}[\mathrm{T}_\pi^P(z)]$ be the distribution over all possible trajectories starting from any initial state $z \in d_0^{\mathcal{M}} \times \{b_{-1}\}$ where $\mathrm{T}_\pi^P(z)$ is the (conditional)

distribution over all possible trajectories starting from $z \in \mathcal{Z}$ generated by π under relation P (given in (8.1)). The probability of LTL satisfaction results from counting how many of the trajectories satisfy the LTL specification:

Definition 8.1.2 (State Satisfaction, $z \models \varphi$)**.** $\mathbb{P}_\pi[z \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P(z)}[\mathbf{1}_{\{\tau \models \varphi\}}] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\mathbf{1}_{\{\tau \models \varphi\}} | z_0 = z]$.

Definition 8.1.3 (Policy Satisfaction, $\pi \models \varphi$)**.** $\mathbb{P}[\pi \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\mathbf{1}_{\{\tau \models \varphi\}}]$ where $\mathbf{1}_X$ is the indicator for X.

Ideally we would like to find a policy with highest probability of LTL specification satisfaction: one that generates the most number of LTL-satisfying runs. Formally,

$$\pi^* \in \arg\max_{\pi \in \Pi} \mathbb{P}[\pi \models \varphi]. \tag{8.2}$$

We note that Eq (8.2) is the standard starting point for formulating policy optimization for LTL satisfaction [224, 27, 32, 83, 84, 215].

8.2 RL-Friendly Form: Eventual Discounting

Unfortunately, the maximization problem in Eq (8.2) is not easily optimized since we dont have a direct signal on $\mathbb{P}[\pi \models \varphi]$. Without any additional assumptions (such as structured knowledge of the MDP), any finite subsequence can only give evidence on whether $\tau \models \varphi$ but not a concrete proof.

Eventual Discounting. To address the above issue, we develop a modified value-function based surrogate as follows. Given a trajectory $\tau = (s_0, b_0, a_0, \ldots)$, we keep track of how often $b_i \in \mathcal{S}^{\mathcal{B}^*}$ and incentivize an agent to visit $\mathcal{S}^{\mathcal{B}^*}$ as many times as possible. In particular, under eventual discounting, the value function will give the agent a reward of 1 when in a state $b_i \in \mathcal{S}^{\mathcal{B}^*}$ and not discount length of time between visits to $\mathcal{S}^{\mathcal{B}^*}$. Formally, we will be seeking

$$\pi^*_\gamma \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \mathrm{T}_\pi^P}[\sum_{i=0}^{\infty} \Gamma_i \mathbf{1}_{\{b_i \in \mathcal{S}^{\mathcal{B}^*}\}}] \quad (\equiv V_\pi^\gamma), \tag{8.3}$$

where $\Gamma_0 = 1$ and

$$\Gamma_i = \prod_{t=0}^{i-1} \gamma(b_t), \quad \gamma(b_t) = \begin{cases} \gamma, & b_t \in \mathcal{S}^{\mathcal{B}^*} \\ 1, & \text{otherwise} \end{cases}. \tag{8.4}$$

Intuition for Γ_i. At first glance setting $\Gamma_i = \gamma^i$ to be the traditional RL exponential discount rate would seem reasonable. Unfortunately, $\nexists \gamma \in [0, 1]$ with $\Gamma_i = \gamma^i$ that avoids catastrophic myopic behavior. In particular, take

Figure 8.1 (Fourth). The agent starts in state 0 and only has two actions A and B. Taking action A transitions directly to an accepting state from which point the accepting state is visited every 2 steps. On the other hand, action B transitions to an accepting state with probability α and a sink state with probability $1-\alpha$. The accepting state reached by action B is revisited every step. Suppose $\beta = \pi(A) = 1 - \pi(B)$ then we can calculate:

$$\mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}[\sum_{i=0}^{\infty} \gamma^i \mathbf{1}_{\{b_i \in \mathcal{S}^{\mathcal{B}^*}\}}] = \frac{\beta}{1-\gamma^2} + \frac{(1-\beta)\alpha}{1-\gamma}. \tag{8.5}$$

For $\alpha > 1/2$, the optimal choice β is $\beta = 0$ implying that $P(\pi \models \varphi) = \alpha$. When $\alpha \in (1/2, 1)$ then this implies that π is not probability optimal. Indeed, $P(\pi \models \varphi) = \alpha < 1$ when $\beta = 0$ but $P(\pi^* \models \varphi) = 1$ by selecting $\beta = 1$. The intuition here, which can be formalized by taking $\gamma \to 1$, is that the average reward for taking action A is $\frac{1}{2}$ while the average reward for taking action B is 1 with probability α, which is worth the risk for large $\alpha > 1/2$.

To avoid this myopic behavior, we must avoid discriminating between return times between good states. The number steps (on average) it takes to return to $\mathcal{S}^{\mathcal{B}^*}$ is irrelevant: we only require that the system does return. For this reason we do not count time (hence $\gamma = 1$) in our definition of Γ_i when the system is not in $\mathcal{S}^{\mathcal{B}^*}$. We call this *eventual discounting*.

Analysis of π^*_γ

In this section we analyze how the probability of π^*_γ satisfying φ compares to that of the best possible one π^*.

Let the set $O(\tau) = \{i : b_i \in \mathcal{S}^{\mathcal{B}^*}\}$ denote the occurences (time steps) when a good state is reached. This quantity is natural since $|O(\tau)| = \infty$ if and only if $\tau \models \varphi$.

Lemma 8.2.1. *For any $\pi \in \Pi$ and $\gamma \in (0,1)$, we have*

$$|(1-\gamma)V^\gamma_\pi - \mathbb{P}[\pi \models \varphi]| \leq \log(\frac{1}{\gamma})O_\pi,$$

where $O_\pi = \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[|O(\tau)| \middle| \tau \not\models \varphi\right]$ is the expected number of visits to an accepting state for the trajectories that do not satisfy φ.

Proof. Fix some state $z = (s, b) \in \mathcal{Z}$.

$$V_\pi^\gamma(z) = \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}[\sum_{i=0}^{\infty} \Gamma_i \mathbf{1}_{\{b_i \in \mathcal{S}^{\mathcal{B}^*}\}} | z_0 = z]$$
$$= \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[\sum_{j=0}^{|O(\tau)|} \gamma^j | z_0 = z\right].$$

Using the fact that $\sum_{j=0}^{k} \gamma^j = \frac{1-\gamma^k}{1-\gamma}$, we have

$$V_\pi^\gamma(z) = \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[\frac{1-\gamma^{|O(\tau)|}}{1-\gamma}\middle|_{z_0=z}^{\tau\models\varphi}\right] \mathbb{P}_\pi[z \models \varphi]$$
$$+ \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[\frac{1-\gamma^{|O(\tau)|}}{1-\gamma}\middle|_{z_0=z}^{\tau\not\models\varphi}\right] \mathbb{P}_\pi[z \not\models \varphi]. \tag{8.6}$$

Since $|O(\tau)| = \infty$ for any $\tau \models \varphi$,

$$\mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[\frac{1-\gamma^{|O(\tau)|}}{1-\gamma}\middle|_{z_0=z}^{\tau\models\varphi}\right] = \frac{1}{1-\gamma}, \tag{8.7}$$

together with $\mathbb{P}_\pi[z \not\models \varphi] \geq 0$ implies

$$V_\pi^\gamma(z) \geq \frac{1}{1-\gamma}\mathbb{P}_\pi[z \models \varphi]. \tag{8.8}$$

Taking the expectation over initial states we have

$$V_\pi^\gamma \geq \frac{1}{1-\gamma}\mathbb{P}[\pi \models \varphi]. \tag{8.9}$$

Now we find an upper bound. Let $M_\pi(t) = \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[e^{t|O(\tau)|}\middle|\tau \not\models \varphi\right]$. Starting again with Eq (8.6) and using Eq (8.7), we have

$$V_\pi^\gamma(z) \leq \frac{\mathbb{P}_\pi[z \models \varphi]}{1-\gamma} + \frac{1 - \mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[e^{\log(\gamma)|O(\tau)|}\middle|_{z_0=z}^{\tau\not\models\varphi}\right]}{1-\gamma}, \tag{8.10}$$

where we have used that $\mathbb{P}_\pi[z \not\models \varphi] \leq 1$ for any $z \in \mathcal{Z}$. Taking the expectation with respect to the initial state distribution then we have

$$(1-\gamma)V_\pi^\gamma \leq \mathbb{P}[\pi \models \varphi] + 1 - M_\pi(\log(\gamma)) \tag{8.11}$$

In particular, $M_\pi(t)$ is convex and therefore it lies above its tangents:

$$M_\pi(t) \geq M_\pi(0) + tM_\pi'(0) = 1 + t\mathbb{E}_{\tau\sim \mathrm{T}_\pi^P}\left[|O(\tau)|\middle|\tau \not\models \varphi\right]$$
$$= 1 + tO_\pi.$$

Plugging this inequality into Eq (8.11), together with Eq (8.9),

$$\mathbb{P}[\pi \models \varphi] \leq (1-\gamma)V_\pi^\gamma \leq \mathbb{P}[\pi \models \varphi] + \log(\frac{1}{\gamma})O_\pi. \tag{8.12}$$

Subtracting $\mathbb{P}[\pi \models \varphi]$ from both sides and taking the absolute value completes the proof. □

Theorem 8.2.2. *(Non-asymptotic guarantee) For any $\gamma \in (0,1)$,*

$$\sup_{\pi\in\Pi} \mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^*_\gamma \models \varphi] \leq 2\log(\frac{1}{\gamma}) \sup_{\pi\in\Pi} O_\pi, \tag{8.13}$$

where $O_\pi = \mathbb{E}_{\tau\sim \mathrm{T}^P_\pi}\left[|O(\tau)| \middle| \tau \not\models \varphi\right]$.

Proof. Consider any sequence $\{\pi_i\}_{i=1}^\infty$ such that $\mathbb{P}[\pi_i \models \varphi] \to \sup_\pi \mathbb{P}[\pi \models \varphi]$ as $i \to \infty$. Then we have for any π_i,

$$\begin{aligned}
\mathbb{P}[\pi_i \models \varphi] - \mathbb{P}[\pi^*_\gamma \models \varphi] &= \mathbb{P}[\pi_i \models \varphi] - (1-\gamma)V^\gamma_{\pi_i} \\
&\quad + (1-\gamma)V^\gamma_{\pi_i} - (1-\gamma)V^\gamma_{\pi^*_\gamma} \\
&\quad + (1-\gamma)V^\gamma_{\pi^*_\gamma} - \mathbb{P}[\pi^*_\gamma \models \varphi] \\
&\overset{(a)}{\leq} |\mathbb{P}[\pi_i \models \varphi] - (1-\gamma)V^\gamma_{\pi_i}| \\
&\quad + |\mathbb{P}[\pi^*_\gamma \models \varphi] - (1-\gamma)V^\gamma_{\pi^*_\gamma}| \\
&\overset{(b)}{\leq} \log(\frac{1}{\gamma})(O_{\pi_i} + O_{\pi^*_\gamma}) \\
&\overset{(c)}{\leq} 2\log(\frac{1}{\gamma}) \sup_{\pi\in\Pi} O_\pi,
\end{aligned}$$

where (a) is triangle inequality together with removing the term $(1-\gamma)V^\gamma_{\pi_i} - (1-\gamma)V^\gamma_{\pi^*_\gamma}$ since it is nonpositive by definition of π^*_γ, (b) is an application of Lemma 8.2.1, and (c) is a supremum over all policies. Taking the limit on both sides as $i \to \infty$ completes the proof. □

Corollary 8.2.3. *If the number of policies in Π is finite then $\sup_{\pi\in\Pi} O_\pi = m < \infty$ is attained, is a finite constant and*

$$\sup_{\pi\in\Pi} \mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^*_\gamma \models \varphi] \leq 2m\log(\frac{1}{\gamma}).$$

Corollary 8.2.4. *In the case that $\mathcal{S}^\mathcal{M}$ and $\mathcal{A}^\mathcal{M}$ are finite, then $\mathcal{Z}$ and $\mathcal{A}$ are finite. It is known that optimal policies are deterministic [166] and therefore there we need only consider deterministic policies, for which there are a finite number. Thus $\sup_{\pi\in\Pi} O_\pi = m < \infty$ is attained, is a finite constant and*

$$\sup_{\pi\in\Pi} \mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^*_\gamma \models \varphi] \leq 2m\log(\frac{1}{\gamma}).$$

Interpretation

Theorem 8.2.2 relies on the quantity $\sup_{\pi\in\Pi} O_\pi$ to be finite for the bound to have meaning. In fact, we need only make requirements on $M_\pi(\log(\gamma))$ but the requirements are more easily understood on O_π. As an aside, $M_\pi(\log(\gamma))$

can be interpreted as the moment generating function of the random variable which is the number of visits to $\mathcal{S}^{\mathcal{B}^*}$. Instead we consider the equally natural quantity O_π. O_π is the (average) number of times that a good state is visited by a trajectory that does not satisfy the specification. Ideally, this number would be small and it would be easy to discriminate against good and bad policies.

The bad news. In the case that Π is an infinite class, conditions for ensuring $\sup_{\pi\in\Pi} O_\pi$ is finite is nontrivial and is dependent on the landscape of the transition function P of the MDP and Π.

Let us suppose $\sup_{\pi\in\Pi} O_\pi$ is infinite. This means there are policies that induce bad trajectories that eventually fail to reach $\mathcal{S}^{\mathcal{B}^*}$, but along the way visited $\mathcal{S}^{\mathcal{B}^*}$ an arbitrarily large (but finite) number of times. In other words, they are policies that are indistinguishable from actual probability-optimal policies until the heat death of the universe.

For example, let us consider the specification in Figure 8.1 (Third), given by "indefinitely cycle between red and yellow while avoiding blue". A good-looking bad policy is one that accomplishes the task frequently but, amongst the times that it fails, it would cycle between red and yellow many times before failing. $\sup_{\pi\in\Pi} O_\pi$ being infinite means that there are policies that will cycle arbitrarily many times before failing.

The good news. Corollary 8.2.4 reveals that discretization suffices to generate probability optimal policies, with suboptimality shrinking at a rate of $\log(\frac{1}{\gamma})$. This suggests that compactness of P and Π and continuity of P may very well be enough but we leave these conditions for future work. Finally, since all computers deal with finite precision, the number of policies is finite and therefore Corollary 8.2.3 similarly applies.

8.3 LTL Counterfactual Experience Replay

One can optimize the formulation in Eq (8.3) using any Q-learning or policy gradient approach, as seen in Algorithm 9 (Line 4). However, doing so is challenging since it suffers from reward sparsity: the agent only receives a signal if it reaches a good state.

We combat reward sparsity by exploiting the LDBA: $P^{\mathcal{B}}$ is completely known. By knowing $P^{\mathcal{B}}$, we can generate multiple off-policy trajectories from a single on-policy trajectory by modifying which stats in the LDBA we start in, which

Algorithm 9 `Learning with LCER`

Require: Maximum horizon T. Replay buffer $D = \{\}$.

1: **for** $k = 1, 2, \ldots$ **do**
2: Run π_{k-1} in the MDP for T timesteps and collect trajectory $\tau = (s_0, b_0, a_0, \ldots, s_{T-1}, b_{T-1}, a_{T-1}, s_T, b_T)$
3: $D_k \leftarrow \texttt{LCER}(D_{k-1}, \tau)$
4: $\pi_k \leftarrow \texttt{Update}(\pi_{k-1}, D_k)$ {Q-learn/Policy grad.}

Algorithm 10 `LCER` for Q-learning

Require: Dataset D. Trajectory τ of length T.

1: **for** $(s_t, a_t, s_{t+1}) \in \tau$ **do**
2: **for** $b \in \mathcal{S}^{\mathcal{B}}$ **do**
3: Set $\tilde{b} \leftarrow P^{\mathcal{B}}(b, L^{\mathcal{M}}(s_{t+1}))$
4: $D \leftarrow D \cup (s_t, b, a_t, \mathbf{1}_{\tilde{b} \in B^*}, s_{t+1}, \tilde{b})$
5: **for** $\epsilon \in \mathcal{A}^{\mathcal{B}}(s)$ **do**
6: Set $\tilde{b} \leftarrow P^{\mathcal{B}}(b, \epsilon)$
7: $D \leftarrow D \cup (s_t, b, \epsilon, \mathbf{1}_{\tilde{b} \in B^*}, s_t, \tilde{b})$
8: **return** D

notably does not require any access to the MDP transition function $P^{\mathcal{M}}$. We call this approach *LTL-guided Counterfactual Experience Replay*, `LCER` (Algorithm 9, Line 3), as it is a modification of standard experience replay [130, 143, 144] to include counterfactual experiences elsewhere in the LDBA. `LCER` is most simply understood through Q-learning, and needs careful modification for policy gradient methods.

Q-learning with `LCER`. See Algorithm 10 for a synopsis of `LCER` for Q-learning. Regardless of whatever state $s \in \mathcal{S}^{\mathcal{M}}$ the agent is in, we can pretend that the agent is in any $b \in \mathcal{S}^{\mathcal{B}}$. Then for any action the agent takes we can store experience tuples:

$$\{(s, b, a, r, s', \tilde{b}') \mid \forall b \in \mathcal{S}^{\mathcal{B}}\}, \tag{8.14}$$

where $\tilde{b}' = P^{\mathcal{B}}(b, L^{\mathcal{M}}(s'))$ is the transition that would have occurred from observing labelled state $L(s')$ in state (s, b) and $r = \mathbf{1}_{\tilde{b}' \in B^*}$. Furthermore we can add all jump transitions:

$$\{(s, b, \epsilon, r, s, \tilde{b}') \mid \forall b \in \mathcal{S}^{\mathcal{B}}, \forall \epsilon \in \mathcal{A}^{\mathcal{B}}(b)\}, \tag{8.15}$$

since jumps also do not affect the MDP. Notice when we add the jumps that $s' = s$, since only the LDBA state shifts in a jump.

Policy Gradient with `LCER`. See Algorithm 11 for a summary of `LCER` for policy gradient. For policy gradient, unlike Q-learning, it is necessary to calculate future reward-to-go: $R_k(\tau) = \sum_{i=k}^{T} \Gamma_i \mathbf{1}_{\{b_i \in \mathcal{S}^{\mathcal{B}^*}\}}$. Thus, we have to

Algorithm 11 `LCER` for Policy Gradient

Require: Dataset D. Trajectory τ of length T.

Set $\tilde{\mathcal{T}}_0 \leftarrow \tilde{\mathcal{T}}(\tau)$
for $k = 1, \ldots, T-1$ **do**
  $\tilde{\mathcal{T}}_k \leftarrow \mathcal{E}(\tilde{\mathcal{T}}_{k-1})$
  if $\tilde{\mathcal{T}}_k == \tilde{\mathcal{T}}_{k-1}$ **then**
    Set $\tilde{\mathcal{T}}_{T-1} \leftarrow \tilde{\mathcal{T}}_k$
    break
Set $D \leftarrow D \cup \tilde{\mathcal{T}}_{T-1}$
return D

generate entire trajectories that are consistent with $P^{\mathcal{B}}$ rather than independent transition tuples as in Eq (8.14). We will show how to generate all feasible trajectories.

Consider a trajectory $\tau = (s_0, b_0, a_0, \ldots, s_T, b_T)$ was collected. Let us remove jump transitions (s_i, b_i, a_i) where $a_i \in \mathcal{A}^{\mathcal{B}}(b_i)$ and consider the projection of the trajectory to the MDP $\tau_{\mathcal{M}} = (s_0, s_1, \ldots, s_T)$. We should only have control over the initial LDBA state b_0 as all other automaton states $(b_1, \ldots, b_T)$ in a trajectory sequence are determined by $\tau_{\mathcal{M}}$ and $b_{i+1} = P^{\mathcal{B}}(b_i, L^{\mathcal{M}}(s_i))$.

Therefore we add

$$\tilde{\mathcal{T}}(\tau) = \{(s_0, \tilde{b}_0, a_0, \ldots, s_T, \tilde{b}_T) \mid \forall \tilde{b}_0 \in \mathcal{S}^{\mathcal{B}}, \tilde{b}_i = P^{\mathcal{B}}(\tilde{b}_{i-1}, L^{\mathcal{M}}(s_i))\},$$

where only the LDBA states are different between the trajectories.

Now we handle jump transitions. Consider some $\tilde{\tau} \in \tilde{\mathcal{T}}(\tau)$. Recall, a jump transition can occur whenever $\mathcal{A}^{\mathcal{B}}(\tilde{b}_i)$ is non-empty. This involves adding a trajectory that is identical to $\tilde{\tau}$ all the way until the jump occurs. The jump occurs and then the same action sequence and MDP state sequence follows but with different LDBA states. Specifically, suppose $\tilde{b}_i$ had an available jump transitions, $\epsilon \in \mathcal{A}^{\mathcal{B}}(\tilde{b}_i)$. Then:

$$\tilde{\tau}_{i,\epsilon} = (s_0, \tilde{b}'_i, a_0, \ldots, s_i, \tilde{b}'_i, \epsilon, s_i, \tilde{b}'_{i+1}, a_i, \ldots, s_T, \tilde{b}'_T), \tag{8.16}$$

where $\tilde{b}'_k = \tilde{b}_i$ for $k \leq i$ and $\tilde{b}'_k = P^{\mathcal{B}}(\tilde{b}'_{k-1}, L^{\mathcal{M}}(s_k))$ otherwise.

We have to add all possible $\tilde{\tau}'_{i,\epsilon}$ that exist. Let $\mathcal{E}$ be the operator that adds jumps to existing sequences:

$$\mathcal{E}(\tilde{\mathcal{T}}(\tau)) = \tilde{\mathcal{T}}(\tau) \cup \{\tilde{\tau}_{i,\epsilon} \text{ from Eq (8.16)} | \forall \tilde{\tau} \in \tilde{\mathcal{T}}(\tau), \exists b_i \in \tilde{\tau} \text{ s.t. } \exists \epsilon \in \mathcal{A}^{\mathcal{B}}(b_i)\}. \tag{8.17}$$

We can only apply $\mathcal{E}(\mathcal{E}(\ldots(\mathcal{E}(\tilde{\mathcal{T}}(\tau)))))$ at most T times since the original length of τ is T.

Remark 8.3.1. *The length of τ has to be sufficiently large to make sure the LDBA has opportunity to reach $\mathcal{S}^{\mathcal{B}^*}$. A sufficient condition is $T \geq |\{b|\mathcal{A}^{\mathcal{B}}(b) \neq \varnothing\}|$, the number of LDBA states with jump transitions.*

It is possible to constructively generate feasible trajectories during the rollout of a policy rather than after-the-fact; see Appendix F.2.

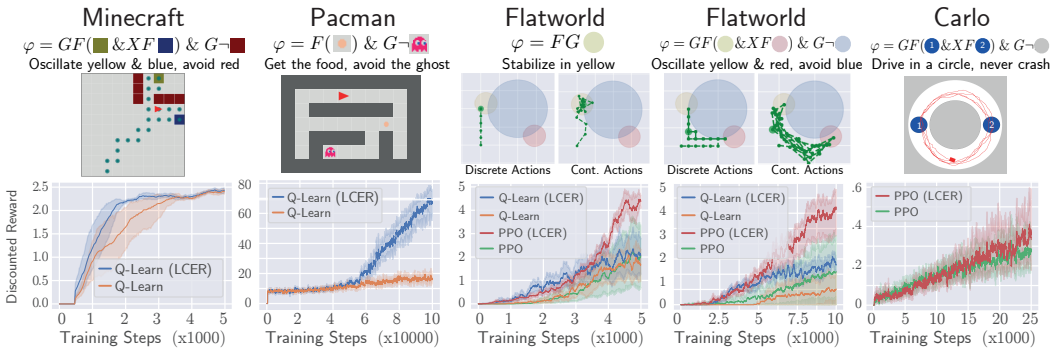


Figure 8.2: *Results.* Each column is an environment and a LTL formula we'd like an agent to satisfy. The environment and a trajectory from the final policy is illustrated in the center of the column (except for Pacman, which is the initial state). The learning curves at the bottom of each column show that adding off-policy data using `LCER` has strong benefits for empirical performance. ***First Column:*** Minecraft, where an agent should visit the yellow and blue areas while avoiding the red. The final policy is illustrated via blue dots. ***Second Column:*** Pacman, where an agent should collect the food while avoiding a ghost. ***Third Column:*** Flatword, where an agent should eventually stabilize in the yellow region. When the actions are discrete we use Q-learning, when the actions are continuous we use PPO. ***Fourth Column:*** Same as the third column except an agent should oscillate between the yellow and red regions while avoiding the blue. ***Fifth Column:*** Carlo, where an agent should drive in a circle without crashing by visiting the blue regions labelled 1 and 2 indefinitely.

8.4 Experiments

We perform experiments in four domains with varying LTL formulas, state spaces, action spaces, and environment stochasticity summarized in the following section. Our aim is to answer the following two questions: (1) Can we achieve policies that behave the way we expect an LTL-satisfying policy to behave? (2) How does `LCER` impact the performance of learning.

Environment Details

Minecraft The Minecraft environment is a 10×10 deterministic gridworld with 5 available actions: left, right, up, down, nothing. The agent, given by a red triangle starts in the cell $(9, 2)$. The environment, as well as the final behavior of the agent (given by blue dots) can be seen in Figure 8.2 (First).

Pacman The Pacman environment is a 5×8 deterministic gridworld with 5 available actions: left, right, up, down, nothing. The agent, given by a red triangle starts in the cell $(0, 3)$. The ghost chases the agent with probability 0.8 and takes a random action with probability 0.2, for this reason the environment is stochastic. The starting position of the environment can be seen in Figure 8.2 (Second).

Flatworld The Flatworld environment (seen in Figure 8.2 Third and Fourth) is a two dimensional continuous world. The agent (given by a green dot) starts at $(-1, -1)$. The dynamics of the world are given by $x' = x + a/10$ where both $x \in \mathbb{R}^2$ and $a \in [0, 1]^2$. We also allow the action space to be discrete by letting there be 5 actions (right, up, left, down, nothing) where the agent takes a full-throttle action in each respective direction.

Carlo The Carlo environment (seen in Figure 8.2 Fifth)is a simplified self-driving simulator that uses a bicycle model for the dynamics. The agent observes its position, velocity, and heading in radians for a total of 5 dimensions. The agent has control over its heading and throttle, for an action space of $[-1, 1]^2$. For this domain, we have chosen to use a circular track where the agent starts in the center of the road at an angle of $\{\pi(1+2i)/4\}_{i=0}^{3}$ and drive counterclockwise around in a circle without crashing.

Methods and Baseline

When the action space is discrete, we use Q-learning with `LCER` otherwise we use PPO with `LCER`. The baseline we compare against is the same method without `LCER`. This allows us to verify the extent to which `LCER` impacts performance. We also plot a trajectory from the final policy for each environment in the middle of each column of Figure 8.2, except for Pacman as it is difficult to visualize the interaction between the ghost and pacman outside of video.

Dealing with $\mathcal{A}$. For PPO, the agent's policy is a Gaussian (as in standard implementations) over the continuous action space. In order to deal with jump transitions (in the LDBA) when in a continuous action space (in the MDP),

we first let the agent decide whether to execute a jump transition or not (i.e. a probabilistic coin flip). If the agent chooses to not, then we take the action according to the Gaussian. The coin flip probability is learned, as well as the Gaussian. For the importance sampling term of PPO, the density of π is modified to account for the coin flip. For more details see Appendix F.1.

Results

Can we achieve desired behavior? The answer here is a resounding yes. For each environment (except Pacman) we illustrate the trajectory of the final policy above each learning curve in Figure 8.2. Determining the probability of satisfaction of the final policy is currently a challenging open problem (except in finite-state action spaces). Nevertheless, in each environment the agent qualitatively accomplishes the task. Even for challenging tasks with continuous action spaces, the agent is able to learn to accomplish the LTL specification.

Does `LCER` help in the learning process? According to the learning curves in the last row of Figure 8.2, `LCER` demonstrably expedites learning. In every environment with the exception of Carlo, `LCER` generates significant lift over lack of experience replay.

Intuition for why `LCER` helps? One way of viewing an LDBA is as a curriculum for what steps need to be taken in order to accomplish a task. By replacing the LDBA state of the agent with some other dream LDBA state, we are allowing the agent to "pretend" that it has already accomplished some portion of the task.

As an example, consider the Flatworld example in Figure 8.2 with $\varphi = GF(y \ \& \ XF(r)) \ \& \ (G\neg b)$. A baseline agent (without `LCER`) would need to accomplish the entirety of the task in order to see any reward. However, an agent with counterfacual data, need only visit y from state 0 of the LDBA (see figure 8.1 for the LDBA). Then once the agent is really good at getting to y, it needs to learn how to reach r from state 2. After both of these tasks are accomplished, independently, the agent has solved the whole task. Placing the agent in state 0 of the LDBA effectively lets the agent pretend that it has already visited r. In this sense, part of the task has been accomplished.

8.5 Related Work

Finding LTL-satisfying policies. Among the attempts at finding LTL-satisfying policies, Q-learning approaches have been the primary method of choice when the dynamics are unknown and Linear Programming methods when the dynamics are known [176, 83, 27, 33, 50]. The Q-learning approaches are predominantly constrained to finite state-action spaces. Among the works that extend to continuous action spaces [84], DDPG is used and takes the form of hierarchical RL which is known to potentially find myopic policies [205].

Handling a subset of LTL specifications involving those expressible as finite expressions can also be addressed with Reward machines [205, 34, 210]. Our work handles ω-regular expressions, subsuming regular expressions. Many problems are ω-regular problems, but not regular, such as liveness (something good will happen eventually) and safety (nothing bad will happen forever).

On the formulation in Eq (8.3)**.** Notable prior work on defining the value function as a function of the number of visits to $\mathcal{S}^{\mathcal{B}}$ and a state-dependent Γ_i function include Bozkurt et al. [27] and Cai et al. [32]. Most notably, these authors use multiple different state-dependent discount rates that have a complicated relationships between them that needs to be satisfied in the limit as $\gamma \to 1^-$. Our work drastically simplifies this, getting rid of the technical assumptions, while strengthening the guarantees. This allows us to find a non-asymptotic dependence on the suboptimality of a policies' probability of LTL satisfaction as a function of γ.

Off-policy data. One may view the counterfactual samples in Toro Icarte et al. [205] as an instantiation of LCER, limited to finite LTL expressions and discrete action spaces. Extension to continuous action space and full LTL requires a careful treatment. In the continuous action and full LTL setting, [218] incorporate starting the agent from a different initial LDBA state (than b_{-1}) which is still on-policy but from a different starting state and doesn't take advantage of the entire LDBA structure. This work can be seen as complimentary to our own.

Theory. Works with strong theoretical guarantees on policy satisfaction include Fu and Topcu [67], Wolff et al. [221], and Voloshin et al. [215] but are once again limited to discrete state/action spaces. Extensions of these work to continuous state space are not trivial as they make heavy use of the discrete Markov chain structure afforded to them.

8.6 Discussion

Our work, to the best of our knowledge, is the first to make full use of the LDBA as a form of experience replay and first to use policy gradient to learn LTL-satisfying policies. Our eventual discounting formulation is unrestricted to Finitary fragments of LTL like most prior work.

Despite the guarantees afforded to us by eventual discounting, in general the problem given in Eq (8.2) is not PAC learnable [224]. Though, like SAT solvers, it is still useful to find reasonable heuristics to problems that are difficult. We show that under particular circumstances, eventual discounting gives a signal on the quantity of interest in (8.2) and even when it fails, it selects a policy that is difficult to differentiate from a successful one. Further, the bad news discussed in Section 8.2 we speculate is unavoidable in general LTL specifications, without significant assumptions on the MDP. For example, for stability problems in LTL and assuming control-affine dynamics then Lyapunov functions can serve as certificates for a policies' LTL satisfaction. A reasonable relaxation to this would be require a system to behave a certain way for a long, but finite amount of time.

BIBLIOGRAPHY

[1] Romina Abachi, Mohammad Ghavamzadeh, and Amir-massoud Farahmand. *Policy-Aware Model Learning for Policy Gradient Methods*. 2020. arXiv: 2003.00030 [cs.AI].

[2] David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. "On the Expressivity of Markov Reward". In: *Advances in Neural Information Processing Systems*. 2021. URL: https://proceedings.neurips.cc/paper/2021/file/4079016d940210b4ae9ae7d41c4a2065-Paper.pdf.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. "Constrained Policy Optimization". In: *International Conference on Machine Learning*. 2017, pp. 22–31.

[4] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. "Constrained Policy Optimization". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017. URL: https://proceedings.mlr.press/v70/achiam17a.html (visited on 04/08/2022).

[5] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudıók, John Langford, and Hanna Wallach. "A reductions approach to fair classification". In: *arXiv preprint arXiv:1803.02453* (2018).

[6] Alekh Agarwal, Sham Kakade, and Lin F Yang. "Model-based reinforcement learning with a generative model is minimax optimal". In: *Conference on Learning Theory*. 2020.

[7] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. "Optimality and approximation with policy gradient methods in markov decision processes". In: *Conference on Learning Theory*. 2020.

[8] Gul Agha and Karl Palmskog. "A Survey of Statistical Model Checking". In: *ACM Trans. Model. Comput. Simul.* 28.1 (Jan. 2018). ISSN: 1049-3301. DOI: 10.1145/3158668. URL: https://doi.org/10.1145/3158668.

[9] Eitan Altman. *Constrained Markov decision processes*. Vol. 7. CRC Press, 1999.

[10] Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. en. 1st ed. Boca Raton: Routledge, Dec. 2021. ISBN: 978-1-315-14022-3. DOI: 10.1201/9781315140223. URL: https://www.taylorfrancis.com/books/9781315140223 (visited on 04/08/2022).

[11] András Antos, Csaba Szepesvári, and Rémi Munos. "Fitted Q-iteration in continuous action-space MDPs". In: *Advances in neural information processing systems*. 2008, pp. 9–16.

[12] András Antos, Csaba Szepesvári, and Rémi Munos. "Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path". In: *Machine Learning* 71.1 (2008), pp. 89–129.

[13] Pranav Ashok, Jan Křetínský, and Maximilian Weininger. "PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games". In: *Computer Aided Verification*. Ed. by Isil Dillig and Serdar Tasiran. Cham: Springer International Publishing, 2019, pp. 497–519. ISBN: 978-3-030-25540-4.

[14] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. "Model-based reinforcement learning with value-targeted regression". In: *International Conference on Machine Learning*. 2020.

[15] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. "Reinforcement learning of POMDPs using spectral methods". In: *Conference on Learning Theory*. 2016.

[16] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. en. Cambridge, Mass: The MIT Press, 2008. ISBN: 978-0-262-02649-9.

[17] Heejung Bang and James M. Robins. "Doubly Robust Estimation in Missing Data and Causal Inference Models". In: *Biometrics* 61.4 (2005), pp. 962–973. DOI: 10.1111/j.1541-0420.2005.00377.x.

[18] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. "Nearly-tight vc-dimension bounds for piecewise linear neural networks". In: *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2017)*. Ed. by Satyen Kale and Ohad Shamir. Vol. 65. Proceedings of Machine Learning Research. PMLR, July 2017, pp. 1064–1068. URL: https://proceedings.mlr.press/v65/harvey17a.html.

[19] Peter L. Bartlett and Shahar Mendelson. "Rademacher and Gaussian Complexities: Risk Bounds and Structural Results". In: *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*. COLT '01/EuroCOLT '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 224–240. ISBN: 3540423435.

[20] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[21] Dimitri P Bertsekas et al. "Dynamic programming and optimal control 3rd edition, volume ii". In: *Belmont, MA: Athena Scientific* (2011).

[22] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*. Vol. 1. 3. Athena scientific Belmont, MA, 2005.

[23] Lars Blackmore, Masahiro Ono, and Brian C Williams. "Chance-constrained optimal path planning with obstacles". In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.

[24] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. "Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising". In: *Journal of Machine Learning Research (JMLR)* 14 (2013), pp. 3207–3260.

[25] Philippe Bougerol and Nico Picard. "Strict stationarity of generalized autoregressive processes". In: *The Annals of Probability* 20.4 (1992), pp. 1714–1730. DOI: 10.1214/aop/1176989526. URL: https://doi.org/10.1214/aop/1176989526.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. "Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10349–10355. DOI: 10.1109/ICRA40945.2020.9196796.

[28] Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Kwiatkowska, David Parker, and Mateusz Ujma. "Verification of Markov Decision Processes Using Learning Algorithms". In: *Automated Technology for Verification and Analysis*. Ed. by Franck Cassez and Jean-François Raskin. Cham: Springer International Publishing, 2014, pp. 98–114. ISBN: 978-3-319-11936-6.

[29] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[30] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540.

[31] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[32] Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. "Modular deep reinforcement learning for continuous motion planning with temporal logic". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7973–7980.

[33] Mingyu Cai, Shaoping Xiao, Zhijun Li, and Zhen Kan. "Optimal Probabilistic Motion Planning with Potential Infeasible LTL Constraints". In: *IEEE Transactions on Automatic Control* (2021), pp. 1–1. DOI: 10.1109/TAC.2021.3138704.

[34] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. "LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 6065–6073. DOI: 10.24963/ijcai.2019/840. URL: https://doi.org/10.24963/ijcai.2019/840.

[35] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. "An empirical evaluation of supervised learning in high dimensions". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 96–103.

[36] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 161–168.

[37] Olivier Chapelle and Lihong Li. "An empirical evaluation of thompson sampling". In: *Advances in neural information processing systems*. 2011, pp. 2249–2257.

[38] Jinglin Chen and Nan Jiang. "Information-Theoretic Considerations in Batch Reinforcement Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 1042–1051.

[39] Jinglin Chen and Nan Jiang. "Information-Theoretic Considerations in Batch Reinforcement Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, June 2019, pp. 1042–1051.

[40] Ching-An Cheng, Xinyan Yan, Evangelos Theodorou, and Byron Boots. "Accelerating Imitation Learning with Predictive Models". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[41] Grace E. Cho and Carl D. Meyer. "Comparison of perturbation bounds for the stationary distribution of a Markov chain". In: *Linear*

Algebra and its Applications 335.1 (2001), pp. 137–150. ISSN: 0024-3795. DOI: https://doi.org/10.1016/S0024-3795(01)00320-2.

[42] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 4754–4765.

[43] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. "Model-Based Reinforcement Learning via Meta-Policy Optimization". In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 617–629.

[44] Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. "Coindice: Off-policy confidence interval estimation". In: *arXiv preprint arXiv:2010.11652* (2020).

[45] Christoph Dann and Emma Brunskill. "Sample complexity of episodic fixed-horizon reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 2818–2826.

[46] Christoph Dann, Gerhard Neumann, and Jan Peters. "Policy evaluation with temporal differences: A survey and comparison". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 809–883.

[47] Thomas Degris, Martha White, and Richard S Sutton. "Off-Policy Actor-Critic". In: (2012).

[48] Marc Peter Deisenroth and Carl Edward Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 465–472. ISBN: 9781450306195.

[49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.

[50] Xuchu Ding, Stephen L. Smith, Calin Belta, and Daniela Rus. "Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints". en. In: *IEEE Transactions on Automatic Control* 59.5 (May 2014), pp. 1244–1257. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2014.2298143.

[51] Victor Dorobantu and Andrew Taylor. *LyaPy*. https://github.com/vdorobantu/lyapy. 2020.

[52] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An open urban driving simulator”. In: *Conference on robot learning*. 2017.

[53] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. “Benchmarking deep reinforcement learning for continuous control”. In: *International Conference on Machine Learning (ICML)*. 2016.

[54] Miroslav Dudıók, John Langford, and Lihong Li. “Doubly robust policy evaluation and learning”. In: *International Conference on Machine Learning (ICML)*. 2011.

[55] Damien Ernst, Pierre Geurts, and Louis Wehenkel. “Tree-Based Batch Mode Reinforcement Learning”. In: *J. Mach. Learn. Res.* 6 (Dec. 2005), pp. 503–556. ISSN: 1532-4435.

[56] Damien Ernst, Pierre Geurts, and Louis Wehenkel. “Tree-based batch mode reinforcement learning”. In: *Journal of Machine Learning Research* 6.4 (2005), pp. 503–556.

[57] Amir M Farahmand, Mohammad Ghavamzadeh, Shie Mannor, and Csaba Szepesvári. “Regularized policy iteration”. In: *Advances in Neural Information Processing Systems*. IEEE. 2009, pp. 441–448.

[58] Amir-massoud Farahmand. “Iterative Value-Aware Model Learning”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 9072–9083.

[59] Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. “Value-Aware Loss Function for Model-based Reinforcement Learning”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 1486–1494.

[60] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. “More Robust Doubly Robust Off-policy Evaluation”. In: *International Conference on Machine Learning (ICML)*. 2018.

[61] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. “More Robust Doubly Robust Off-policy Evaluation”. In: *arXiv preprint arXiv:1802.03493* (2018).

[62] Yihao Feng, Lihong Li, and Qiang Liu. “A kernel loss for solving the bellman equation”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 15430–15441.

[63] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 49–58.

[64] Yoav Freund and Robert E Schapire. "Adaptive Game Playing Using Multiplicative Weights". In: *Games and Economic Behavior* 29 (1999), pp. 79–103.

[65] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer Series in Statistics. New York, NY, USA: Springer, 2001.

[66] Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. "Improved analysis of ucrl2 with empirical bernstein inequality". In: *arXiv preprint arXiv:2007.05456* (2020).

[67] Jie Fu and Ufuk Topcu. "Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints". In: *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*. Ed. by Dieter Fox, Lydia E. Kavraki, and Hanna Kurniawati. 2014. DOI: 10.15607/RSS.2014.X.039. URL: http://www.roboticsproceedings.org/rss10/p39.html.

[68] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2020. arXiv: 2004.07219 [cs.LG].

[69] Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, ziyu wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Thomas Paine. "Benchmarks for Deep Off-Policy Evaluation". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=kWSeGEeHvF8.

[70] Javier García and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[71] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Zhengxing Chen, Yuchen He, Zachary Kaden, Vivek Narayanan, and Xiaohui Ye. "Horizon: Facebook's Open Source Applied Reinforcement Learning Platform". In: *arXiv preprint arXiv:1811.00260* (2018).

[72] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. "Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model". In: *Machine learning* 91.3 (2013), pp. 325–349.

[73] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[74] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680.

[75] Zhaohan Guo, Philip S Thomas, and Emma Brunskill. "Using Options and Covariance Testing for Long Horizon Off-Policy Policy Evaluation". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2492–2501.

[76] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[77] David Ha and Jürgen Schmidhuber. "World Models". In: *arXiv preprint arXiv:1803.10122* (2018).

[78] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. "Reinforcement Learning with Deep Energy-Based Policies". In: *International Conference on Machine Learning*. 2017, pp. 1352–1361.

[79] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. "Cooperative inverse reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3909–3917.

[80] Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. "Lazy probabilistic model checking without determinisation". In: *arXiv preprint arXiv:1311.2928* (2013).

[81] John Michael Hammersley and David Christopher Handscomb. "Monte Carlo methods". In: (1964).

[82] Anna Harutyunyan, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. "Q(lambda) with Off-Policy Corrections". In: *Conference on Algorithmic Learning Theory (ALT)*. 2016.

[83] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. *Logically-Constrained Reinforcement Learning*. 2018. DOI: 10.48550/ARXIV.1801.08099. URL: https://arxiv.org/abs/1801.08099.

[84] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. "Deep reinforcement learning with temporal logics". In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2020, pp. 1–22.

[85] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.

[86] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.

[87] David Haussler. "Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension". In: *Journal of Combinatorial Theory, Series A* 69.2 (1995), pp. 217–232.

[88] Mikael Henaff, Alfredo Canziani, and Yann LeCun. "Model-Predictive Policy Learning with Uncertainty Regularization for Driving in Dense Traffic". In: *arXiv preprint arXiv:1901.02705* (2019).

[89] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[90] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. "Approximate Probabilistic Model Checking". In: *Verification, Model Checking, and Abstract Interpretation*. Ed. by Bernhard Steffen and Giorgio Levi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 73–84. ISBN: 978-3-540-24622-0.

[91] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. "Deep q-learning from demonstrations". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[92] Daniel G Horvitz and Donovan J Thompson. "A generalization of sampling without replacement from a finite universe". In: *Journal of the American statistical Association* 47.260 (1952), pp. 663–685.

[93] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. "Reward learning from human preferences and demonstrations in atari". In: *Advances in neural information processing systems* 31 (2018).

[94] Thomas Jaksch, Ronald Ortner, and Peter Auer. "Near-optimal Regret Bounds for Reinforcement Learning". In: *Journal of Machine Learning Research* 11.51 (2010), pp. 1563–1600. ISSN: 1533-7928.

[95] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. "When to Trust Your Model: Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems 32*. Ed. by H.

Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 12519–12530. URL: http://papers.nips.cc/paper/9416-when-to-trust-your-model-model-based-policy-optimization.pdf.

[96] Nan Jiang and Jiawei Huang. "Minimax Value Interval for Off-Policy Evaluation and Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 2747–2758.

[97] Nan Jiang and Lihong Li. "Doubly Robust Off-policy Value Evaluation for Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. 2016.

[98] Nan Jiang and Lihong Li. "Doubly Robust Off-policy Value Evaluation for Reinforcement Learning". In: *International Conference on Machine Learning*. 2016, pp. 652–661.

[99] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. "A composable specification language for reinforcement learning tasks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[100] Sham Kakade and John Langford. "Approximately optimal approximate reinforcement learning". In: *ICML*. Vol. 2. 2002, pp. 267–274.

[101] Sham Machandranath Kakade. "On the sample complexity of reinforcement learning". PhD thesis. University of College London, 2003.

[102] Nathan Kallus and Masatoshi Uehara. "Double reinforcement learning for efficient off-policy evaluation in markov decision processes". In: *arXiv preprint arXiv:1908.08526* (2019).

[103] Nathan Kallus and Masatoshi Uehara. "Efficiently breaking the curse of horizon: Double reinforcement learning in infinite-horizon processes". In: *arXiv preprint arXiv:1909.05850* (2019).

[104] Joseph DY Kang, Joseph L Schafer, et al. "Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data". In: *Statistical science* 22.4 (2007), pp. 523–539.

[105] Abbas Kazerouni, Mohammad Ghavamzadeh, Yasin Abbasi Yadkori, and Benjamin Van Roy. "Conservative Contextual Linear Bandits". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. 2017.

[106] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. *MOReL : Model-Based Offline Reinforcement Learning*. 2020. arXiv: 2005.05951 [cs.LG].

[107] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[108] Jyrki Kivinen and Manfred K Warmuth. "Exponentiated Gradient versus Gradient Descent for Linear Predictors". In: *Information and Computation* 132.1 (1997), pp. 1–63.

[109] Sven Koenig and Reid G. Simmons. "The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms". In: *Machine Learning* 22.1 (Mar. 1996), pp. 227–250. ISSN: 1573-0565. DOI: 10.1007/BF00114729.

[110] Andrey Kolobov, Mausam, and Daniel S. Weld. "A Theory of Goal-Oriented MDPs with Dead Ends". In: *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. UAI'12. Catalina Island, CA: AUAI Press, 2012, pp. 438–447.

[111] Jan Křetıónskỳ, Tobias Meggendorfer, and Salomon Sickert. "Owl: a library for ω-words, automata, and LTL". In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2018, pp. 543–550.

[112] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[113] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. "Model-Ensemble Trust-Region Policy Optimization". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[114] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.

[115] Michail G Lagoudakis and Ronald Parr. "Least-squares policy iteration". In: *Journal of machine learning research* 4.12 (2003), pp. 1107–1149.

[116] Sascha Lange, Thomas Gabel, and Martin Riedmiller. "Batch reinforcement learning". In: *Reinforcement learning*. Springer, 2012, pp. 45–73.

[117] Richard Lassaigne and Sylvain Peyronnet. "Probabilistic Verification and Approximation". In: *Electron. Notes Theor. Comput. Sci.* 143 (Jan. 2006), pp. 101–114. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2005.05.031. URL: https://doi.org/10.1016/j.entcs.2005.05.031.

[118] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. "Finite-sample analysis of least-squares policy iteration". In: *Journal of Machine Learning Research* 13.Oct (2012), pp. 3041–3074.

[119] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. "Finite-sample analysis of LSTD". In: *ICML-27th International Conference on Machine Learning*. 2010, pp. 615–622.

[120] Alessandro Lazaric and Marcello Restelli. "Transfer from multiple MDPs". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1746–1754.

[121] Hoang M Le, Nan Jiang, Alekh Agarwal, Miro Dudík, Yisong Yue, and Hal Daumé III. "Hierarchical Imitation and Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. July 2018.

[122] Hoang M Le, Andrew Kang, Yisong Yue, and Peter Carr. "Smooth imitation learning for online sequence prediction". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. JMLR. org. 2016, pp. 680–688.

[123] Hoang M Le, Cameron Voloshin, and Yisong Yue. "Batch Policy Learning under Constraints". In: *International Conference on Machine Learning (ICML)*. 2019.

[124] Wee Sun Lee, Peter L Bartlett, and Robert C Williamson. "Efficient agnostic learning of neural networks with bounded fan-in". In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 2118–2132.

[125] Sergey Levine and Pieter Abbeel. "Learning neural network policies with guided policy search under unknown dynamics". In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079.

[126] Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. "Breaking the sample size barrier in model-based reinforcement learning with a generative model". In: *Advances in neural information processing systems* 33 (2020), pp. 12861–12872.

[127] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. "Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2011.

[128] Lihong Li, Rémi Munos, and Csaba Szepesvári. "Toward Minimax Off-policy Value Estimation". In: *Artificial Intelligence and Statistics*. 2015, pp. 608–616.

[129] 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". In: *International Conference on Learning Representations (ICLR)*. 2016.

[130] Long-Ji Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine learning* 8.3-4 (1992), pp. 293–321.

[131] Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. *Environment-Independent Task Specifications via GLTL*. 2017. DOI: 10.48550/ARXIV.1704.04341. URL: https://arxiv.org/abs/1704.04341.

[132] Anqi Liu, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue. "Robust regression for safe exploration in control". In: *Learning for Dynamics and Control (L4DC)*. 2020.

[133] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. "Breaking the curse of horizon: Infinite-horizon off-policy estimation". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[134] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. "Breaking the curse of horizon: Infinite-horizon off-policy estimation". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5361–5371.

[135] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. "Off-Policy Policy Gradient with State Distribution Correction". In: *arXiv preprint arXiv:1904.08473* (2019).

[136] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *International Conference on Machine Learning*. 2019, pp. 4114–4124.

[137] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. "Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[138] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. "Off-policy Learning in Two-stage Recommender Systems". In: *International World Wide Web Conference (WWW)*. 2020.

[139] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002. ISBN: 0521642981.

[140] Odalric-Ambrym Maillard, Rémi Munos, Alessandro Lazaric, and Mohammad Ghavamzadeh. "Finite-sample analysis of Bellman residual minimization". In: *Proceedings of 2nd Asian Conference on Machine Learning*. 2010, pp. 299–314.

[141] Andreas Maurer and Massimiliano Pontil. *Empirical Bernstein Bounds and Sample Variance Penalization*. 2009. DOI: 10.48550/ARXIV.0907.3740. URL: https://arxiv.org/abs/0907.3740.

[142] Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E Schapire. "Reinforcement learning with convex constraints". In: *Neural Information Processing Systems (NeurIPS)*. 2019.

[143] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[144] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[145] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[146] William H Montgomery and Sergey Levine. "Guided policy search via approximate mirror descent". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4008–4016.

[147] Rémi Munos. "Error bounds for approximate policy iteration". In: *ICML*. Vol. 3. 2003, pp. 560–567.

[148] Rémi Munos. "Performance bounds in l_p-norm for approximate value iteration". In: *SIAM journal on control and optimization* 46.2 (2007), pp. 541–561.

[149] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. "Safe and Efficient Off-Policy Reinforcement Learning". In: *Neural Information Processing Systems (NeurIPS)*. 2016.

[150] Rémi Munos and Csaba Szepesvári. "Finite-time bounds for fitted value iteration". In: *Journal of Machine Learning Research* 9.5 (2008), pp. 815–857.

[151] Susan A Murphy, Mark J van der Laan, James M Robins, and Conduct Problems Prevention Research Group. "Marginal mean models for dynamic regimes". In: *Journal of the American Statistical Association* 96.456 (2001), pp. 1410–1423.

[152] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections". In: *Advances in Neural Information Processing Systems*. 2019, pp. 2315–2325.

[153] Arkadii Semenovich Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. Wiley, 1983.

[154] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *Icml*. Vol. 99. 1999, pp. 278–287.

[155] Xinkun Nie, Emma Brunskill, and Stefan Wager. "Learning When-to-Treat Policies". In: *arXiv preprint arXiv:1905.09751* (2019).

[156] Michael Oberst and David Sontag. "Counterfactual Off-Policy Evaluation with Gumbel-Max Structural Causal Models". In: *International Conference on Machine Learning*. 2019, pp. 4881–4890.

[157] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. "Self-Imitation Learning". In: *International Conference on Machine Learning*. 2018.

[158] Masahiro Ono, Marco Pavone, Yoshiaki Kuwata, and J Balaram. "Chance-constrained dynamic programming with application to risk-aware robotic space exploration". In: *Autonomous Robots* 39.4 (2015), pp. 555–571.

[159] Dirk Ormoneit and Śaunak Sen. "Kernel-based reinforcement learning". In: *Machine learning* 49.2-3 (2002), pp. 161–178.

[160] Ian Osband, Daniel Russo, and Benjamin Van Roy. "(More) efficient reinforcement learning via posterior sampling". In: *Advances in Neural Information Processing Systems* 26 (2013).

[161] Cosmin Paduraru. "Off-policy evaluation in Markov decision processes". PhD thesis. McGill University Libraries, 2013.

[162] Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. "Sample-efficient batch reinforcement learning for dialogue management optimization". In: *ACM Transactions on Speech and Language Processing (TSLP)* 7.3 (2011), p. 7.

[163] Michael JD Powell and J Swann. "Weighted uniform sampling—a Monte Carlo technique for reducing variance". In: *IMA Journal of Applied Mathematics* 2.3 (1966), pp. 228–236.

[164] Doina Precup, Richard S Sutton, and Satinder P Singh. "Eligibility Traces for Off-Policy Policy Evaluation". In: *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc. 2000, pp. 759–766.

[165] Doina Precup, Richard S. Sutton, and Satinder P. Singh. "Eligibility Traces for Off-Policy Policy Evaluation". In: *International Conference on Machine Learning (ICML)*. 2000.

[166] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[167] Guannan Qu and Adam Wierman. "Finite-time analysis of asynchronous stochastic approximation and Q-learning". In: *Conference on Learning Theory*. 2020.

[168] Guannan Qu, Chenkai Yu, Steven Low, and Adam Wierman. "Exploiting Linear Models for Model-Free Nonlinear Control: A Provably Convergent Policy Gradient Approach". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 6539–6546.

[169] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. *Stable Baselines3*. `https://github.com/DLR-RM/stable-baselines3`. 2019.

[170] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. *A Game Theoretic Framework for Model Based Reinforcement Learning*. 2020. arXiv: `2004.07804 [cs.LG]`.

[171] Jette Randlòv and Preben Alstròm. "Learning to Drive a Bicycle Using Reinforcement Learning and Shaping". In: *ICML*. 1998.

[172] Martin Riedmiller. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method". In: *Machine Learning: ECML 2005*. Springer, 2005, pp. 317–328.

[173] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. "Reinforcement learning for robot soccer". In: *Autonomous Robots* 27.1 (2009), pp. 55–73.

[174] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. "A survey of multi-objective sequential decision-making". In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 67–113.

[175] Stephane Ross and J Andrew Bagnell. "Reinforcement and imitation learning via interactive no-regret learning". In: *arXiv preprint arXiv:1406.5979* (2014).

[176] Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications". In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 1091–1096. DOI: 10.1109/CDC.2014.7039527.

[177] Yuta Saito, Shunsuke Aihara, Megumi Matsutani, and Yusuke Narita. "A Large-scale Open Dataset for Bandit Algorithms". In: *arXiv preprint arXiv:2008.07146* (2020).

[178] Florian Schaefer and Anima Anandkumar. "Competitive Gradient Descent". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 7625–7635. URL: http://papers.nips.cc/paper/8979-competitive-gradient-descent.pdf.

[179] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[180] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[181] E. Seneta. "Perturbation of the stationary distribution measured by ergodicity coefficients". In: *Advances in Applied Probability* 20.1 (1988), pp. 228–230. DOI: 10.2307/1427277.

[182] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[183] Jack Sherman and Winifred J. Morrison. "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix". In: *Ann. Math. Statist.* 21.1 (Mar. 1950), pp. 124–127. DOI: 10.1214/aoms/1177729893.

[184] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínský. "Limit-Deterministic Büchi Automata for Linear Temporal Logic". In: *Computer Aided Verification*. Ed. by Swarat Chaudhuri and Azadeh Farzan. Cham: Springer International Publishing, 2016, pp. 312–332. ISBN: 978-3-319-41540-6.

[185] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[186] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. "Reward is enough". In: *Artificial Intelligence* 299 (2021), p. 103535.

[187] Jonathan Sorg. "The Optimal Reward Problem: Designing Effective Reward for Bounded Agents". PhD thesis. University of Michigan, USA, 2011. URL: https://hdl.handle.net/2027.42/89705.

[188] Jonathan Sorg, Satinder Singh, and Richard L. Lewis. "Reward Design via Online Gradient Ascent". In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*. NIPS'10. Vancouver, British Columbia, Canada: Curran Associates Inc., 2010, pp. 2190–2198.

[189] R.S. Sutton, D. Precup, and S. Singh. "Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". In: *Artificial Intelligence* 112 (1999), pp. 181–211.

[190] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, Mar. 1998. ISBN: 0-262-19398-1.

[191] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[192] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[193] Richard S. Sutton. "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming". In: *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 1990, pp. 216–224.

[194] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[195] Adith Swaminathan and Thorsten Joachims. "Batch learning from logged bandit feedback through counterfactual risk minimization." In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1731–1755.

[196] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. "Off-policy evaluation for slate recommendation". In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[197] Csaba Szepesvári. "Algorithms for reinforcement learning". In: *Synthesis lectures on artificial intelligence and machine learning* 4.1 (2010), pp. 1–103.

[198] Jean Tarbouriech, Matteo Pirotta, Michal Valko, and Alessandro Lazaric. "Sample Complexity Bounds for Stochastic Shortest Path with a Generative Model". en. In: *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*. Mar. 2021. URL: https://proceedings.mlr.press/v132/tarbouriech21a.html.

[199] Jean Tarbouriech, Runlong Zhou, Simon Shaolei Du, Matteo Pirotta, Michal Valko, and Alessandro Lazaric. "Stochastic Shortest Path: Minimax, Parameter-Free and Towards Horizon-Free Regret". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=cc_AXK6rWPJ.

[200] Guy Tennenholtz, Shie Mannor, and Uri Shalit. "Off-Policy Evaluation in Partially Observable Environments". In: *arXiv preprint arXiv:1909.03739* (2019).

[201] Philip Thomas and Emma Brunskill. "Data-efficient off-policy policy evaluation for reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2016.

[202] Philip Thomas and Emma Brunskill. "Data-efficient off-policy policy evaluation for reinforcement learning". In: *International Conference on Machine Learning*. 2016, pp. 2139–2148.

[203] Philip S Thomas, Georgios Theocharous, Mohammad Ghavamzadeh, Ishan Durugkar, and Emma Brunskill. "Predictive off-policy policy evaluation for nonstationary decision problems, with applications to digital marketing". In: *AAAI Conference on Innovative Applications (IAA)*. 2017.

[204] 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*. IEEE. 2012, pp. 5026–5033.

[205] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning". In: *J. Artif. Int. Res.* 73 (May 2022). ISSN: 1076-9757. DOI: 10.1613/jair.1.12440. URL: https://doi.org/10.1613/jair.1.12440.

[206] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. "Is deep reinforcement learning really superhuman on atari? leveling the playing field". In: *arXiv preprint arXiv:1908.04683* (2019).

[207] John Tsitsiklis and Benjamin Van Roy. "Analysis of temporal-diffference learning with function approximation". In: *Advances in neural information processing systems* 9 (1996).

[208] Masatoshi Uehara, Jiawei Huang, and Nan Jiang. "Minimax Weight and Q-Function Learning for Off-Policy Evaluation". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 1023–1032.

[209] Masatoshi Uehara, Jiawei Huang, and Nan Jiang. "Minimax weight and q-function learning for off-policy evaluation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9659–9668.

[210] Pashootan Vaezipoor, Andrew C. Li, Rodrigo Toro Icarte, and Sheila A. McIlraith. "LTL2Action: Generalizing LTL Instructions for Multi-Task RL". In: *Proceedings of the 38th International Conference on Machine Learning, ICML*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 10497–10508. URL: http://proceedings.mlr.press/v139/vaezipoor21a.html.

[211] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In: *AAAI*. Vol. 2. Phoenix, AZ. 2016, p. 5.

[212] Kristof Van Moffaert and Ann Nowé. "Multi-objective reinforcement learning using sets of pareto dominating policies". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3483–3512.

[213] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[214] Cameron Voloshin, Nan Jiang, and Yisong Yue. "Minimax Model Learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1612–1620.

[215] Cameron Voloshin, Hoang Minh Le, Swarat Chaudhuri, and Yisong Yue. "Policy Optimization with Linear Temporal Logic Constraints". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=yZcPRIZEwOG.

[216] Cameron Voloshin, Hoang Minh Le, Nan Jiang, and Yisong Yue. "Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. 2021. URL: https://openreview.net/forum?id=IsK8iKbL-I.

[217] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[218] Chuanzheng Wang, Yinan Li, Stephen L. Smith, and Jun Liu. *Continuous Motion Planning with Temporal Logic Specifications using Deep Neural Networks*. 2020. DOI: 10.48550/ARXIV.2004.02610. URL: https://arxiv.org/abs/2004.02610.

[219] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudík. "Optimal and Adaptive Off-policy Evaluation in Contextual Bandits". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 3589–3597.

[220] MA Wiering. "Multi-agent reinforcement learning for traffic light control". In: *International Conference on Machine Learning (ICML)*. 2000.

[221] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. "Robust control of uncertain Markov Decision Processes with temporal logic specifications". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. 2012, pp. 3372–3379. DOI: 10.1109/CDC.2012.6426174.

[222] Yihong Wu and Pengkun Yang. "Chebyshev polynomials, moment matching, and optimal estimation of the unseen". In: *The Annals of Statistics* 47.2 (2019), pp. 857–883. DOI: 10.1214/17-AOS1665. URL: https://doi.org/10.1214/17-AOS1665.

[223] Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. "Towards Optimal Off-Policy Evaluation for Reinforcement Learning with Marginalized Importance Sampling". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9665–9675.

[224] Cambridge Yang, Michael L. Littman, and Michael Carbin. "Reinforcement Learning for General LTL Objectives Is Intractable". In: *CoRR* abs/2111.12679 (2021). arXiv: 2111.12679. URL: https://arxiv.org/abs/2111.12679.

[225] Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. "Off-Policy Evaluation via the Regularized Lagrangian". In: *Advances in Neural Information Processing Systems* 33 (2020).

[226] Håkan L. S. Younes, Edmund M. Clarke, and Paolo Zuliani. "Statistical Verification of Probabilistic Properties with Unbounded Until". In: *Formal Methods: Foundations and Applications*. Ed. by Jim Davies, Leila Silva, and Adenilso Simao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 144–160. ISBN: 978-3-642-19829-8.

[227] Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. "On the importance of hyperparameter optimization for model-based reinforcement learning". In: *International Conference on Artificial Intelligence and Statistics*. 2021.

[228] Michael R Zhang, Tom Le Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, Ziyu Wang, and Mohammad Norouzi. "Autoregressive Dynamics Models for Offline Policy Evaluation and Optimization". In: *arXiv preprint arXiv:2104.13877* (2021).

[229] Shangtong Zhang, Bo Liu, and Shimon Whiteson. "GradientDICE: Rethinking Generalized Offline Estimation of Stationary Values". In: *arXiv preprint arXiv:2001.11113* (2020).

[230] Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C Parkes, and Richard Socher. "The ai economist: Improving equality and productivity with ai-driven tax policies". In: *arXiv preprint arXiv:2004.13332* (2020).

[231] Brian D Ziebart. "Modeling purposeful adaptive behavior with the principle of maximum causal entropy". In: (2010).

[232] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. "Maximum Entropy Inverse Reinforcement Learning". In: *AAAI*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[233] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 9781577353683.

[234] Martin Zinkevich. "Online convex programming and generalized infinitesimal gradient ascent". In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 928–936.

Part IV

APPENDIX

Appendix A

CHAPTER 1 APPENDIX

A.1 Glossary of Terms

Acronym	Term
OPE	Off-Policy Policy Evaluation
X	State Space
A	Action Space
P	Transition Function
R	Reward Function
γ	Discount Factor
d_0	Initial State Distribution
D	Dataset
τ	Trajectory/Episode
T	Horizon/Episode Length
N	Number of episodes in D
π_b	Behavior Policy
π_e	Evaluation Policy
V, Q	Value (ex: $V(s)$), Action-Value (ex: $Q(s,a)$)
$\rho^i_{j:j'}$	Cumulative Importance Weight, $\prod_{t=j}^{\min(j',T-1)} \frac{\pi_e(a^i_t\|s^i_t)}{\pi_b(a^i_t\|s^i_t)}$. If $j > j'$ then $\rho = 1$.
IPS	Inverse Propensity Scoring
DM	Direct Method
HM	Hybrid Method
IS	Importance Sampling
PDIS	Per-Decision Importance Sampling
WIS	Weighted Importance Sampling
PDWIS	Per-Decision Weighted Importance Sampling
FQE	Fitted Q Evaluation [123]
IH	Infinite Horizon [133]
Q-Reg	Q Regression [60]
MRDR	More Robust Doubly Robst [60]
AM	Approximate Model (Model Based)
$Q(\lambda)$	$Q^\pi(\lambda)$ [82]
$R(\lambda)$	Retrace(λ) [149]
Tree	Tree-Backup(λ) [165]
DR	Doubly-Robust [97, 54]
WDR	Weighted Doubly-Robust [54]
MAGIC	Model And Guided Importance Sampling Combining (Estimator) [201]
Graph	Graph Environment
Graph-MC	Graph Mountain Car Environment
MC	Mountain Car Environment
Pix-MC	Pixel-Based Mountain Car Environment
Enduro	Enduro Environment
Graph-POMDP	Graph-POMDP Environment
GW	Gridworld Environment
Pix-GW	Pixel-Based Gridworld Environment

A.2 Ranking of Methods

A method that is within 10% of the method with the lowest Relative MSE is counted as a top method, called Near-top Frequency, and then we aggregate across all experiments. See Table A.1 for a sorted list of how often the methods appear within 10% of the best method.

Table A.1: Fraction of time among the top estimators across all experiments

Method	Near-top Frequency
MAGIC FQE	0.300211
DM FQE	0.236786
IH	0.190275
WDR FQE	0.177590
MAGIC $Q^\pi(\lambda)$	0.173362
WDR $Q^\pi(\lambda)$	0.173362
DM $Q^\pi(\lambda)$	0.150106
DR $Q^\pi(\lambda)$	0.135307
WDR R(λ)	0.133192
DR FQE	0.128964
MAGIC R(λ)	0.107822
WDR Tree	0.105708
DR R(λ)	0.105708
DM R(λ)	0.097252
DM Tree	0.084567
MAGIC Tree	0.076110
DR Tree	0.073996
DR MRDR	0.073996
WDR Q-Reg	0.071882
DM AM	0.065539
IS	0.063425
WDR MRDR	0.054968
PDWIS	0.046512
DR Q-Reg	0.044397
MAGIC AM	0.038055
MAGIC MRDR	0.033827
DM MRDR	0.033827
PDIS	0.033827
MAGIC Q-Reg	0.027484
WIS	0.025370
NAIVE	0.025370
DM Q-Reg	0.019027
DR AM	0.012685
WDR AM	0.006342

Decision Tree Support

Tables A.2-A.9 provide a numerical support for the decision tree in Figure 3.3. Each table refers to a child node in the decision tree, ordered from left to right, respectively. For example, Table A.2 refers to the left-most child node (propery specified, short horizon, small policy mismatch) while Table A.9 refers to the right-most child node (misspecified, good representation, long horizon, good π_b estimate).

Table A.2: Near-top Frequency. Properly specified, short horizon, small policy mismatch experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	4.7%	4.7%	3.1%	4.7%
Q-Reg	0.0%	4.7%	6.2%	4.7%
MRDR	7.8%	14.1%	7.8%	7.8%
FQE	**40.6**%	23.4%	21.9%	**34.4**%
R(λ)	17.2%	20.3%	20.3%	14.1%
Q$^\pi$(λ)	21.9%	18.8%	18.8%	17.2%
Tree	15.6%	12.5%	12.5%	14.1%
IH	17.2%	-	-	-

	IPS	
	Standard	Per-Decision
IS	**4.7**%	4.7%
WIS	3.1%	3.1%
NAIVE	1.6%	-

Table A.3: Near-top Frequency. Properly specified, short horizon, large policy mismatch experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	20.3%	1.6%	0.0%	7.8%
Q-Reg	1.6%	1.6%	3.1%	1.6%
MRDR	3.1%	1.6%	6.2%	1.6%
FQE	**35.9**%	14.1%	17.2%	**37.5**%
R(λ)	23.4%	14.1%	20.3%	23.4%
Q$^\pi$(λ)	15.6%	15.6%	14.1%	20.3%
Tree	21.9%	12.5%	18.8%	21.9%
IH	29.7%	-	-	-

	IPS	
	Standard	Per-Decision
IS	0.0%	0.0%
WIS	0.0%	**1.6**%
NAIVE	3.1%	-

Table A.4: Near-top Frequency. Properly specified, long horizon, small policy mismatch experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	6.9%	0.0%	0.0%	5.6%
Q-Reg	0.0%	1.4%	1.4%	1.4%
MRDR	1.4%	0.0%	1.4%	2.8%
FQE	**50.0**%	22.2%	23.6%	**50.0**%
R(λ)	13.9%	12.5%	11.1%	9.7%
Q$^\pi$(λ)	20.8%	18.1%	18.1%	18.1%
Tree	2.8%	1.4%	0.0%	2.8%
IH	29.2%	-	-	-

	IPS	
	Standard	Per-Decision
IS	**0.0**%	0.0%
WIS	0.0%	0.0%
NAIVE	5.6%	-

Table A.5: Near-top Frequency. Properly specified, long horizon, large policy mismatch, deterministic env/rew experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	3.5%	3.5%	1.8%	1.8%
Q-Reg	3.5%	1.8%	0.0%	0.0%
MRDR	3.5%	1.8%	0.0%	0.0%
FQE	15.8%	17.5%	29.8%	28.1%
R(λ)	1.8%	3.5%	0.0%	0.0%
Q$^\pi$(λ)	**22.8**%	15.8%	**38.6**%	24.6%
Tree	3.5%	3.5%	1.8%	1.8%
IH	21.1%	-	-	-

	IPS	
	Standard	Per-Decision
IS	5.3%	3.5%
WIS	0.0%	**8.8**%
NAIVE	0.0%	-

Table A.6: Near-top Frequency. Properly specified, long horizon, large policy mismatch, stochastic env/rew experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	14.6%	0.0%	0.0%	8.3%
Q-Reg	4.2%	2.1%	0.0%	2.1%
MRDR	4.2%	2.1%	0.0%	0.0%
FQE	31.2%	2.1%	0.0%	**25.0**%
R(λ)	4.2%	6.2%	0.0%	0.0%
Q$^\pi$(λ)	2.1%	0.0%	0.0%	2.1%
Tree	4.2%	6.2%	0.0%	0.0%
IH	**41.7**%	-	-	-

	IPS	
	Standard	Per-Decision
IS	**25.0**%	4.2%
WIS	0.0%	0.0%
NAIVE	2.1%	-

Table A.7: Near-top Frequency. Potentially misspecified, insufficient representation experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	-	-	-	-
Q-Reg	3.9%	13.7%	**25.5**%	6.9%
MRDR	0.0%	18.6%	15.7%	5.9%
FQE	0.0%	5.9%	13.7%	24.5%
R(λ)	-	-	-	-
Q$^\pi$(λ)	-	-	-	-
Tree	-	-	-	-
IH	**6.9**%	-	-	-

	IPS	
	Standard	Per-Decision
IS	10.8%	8.8%
WIS	9.8%	**13.7**%
NAIVE	3.9%	-

Table A.8: Near-top Frequency. Potentially misspecified, sufficient representation, poor π_b estimate experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	0.0%	0.0%	0.0%	0.0%
Q-Reg	0.0%	0.0%	3.3%	0.0%
MRDR	13.3%	6.7%	0.0%	0.0%
FQE	0.0%	3.3%	6.7%	10.0%
R(λ)	16.7%	0.0%	6.7%	**20.0**%
Q$^\pi$(λ)	6.7%	0.0%	0.0%	3.3%
Tree	**20.0**%	0.0%	6.7%	6.7%
IH	0.0%	-	-	-

	IPS	
	Standard	Per-Decision
IS	**3.3**%	0.0%
WIS	0.0%	0.0%
NAIVE	0.0%	-

Table A.9: Near-top Frequency. Potentially misspecified, sufficient representation, good π_b estimate experiments.

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	0.0%	0.0%	0.0%	2.8%
Q-Reg	0.0%	0.0%	0.0%	0.0%
MRDR	0.0%	5.6%	0.0%	5.6%
FQE	**8.3**%	8.3%	**25.0**%	11.1%
R(λ)	2.8%	8.3%	8.3%	19.4%
Q$^\pi$(λ)	5.6%	5.6%	8.3%	0.0%
Tree	5.6%	8.3%	16.7%	5.6%
IH	0.0%	-	-	-

	IPS	
	Standard	Per-Decision
IS	**0.0**%	0.0%
WIS	0.0%	0.0%
NAIVE	0.0%	-

A.3 Supplementary Folklore Backup

The following tables represent the numerical support for how horizon and policy difference affect the performance of the OPE estimators when policy mismatch is held constant. Notice that the policy mismatch for table A.11 and A.12 are identical: $\left(\frac{.124573...}{.1}\right)^{100} = \left(\frac{.9}{.1}\right)^{10}$. What we see here is that despite identical policy mismatch, the longer horizon does not impact the error as much (compared to the baseline, Table A.10) as moving π_e to .9, far from .1 and keeping the horizon the same.

For the tables in this section we have $T = 10$, $N = 50$, $\pi_b(a = 0) = 0.1$, $\pi_e(a = 0) = 0.1246$, unless otherwise specified.

Table A.10: Graph, relative MSE. Dense rewards. *Baseline.*

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	1.9E-3	4.9E-3	5.0E-3	3.4E-3
Q-Reg	2.4E-3	4.3E-3	4.2E-3	4.5E-3
MRDR	5.8E-3	8.9E-3	9.4E-3	9.2E-3
FQE	1.8E-3	1.8E-3	1.8E-3	1.8E-3
R(λ)	1.8E-3	1.8E-3	1.8E-3	**1.8E-3**
Q$^\pi$(λ)	1.8E-3	1.8E-3	1.8E-3	1.8E-3
Tree	1.8E-3	1.8E-3	1.8E-3	1.8E-3
IH	**1.6E-3**	-	-	-

	IPS	
	Standard	Per-Decision
IS	**5.6E-4**	8.4E-4
WIS	1.4E-3	1.4E-3
NAIVE	6.1E-3	-

Table A.11: Graph, relative MSE. Dense rewards. $T = 100$. *Increasing horizon compared to baseline, fixed π_e.*

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	5.6E-2	5.9E-2	5.9E-2	5.3E-2
Q-Reg	3.4E-3	1.1E-1	1.2E-1	9.2E-2
MRDR	1.1E-2	2.5E-1	2.9E-1	3.1E-1
FQE	6.0E-2	6.0E-2	6.0E-2	6.0E-2
R(λ)	6.0E-2	6.0E-2	6.0E-2	6.0E-2
Q$^\pi$(λ)	6.0E-2	6.0E-2	6.0E-2	6.0E-2
Tree	3.4E-1	7.0E-3	**1.6E-3**	2.3E-3
IH	**4.7E-4**	-	-	-

	IPS	
	Standard	Per-Decision
IS	1.7E-2	2.5E-3
WIS	9.5E-4	**4.9E-4**
NAIVE	5.4E-3	-

Table A.12: Graph, relative MSE. $\pi_e(a = 0) = 0.9$. Dense rewards. *Increasing π_e compared to baseline, fixed horizon.*

	DM	Hybrid		
	Direct	DR	WDR	MAGIC
AM	6.6E-1	6.7E-1	6.6E-1	6.6E-1
Q-Reg	5.4E-1	**6.3E-1**	1.3E0	9.3E-1
MRDR	5.4E-1	7.3E-1	2.0E0	2.0E0
FQE	6.6E-1	6.6E-1	6.6E-1	6.6E-1
R(λ)	6.7E-1	6.6E-1	9.3E-1	1.0E0
Q$^\pi$(λ)	6.6E-1	6.6E-1	6.6E-1	6.6E-1
Tree	6.7E-1	6.6E-1	9.4E-1	1.0E0
IH	**1.4E-2**	-	-	-

	IPS	
	Standard	Per-Decision
IS	1.0E0	**5.4E-1**
WIS	2.0E0	9.7E-1
NAIVE	4.0E0	-

A.4 Model Selection Guidelines

For the definition of near-top frequency, see the definition in Section 3.1. For support of the guideline, see Table A.1)

Table A.13: Model Selection Guidelines.

Class	Recommendation	When to use	Prototypical env.
Direct	FQE	Stochastic env, severe policy mismatch	Graph, MC, Pix-MC
	$Q(\lambda)$	Compute non-issue, moderate policy mismatch	GW/Pix-GW
	IH	Long horizon, mild policy mismatch, good kernel	Graph-MC
IPS	PDWIS	Short horizon, mild policy mismatch	Graph
Hybrid	MAGIC FQE	Severe model misspecification	Graph-POMDP, Enduro
	MAGIC $Q(\lambda)$	Compute non-issue, severe model misspecification	Graph-POMDP

Table A.14: Model Selection Guideline Cont.

Class	Recommendation	Near-top Freq.
Direct	FQE	23.7%
	$Q(\lambda)$	15.0%
	IH	19.0%
IPS	PDWIS	4.7%
Hybrid	MAGIC FQE	30.0%
	MAGIC $Q(\lambda)$	17.3%

A.5 Methods

Below we include a description of each of the methods we tested. Let $\tilde{T} = T - 1$.

Inverse Propensity Scoring (IPS) Methods

Table A.15: IPS methods. [54, 97]

	Standard	Per-Decision
IS	$\sum_{i=1}^{N} \frac{\rho^i_{0:\tilde{T}}}{N} \sum_{t=0}^{\tilde{T}} \gamma^t r_t$	$\sum_{i=1}^{N} \sum_{t=0}^{\tilde{T}} \gamma^t \frac{\rho^i_{0:t}}{N} r_t$
WIS	$\sum_{i=1}^{N} \frac{\rho^i_{0:\tilde{T}}}{w_{0:\tilde{T}}} \sum_{t=0}^{\tilde{T}} \gamma^t r_t$	$\sum_{i=1}^{N} \sum_{t=0}^{\tilde{T}} \gamma^t \frac{\rho^i_{0:t}}{w_{0:t}} r_t$

Table A.15 shows the calculation for the four traditional IPS estimators: $V_{IS}, V_{PDIS}, V_{WIS}, V_{PDWIS}$. In addition, we include the following method as well since it is a Rao-Blackwellization [133] of the IPS estimators:

Hybrid Methods

Hybrid rely on being supplied an action-value function $\widehat{Q}$, an estimate of Q, from which one can also yield $\widehat{V}(s) = \sum_{a \in A} \pi(a|s) \widehat{Q}(s,a)$.

Doubly-Robust (DR): [201, 97]

$$V_{DR} = \frac{1}{N} \sum_{i=1}^{N} \widehat{V}(s^i_0) + \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\infty} \gamma^t \rho^i_{0:t} [r^i_t - \widehat{Q}(s^i_t, a^i_t) + \gamma \widehat{V}(s^i_{t+1})].$$

Weighted Doubly-Robust (WDR): [201]

$$V_{WDR} = \frac{1}{N} \sum_{i=1}^{N} \widehat{V}(s^i_0) + \sum_{i=1}^{N} \sum_{t=0}^{\infty} \gamma^t \frac{\rho^i_{0:t}}{w_{0:t}} [r^i_t - \widehat{Q}(s^i_t, a^i_t) + \gamma \widehat{V}(s^i_{t+1})].$$

MAGIC: [201] Given $g_J = \{g^i | i \in J \subseteq \mathbb{N} \cup \{-1\}\}$ where

$$g^j(D) = \sum_{i=1}^{N} \sum_{t=0}^{j} \gamma^t \frac{\rho^i_{0:t}}{w_{0:t}} r^i_t + \sum_{i=1}^{N} \gamma^{j+1} \frac{\rho^i_{0:t}}{w_{0:t}} \widehat{V}(s^i_{j+1}) - \sum_{i=1}^{N} \sum_{t=0}^{j} \gamma^t \left(\frac{\rho^i_{0:t}}{w_{0:t}} \widehat{Q}(s^i_t, a^i_t) - \frac{\rho^i_{0:\tilde{T}}}{w_{0:\tilde{T}}} \widehat{V}(s^i_t)\right),$$

then define $dist(y, Z) = \min_{z \in Z} |y - z|$ and

$$\widehat{b}_n(j) = dist(g^J_j(D), CI(g^\infty(D), 0.5))$$
$$\widehat{\Omega}_n(i,j) = Cov(g^J_i(D), g^J_j(D)),$$

then for a $|J|-$simplex $\Delta^{|J|}$ we can calculate

$$\widehat{x}^* \in \arg\min_{x \in \Delta^{|J|}} x^T [\widehat{\Omega}_n + \widehat{b}\widehat{b}^T] x,$$

yielding

$$V_{MAGIC} = (\widehat{x}^*)^T g_J.$$

MAGIC can be thought of as a weighted average of different blends of the DM

and Hybrid. In particular, for some $i \in J$, g^i represents estimating the first i steps of $V(\pi_e)$ according to DR (or WDR) and then estimating the remaining steps via $\widehat{Q}$. Hence, V_{MAGIC} finds the most appropriate set of weights which trades off between using a direct method and a Hybrid.

Direct Methods (DM)

Model-Based

Approximate Model (AM): [97] An approach to model-based value estimation is to directly fit the transition dynamics $P(s_{t+1}|s_t, a_t)$, reward $R(s_t, a_t)$, and terminal condition $P(s_{t+1} \in \mathcal{S}_{terminal}|s_t, a_t)$ of the MDP using some for of maximum likelihood or function approximation. This yields a simulation environment from which one can extract the value of a policy using an average over rollouts. Thus, $V(\pi) = \mathbb{E}[\sum_{t=1}^{T} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = \pi(s_0)]$ where the expectation is over initial conditions $s \sim d_0$ and the transition dynamics of the simulator.

Model-Free

Every estimator in this section will approximate Q with $\widehat{Q}(\cdot;\theta)$, parametrized by some θ. From $\widehat{Q}$ the OPE estimate we seek is

$$V = \frac{1}{N}\sum_{i=1}^{N}\sum_{a\in A}\pi_e(a|s)\widehat{Q}(s_0^i, a;\theta).$$

Note that $\mathbb{E}_{\pi_e}Q(s_{t+1},\cdot) = \sum_{a\in A}\pi_e(a|s_{t+1})Q(s_{t+1},a)$.

Direct Model Regression (Q-Reg): [60]

$$\widehat{Q}(\cdot,\theta) = \min_{\theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\tilde{T}}\gamma^t\rho_{0:t}^i\left(R_{t:\tilde{T}}^i - \widehat{Q}(s_t^i, a_t^i;\theta)\right)^2$$

$$R_{t:\tilde{T}}^i = \sum_{t'=t}^{\tilde{T}}\gamma^{t'-t}\rho_{(t+1):t'}^i r_{t'}^i$$

Fitted Q Evaluation (FQE): [123] $\widehat{Q}(\cdot,\theta) = \lim_{k\to\infty}\widehat{Q}_k$ where

$$\widehat{Q}_k = \min_{\theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\tilde{T}}(\widehat{Q}_{k-1}(s_t^i, a_t^i;\theta) - y_t^i)^2$$

$$y_t^i \equiv r_t^i + \gamma\mathbb{E}_{\pi_e}\widehat{Q}_{k-1}(s_{t+1}^i,\cdot;\theta)$$

Retrace(λ) (R(λ)), Tree-Backup (Tree), $Q^\pi(\lambda)$: [149, 165, 82]

$$\widehat{Q}(\cdot,\theta) = \lim_{k\to\infty}\widehat{Q}_k$$

where

$$\widehat{Q}_k(s,a;\theta) = \widehat{Q}_{k-1}(s,a;\theta) + \mathbb{E}_{\pi_b}[\sum_{t\geq 0}\gamma^t \prod_{k=1}^{t} c_k y_t | s_0 = s, a_0 = a]$$

and

$$y_t = r^t + \gamma \mathbb{E}_{\pi_e}\widehat{Q}_{k-1}(s_{t+1},\cdot;\theta) - \widehat{Q}_{k-1}(s_t,a_t;\theta)$$

$$c_k = \begin{cases} \lambda \min(1, \frac{\pi_e(a_k|s_k)}{\pi_b(a_k|s_k)}) & R(\lambda) \\ \lambda \pi_e(a_k|s_k) & Tree \\ \lambda & Q^\pi(\lambda) \end{cases}.$$

More Robust Doubly-Robust (MRDR): [60] Given

$$\Omega_{\pi_b}(s) = diag[1/\pi_b(a|s)]_{a\in A} - ee^T$$
$$e = [1,\ldots,1]^T$$
$$R^i_{t:\tilde{T}} = \sum_{j=t}^{\tilde{T}} \gamma^{j-t}\rho^i_{(t+1):j} r(s^i_j, a^i_j)$$

and

$$q_\theta(s,a,r) = diag[\pi_e(a'|s)]_{a'\in A}[\widehat{Q}(s,a';\theta)]_{a'\in A} - r[\mathbf{1}\{a'=a\}]_{a'\in A}$$

where $\mathbf{1}$ is the indicator function, then

$$\widehat{Q}(\cdot,\theta) = \min_\theta \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\tilde{T}} \gamma^{2t}(\rho^i_{0:\tilde{T}})^2 \times \rho^i_t q_\theta(s^i_t, a^i_t, R^i_{t:\tilde{T}})^T \Omega_{\pi_b}(s^i_t) q_\theta(s^i_t, a^i_t, R^i_{t:\tilde{T}}).$$

State Density Ratio Estimation (IH): [133]

$$V_{IH} = \sum_{i=1}^{N}\sum_{t=0}^{\tilde{T}} \frac{\gamma^t \omega(s^i_t)\rho_{t:t} r^i_t}{\sum_{i'=0}^{N}\sum_{t'=1}^{\tilde{T}} \gamma^{t'}\omega(s^{i'}_{t'})\rho_{t':t'}}$$
$$\omega(s^i_t) = \lim_{t\to\infty} \frac{\sum_{t=0}^{T}\gamma^t d_{\pi_e}(s^i_t)}{\sum_{t=0}^{T}\gamma^t d_{\pi_b}(s^i_t)},$$

where π_b is assumed to be a fixed data-generating policy, and d_π is the distribution of states when executing π from $s_0 \sim d_0$. The details for how to find ω can be found in Algorithm 1 and 2 of [133].

A.6 Environments

For every environment, we initialize the environment with a fixed horizon length T. If the agent reaches a goal before T or if the episode is not over by step T, it will transition to an environment-dependent absorbing state where it will stay until time T. For a high level description of the environment features, see Table 3.1.

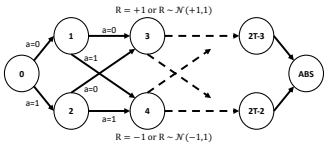


Figure A.1: Graph Environment.

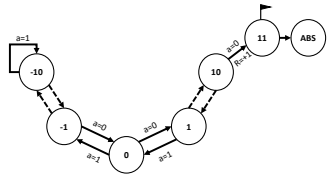


Figure A.2: Graph-MC Environment.

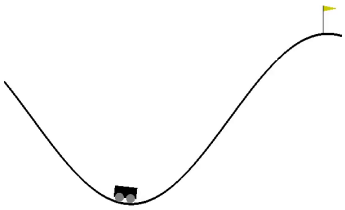


Figure A.3: MC Environment, pixel-version. The non-pixel version involves representing the state of the car as the position and velocity.

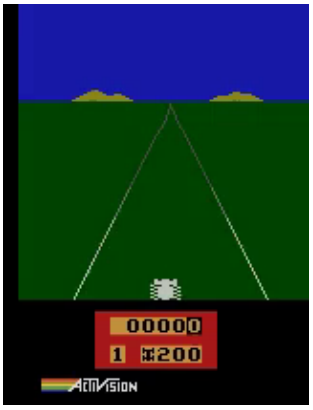


Figure A.4: Enduro Environment.

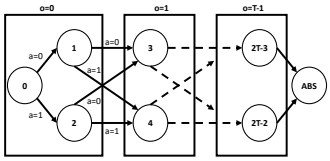


Figure A.5: Graph-POMDP Environment. Model-Fail [201] is a special case of this environment where T=2. We also extend the environment to arbitrary horizon which makes it a semi-mdp.

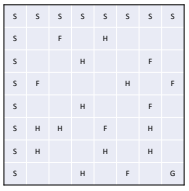


Figure A.6: Gridworld environment. Blank spaces indicate areas of a small negative reward, S indicates the starting states, F indicates a field of slightly less negative reward, H indicates a hole of severe penalty, G indicates the goal of positive reward.

Environment Descriptions

Graph

Figure A.1 shows a visualization of the Toy-Graph environment. The graph is initialized with horizon T and with absorbing state $s_{abs} = 2T$. In each episode,

the agent starts at a single starting state $s_0 = 0$ and has two actions, $a = 0$ and $a = 1$. At each time step $t < T$, the agent can enter state $s_{t+1} = 2t + 1$ by taking action $a = 0$, or $s_{t+1} = 2t + 2$ by taking action $a = 1$. If the environment is stochastic, we simulate noisy transitions by allowing the agent to slip into $s_{t+1} = 2t + 2$ instead of $s_{t+1} = 2t + 1$ and vice-versa with probability .25. At the final time $t = T$, the agent always enters the terminal state s_{abs}. The reward is $+1$ if the agent transitions to an odd state, otherwise is -1. If the environment provides sparse rewards, then $r = +1$ if s_{T-1} is odd, $r = -1$ if s_{T-1} is even, otherwise $r = 0$. Similarly to deterministic rewards, if the environment's rewards are stochastic, then the reward is $r \sim N(1, 1)$ if the agent transitions to an odd state, otherwise $r \sim N(-1, 1)$. If the rewards are sparse and stochastic then $r \sim N(1, 1)$ if s_{T-1} is odd, otherwise $r \sim N(-1, 1)$ and $r = 0$ otherwise.

Graph-POMDP

Figure A.5 shows a visualization of the Graph-POMDP environment. The underlying state structure of Graph-POMDP is exactly the Graph environment. However, the states are grouped together based on a choice of Graph-POMDP horizon length, H. This parameter groups states into H observable states. The agent only is able to observe among these states, and not the underlying MDP structure. Model-Fail [201] is a special case of this environment when $H = T = 2$.

Graph Mountain Car (Graph-MC)

Figure A.2 shows a visualization of the Toy-MC environment. This environment is a 1-D graph-based simplification of Mountain Car. The agent starts at $s_0 = 0$, the center of the valley and can go left or right. There are 21 total states, 10 to the left of the starting position and 11 to the right of the starting position, and a terminal absorbing state $s_{abs} = 22$. The agent receives a reward of $r = -1$ at every timestep. The reward becomes zero if the agent reaches the goal, which is state $x = +11$. If the agent reaches $x = -10$ and continues left then the agent remains in $x = -10$. If the agent does not reach state $x = +11$ by step T then the episode terminates and the agent transitions to the absorbing state.

Mountain Car (MC)

We use the OpenAI version of Mountain Car with a few simplifying modifications [29, 192]. The car starts in a valley and has to go back and forth to gain enough momentum to scale the mountain and reach the end goal. The state space is given by the position and velocity of the car. At each time step, the car has the following options: accelerate backwards, forwards or do nothing. The reward is $r = -1$ for every time step until the car reaches the goal. While the original trajectory length is capped at 200, we decrease the effective length by applying every action a_t five times before observing s_{t+1}. Furthermore, we modify the random initial position from being uniformly between $[-.6, -.4]$ to being one of $\{-.6, -.5, -.4\}$, with no velocity. The environment is initialized with a horizon T and absorbing state $s_{abs} = [.5, 0]$, position at .5 and no velocity.

Pixel-based Mountain Car (Pix-MC)

This environment is identical to Mountain Car except the state space has been modified from position and velocity to a pixel based representation of a ball, representing a car, rolling on a hill, see Figure A.3. Each frame f_t is a 80×120 image of the ball on the mountain. One cannot deduce velocity from a single frame, so we represent the state as $s_t = \{f_{t-1}, f_t\}$ where $f_{-1} = f_0$, the initial state. Everything else is identical between the pixel-based version and the position-velocity version described earlier.

Enduro

We use OpenAI's implementation of Enduro-v0, an Atari 2600 racing game. We downsample the image to a grayscale of size (84,84). We apply every action one time and we represent the state as $s_t = \{f_{t-3}, f_{t-2}, f_{t-1}, f_t\}$ where $f_i = f_0$, the initial state, for $i < 0$. See Figure A.4 for a visualization.

Gridworld (GW)

Figure A.6 shows a visualization of the Gridworld environment. The agent starts at a state in the first row or column (denoted S in the figure), and proceeds through the grid by taking actions, given by the four cardinal directions, for $T = 25$ timesteps. An agent remains in the same state if it chooses an action which would take it out of the environment. If the agent reaches the goal state

Table A.16: Environment parameters - Full description.

Environment	Graph	Graph-MC	MC	Pix-MC	Enduro	Graph-POMDP	GW	Pix-GW
Is MDP?	yes	yes	yes	yes	yes	no	yes	yes
State desc.	position	position	[pos, vel]	pixels	pixels	position	position	pixels
T	4 or 16	250	250	250	1000	2 or 8	25	25
Stoch Env?	variable	no	no	no	no	no	no	variable
Stoch Rew?	variable	no	no	no	no	no	no	no
Sparse Rew?	variable	terminal	terminal	terminal	dense	terminal	dense	dense
$\hat{Q}$ Func. Class	tabular	tabular	linear/NN	NN	NN	tabular	tabular	NN
Initial state	0	0	variable	variable	gray img	0	variable	variable
Absorb. state	2T	22	[.5,0]	img([.5,0])	zero img	2T	64	zero img
Frame height	1	1	2	2	4	1	1	1
Frame skip	1	1	5	5	1	1	1	1

G, in the bottom right corner of the environment, it transitions to a terminal state $x = 64$ for the remainder of the trajectory and receives a reward of $+1$. In the grid, there is a field (denoted F) which gives the agent a reward of $-.005$ and holes (denoted H) which give $-.5$. The remaining states give a reward of $-.01$.

Pixel-Gridworld (Pixel-GW)

This environment is identical to Gridworld except the state space has been modified from position to a pixel based representation of the position: 1 for the agent's location, 0 otherwise. We use the same policies as in the Gridworld case.

A.7 Experimental Setup

Description of the policies

Graph, Graph-POMDP and Graph-MC use static policies with some probability of going left and another probability of going right, ex: $\pi(a = 0) = p, \pi(a = 1) = 1 - p$, independent of state. We vary p in our experiments.

GW, Pix-GW, MC, Pixel-MC, and Enduro all use an $\epsilon-$Greedy policy. In other words, we train a policy Q^* (using value iteration or DDQN) and then vary the deviation away from the policy. Hence $\epsilon - Greedy(Q^*)$ implies we follow a mixed policy $\pi = \arg\max_a Q^*(s, a)$ with probability $1 - \epsilon$ and uniform with probability ϵ. We vary ϵ in our experiments.

Enumeration of Experiments

The set of experiments per table is the Cartesian product of the parameters in the table.

Table A.17: Graph parameters.

	Parameters
γ	.98
N	$2^{3:11}$
T	$\{4, 16\}$
$\pi_b(a=0)$	$\{.2, .6\}$
$\pi_e(a=0)$	.8
Stochastic Env	{True, False}
Stochastic Rew	{True, False}
Sparse Rew	{True, False}
Seed	{10 random}
ModelType	Tabular
Regress π_b	False

Table A.18: Graph-POMDP parameters.

	Parameters
γ	.98
N	$2^{8:11}$
(T,H)	$\{(2, 2), (16, 6)\}$
$\pi_b(a=0)$	$\{.2, .6\}$
$\pi_e(a=0)$	.8
Stochastic Env	{True, False}
Stochastic Rew	{True, False}
Sparse Rew	{True, False}
Seed	{10 random}
ModelType	Tabular
Regress π_b	False

Table A.19: Gridworld parameters.

	Parameters
γ	.98
N	$2^{6:11}$
T	25
$\epsilon-$ Greedy, π_b	$\{.2, .4, .6, .8, 1.\}$
$\epsilon-$ Greedy, π_e	.1
Stochastic Env	False
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	Tabular
Regress π_b	True

Table A.20: Pix-GW parameters.

	Parameters
γ	.96
N	$2^{6:9}$
T	25
$\epsilon-$ Greedy, π_b	$\{.2, .4, .6, .8, 1.\}$
$\epsilon-$ Greedy, π_e	.1
Stochastic Env	{True, False}
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	NN
Regress π_b	{True, False}

Table A.21: Graph-MC parameters.

	Parameters
γ	.99
N	$2^{7:11}$
T	250
$(\pi_b(a=0), \pi_e(a=0))$	$\{(.45, .45), (.6, .6), (.45.6)$ $(.6, .45), (.8, .2), (.2, .8)\}$
Stochastic Env	False
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	Tabular
Regress π_b	False

Table A.22: MC parameters.

	Parameters
γ	.99
N	$2^{7:10}$
T	250
$\epsilon-$ Greedy, (π_b, π_e)	$\{(.1, 0), (1, 0)$ $(1, .1), (.1, 1)\}$
Stochastic Env	False
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	{Tabular, NN}
Regress π_b	False

Table A.23: Pix-MC parameters.

	Parameters
γ	.97
N	512
T	500
$\epsilon-$Greedy, (π_b, π_e)	$\{(.25, 0), (.1, 0)$ $(.25, .1)\}$
Stochastic Env	False
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	{Tabular, NN}
Regress π_b	False

Table A.24: Enduro parameters.

	Parameters
γ	.9999
N	512
T	500
$\epsilon-$Greedy, (π_b, π_e)	$\{(.25, 0), (.1, 0)$ $(.25, .1)\}$
Stochastic Env	False
Stochastic Rew	False
Sparse Rew	False
Seed	{10 random}
ModelType	{Tabular, NN}
Regress π_b	False

Representation and Function Class

For the simpler environments, we use a tabular representation for all the methods. AM amounts to solving for the transition dynamics, rewards, terminal state, etc. through maximum likelihood. FQE, Retrace(λ), $Q^\pi(\lambda)$, and Tree-Backup are all implemented through dynamics programming with Q tables. MRDR and Q-Reg used the Sherman Morrison [183] method to solve the weighted-least square problem, using a basis which spans a table.

In the cases where we needed function approximation, we did not directly fit the dynamics for AM; instead, we fit on the difference in states $P(s' - s|s, a)$, which is common practice.

For the MC environment, we ran experiments with both a linear and NN function class. In both cases, the representation of the state was not changed and remained [position, velocity]. The NN architecture was dense with [16,8,4,2] as the layers. The layers had relu activations (except the last, with a linear activation) and were all initialized with truncated normal centered at 0 with a standard deviation of 0.1.

For the pixel-based environments (MC, Enduro), we use a convolutional NN. The architechure is a layer of size 8 with filter (7,7) and stride 3, followed by maxpooling and a layer of size 16 with filter (3,3) and stride 1, followed by max pooling, flattening and a dense layer of size 256. The final layer is a dense layer with the size of the action space, with a linear activation. The layers had elu activations and were all initialized with truncated normal centered at 0 with a standard deviation of 0.1. The layers also have kernel L2 regularizers with weight 1e-6.

When using NNs for the IH method, we used the radial-basis function and a shallow dense network for the kernel and density estimate respectively.

Datasets

Datasets are not included as part of COBS since our benchmark is completely simulation based. To recreate any dataset, select the appropriate choice of environment parameters from the experiments enumerated in Section A.7.

Choice of hyperparameters.

Many methods require selection of convergence criteria, regularization parameters, batch sizes, and a whole host of other hyperparameters. Often there is a trade-off between computational cost and the accuracy of the method. Hyperparameter search is not feasible in OPE since there is no proper validation (like game score in learning). See Tables A.7, A.8 for a list of hyperparameters that were chosen for the experiments.

Figure A.7: Hyperparameters for each model by Environment.

Method	Parameter	Graph	TMC	MC	Pix-MC	Enduro	Graph-POMDP	GW	Pix-GW
AM	Max Traj Len	T	T	50	50	-	T	T	T
	NN Fit Epochs	-	-	100	100	-	-	-	100
	NN Batchsize	-	-	32	32	-	-	-	25
	NN Train size	-	-	.8	.8	-	-	-	.8
	NN Val size	-	-	.2	.2	-	-	-	.2
	NN Early Stop delta	-	-	1e-4	1e-4	-	-	-	1e-4
Q-Reg	Omega regul.	1	1	-	-	-	1	1	-
	NN Fit Epochs	-	-	80	80	80	-	-	80
	NN Batchsize	-	-	32	32	32	-	-	32
	NN Train size	-	-	.8	.8	.8	-	-	.8
	NN Val size	-	-	.2	.2	.2	-	-	.2
	NN Early Stop delta	-	-	1e-4	1e-4	1e-4	-	-	1e-4
FQE	Convergence ϵ	1e-5	1e-5	1e-4	1e-4	1e-4	1e-5	4e-4	1e-4
	Max Iter	-	-	160	160	600	-	50	80
	NN Batchsize	-	-	32	32	32	-	-	32
	Optimizer Clipnorm	-	-	1.	1.	1.	-	-	1.
IH	Quad. prog. regular.	1e-3	1e-3	-	-	-	1e-3	1e-3	-
	NN Fit Epochs	-	-	10001	10001	10001	-	-	1001
	NN Batchsize	-	-	1024	128	128	-	-	128

Figure A.8: Hyperparameters for each model by Environment, Cont.

Method	Parameter	Graph	TMC	MC	Pix-MC	Enduro	Graph-POMDP	GW	Pix-GW
MRDR	Omega regul.	1	1	-	-	-	1	1	-
	NN Fit Epochs	-	-	80	80	80	-	-	80
	NN Batchsize	-	-	1024	1024	1024	-	-	32
	NN Train size	-	-	.8	.8	.8	-	-	.8
	NN Val size	-	-	.2	.2	.2	-	-	.2
	NN Early Stop delta	-	-	1e-4	1e-4	1e-4	-	-	1e-4
$R(\lambda)$	λ	.9	.9	.9	-	-	.9	.9	.9
	Convergence ϵ	1e-3	2e-3	1e-3	-	-	1e-3	2e-3	1e-3
	Max Iter	500	500	-	-	-	500	50	-
	NN Fit Epochs	-	-	80	-	-	-	-	80
	NN Batchsize	-	-	4	-	-	-	-	4
	NN Train Size	-	-	.03	-	-	-	-	.03
	NN ClipNorm	-	-	1.	-	-	-	-	1.
$Q^\pi(\lambda)$	λ	.9	.9	.9	-	-	.9	.9	.9
	Convergence ϵ	1e-3	2e-3	1e-3	-	-	1e-3	2e-3	1e-3
	Max Iter	500	500	-	-	-	500	50	-
	NN Fit Epochs	-	-	80	-	-	-	-	80
	NN Batchsize	-	-	4	-	-	-	-	4
	NN Train Size	-	-	.03	-	-	-	-	.03
	NN ClipNorm	-	-	1.	-	-	-	-	1.
Tree	λ	.9	.9	.9	-	-	.9	.9	.9
	Convergence ϵ	1e-3	2e-3	1e-3	-	-	1e-3	2e-3	1e-3
	Max Iter	500	500	-	-	-	500	50	-
	NN Fit Epochs	-	-	80	-	-	-	-	80
	NN Batchsize	-	-	4	-	-	-	-	4
	NN Train Size	-	-	.03	-	-	-	-	.03
	NN ClipNorm	-	-	1.	-	-	-	-	1.

A.8 Additional Supporting Figures

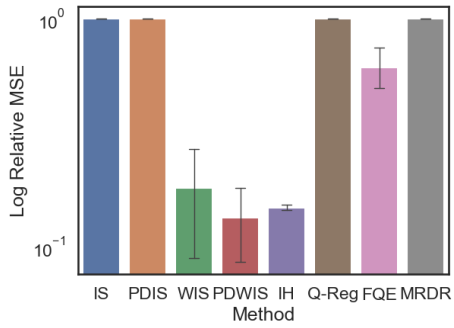


Figure A.9: Enduro DM vs IPS. π_b is a policy that deviates uniformly from a trained policy 25% of the time, π_e is a policy trained with DDQN. *IH has relatively low error mainly due to tracking the simple average, since the kernel function did not learn useful density ratio. The computational time required to calculate the multi-step rollouts of AM, Retrace(λ), $Q^\pi(\lambda)$, Tree-Backup(λ) exceeded our compute budget and were thus excluded.*

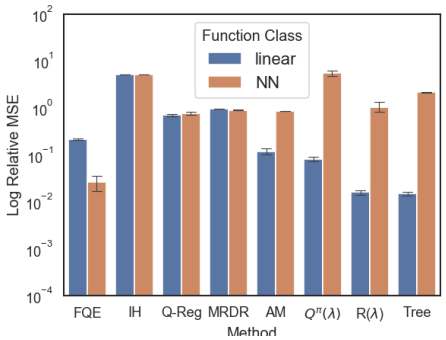


Figure A.10: MC comparison. $N = 256$. π_b is a uniform random policy, π_e is a policy trained with DDQN.

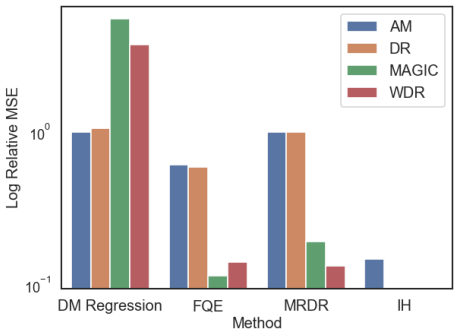


Figure A.11: Enduro DM vs HM. π_b is a policy that deviates uniformly from a trained policy 25% of the time, π_e is a policy trained with DDQN.

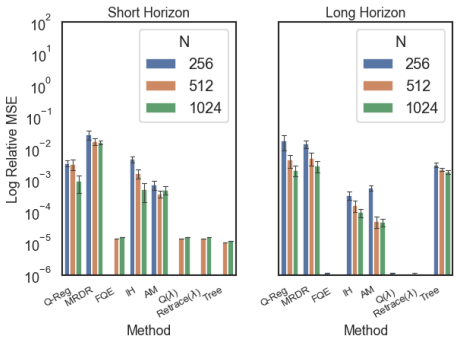


Figure A.12: Comparison of Direct methods' performance across horizon and number of trajectories in the Graph environment. Small policy mismatch under a deterministic environment.

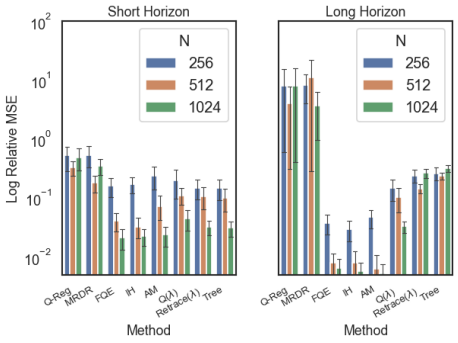


Figure A.13: *(Graph domain) Comparing DMs across horizon length and number of trajectories. Large policy mismatch and a stochastic environment setting.*

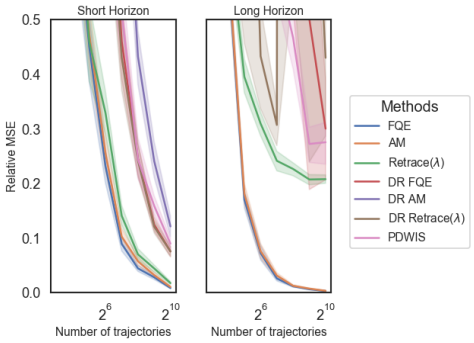


Figure A.14: Comparing DM to DR in a stochastic environment with large policy mismatch. (Graph)

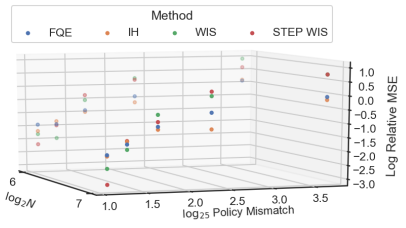


Figure A.15: Comparison between FQE, IH, and WIS in a low data regime. For low policy mismatch, IPS is competitive to DM in low data, but as the policy mismatch grows, the top DM outperform. Experiments ran in the Gridworld Environment.

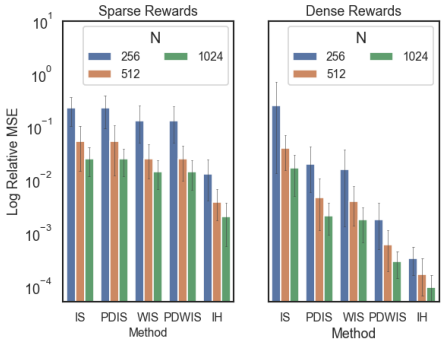


Figure A.16: Comparison between IPS methods and IH with dense vs sparse rewards. Per-Decision IPS methods see substantial improvement when the rewards are dense. Experiments ran in the Graph environment with $\pi(a = 0) = .6, \pi_e(a = 0) = .8$

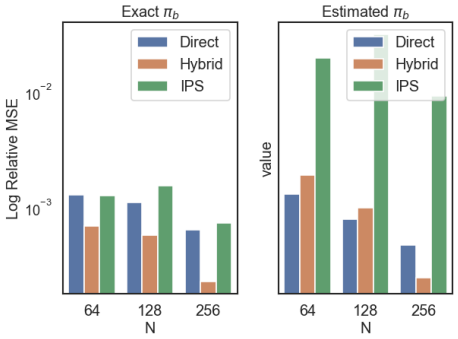


Figure A.17: Exact vs Estimated π_b. Exact π_b = .2−Greedy, π_e = .1−Greedy. Min error per class. (Pixel Gridworld, deterministic)

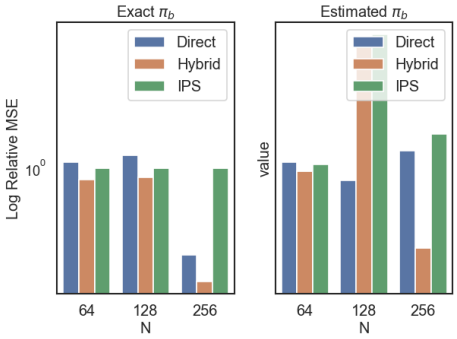


Figure A.18: Exact vs Estimated π_b. Exact π_b =uniform, π_e = .1−Greedy. Min error per class. (Pixel Gridworld, deterministic)

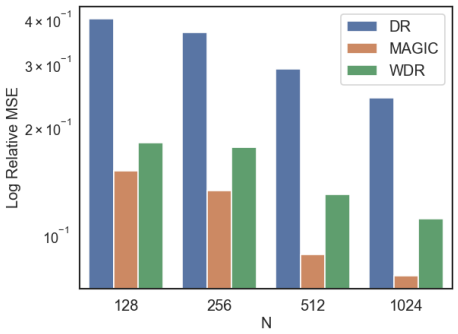


Figure A.19: Hybrid Method comparison. $\pi_b(a=0) = .2, \pi_e(a=0) = .8$. Min error per class. (Graph-MC)

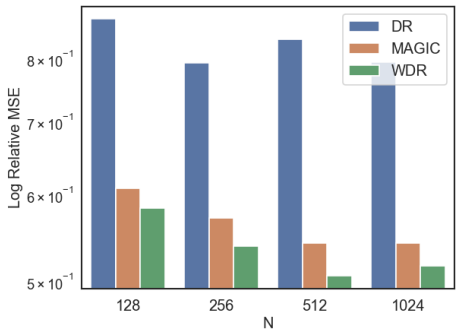


Figure A.20: Hybrid Method comparison. $\pi_b(a=0) = .8, \pi_e(a=0) = .2$. Min error per class. (Graph-MC)

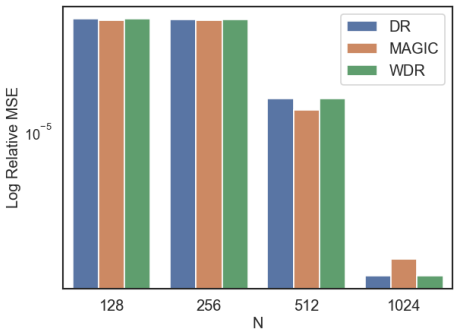


Figure A.21: Hybrid Method comparison. $\pi_b(a=0) = .6, \pi_e(a=0) = .6$. Min error per class. (Graph-MC)

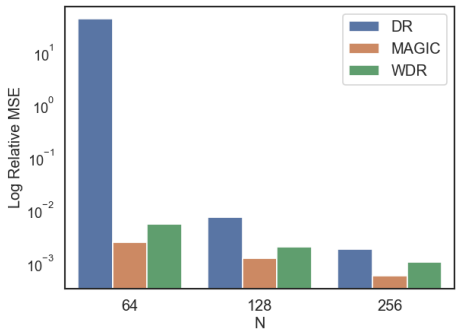


Figure A.22: Hybrid Method comparison. Exact $\pi_b = .2-$Greedy, $\pi_e = .1-$Greedy. Min error per class. (Pixel Gridworld)

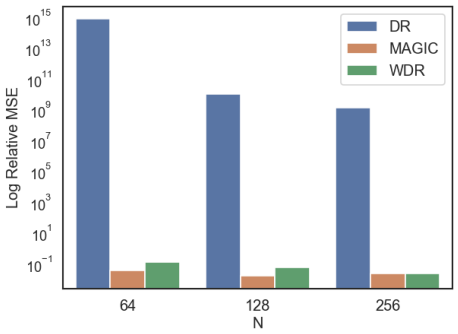


Figure A.23: Hybrid Method comparison. $\pi_b = .8-$Greedy(optimal), $\pi_e = .1-$Greedy(optimal). Min error per class. (Pixel Gridworld)

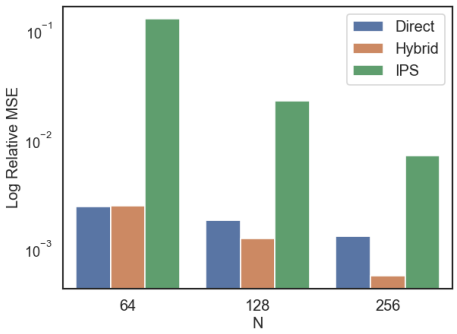


Figure A.24: Class comparison with unknown π_b. At first, HM underperform DM because π_b is more difficult to calculate leading to imprecise importance sampling estimates. Exact $\pi_b = .2-$Greedy(optimal), $\pi_e = .1-$Greedy(optimal). Min error per class. (Pixel Gridworld, stochastic env with .2 slippage)

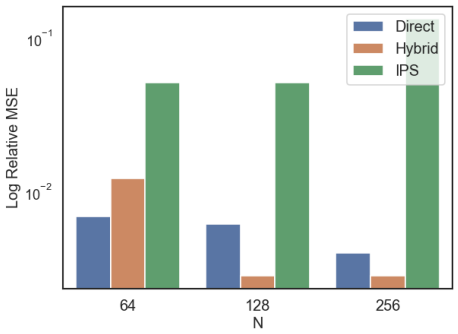


Figure A.25: Class comparison with unknown π_b. At first, HM underperform DM because π_b is more difficult to calculate leading to imprecise importance sampling estimates. Exact $\pi_b = .6-$Greedy(optimal), $\pi_e = .1-$Greedy(optimal). Min error per class. (Pixel Gridworld, stochastic env with .2 slippage)

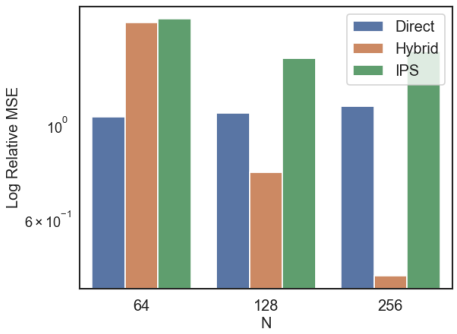


Figure A.26: Class comparison with unknown π_b. At first, HM underperform DM because π_b is more difficult to calculate leading to imprecise importance sampling estimates. Exact π_b =uniform, $\pi_e = .1-$Greedy(optimal). Min error per class. (Pixel Gridworld, stochastic env with .2 slippage)

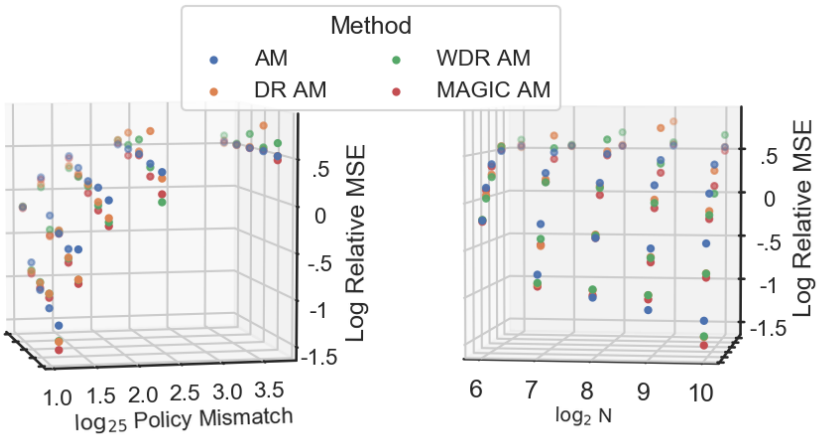


Figure A.27: AM Direct vs Hybrid comparison for AM. (Gridworld)

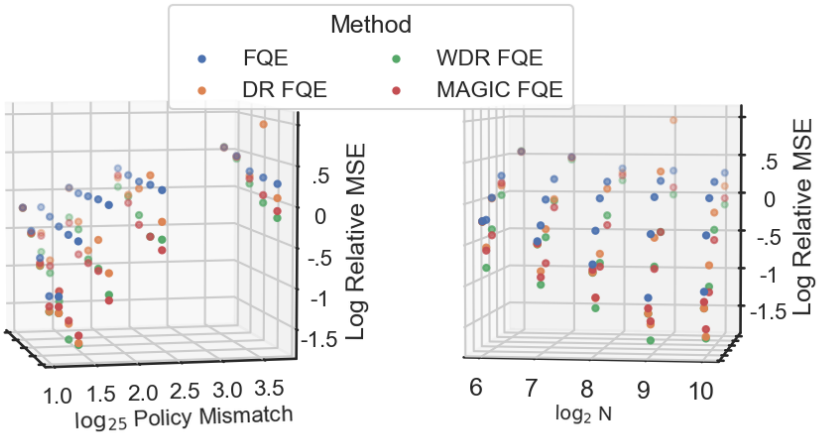


Figure A.28: FQE Direct vs Hybrid comparison. (Gridworld)

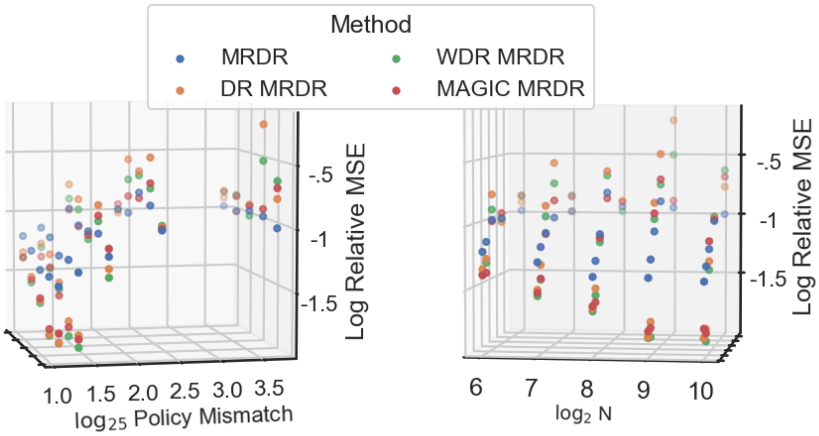


Figure A.29: MRDR Direct vs Hybrid comparison. (Gridworld)

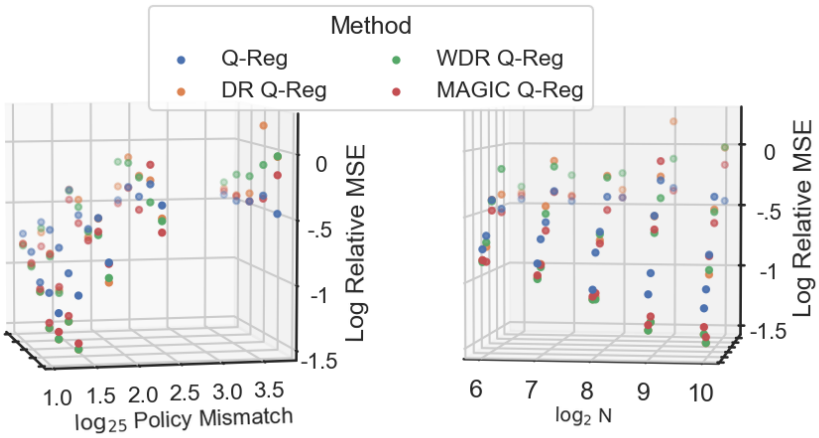


Figure A.30: Q-Reg Direct vs Hybrid comparison. (Gridworld)

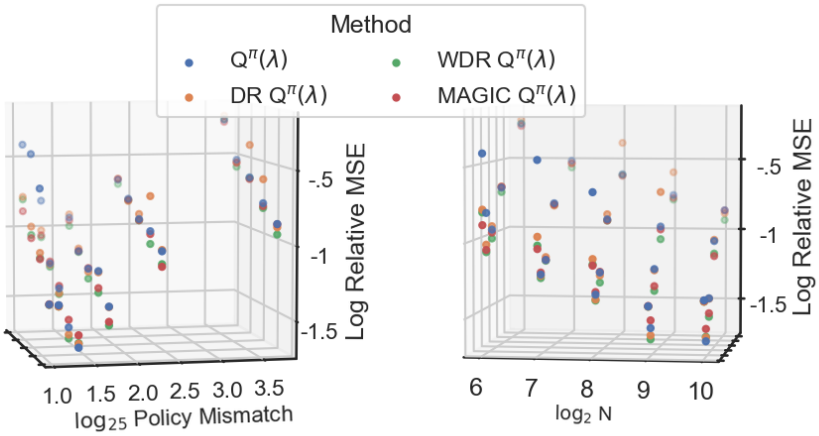


Figure A.31: $Q^{\pi}(\lambda)$ Direct vs Hybrid comparison. (Gridworld)

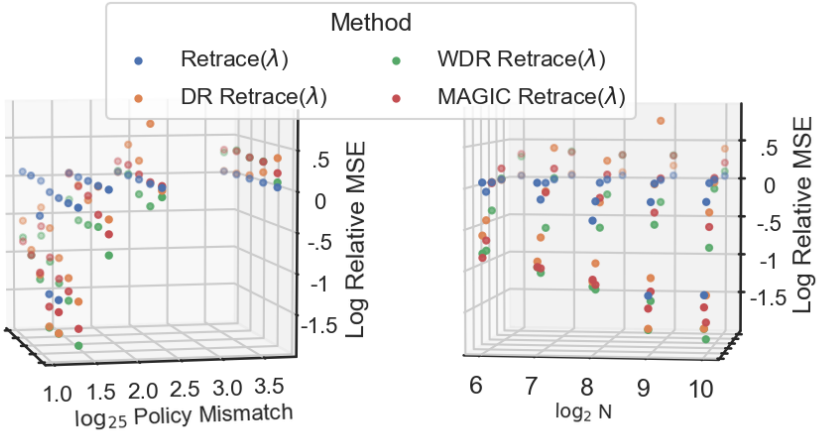


Figure A.32: Retrace(λ) Direct vs Hybrid comparison. (Gridworld)

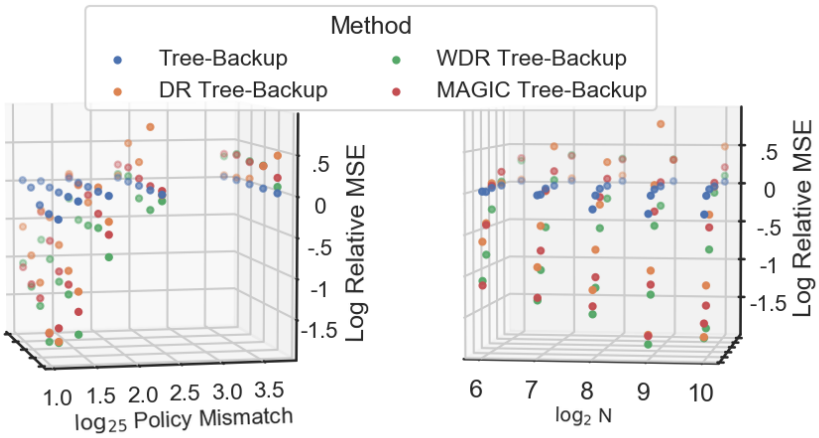


Figure A.33: Tree-Backup Direct vs Hybrid comparison. (Gridworld)

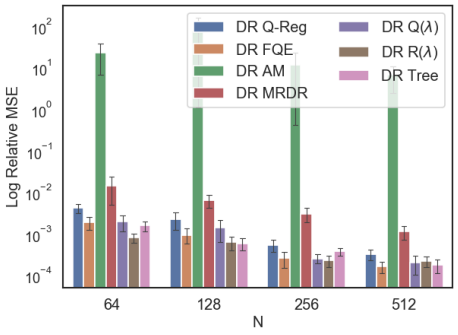


Figure A.34: DR comparison with π_b = .2−Greedy(optimal), π_e = 1.−Greedy(optimal). (Pixel Gridworld)

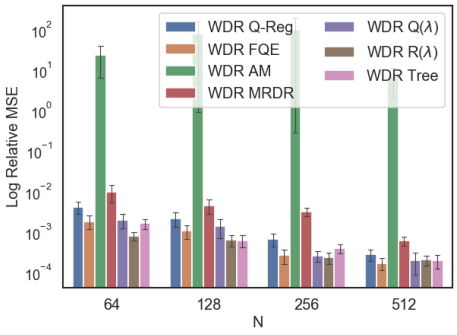


Figure A.35: WDR comparison with π_b = .2−Greedy(optimal), π_e = 1.−Greedy(optimal). (Pixel Gridworld)

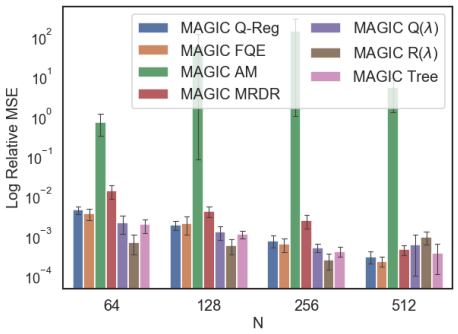


Figure A.36: MAGIC comparison with π_b = .2−Greedy(optimal), π_e = 1.−Greedy(optimal). (Pixel Gridworld)

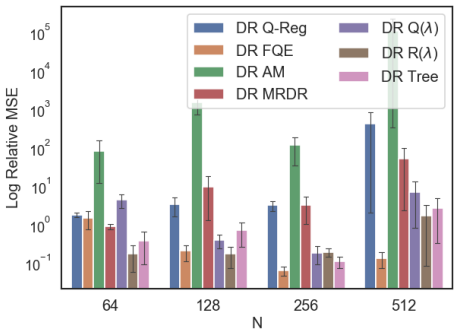


Figure A.37: DR comparison with π_b = .8−Greedy(optimal), π_e = 1.−Greedy(optimal). (Pixel Gridworld)

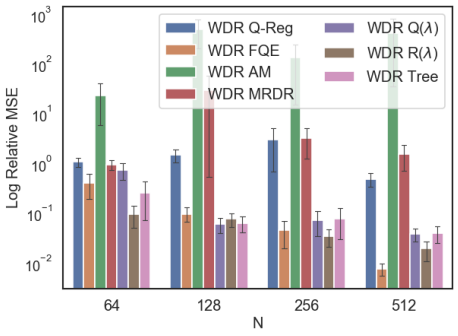


Figure A.38: WDR comparison with $\pi_b = .8-$Greedy(optimal), $\pi_e = 1.-$Greedy(optimal). (Pixel Grid-world)

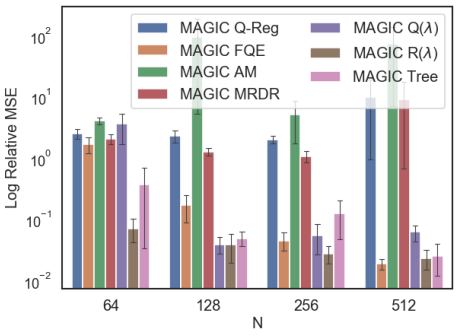


Figure A.39: MAGIC comparison with $\pi_b = .8-$Greedy(optimal), $\pi_e = 1.-$Greedy(optimal). (Pixel Grid-world)

Appendix B

CHAPTER 2 APPENDIX

Acronym	Term
OPE	Off Policy (Policy) Evaluation
OPO	Off Policy (Policy) Optimization. Also goes by batch off-policy reinforcement learning.
$\mathcal{S}$	State Space
$\mathcal{A}$	Action Space
P	Transition Function
P^*	True Transition Function
$\mathcal{R}$	Reward Function
$\mathcal{X}$	State-Action Space $\mathcal{S} \times \mathcal{A}$
γ	Discount Factor
π	Policy
$J(\pi, P)$	Performance of π in P
V_π^P	Value Function of π with respect to P
d_0	Initial State Distribution
$d_\pi^{P,\gamma}$	(Discounted) Distribution of State-Action Pairs Induced by Running π in P
w_π^P	Distribution Shift ($w_\pi^P(s,a) = \frac{d_\pi^{P,\gamma}(s,a)}{D_{\pi_b}(s,a)}$)
ν	Lebesgue measure
d_{π_b}	Behavior state distribution
π_b	Behavior policy
D_{π_b}	Behavior data ($d_{\pi_b}\pi_b$)
D	Dataset containing samples from $D_{\pi_b}P^*$
$E_n[\cdot]$	Empirical approximation using D
$E[\cdot]$	Exact expectation
$\mathcal{W}$	Distribution Shifts Function Class (e.g. $\frac{d_\pi^P(s,a)}{D_\pi(s,a)}$)
$\mathcal{V}$	Value Function Class (e.g. $V_\pi^P \in \mathcal{V}$)
$\mathcal{P}$	Model Function Class (e.g. $P \in \mathcal{P}$)
$\mathcal{L}$	Model Learning Loss Function
$\widehat{P}$	Best Model w.r.t $\mathcal{L}$
$\epsilon_{\mathcal{H}}$	Misspecification Error
π_P^*	Optimal Policy in P
RKHS	Reproducing Kernel Hilbert Space
LQR	Linear Quadratic Regulator
IP	Inverted Pendulum
MML	Minimax Model Learning (Ours)
MLE	Maximum Likelihood Estimation
VAML	Value-Aware Model Learning

B.1 OPE

In this section we explore the OPE results in the order in which they were presented in the chapter.

Main Result

Proof for Theorem 4.3.1. Assume $(w_\pi^{P^*}, V_\pi^P) \in \mathcal{W} \times \mathcal{V}$. Fix some $P \in \mathcal{P}$. We use both definitions of J as follows:

$$
\begin{aligned}
&J(\pi, P) - J(\pi, P^*) \\
&= E_{d_0}[V_\pi^P] - E_{(s,a)\sim d_{\pi,\gamma}^{P^*}, r\sim\mathcal{R}(\cdot|s,a)}[r] \\
&= E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]] + E_{d_0}[V_\pi^P] - E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s)] \\
&= E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[V_\pi^P(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]] - \sum_{t=1}^{\infty}\gamma^t \int d_{\pi,t}^{P^*}(s,a) V_\pi^P(s) d\nu(s,a) \\
&= E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[\gamma E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')]] - \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t+1}^{P^*}(s,a) V_\pi^P(s) d\nu(s,a) \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')]] - \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t}^{P^*}(\tilde{s},\tilde{a}) P^*(s|\tilde{s},\tilde{a})\pi(a|s) V_\pi^P(s) d\nu \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')]] - \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t}^{P^*}(s,a) P^*(s'|s,a) V_\pi^P(s') d\nu \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')]] - \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P^*(\cdot|s,a)}[V_\pi^P(s')]] \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^{P^*}}[E_{s'\sim P(\cdot|s,a)}[V_\pi^P(s')] - E_{s'\sim P^*(\cdot|s,a)}[V_\pi^P(s')]] \\
&= \gamma E_{(s,a,s')\sim D_{\pi_b}P^*(\cdot|s,a)}[\frac{d_{\pi,\gamma}^{P^*}(s,a)}{D_{\pi_b}(s,a)}\left(E_{x\sim P(\cdot|s,a)}[V_\pi^P(x)] - V_\pi^P(s')\right)]] \\
&= \gamma E_{(s,a,s')\sim D_{\pi_b}P^*(\cdot|s,a)}[w_\pi^{P^*}(s,a)\left(E_{x\sim P(\cdot|s,a)}[V_\pi^P(x)] - V_\pi^P(s')\right)]] \\
&= \gamma\mathcal{L}(w_\pi^{P^*}, V_\pi^P, P),
\end{aligned}
$$

where the first equality is definition. The second equality is addition of 0. The third equality is simplification. The fourth equality is change of bounds. The fifth is definition. The sixth is relabeling of the integration variables. The seventh and eighth are simplification. The ninth is importance sampling. The tenth and last is definition. Since $(w_\pi^{P^*}, V_\pi^P) \in \mathcal{W} \times \mathcal{V}$ then

$$
\begin{aligned}
|J(\pi, P) - J(\pi, P^*)| &= \gamma|\mathcal{L}(w_\pi^{P^*}, V_\pi^P, P)| \\
&\leq \gamma \max_{w\in\mathcal{W}, V\in\mathcal{V}} |\mathcal{L}(w, V, P)| \leq \gamma \min_{P\in\mathcal{P}} \max_{w\in\mathcal{W}, V\in\mathcal{V}} |\mathcal{L}(w, V, P)|,
\end{aligned}
$$

where the last inequality holds because P was selected in $\mathcal{P}$ arbitrarily.

Now, instead, assume $(w_\pi^P, V_\pi^{P^*}) \in \mathcal{W} \times \mathcal{V}$. Fix some $P \in \mathcal{P}$. Then, similarly,

$$
\begin{aligned}
&J(\pi, P) - J(\pi, P^*) = E_{(s,a)\sim d_{\pi,\gamma}^P, r\sim\mathcal{R}(\cdot|s,a)}[r] - E_{d_0}[V_\pi^{P^*}] \\
&= E_{(s,a)\sim d_{\pi,\gamma}^P}[V_\pi^{P^*}(s)] - E_{d_0}[V_\pi^{P^*}] - E_{(s,a)\sim d_{\pi,\gamma}^P}[V_\pi^{P^*}(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]] \\
&= \sum_{t=1}^{\infty}\gamma^t \int d_{\pi,t}^P(s,a) V_\pi^{P^*}(s) d\nu(s,a) - E_{(s,a)\sim d_{\pi,\gamma}^P}[V_\pi^{P^*}(s) - E_{r\sim\mathcal{R}(\cdot|s,a)}[r]] \\
&= \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t+1}^P(s,a) V_\pi^{P^*}(s) d\nu(s,a) - E_{(s,a)\sim d_{\pi,\gamma}^P}[\gamma E_{s'\sim P^*(\cdot|s,a)}[V_\pi^{P^*}(s')]] \\
&= \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t}^P(\tilde{s},\tilde{a}) P(s|\tilde{s},\tilde{a})\pi(a|s) V_\pi^{P^*}(s) d\nu - \gamma E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{s'\sim P^*(\cdot|s,a)}[V_\pi^{P^*}(s')]] \\
&= \gamma\sum_{t=0}^{\infty}\gamma^t \int d_{\pi,t}^P(s,a) P(s'|s,a) V_\pi^{P^*}(s') d\nu - \gamma E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{s'\sim P^*(\cdot|s,a)}[V_\pi^{P^*}(s')]] \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{s'\sim P(\cdot|s,a)}[V_\pi^{P^*}(s')]] - \gamma E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{s'\sim P^*(\cdot|s,a)}[V_\pi^{P^*}(s')]] \\
&= \gamma E_{(s,a)\sim d_{\pi,\gamma}^P}[E_{s'\sim P(\cdot|s,a)}[V_\pi^{P^*}(s')] - E_{s'\sim P^*(\cdot|s,a)}[V_\pi^{P^*}(s')]] \\
&= \gamma E_{(s,a,s')\sim D_{\pi_b}P^*(\cdot|s,a)}[\frac{d_{\pi,\gamma}^P(s,a)}{D_{\pi_b}(s,a)}\Big(E_{x\sim P(\cdot|s,a)}[V_\pi^{P^*}(x)] - V_\pi^{P^*}(s')\Big)]] \\
&= \gamma E_{(s,a,s')\sim D_{\pi_b}P^*(\cdot|s,a)}[w_\pi^P(s,a)\Big(E_{x\sim P(\cdot|s,a)}[V_\pi^{P^*}(x)] - V_\pi^{P^*}(s')\Big)]] \\
&= \gamma\mathcal{L}(w_\pi^P, V_\pi^{P^*}, P),
\end{aligned}
$$

where we follow the same steps as in the previous derivation. Since $(w_\pi^P, V_\pi^{P^*}) \in \mathcal{W} \times \mathcal{V}$ then

$$
\begin{aligned}
|J(\pi, P) - J(\pi, P^*)| &= \gamma|\mathcal{L}(w_\pi^P, V_\pi^{P^*}, P)| \\
&\leq \gamma \max_{w\in\mathcal{W}, V\in\mathcal{V}} |\mathcal{L}(w, V, P)| \leq \gamma \min_{P\in\mathcal{P}} \max_{w\in\mathcal{W}, V\in\mathcal{V}} |\mathcal{L}(w, V, P)|,
\end{aligned}
$$

where the last inequality holds because P was selected in $\mathcal{P}$ arbitrarily. □

Sample Complexity for OPE

We do not have access to exact expectations, so we must work with $\widehat{P}_n = \arg\min_P \max_{w,V} E_n[\ldots]$ instead of $\widehat{P} = \arg\min_P \max_{w,V} E[\ldots]$. Furthermore, $J(\pi, \widehat{P})$ requires exact expectation of an infinite sum: $E_{d_0}[\sum_{t=0}^{\infty}\gamma^t r_t]$ where we collect r_t by running π in simulation $\widehat{P}$. Instead, we can only estimate an empirical average over a finite sum in $\widehat{P}_n$: $J_{T,m}(\pi, \widehat{P}_n) = \frac{1}{m}\sum_{j=1}^{m}\sum_{t=0}^{T}\gamma^t r_t^j$, where each j indexes rollouts starting from $s_0 \sim d_0$ and the simulation is over $\widehat{P}_n$. Our OPE estimate is therefore bounded as follows:

Theorem B.1.1. *[OPE Error] Let the functions in $\mathcal{V}$ and $\mathcal{W}$ be uniformly bounded by $C_\mathcal{V}$ and $C_\mathcal{W}$ respectively. Assume the conditions of Theorem 4.3.1*

hold and $|\mathcal{R}| \leq \bar{R}, \gamma \in [0,1)$. *Then, with probability* $1-\delta$,

$$|J_{T,m}(\pi, \widehat{P}_n) - J(\pi, P^*)| \leq \gamma \min_P \max_{w,V} |\mathcal{L}(w, V, P)|$$
$$+ 4\gamma \mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P}) + \frac{2\bar{R}}{1-\gamma}\gamma^{T+1}$$
$$+ \frac{2\bar{R}}{1-\gamma}\sqrt{\log(2/\delta)/(2m)} + 4\gamma C_{\mathcal{W}} C_{\mathcal{V}} \sqrt{\log(2/\delta)/n},$$

where $\mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P})$ *is the Rademacher complexity of the function class*

$$\{(s,a,s') \mapsto w(s,a)(E_{x\sim P}[V(x)] - V(s')) :$$
$$w \in \mathcal{W}, V \in \mathcal{V}, P \in \mathcal{P}\}.$$

Proof for Theorem B.1.1. By definition and triangle inequality,

$$|J_{T,m}(\pi, \widehat{P}_n) - J(\pi, P^*)| = |J_{T,m}(\pi, \widehat{P}_n) - J(\pi, \widehat{P}_n) + J(\pi, \widehat{P}_n) - J(\pi, P^*)|$$
$$\leq \underbrace{|J_{T,m}(\pi, \widehat{P}_n) - J(\pi, \widehat{P}_n)|}_{(a)} + \underbrace{|J(\pi, \widehat{P}_n) - J(\pi, P^*)|}_{(b)}. \tag{B.1}$$

Define $\widehat{V}^P_{\pi,T}(s_0^i) \equiv \sum_{t=0}^T \gamma^t r_t^i$ for some trajectory indexed by $i \in \mathbb{N}$ where r_t^i is the reward obtained by running π in P at time $t \leq T$ starting at s_0^i. For (a),

$$|J_{T,m}(\pi, \widehat{P}_n) - J(\pi, \widehat{P}_n)|$$
$$= \left|\frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,T}(s_0^i) - \frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,\infty}(s_0^i) + \frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,\infty}(s_0^i) - E_{d_0}[V_\pi^{\widehat{P}_n}]\right|$$
$$\leq \left|\frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,T}(s_0^i) - \frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,\infty}(s_0^i)\right| + \left|\frac{1}{m}\sum_{i=1}^m \widehat{V}^{\widehat{P}_n}_{\pi,\infty}(s_0^i) - E_{d_0}[V_\pi^{\widehat{P}_n}]\right|$$
$$\leq \frac{2\bar{R}}{1-\gamma}\gamma^{T+1} + \frac{2\bar{R}}{1-\gamma}\sqrt{\log(2/\delta)/(2m)}, \tag{B.2}$$

with probability $1-\delta$, where the last inequality is definition of $\widehat{V}_{\pi,T}$ and Hoeffding's inequality.

For (b), by Theorem 4.3.1,

$$|J(\pi, \widehat{P}_n) - J(\pi, P^*)| = \gamma |L(w_\pi^{P^*}, V^{\widehat{P}_n}, \widehat{P}_n)| \leq \gamma \max_{w,V} |L(w, V, \widehat{P}_n)|$$
$$= \gamma(\max_{w,V} |L(w, V, \widehat{P}_n)| - \max_{w,V} |L_n(w, V, \widehat{P}_n)| + \max_{w,V} |L_n(w, V, \widehat{P}_n)|$$
$$- \max_{w,V} |L(w, V, \widehat{P})| + \max_{w,V} |L(w, V, \widehat{P})|)$$
$$\leq \gamma(2\max_{w,V,P} ||L(w, V, P)| - |L_n(w, V, P)|| + \min_P \max_{w,V} |L(w, V, P)|)$$
$$\leq \gamma(2\mathfrak{R}'_n(\mathcal{W}, \mathcal{V}, \mathcal{P}) + 2K\sqrt{\log(2/\delta)/n} + \min_P \max_{w,V} |L(w, V, P)|)$$
$$\leq \gamma(4\mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P}) + 2K\sqrt{\log(2/\delta)/n} + \min_P \max_{w,V} |L(w, V, P)|), \tag{B.3}$$

where $\mathfrak{R}'_n(\mathcal{W}, \mathcal{V}, \mathcal{P})$ is the Rademacher complexity of the function class

$$\{(s, a, s') \mapsto |w(s, a)(E_{x\sim P}[V(x)] - V(s'))| : w \in \mathcal{W}, V \in \mathcal{V}, P \in \mathcal{P}\},$$

noting that $K = 2C_wC_V$ uniformly bounds $|w(s, a)(E_{x\sim P(\cdot|s,a)}[V(x)] - V(s'))|$ (Theorem 8 [19]). Furthermore since absolute value is 1-Lipshitz (by reverse triangle ineq), then $\mathfrak{R}'_n < 2\mathfrak{R}_n$ (Theorem 12 [19]) where $\mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P})$ is the Rademacher complexity of the function class

$$\{(s, a, s') \mapsto w(s, a)(E_{x\sim P(\cdot|s,a)}[V(x)] - V(s'))) : w \in \mathcal{W}, V \in \mathcal{V}, P \in \mathcal{P}\}.$$

Altogether, combining (1), (2), (3) we get our result. □

The first term can be thought of as the estimate under infinite data, the second term as the penalty for using function classes that are too rich, and the remaining terms as the price we pay for finite data/ finite calculations.

Misspecification for OPE

When the assumptions behind MML do not hold, our method underbounds the true error. The following is the proof for this Proposition.

Proof for Prop. 4.3.5. We have shown already that $J(\pi, \widehat{P}) - J(\pi, P^*) = \gamma\mathcal{L}(w_\pi^{P^*}, V_\pi^P, P)$ $(= \gamma\mathcal{L}((WV)^*, P))$. Therefore, by linearity of $\mathcal{L}$ in $\mathcal{H}$, we have

$$\begin{aligned}
|\mathcal{L}((WV)^*, P)| &= |\mathcal{L}(h, P) + \mathcal{L}((WV)^* - h, P)| \quad \forall h \in \mathcal{H}, P \in \mathcal{P} \\
&\leq |\mathcal{L}(h, P)| + |\mathcal{L}((WV)^* - h, P)| \\
&\leq \min_P \max_h |\mathcal{L}(h, P)| + |\mathcal{L}(h - (WV)^*, P)| \\
&\leq \min_P \max_h |\mathcal{L}(h, P)| + \max_P \min_h |\mathcal{L}((WV)^* - h, P)|,
\end{aligned}$$

where $\epsilon_{\mathcal{H}} = \max_P \min_h |\mathcal{L}((WV)^* - h, P)|$. Therefore $|J(\pi, \widehat{P}) - J(\pi, P^*)| \leq \gamma(\min_P \max_h |\mathcal{L}(h, P)| + \epsilon_{\mathcal{H}})$, as desired. □

Application to the Online Setting and Brief VAML Comparison

Algorithm 12 is the prototypical online model-based RL algorithm. In contrast to the batch setting, we allow for online data collection. We require a function called PLANNER, which can take a model P_k and find the optimal solution π_k in P_k.

Algorithm 12 Online Model-Based RL

Require: $\pi_0 = \pi_b$. PLANNER$(\cdot)$

1: **for** $k = 0, 1, \ldots, K$ **do**
2: Collect data D_k by interacting with the true environment using π_k.
3: Fit $P_k \leftarrow \arg\min_{P \in \mathcal{P}} \max_{w, V \in \mathcal{W}, \mathcal{V}} \mathcal{L}_{MML}(w, V, P)$ where $D_{\pi_b} = D_k$
4: Fit $\pi_k \leftarrow$ PLANNER(P_k)
5: **return** (P_K, π_K)

Here we show that MML lower bounds the VAML error in online model-based RL, where VAML is designed.

Proposition B.1.2. *Let* $\mathcal{W} = \{1\}$. *Then*

$$\min_{P \in \mathcal{P}} \max_{w \in \mathcal{W}, V \in \mathcal{V}} \mathcal{L}_{MML}(w, V, P)^2 \leq \min_{P \in \mathcal{P}} \mathcal{L}_{VAML}(\mathcal{V}, P),$$

for every $\mathcal{V}, \mathcal{P}$.

Proof. Fix $P \in \mathcal{P}$. Then, by definition,

$$\mathcal{L}_{MML}(w, V, P) = E_{(s,a,s') \sim D_{\pi_b} P^*}[w(s,a)(E_{x \sim P(\cdot|s,a)}[V(x)] - V(s'))]$$

. Since $\mathcal{W} = \{1\}$, then we can eliminate this dependence and get $\mathcal{L}_{MML}(1, V, P) = E_{(s,a,s') \sim D_{\pi_b} P^*}[E_{x \sim P(\cdot|s,a)}[V(x)] - V(s')]$. Explicitly,

$$\begin{aligned}
&\mathcal{L}_{MML}(1, V, P)^2 \\
&= \Big(\int \left(\int P(x|s,a) V(x) d\nu(x) - \int P^*(s'|s,a) V(s') d\nu(s') \right) d\nu(s,a)\Big)^2 \\
&= \Big(\int \left(\int (P(x|s,a) - P^*(x|s,a)) V(s') d\nu(x) \right) d\nu(s,a)\Big)^2 \\
&\leq \int \left(\int (P(x|s,a) - P^*(x|s,a)) V(x) d\nu(x) \right)^2 d\nu(s,a), \qquad \text{Cauchy Schwarz}
\end{aligned}$$

Taking the $\max_{V \in \mathcal{V}}$ on both sides and noting $\max_V \int f(V) \leq \int \max_V f(V)$ for any f, V then

$$\begin{aligned}
\max_{V \in \mathcal{V}} \mathcal{L}_{MML}(1, V, P)^2 &\leq \int \max_{V \in \mathcal{V}} \left(\int (P(x|s,a) - P^*(x|s,a)) V(x) d\nu(x) \right)^2 d\nu(s,a) \\
&= \mathcal{L}_{VAML}(\mathcal{V}, P). \qquad \text{(B.4)}
\end{aligned}$$

Since we chose P arbitrarily, then Eq B.4 holds for any $P \in \mathcal{P}$. In particular, if $\widehat{P}_{VAML} = \arg\min_{P \in \mathcal{P}} \mathcal{L}_{VAML}(\mathcal{V}, P)$ then

$$\min_{P \in \mathcal{P}} \max_{V \in \mathcal{V}} \mathcal{L}_{MML}(1, V, P)^2 \leq \max_{V \in \mathcal{V}} \mathcal{L}_{MML}(1, V, \widehat{P}_{VAML})^2 \leq \min_{P \in \mathcal{P}} \mathcal{L}_{VAML}(\mathcal{V}, P).$$

□

Prop B.1.2 reflects that the MML loss function is a tighter loss in the online model-based RL case than VAML. In a sense, this reflects that MML should be the preferred decision-aware loss function even in online model-based RL. An argument in favor of VAML is that it is more computationally tractable given an assumption that $\mathcal{V}$ is the set of linear function approximators. However, if we desire to use more powerful function approximation VAML suffers the same computational issues as MML. In general the pointwise supremum within VAML presents a substantial computational challenge while the uniform supremum from MML is much more mild, can be formulated as a two player game and solved via higher-order gradient descent (see Section B.4).

Lastly, VAML defines the pointwise loss with respect to the L^2 norm of the difference between P and P^*. The choice is justified in that it is computationally friendlier but it is noted that L^1 may also be reasonable [59]. We show in the following example that, actually, VAML may not work with a pointwise L^1 error.

Example B.1.1. Let $\mathcal{S} = A \cup B$, a disjoint partition of the state space. For simplicity, assume no dependence on actions. Suppose our models $\mathcal{P} = \{P_\alpha\}_{\alpha\in[0,1]}$ take the form

$$P_\alpha(s'|s) = \begin{cases} \alpha & s' \in A \\ 1-\alpha & s' \in B \end{cases}.$$

Suppose also that $P^*_{\alpha^*} \in \mathcal{P}$ for some $\alpha^* \in [0,1]$. Let $\mathcal{V} = \{x\mathbf{1}_{s\in A}(s) + y\mathbf{1}_{s\in B}(s) | x, y < M \in \mathbb{R}^+\}$ be all bounded piecewise constant value functions with $\|V\|_\infty = M \in \mathbb{R}^+$. Then the empirical VAML loss with L^1 pointwise distance does not choose P^* when $\alpha \neq \frac{1}{2}$ and cannot differentiate between P^* and any other $P \in \mathcal{P}$ when $\alpha^* = \frac{1}{2}$. MML does not have this issue.

Proof. To show this, first fix $P \in \mathcal{P}$. Then the empirical VAML loss (in expectation) is given by

$$\begin{aligned}
&E_{s\sim P^*}[\max_V |E_{x\sim P}[V(x)] - V(s)|] \\
&= \alpha^* \max_V |E_{x\sim P}[V(x)] - V(A)| + (1-\alpha^*) \max_V |E_{x\sim P}[V(x)] - V(B)| \\
&= \alpha^* \max_{x,y\in[0,M]} |\alpha x + (1-\alpha)y - x| + (1-\alpha^*) \max_{x,y\in[0,M]} |\alpha x + (1-\alpha)y - y| \\
&= \alpha^* \max_{x,y\in[0,M]} |(\alpha-1)(x-y)| + (1-\alpha^*) \max_{x,y\in[0,M]} |\alpha(x-y)| \\
&= (\alpha^*|\alpha-1| + (1-\alpha^*)|\alpha|)M.
\end{aligned}$$

If $\alpha^* < .5$ then the minimizer of the above quantity is $\alpha = 0$, if $\alpha^* > .5$ then the minimizer is $\alpha = 1$. Therefore, if $\alpha^* \in (0, .5) \cup (.5, 1)$ then VAML picks the wrong model $\alpha \neq \alpha^*$. Additionally, in the case that $\alpha^* = .5$ then the loss is $\frac{M}{2}$ for every $P \in \mathcal{P}$. In this case, VAML with L^1 cannot differentiate between any model; all models are perfectly identical.

On the other hand, we repeat this process with MML:

$$
\begin{aligned}
&|E_{s\sim P^*}[E_{x\sim P}[V(x)] - V(s)]| \\
&= |\alpha^*(E_{x\sim P}[V(x)] - V(A)) + (1-\alpha^*)(E_{x\sim P}[V(x)] - V(B))| \\
&= |\alpha^*(\alpha x + (1-\alpha)y - x) + (1-\alpha^*)(\alpha x + (1-\alpha)y - y)| \\
&= |\alpha^*(\alpha - 1)(x-y) + (1-\alpha^*)\alpha(x-y)| \\
&= |\alpha - \alpha^*||x-y|.
\end{aligned}
$$

Clearly $\min_{\alpha\in[0,1]} \max_{x,y\in[0,M]} |\alpha - \alpha^*||x-y| = 0$ where $\alpha = \alpha^*$. □

We do not have to worry about the choice of norm for MML because we know that the OPE error is precisely $\mathcal{L}_{MML}$. On the other hand, as shown in the example, this is not the case for VAML.

B.2 OPO

In this section we explore the OPO results, as presented in the chapter.

Main Result

Proof for Theorem 4.4.1. Fix some $P \in \mathcal{P}$. Through addition of 0, we get

$$J(\pi^*_{P^*}, P^*) - J(\pi^*_P, P^*) = J(\pi^*_{P^*}, P^*) - J(\pi^*_{P^*}, P) + J(\pi^*_{P^*}, P) - J(\pi^*_P, P) + J(\pi^*_P, P) - J(\pi^*_P, P^*).$$

Since π^*_P is optimal in P then $J(\pi^*_{P^*}, P) - J(\pi^*_P, P) \leq 0$ which implies

$$J(\pi^*_{P^*}, P^*) - J(\pi^*_P, P^*) \leq J(\pi^*_{P^*}, P^*) - J(\pi^*_{P^*}, P) + J(\pi^*_P, P) - J(\pi^*_P, P^*)$$

Taking the absolute value of both sides, triangle inequality and invoking Lemma 4.3.1 yields:

$$|J(\pi^*_{P^*}, P^*) - J(\pi^*_{\widehat{P}}, P^*)| \leq 2\gamma \max_{w,V} |L(w, V, \widehat{P})| = 2\gamma \min_P \max_{w,V} |L(w, V, P)|$$

when $w^{P^*}_{\pi^*_{P^*}}, w^{P^*}_{\pi^*_P} \in \mathcal{W}$ and $V^P_{\pi^*_{P^*}}, V^P_{\pi^*_P} \in V$ for every $P \in \mathcal{P}$, or alternatively $w^P_{\pi^*_{P^*}}, w^P_{\pi^*_P} \in \mathcal{W}$ and $V^{P^*}_{\pi^*_{P^*}}, V^{P^*}_{\pi^*_P} \in V$ for every $P \in \mathcal{P}$. □

Sample Complexity for OPO

Since we will only have access to the empirical version $\widehat{P}_n$ rather than $\widehat{P}$, we provide the following bound

Theorem B.2.1 (Learning Error)**.** *Let the functions in $\mathcal{V}$ and $\mathcal{W}$ be uniformly bounded by C_V and C_W respectively. Assume the conditions of Theorem 4.4.1 hold and $|\mathcal{R}| \leq \bar{R}, \gamma \in [0, 1)$. Then, with probability $1 - \delta$,*

$$|J(\pi^*_{\widehat{P}_n}, P^*) - J(\pi^*_{P^*}, P^*)| \leq 2\gamma \min_P \max_{w,V} |L(w, V, P)| + 8\gamma \mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P}) + 8\gamma C_{\mathcal{W}} C_{\mathcal{V}} \sqrt{\log(2/\delta)/n},$$

where $\mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P})$ is the Rademacher complexity of the function class

$$\{(s, a, s') \mapsto w(s, a)(E_{x \sim P}[V(x)] - V(s')) : w \in \mathcal{W}, P \in \mathcal{P}, V \in \mathcal{V}\}.$$

Proof for Theorem B.2.1. By Theorem 4.4.1,

$$|J(\pi^*_{\widehat{P}_n}, P^*) - J(\pi^*_{P^*}, P^*)| \leq 2\gamma \max_{w,V} |L(w, V, \widehat{P}_n)|.$$

We have shown in the proof of Theorem 4.3.1 that

$$\max_{w,V} |L(w, V, \widehat{P}_n)| \leq \min_P \max_{w,V} |L(w, V, P)| + 4\mathfrak{R}_n(\mathcal{W}, \mathcal{V}, \mathcal{P}) + 4C_{\mathcal{W}} C_{\mathcal{V}} \sqrt{\log(2/\delta)/n}.$$

Combining the two completes the proof. □

This bound has the same interpretation as in the OPO case; see Section B.1.

Misspecification

Similarly as in Section B.1, we show the misspecification gap for OPO in the following result.

Lemma B.2.2 (OPO Misspecification)**.** *Let $\mathcal{H} \subset (\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R})$ be functions on (s, a, s'). Denote $(WV)^*_{P^*} = w^{P^*}_{\pi^*_{P^*}}(s,a) V^P_{\pi^*_{P^*}}(s')$ and $(WV)^*_P = w^{P^*}_{\pi^*_P}(s,a) V^P_{\pi^*_P}(s')$. Then:*

$$|J(\pi, \widehat{P}) - J(\pi, P^*)| \leq 2\gamma \left(\min_{P \in \mathcal{P}} \max_{h \in \mathcal{H}} |\mathcal{L}(h, P)| + \epsilon_{\mathcal{H}} \right), \tag{B.5}$$

where

$$\epsilon_{\mathcal{H}} = \max(\max_{P \in \mathcal{P}} \min_{h \in \mathcal{H}} |\mathcal{L}((WV)^*_{P^*} - h, P)|, \max_{P \in \mathcal{P}} \min_{g \in \mathcal{H}} |\mathcal{L}((WV)^*_P - g, P)|).$$

Proof for Lemma B.2.2. From the proof of Theorem 4.4.1,

$$\begin{aligned} J(\pi^*_{P^*}, P^*) - J(\pi^*_P, P^*) &\leq J(\pi^*_{P^*}, P^*) - J(\pi^*_{P^*}, P) + J(\pi^*_P, P) - J(\pi^*_P, P^*) \\ &= \mathcal{L}(w^{P^*}_{\pi^*_{P^*}}, V^P_{\pi^*_{P^*}}, P) + \mathcal{L}(w^{P^*}_{\pi^*_P}, V^P_{\pi^*_P}, P). \end{aligned}$$

Using the result from proof of Lemma 4.3.5,

$$\begin{aligned} &|\mathcal{L}(w^{P^*}_{\pi^*_{P^*}}, V^P_{\pi^*_{P^*}}, P) + \mathcal{L}(w^{P^*}_{\pi^*_P}, V^P_{\pi^*_P}, P)| \\ &\leq |\mathcal{L}(h, P) + \mathcal{L}((WV)^*_{P^*} - h, P)| + |\mathcal{L}(g, P) + \mathcal{L}((WV)^*_P - g, P)| \\ &\leq 2 \min_P \max_{h \in \mathcal{H}} |\mathcal{L}(h, P)| + \max_P \min_{h \in \mathcal{H}} |\mathcal{L}((WV)^*_{P^*} - h, P)| \\ &\qquad\qquad + \max_P \min_{g \in \mathcal{H}} |\mathcal{L}((WV)^*_P - g, P)| \\ &\leq 2(\min_P \max_{h \in \mathcal{H}} |\mathcal{L}(h, P)| + \epsilon_{\mathcal{H}}), \end{aligned}$$

where

$$\epsilon_{\mathcal{H}} = \max(\max_P \min_h |\mathcal{L}((WV)^*_{P^*} - h, P)|, \max_P \min_g |\mathcal{L}((WV)^*_P - g, P)|).$$

Therefore $|J(\pi, \widehat{P}) - J(\pi, P^*)| \leq 2\gamma(\min_P \max_h |\mathcal{L}(h, P)| + \epsilon_{\mathcal{H}})$, as desired. $\square$

B.3 Additional theory

In this section, we provide additional results that were not covered in the chapter. Specifically, we show that as we make $\mathcal{W}, \mathcal{V}$ too rich then the only model with zero loss is P^* itself, which may not be in $\mathcal{P}$.

Necessary and sufficient conditions for uniqueness of $|\mathcal{L}(w, V, P)| = 0$

When $\mathcal{W}, \mathcal{V}$ are in L^2 then $|\mathcal{L}| = 0$ is uniquely determined:

Lemma B.3.1 (Necessary and Sufficient)**.** $\mathcal{L}(w, V, P) = 0$ *for all* $w \in L^2(\mathcal{X}, \nu) = \{g : \int g^2(x, a) d\nu(x, a) < \infty\}, V \in L^2(\mathcal{S}, \nu) = \{f : \int f^2(x) d\nu(x) < \infty\}$ *if and only if* $P = P^*$ *wherever* $D_{\pi_b}(s, a) \neq 0$.

Corollary B.3.2. *The same result holds if* $w \cdot V \in L^2(\mathcal{X} \times \mathcal{S}, \nu) = \{h : \int h^2(x, a, x') d\nu(x, a, x') < \infty\}$.

Proof for Lemma B.3.1 and Corollary B.3.2. We begin with definition 4.5.1 and expand the expectation.

$$\begin{aligned} L(w, V, P) =& E_{(s,a,s') \sim D_{\pi_b}(\cdot,\cdot) P^*(\cdot|s,a)}[w(s, a) \left(E_{x \sim P(\cdot|s,a)}[V(x)] - V(s') \right)] \\ &= E_{(s,a) \sim D_{\pi_b}(\cdot,\cdot)}[w(s, a) \left(E_{s' \sim P(\cdot|s,a)}[V(s')] - E_{s' \sim P(\cdot|s,a)}[V(s')] \right)] \\ &= \int D_{\pi_b}(s, a) w(s, a) (V(s')(P(s'|s, a) - P^*(s'|s, a)) \, d\nu(s, a, s'). \end{aligned}$$

($\Rightarrow$) Clearly if $P = P^*$ then $L(w, V, P) = 0$. ($\Leftarrow$) For the other direction, suppose $L(w, V, P) = 0$. By assumption, $w(s, a)$ can take on any function in $L^2(\mathcal{X}, \nu)$ and therefore if $L(w, V, P) = 0$ then

$$\int V(s')(P(s'|s, a) - P^*(s'|s, a)) \, d\nu(s') = 0, \tag{B.6}$$

wherever $D_{\pi_b}(s, a) \neq 0$. Similarly, $V(s')$ can take on any function in $L^2(\mathcal{S}, \nu)$ and therefore if equation (B.6) holds then $P = P^*$. For the corollary, let $(w, V) \in \mathcal{W}\mathcal{V}$ take on any function in $L^2(\mathcal{X} \times \mathcal{S}, \nu)$. If $L(w, V, P) = 0$ then $P(s'|s, a) - P^*(s'|s, a) = 0$, as desired. □

In an RKHS, when the kernel corresponds to an integrally strict positive definite kernel (ISPD), $P = P^*$ remains the unique minimizer of the MML Loss:

Lemma B.3.3 (Realizability means zero loss even in RKHS)**.** $\mathcal{L}(w, f, P) = 0$ *if and only if* $P = P^*$ *for all* $(w, V) \in \{(w(s, a), V(s')) : \langle wV, wV \rangle_{\mathcal{H}_k} \leq 1, w : X \times A \to \mathbb{R}, V : X \to \mathbb{R}\}$ *in an RKHS with an integrally strict positive definite (ISPD) kernel.*

Proof for Lemma B.3.3. [208] prove an analogous result and proof here is included for reader convenience. From Mercer's theorem [145], there exists an orthonormal basis $(\varphi_j)_{j=1}^{\infty}$ of $L^2(\mathcal{X} \times \mathcal{S}, \nu)$ such that RKHS is represented as

$$\mathcal{WV} = \left\{ w \cdot V = \sum_{j=1}^{\infty} b_j \varphi_j \,\middle|\, (b_j)_{j=1}^{\infty} \in l^2(\mathbb{N}) \text{ with } \sum_{j=1}^{\infty} \frac{b_j^2}{\mu_j} < \infty \right\},$$

where each μ_j is a positive value since kernel is ISPD. Suppose there exists some $P \in \mathcal{P}$ such that $L(w, V, P) = 0$ for all $(w, V) \in \mathcal{WV}$ and $P \neq P^*$. Then, by taking $b_j = 1$ when $(j = j')$ and $b_j = 0$ when $(j \neq j')$ for any $j' \in \mathbb{N}$, we have $L(\varphi_j, P) = 0$ where we treat $w \cdot V$ as a single input to L. This implies $L(w, V, P) = 0$ for all $w \cdot V \in L^2(\mathcal{X} \times \mathcal{S}, \nu) = 0$. This contradicts corollary B.3.2, concluding the proof. □

B.4 Scenarios & Considerations

In this section we give proof for the various propositions for the corresponding section in the chapter.

Linear Function Classes

Proof for Prop. 4.5.1. Given $w(s,a)V(s') = \psi(s,a,s')^T\beta$ and $P(s'|s,a) = \varphi(s,a,s')^T\alpha$ then

$$\begin{aligned} L_n(w,V,P) &= E_n[E_{x\sim P}[\psi(s,a,x)^T\beta] - \psi(s,a,s')^T\beta], \\ &= E_n\left[\int \alpha\varphi(s,a,x)^T\psi(s,a,x)^T\beta d\nu(x) - \psi(s,a,s')^T\beta\right], \\ &= E_n[\alpha^T\left(\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right)\beta - \psi(s,a,s')^T\beta], \end{aligned}$$

which is linear in β. $L_n^2(w,V,P) = 0$ is achieved through

$$E_n[\alpha^T\left(\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right) - \psi(s,a,s')^T] = 0.$$

Thus,

$$\widehat{\alpha}^T = E_n[\psi(s,a,s')^T]E_n\left[\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right]^{-1},$$

assuming $E_n\left[\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(s')\right]$ is full rank. Taking the transpose completes the proof. □

Proof for Prop. 4.5.2. We begin with $\varphi(s,a,s') = e_{(s,a,s')}$, the (s,a,s')-th standard basis vector and $\psi = \varphi$. Then

$$X(s,a) = (\sum_{x\in\mathcal{S}} \varphi(s,a,x)\varphi(s,a,x)^T)_{i,j} = \begin{cases} 1 & i = s|\mathcal{A}||\mathcal{S}| + a|\mathcal{S}|, i = j \\ 0 & \text{otherwise} \end{cases}.$$

Notice that $X(s,a)$ is a diagonal matrix and is the discrete counter-part to $\int \varphi(s,a,s')\psi(s,a,s')^T d\nu(x)$. Therefore, $E_n[X(s,a)] = \frac{1}{N}\sum_{(s,a,s')\in D} X(s,a)$, which is a diagonal matrix of the average number of times (s,a) appears in the dataset D. Similarly, $E_n[\varphi(s,a,s')]$ is the average number of times that (s,a,s') appears in the dataset D. Hence, by Prop 4.5.1,

$$\widehat{\alpha}_{s,a,s'} = \frac{\#\{(s,a,s') \in D\}}{\#\{(s,a,x) \in D : \forall x \in \mathcal{S}\}}.$$

Therefore $P(s'|s,a) = \varphi(s,a,s')^T\widehat{\alpha} = \widehat{\alpha}_{s,a,s'}$, as desired.

□

LQR

In order to provide proof that MML gives the LQR-optimal solution, we begin with a few Lemmas. First, we show that the value function is quadratic.

Lemma B.4.1 (Value Function is Quadratic)**.** *Let $s_{t+1} = As_t + Ba_t + w$ with $w \sim N(0, \sigma^{*2} I)$ be the dynamics, $\pi_K(a|s) = -Ks + w_K$ where $w_K \sim N(0, \sigma_K^2 I)$ be the policy. Let $\gamma \in (0, 1]$ be the discount factor. Then $V(s) = s^T U s + q$ where*

$$U = Q + K^T R K + \gamma (A - BK)^T U (A - BK)$$
$$q = \frac{1}{1-\gamma}(\sigma_K^2 tr(R) + \gamma \sigma_K^2 tr(B^T U B) + \gamma \sigma^{*2} tr(U)).$$

Proof for Lemma B.4.1. The value function is given by:

$$\begin{aligned}
&x^T U x + q \\
&= x^T Q x + E_{N(-Kx, \sigma_K^2 I)}[u^T R u + \gamma E_{N(Ax+Bu, \sigma^{*2} I)}[V(s')]] \\
&= x^T Q x + E_{N(-Kx, \sigma_K^2 I)}[u^T R u + \gamma (Ax + Bu)^T U (Ax + Bu) + \gamma q + \gamma \sigma^{*2} tr(U)] \\
&= x^T Q x + x^T K^T R K x + \sigma_K^2 tr(R) + \gamma x^T (A - BK)^T J (A - BK) x \\
&\qquad + \gamma \sigma_K^2 tr(B^T U B) + \gamma q + \gamma \sigma^{*2} tr(U).
\end{aligned}$$

Thus, the quadratic terms satisfies:

$$U = Q + K^T R K + \gamma (A - BK)^T U (A - BK),$$

and the linear term satisfies:

$$q = \frac{1}{1-\gamma}(\sigma_K^2 tr(R) + \gamma \sigma_K^2 tr(B^T U B) + \gamma \sigma^{*2} tr(U)).$$

The final value is given by:

$$J(\pi, P^*) = E_{N(s_0, \sigma_0^2 I)}[U] = s_0^T U s_0 + q + \sigma_0^2 tr(U).$$

Existence and uniqueness of U, q is heavily studied [22]. □

Under the same assumptions as Lemma B.4.1, we simplify $\mathcal{L}$ to:

Lemma B.4.2 (LQR Loss Simplified)**.** *In addition to the assumptions of Lemma B.4.1, let $d_0 = s_0 + w_{d_0}$ where $w_{d_0} \sim N(0, \sigma_{d_0}^2 I)$ be the initial state distribution. Let $P = As + Ba \in \mathcal{P}$ where $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times k}$ and (A, B) is controllable. Let $K \in \mathbb{R}^{k \times n}$ represent all linear policies and $U \in \mathbb{S}_+^n$ be all symmetric positive semi-definite matrices.*

$$\begin{aligned}
&\min_P \max_{w,V} |\mathcal{L}(w, V, P)| \\
&= \min_{A,B} \max_{K,U} \sum_i \gamma^i [s_0^T (A^* - B^* K)^{iT} \Delta (A^* - B^* K)^i s_0 \\
&\qquad + tr(\Delta \Sigma_i)] + \sigma_K^2 tr(B^T U B - B^{*T} U B^*) - \sigma^{*2} tr(U),
\end{aligned}$$

where $\Delta = (A - BK)^T U(A - BK) - (A^* - B^*K)^T U(A^* - B^*K)$ *and* $\Sigma_i = \sigma^*(I + \ldots + F^{i-1}F^{(i-1)T}) + \sigma_K(B^*B^{*T} + \ldots + F^{i-1}B^*B^{*T}F^{(i-1)T}) + \sigma_0 F^i F^{iT}$ *for* $i > 0$ *and* $\Sigma_0 = \sigma_0 I$, $F = A^* - B^*K$.

Proof for Lemma B.4.2. We first show that the evolution of dynamics P^* under gaussian noise, with a linear gaussian controller is a gaussian mixture $\sum_i N((A^* - B^*K)^i s_0, \Sigma_i)$, where $\Sigma_i = \sigma^*(I + \ldots + F^{i-1}F^{(i-1)T}) + \sigma_K(B^*B^{*T} + \ldots F^{i-1}B^*B^{*T}F^{(i-1)T}) + \sigma_0 F^i F^{iT}$ for $i > 0$ and $\Sigma_0 = \sigma_0 I$, $F = A^* - B^*K$.

It's clear $s_0 \sim N(s_0, \sigma_0^2 I)$, the base case. Suppose for induction $s_n \sim N((A^* - B^*K)^n s_0, \Sigma_n)$ holds for some $n \geq 0$. Then

$$\begin{aligned}
s_{n+1} &= A^* s_n + B^*(-Ks_n + w_K) + w^* \\
&= (A^* - B^*K)s_n + B^* w_k + w^* \\
&\sim N((A^* - B^*K)^{n+1} s_0, (A^* - B^*K)\Sigma_n (A^* - B^*K)^T + B^*B^{*T} + \sigma^* I) \\
&= N((A^* - B^*K)^{n+1} s_0, \Sigma_{n+1}),
\end{aligned}$$

completing the inductive step. Every step s_t is gaussian, therefore

$$d_{\pi,\gamma}^{P^*}(s, a) = \sum_{i=0}^{\infty} \gamma^i N(s; F^i s_0, \Sigma_i) N(a; -Ks, \sigma_K^2 I) \tag{B.7}$$

is a gaussian mixture. Let $w = \frac{d_{\pi,\gamma}^{P^*}}{D}$. We know V is quadratic, given by $U \in \mathcal{S}_+^n$. For notational convenience, for matrix Z and vectors x,y define $\langle x, y \rangle_Z \equiv x^T Z y$ and similarly $\|x\|_Z \equiv \langle x, x \rangle_Z = x^T Z x$. Therefore,

$$\begin{aligned}
&\min_P \max_{w,V} \mathcal{L}(w, V, P) = \min_{A,B} \max_{w,V} E_{(s,a)\sim D}[w[E_P[V] - E_{P^*}[V]]] \\
&= \min_{A,B} \max_{w,U} E_{(s,u)\sim D}[w[\|As + Bu\|_U - \|A^*s + B^*u\|_U - \sigma^{*2} tr(U)]] \\
&= \min_{A,B} \max_{K,U} E_{\sum_i \gamma^i N((A^* - B^*K)^i s_0, \Sigma_i)}[E_{u \sim N(-Ks, \sigma_K^2 I)}[\|As + Bu\|_U \\
&\qquad\qquad - \|A^*s + B^*u\|_U - \sigma^{*2} tr(U)]] \\
&= \min_{A,B} \max_{K,U} E_{\sum_i \gamma^i N((A^* - B^*K)^i s_0, \Sigma_i)}[\|(A - BK)s\|_U - \|(A^* - B^*K)s\|_U \\
&\qquad\qquad + \sigma_K^2 tr(B^T U B) - \sigma_K^2 tr(B^{*T} U B^*) - \sigma^{*2} tr(U)] \\
&= \min_{A,B} \max_{K,U} E_{\sum_i \gamma^i N((A^* - B^*K)^i s_0, \Sigma_i)}[s^T[\Delta(A, B, A^*, B^*, U, K)]s \\
&\qquad\qquad + \sigma_K^2 tr(B^T U B - B^{*T} U B^*) - \sigma^{*2} tr(U)] \\
&= \min_{A,B} \max_{K,U} \sum_i \gamma^i [\|(A^* - B^*K)^i s_0\|_\Delta + tr(\Delta \Sigma_i)] \\
&\qquad\qquad + \sigma_K^2 tr(B^T U B - B^{*T} U B^*) - \sigma^{*2} tr(U),
\end{aligned}$$

where $\Delta = (A - BK)^T U(A - BK) - (A^* - B^*K)^T U(A^* - B^*K)$. □

First, Lemma B.4.2 supposes that there is model mismatch $P^* \notin \mathcal{P}$ since $\mathcal{P}$ are deterministic simulators and P^* is stochastic. Second, we notice that K takes the position of w, which is to say that the policy K directly specifies w, as expected. We will need the previous two results in the experiments. We may now prove Prop 4.5.3 that says MML yields the true parameters of LQR in expectation:

Proof for Prop 4.5.3. Consider two linear, controllable systems with parameters $P_1 = (A_1, B_1)$ and $P_2 = (A_2, B_2)$. Then there exists a controller K that stabilizes P_1 (i.e, $J(P_1, K) < \infty$) but destabilizes P_2 (i.e, $J(P_2, K) = \infty$). We show this by analyzing the characteristic polynomial of both $A_1 - B_1K$ and $A_2 - B_2K$. There exists an invertible matrix T_1, T_2 that put $(A_1, B_1), (A_2, B_2)$ into controllable canonical forms (CCF), respectively [22]. Thus, we will assume, wlog, that $(\tilde{A}_1, \tilde{B}_1), (\tilde{A}_2, \tilde{B}_2)$ are already in CCF. Hence,

$$\tilde{A}_1 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix}, \quad \tilde{B}_1 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

and

$$\tilde{A}_2 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & & 1 \\ -b_0 & -b_1 & -b_2 & \dots & -b_{n-1} \end{bmatrix}, \quad \tilde{B}_2 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

We will find a controller in the form $K = K_1T_1 = K_2T_2$ for some K_1, K_2 for T_1, T_2 that put the systems into CCF. Consider a desired characteristic polynomial of $f(s) = (s + \epsilon)^{n-1}(s + \lambda)$ for $\epsilon, \lambda \in \mathbb{R}^+(> 0)$. This polynomial has eigenvalues equal to $-\epsilon, -\lambda$ and therefore a system with this polynomial is asymptotically stable (converges to 0 exponentially fast). Take $K_1 = [k_{1,0}, k_{1,1}, \dots, k_{1,n-1}]$. Then $det(sI - (\tilde{A}_1 - \tilde{B}_1K_1)) = s^n + (a_{n-1} + k_{1,n-1})s^{n-1} + \dots + (a_0 + k_{1,0})$. By selecting $k_{1,i} = \left(\binom{n-1}{i}\lambda + \binom{n-1}{i-1}\epsilon\right)\epsilon^{n-1-i} - a_i$ then $det(sI - (\tilde{A}_1 - \tilde{B}_1K_1)) = f(s)$. Hence, $(\tilde{A}_1, \tilde{B}_1)$ is asymototically stable with eigenvalues $-\lambda, -\epsilon$ for any λ, ϵ strictly positive. Therefore $K = K_1T_1$ makes the system (A_1, B_1) asymptotically stable.

Now we consider $K_2 = K_1T_1T_2^{-1}$. Let us denote $T_1T_2^{-1} = T$ which is also

invertible since T_1, T_2 are invertible. Then by taking the last term of $det(sI - (\tilde{A}_2 - \tilde{B}_2 K_2))$, we can examine the product $\prod_{i=0}^{n-1} \lambda_i$ of the eigenvalues of the closed loop system $\tilde{A}_2 - \tilde{B}_2 K_2$. Namely, $b_0 + \sum_{i=0}^{n-1} k_{1,i} T_{i,n}$ is the product of eigenvalues. We may simplify this via some algebra as follows:

$$\begin{aligned}
\prod_{i=0}^{n-1} \lambda_i &= b_0 + \sum_{i=0}^{n-1} k_{1,i} T_{i,n} \\
&= b_0 + \sum_{i=0}^{n-1} T_{i,n} \left(\left(\binom{n-1}{i} \lambda + \binom{n-1}{i-1} \epsilon \right) \epsilon^{n-1-i} - a_i \right) \\
&= \underbrace{b_0 - \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} T_{i,n} \binom{n-1}{i-1} \epsilon^{n-i}}_{\bar{b}} + \lambda \underbrace{\sum_{i=0}^{n-1} T_{i,n} \binom{n-1}{i} \epsilon^{n-1-i}}_{c} \\
&= \bar{b} + \lambda c.
\end{aligned}$$

We may select $\epsilon > 0$ so that $c \neq 0$ otherwise $T_{i,n} = 0$ for all i which would contradict invertibility of T. Therefore $\prod_{i=0}^{n-1} \lambda_i$ is linear in λ. By driving $\lambda \to \infty$, then $|\prod_{i=0}^{n-1} \lambda_i| \to \infty$ is unbounded. Select λ so that $|\bar{b} + \lambda c| > 1$. By the pigeonhole principle, at least one of the eigenvalues of $\tilde{A}_2 - \tilde{B}_2 K_2$ must have a magnitude greater than 1 and therefore the system is unstable. Therefore the controller $K_2 T_2 = K_1 T_1 T_2^{-1} T_2 = K_1 T_1 = K$ makes the system (A_2, B_2) unstable. Hence, K simultaneously stabilizes (A_1, B_1) but destabilizes (A_2, B_2).

According to Lemma B.4.2, when $(A, B) = (A^*, B^*)$ then for any K:

$$\max_U \mathcal{L}((A, B), K, U) = \max_U |\sigma^{*2} tr(U)| < \infty,$$

since U are bounded by assumption. Furthermore, we have just shown that there always exists a K that destabilizes any controller $(A, B) \neq (A^*, B^*)$ while stabilizing (A^*, B^*). Therefore:

$$\max_{K,U} \mathcal{L}((A, B), K, U) = \infty,$$

for any system $(A, B) \neq (A^*, B^*)$. Therefore:

$$\min_{(A,B)} \max_{K,U} \mathcal{L}((A, B), K, U) = (A^*, B^*).$$

It is well known that ordinary least squares is a consistent estimator when the noise is exogenous, as it is here. Therefore the maximum likelihood solution also yields (A^*, B^*) in expectation. □

RKHS & Practical Implementation

Since $P \in \mathcal{P}$ is a stochastic model in general, then the inner expectation of the loss in def (4.5.1) over P involves sampling x from $P(\cdot|s, a)$ and computing the

empirical average of $V(x)$. In general this can be computationally demanding if $\mathcal{S}$ is high dimensional and P does not have a closed form, requiring MCMC estimates or variational estimates [139, 73]. However, in practice, most parametrizations of models use nice distributions, such as gaussians, from which sampling is efficient. This issue is similarly present in other decision-aware literature [e.g., 59].

The estimator based on Eq (4.8) requires solving a minimax problem which is often computationally challenging. One approach might be to set-up GAN-style neural networks and use higher order gradient descent [74, 178].

If we have access to a kernel, say radial basis function (RBF), then the inner maximization over w, V has a closed form when $\mathcal{W} \times \mathcal{V}$ correspond to a reproducing kernel Hilbert space (RKHS), H_K with kernel K. In particular, in similar spirit to [134, 62, 208] we have

Proposition B.4.3 (Closed form exists in RKHS)**.** *Assume* $\mathcal{WV} = \{(w(s,a), V(s')) : \langle wV, wV\rangle_{\mathcal{H}_K} \leq 1, w : \mathcal{X} \to \mathbb{R}, V : \mathcal{S} \to \mathbb{R}\}$. *Let* $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ *be an inner product on* $\mathcal{H}_K$ *satisfying the reproducing kernel property* $w(s,a)V(s') = \langle wV, K((s,a,s'), \cdot)\rangle_{\mathcal{H}_K}$. *The term* $\max_{(w,V)\in\mathcal{WV}} \mathcal{L}(w,V,P)^2$ *has a closed form:*

$$\max_{(w,V)\in\mathcal{WV}} \mathcal{L}(w,V,P)^2 = E_{(s,a,s')\sim D_{\pi_b}P^*,(\tilde{s},\tilde{a},\tilde{s}')\sim D_{\pi_b}P^*}\Big[E_{x\sim P,\tilde{x}\sim P}[K((s,a,x),(\tilde{s},\tilde{a},\tilde{x}))] - 2E_{x\sim P}[K((s,a,x),(\tilde{s},\tilde{a},\tilde{s}'))] + K((s,a,s'),(\tilde{s},\tilde{a},\tilde{s}'))\Big].$$

Proof for Prop B.4.3. Recall that by the reproducing property of kernel K in the RKHS space H_K then $\langle f, K\rangle_{H_K}$ for any $f \in H_K$. Starting from Def 4.5.1,

$L(w,V,P)^2 = E_{(s,a,s')\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)}[w(s,a)\left(E_{x\sim P(\cdot|s,a)}[V(x)] - V(s')\right)]^2$

$= E_{(s,a,s',x)\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)P(\cdot|s,a)}[w(s,a)V(x) - w(s,a)V(s')]^2$

$= E_{(s,a,s',x)\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)P(\cdot|s,a)}[\langle wV, K((s,a,x),\cdot)\rangle_{\mathcal{H}_k} - \langle wV, K((s,a,s'),\cdot)\rangle_{\mathcal{H}_k}]^2$

$= \langle wV, (wV)^*\rangle^2_{\mathcal{H}_k}$

where $(wV)^*(\cdot) = E_{(s,a,s',x)\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)P(\cdot|s,a)}[K((s,a,x),\cdot) - K((s,a,s',\cdot)]$. By Cauchy-Schwarz and the fact that wV is within a unit ball, then

$\max_{w,V\in\mathcal{WV}} L(w,f,V)^2 = \max_{w,V\in\mathcal{WV}} \langle wV, (wV)^*\rangle^2_{\mathcal{H}_k} = \|(wV)^*\|^2 = \langle (wV)^*, (wV)^*\rangle_{\mathcal{H}_k}$.

Expanding,

$$
\begin{aligned}
&\max_{w,V\in\mathcal{WV}} L(w,f,V)^2 \\
&= \langle (wV)^*, (wV)^* \rangle_{\mathcal{H}_k} \\
&= \langle E_{(s,a,s',x)\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|s,a)P(\cdot|s,a)}[K((s,a,x),\cdot) - K((s,a,s',\cdot)], \\
&\qquad E_{(\tilde{s},\tilde{a},\tilde{s}',\tilde{x})\sim D_{\pi_b}(\cdot,\cdot)P^*(\cdot|\tilde{s},\tilde{a})P(\cdot|\tilde{s},\tilde{a})}[K((\tilde{s},\tilde{a},\tilde{x}),\cdot) - K((\tilde{s},\tilde{a},\tilde{s}',\cdot)]\rangle_{\mathcal{H}_k} \\
&= \Bigg\langle \int D_{\pi_b}(s,a)P^*(s'|s,a)P(x|s,a)(K((s,a,x),\cdot) - K((s,a,s'),\cdot)), \\
&\qquad \int D_{\pi_b}(\tilde{s},\tilde{a})P^*(\tilde{s}'|\tilde{s},\tilde{a})P(\tilde{x}|\tilde{s},\tilde{a})(K((\tilde{s},\tilde{a},\tilde{x}),\cdot) - K((\tilde{s},\tilde{a},\tilde{s}'),\cdot)), \Bigg\rangle_{\mathcal{H}_k} \\
&= \int D_{\pi_b}(s,a)P^*(s'|s,a)P(x|s,a)D_{\pi_b}(\tilde{s},\tilde{a})P^*(\tilde{s}'|\tilde{s},\tilde{a})P(\tilde{x}|\tilde{s},\tilde{a}) \\
&\qquad \times \langle K((s,a,x),\cdot) - K((s,a,s'),\cdot), K((\tilde{s},\tilde{a},\tilde{x}),\cdot) - K((\tilde{s},\tilde{a},\tilde{s}',\cdot)\rangle_{\mathcal{H}_k}.
\end{aligned}
$$

By linearity of the inner product, the reproducing kernel property we get:

$$
\begin{aligned}
&\max_{(w,V)\in\mathcal{WV}} L(w,f,V)^2 \\
&= E_{(s,a,s',x)\sim D_{\pi_b}P^*P,(\tilde{s},\tilde{a},\tilde{s}',\tilde{x})\sim D_{\pi_b}P^*P}[K((s,a,x),(\tilde{s},\tilde{a},\tilde{x})) - K((s,a,x),(\tilde{s},\tilde{a},\tilde{s}')) \\
&\qquad\qquad - K((s,a,s'),(\tilde{s},\tilde{a},\tilde{x})) + K((s,a,s'),(\tilde{s},\tilde{a},\tilde{s}'))] \\
&= E_{(s,a,s',x)\sim D_{\pi_b}P^*P,(\tilde{s},\tilde{a},\tilde{s}',\tilde{x})\sim D_{\pi_b}P^*P}[K((s,a,x),(\tilde{s},\tilde{a},\tilde{x})) - 2K((s,a,x),(\tilde{s},\tilde{a},\tilde{s}')) \\
&\qquad\qquad + K((s,a,s'),(\tilde{s},\tilde{a},\tilde{s}'))],
\end{aligned}
$$

where for the last equality we used the fact that K is symmetric. □

B.5 Experiments

Environment Descriptions

LQR

The LQR domain is a 1D stochastic environment with true dynamics: $P^*(s'|s,a) = s - .5a + w^*$ where $w^* \sim N(0, .01^2)$. We let $x_0 \sim N(1, .1^2)$. The reward function is $R(s,a) = -(s+a)$ and $\gamma = .9$. We use a finite class $\mathcal{P}$ consisting of all deterministic models $\mathcal{P} = \{P_x(s'|s,a) = (1 + x/10)s - (.5 + x/10)a | x \in [0, M]\}$ where we vary $M \in \{2, 3, \ldots, 19\}$. We write $(A^*, B^*) = P_0(s'|s,a) = A^*s + B^*a$, the deterministic version of P^*.

Cartpole

We use the standard Cartpole benchmark (OpenAI, [30]). The state space is a tuple $(x, \dot{x}, \theta, \dot{\theta})$ representing the position of the cart, velocity of the cart, angle of the pole and angular velocity of the pole, respectively. The action space is discrete given by pushing the car to the left or pushing the car to the right. We add $N(0, .001^2)$ Gaussian noise to each component of the state to make the dynamics stochastic. We consider the infinite horizon setting with $\gamma = .98$. The reward function is modified to be a function of angle and location $R(s, \theta) = (2 - \theta/\theta_{\max}) * (2 - s/s_{\max}) - 1)$ rather than 0/1 to make the OPE problem more challenging.

Inverted Pendulum (IP)

We consider the infinite horizon setting with $\gamma = .98$. The state space is a tuple $(\theta, \dot{\theta})$ representing the angle of the pole and angular velocity of the pole, respectively. The action space $\mathcal{A} = \mathbb{R}$ is continuous representing a clockwise or counterclockwise force. The reward function is a clipped quadratic function $R([\theta, \dot{\theta}], a) = \min(((\theta + \pi) \mod 2\pi - \pi)^2 + .1\dot{\theta}^2 + .001u^2, 100)$. This IP environment has a Runge-Kutta(4) integrator [51] rather than Forwrd Euler and, thus, produces more realistic data. The mass of the rod is .25 and the length .5.

Experiment Descriptions

LQR OPE/OPO

OPE. We aim to evaluate $\pi(a|s) = N(1.3s, .1^2)$. We ensure $V_\pi^P \in \mathcal{V}$ for all $P \in \mathcal{P}$ by solving the equations in Lemma B.4.1. We ensure $W_\pi^{P^*} \in \mathcal{W}$ using

Equation (B.7). We derive 1-d equations for VAML analogous to Lemma B.4.2). Finally, we know MLE gives (A^*, B^*) in expectation (see Prop 4.5.3).

Metric: We compute $|(J(\pi, \widehat{P}) - J(\pi, P^*))|$, the OPE error.

OPO. Similarly as in OPE, we ensure that all MML realizability assumptions hold. This means as we increase $\mathcal{P}$ then we have to increase the sizes of both $\mathcal{W}$ and $\mathcal{V}$ now instead of just $\mathcal{V}$ as in OPE. Once again MLE gives (A^*, B^*) in expectation (see Prop 4.5.3) and we evaluate VAML using equations analogous to those in Lemma B.4.2). With this, we produce Figure B.1 (right). By increasing $\mathcal{P}$, we also have more policies $\{\pi^*_P\}_{P\in\mathcal{P}}$ we may consider. Instead of selecting one for OPE, for each $\pi \in \{\pi^*_P\}_{P\in\mathcal{P}}$ we calculate the OPE error. We aggregate across all $\{\pi^*_P\}_{P\in\mathcal{P}}$ by taking the average of the OPE errors and the worst-case, which can be seen in Figure B.1 (left). **Metric:** We compute $|(J(\pi^*_{\widehat{P}}, \widehat{P}) - J(\pi^*_{P^*}, P^*))|$, the OPO error.

Note: All calculations in LQR OPE/OPO are in expectation so no error bars need be included.

Verifiability. With the same setup as in OPE, now randomly sample 100k points in the interval $[-3, 3] \times [-3, 3]$, which is the support of the LQR system. We rerun the same experiment as in OPE except now we add $w \sim \mathcal{N}(0, \epsilon)$ noise to $V \in \mathcal{V}$ where $\epsilon \in \{0, .2, \ldots, .8, 1\}$. We evaluate the error $|(J(\pi, \widehat{P}) - J(\pi, P^*))|$ over the 100k samples rather than in expectation as before. We run 5 seeds and present the mean over the seeds with standard error. We smooth the resulting mean with a moving average filter of size 3. The result can be seen in Figure 4.2 (right).

Cartpole OPE

Each $P \in \mathcal{P}$ takes the form $s' \sim \mathcal{N}(\mu(s, a), \sigma(s, a))$, where a NN outputs a mean, and logvariance representing a normal distribution around the next state. Each model has a two hidden layers and with 64 units each and ReLU activation with final linear layer. We generate the behavior and target policy using a near-perfect DDQN-based policy Q with a final softmax layer and adjustable parameter τ: $\pi(a|s) \propto \exp(Q(s, a)/\tau)$. The behavior policy has $\tau = 1$, while the target policy has $\tau = 1.5$. We truncate all rollouts at 1000 time steps and we calculate the true expected value using the Monte Carlo average of 10000 rollouts.

We model the class $\mathcal{WV}$ as a RKHS as in Lemma B.4.3 with an RBF kernel. We do the same for VAML. The RKHS kernel we use for MML and VAML is given by $K(s, a, s') = K_1(s)K_2(a)K_3(s')$ and $K_3(s')$ respectively where K_i are Gaussian Radial Basis Function (RBF) kernels with a bandwidth equal to the median of the pair-wise distances for each coordinate (s, a, s' independently) over the batch.

For MML, we sample from P a total of 5 times and take the empirical mean to calculate the expectation over P for the RKHS formula given in B.4.3.

We run 20000 batches of size 128 and normalize the data over the batch. Our learning rate is 10^{-3} and we use Adam [107] optimizer. The estimate we use is the mean over the last 10 batches. We run 5 random seeds per dataset size, and plot the log-relative MSE with standard error in Figure 4.3.

Note: These hyperparameters remain the same across the different loss functions.

Metric: We compare the methods using the log-relative MSE metric:

$$\log(\frac{(J(\pi, \widehat{P}) - J(\pi, P^*))^2}{(J(\pi_b, P^*) - J(\pi, P^*))^2}),$$

which is negative when the OPE estimate $J(\pi, \widehat{P})$ is superior to the on-policy estimate $J(\pi_b, \widehat{P})$. The more negative, the better the estimate. To calculate $J(\pi, \widehat{P})$ we run 100 trajectories in $\widehat{P}$ and take the mean.

Inverted Pendulum OPO

We generate the behavior data using a noisy feedback-linearized controller: $\pi_b(a|s)$ is uniformly random with probability .3 and is a feedback-linearized LQR controller (FLC) with probability .7 where we use the FLC corresponding to LQR matrices $Q = 2I_{2\times 2}, R = I_{2\times 2}$. We truncate all rollouts at 200 time steps. We fit 4 feed-forward neural networks representing $P_1, \ldots, P_4$ where each is a deterministic model with two layers of 16 weights and a Tanh activation followed with Linear. We use Adam [107] optimizer with 10^{-3} as the learning rate. Using different batches of size 64 on each P_i and perform 5000 iterations for each model.

The RKHS kernel we use for MML and VAML is given by $K(s, a, s') = K_1(s)K_2(a)K_3(s')$ and $K_3(s')$ respectively where K_i are Gaussian Radial Basis Function (RBF) kernels with a bandwidth equal to 1.

For MML, we only sample from P once to calculate the expectation over P for the RKHS formula given in B.4.3, since P is deterministic.

Now we have $P(s'|s,a) = \frac{1}{4}\sum_{i=1}^{4} P_i(s'|s,a)$. We calculate $\alpha = Median(\{\|P_j(s,a) - s'\|_2 : j \in [1,\ldots,4], (s,a,s') \in X \subset D\})$ where X is 10000 random samples from the dataset. We form an α-USAD (see MOREL Section B.5) and construct a pessimistic MDP $(\tilde{P}, \tilde{R})$ (see Section B.5). We use PPO as our policy optimizer with the default settings from [169]. We run PPO three times in the pessimistic MDP and take the policy that performs the best and report its performance. We keep track of the running maximum as we increase the dataset size. We plot the mean of the running maximums over the five seeds including standard error bars in Figure 4.4.

Note: These hyperparameters remain the same across the different loss functions.

Metric: We look at the performance $J(\pi^*_{\widehat{P}}, P^*)$ of a policy and compare it to π^*, learned from PPO. To calculate $J(\pi^*_{\widehat{P}}, P^*)$ we run 100 trajectories in P^* and take the mean.

MOREL

We give a brief explanation of MOREL [106] and its construction. The objective of MOREL is to make sure that the policy we learn does not take advantage of the errors in the simulator P. If there are errors in P then a policy may think the agent can perform a particular state transition (s,a,s') and $R(s',a')$ has high reward for some action a'. However, it's possible that such a transition (s,a,s') may not occur in the true environment. Therefore, we modify our model $P(s'|s,a)$ in the following way:

$$\tilde{P}(s'|s,a) = \begin{cases} \text{Terminate episode} & U^\alpha(s,a) = 1 \\ P(s'|s,a) & \text{otherwise} \end{cases},$$

where $U^\alpha(s,a) = 1$ if $\max_{i\in\{1,2,3,4\}} \|P_i(s'|s,a) - P(s'|s,a)\| \geq \alpha$, otherwise 0. In other words, we've modified the transition dynamics so that we do not trust our model P unless all the P_i are in agreement. We also modify our reward to be

$$\tilde{R}(s,a) = \begin{cases} \text{-100} & U^\alpha(s,a) = 1 \\ R(s,a) & \text{otherwise} \end{cases},$$

where -100 is chosen this value is well below any reward that the Inverted Pendulum environment generates. Similarly, we penalize our policy for entering a state where we are uncertain. Together, this creates a pessimistic MDP.

Additional Experiments

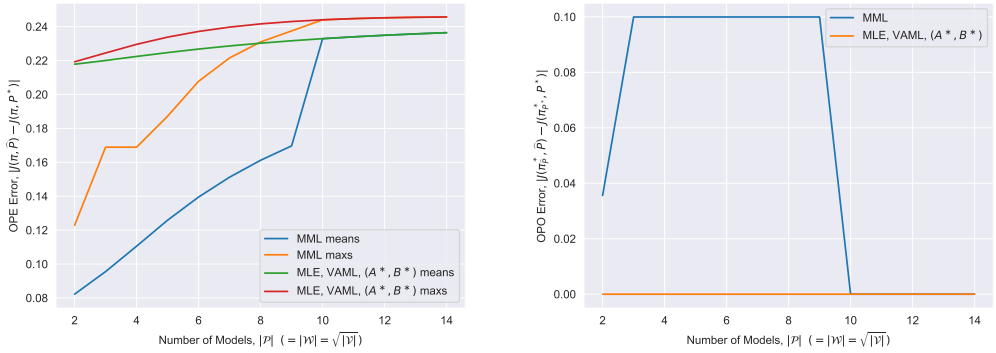


Figure B.1: *(LQR) As we increase $|W|, |V|$ then MML is forced to be robust to too many OPE problems and settles for the system (A^*, B^*) since this is the only system robust to the most OPE problems.*

In the experiments for Figure B.1, we consider what happens when we satisfy the realizability conditions for OPO. As we increase $|\mathcal{P}|$, we must also increase $|\mathcal{W}|, |\mathcal{V}|$ because each $P \in \mathcal{P}$ induces an optimal policy π^*_P to which we have to make sure $w^{P^*}_{\pi^*_P} \in \mathcal{W}$ and $V^{P_i}_{\pi^*_P} \in \mathcal{V}$ for $\forall P_i \in \mathcal{P}$. In a sense, we are adding more OPE problems for MML to be robust to. In particular, we now have more policies $\{\pi^*_P\}_{P \in \mathcal{P}}$ to consider. As described earlier, for each $\pi \in \{\pi^*_P\}_{P \in \mathcal{P}}$ we calculate the OPE error. We aggregate across all $\{\pi^*_P\}_{P \in \mathcal{P}}$ by taking the average of the OPE errors and the worst-case, which can be seen in Figure B.1 (left). We plot the OPO error in Figure B.1 (right). What we see is that while $|\mathcal{P}|$ is small, MML is able to be robust to a certain number of OPE problems. But as we increase the number of OPE problems the average and max error increases until all methods select the same model, which is the OPO-optimal model, (A^*, B^*).

Appendix C

CHAPTER 3 APPENDIX

C.1 Preliminaries to Analysis of Fitted Q Evaluation (FQE)

In this section, we set-up necessary notations and definitions for the theoretical analysis of FQE. To simplify the presentation, we will focus exclusively on weighted ℓ_2 norm for error analysis.

With the definitions and assumptions presented in this section, we will present the sample complexity guarantee of Fitted-Q-Evaluation (FQE) in appendix C.2.

While it is possible to adapt proofs from related algorithms [150, 12] to analyze FQE, in the next two sections we show improved convergence rate from $O(n^{-4})$ to $O(n^{-2})$, where n is the number of samples in data set D.

Our notation for Q function is similar to the RL literature — the only difference is that the optimal policy minimizes $Q(s,a)$ instead of maximizing. We denote the bound on the value function as $\overline{C}$ (alternatively if the single timestep cost is bounded by $\bar{c}$, then $\overline{C} = \frac{\bar{c}}{1-\gamma}$).

Bellman operators

The *Bellman optimality operator* $\mathbb{T} : \mathcal{B}(\mathcal{X}; \overline{C}) \mapsto \mathcal{B}(\mathcal{X}; \overline{C})$ as:

$$(\mathbb{T}Q)(s,a) = c(s,a) + \gamma \int_{\mathcal{S}} \min_{a' \in \mathcal{A}} Q(s',a') P(ds'|s,a). \tag{C.1}$$

The optimal value functions are defined as usual by $C^*(s) = \sup_\pi C^\pi(s)$ and $Q^*(s,a) = \sup_\pi Q^\pi(s,a) \;\; \forall (s,a) \in \mathcal{X}$.

For a given policy π, the *Bellman evaluation operator* $\mathbb{T}^\pi : \mathcal{B}(\mathcal{X}; \overline{C}) \mapsto \mathcal{B}(\mathcal{X}; \overline{C})$ as:

$$(\mathbb{T}^\pi Q)(s,a) = c(s,a) + \gamma \int_{\mathcal{S}} Q(s', \pi(s')) P(ds'|s,a). \tag{C.2}$$

It is well known that $\mathbb{T}^\pi Q^\pi = Q^\pi$, a fixed point of the $\mathbb{T}^\pi$ operator.

Data distribution and weighted ℓ_2 norm

Denote the state-action data generating distribution as μ, induced by some data-generating (behavior) policy π_{D}, that is, $(s_i, a_i) \sim \mu$ for $(s_i, a_i, s_i', c_i) \in \mathrm{D}$.

Note that data set D is formed by multiple trajectories generated by π_{D}. For

each (s_i, a_i), we have $s_i' \sim P(\cdot|s_i, a_i)$ and $c_i = c(s_i, a_i)$. For any (measurable) function $f : \mathcal{X} \mapsto \mathbb{R}$, define the μ-weighted ℓ_2 norm of f as $\|f\|_\mu^2 = \int_{\mathcal{X}} f(s,a)^2 \mu(ds, da) = \int_{\mathcal{X}} f(s,a)^2 \mu_x(ds)\pi_{\mathrm{D}}(a|ds)$. Similarly for any other state-action distribution ρ, $\|f\|_\rho^2 = \int_{\mathcal{X}} f(s,a)^2 \rho(ds, da)$.

Inherent Bellman error

FQE depends on a chosen function class F to approximate $Q(s,a)$. To express how well the Bellman operator $\mathbb{T}g$ can be approximated by a function in the policy class F, when $\mathbb{T}g \notin \mathrm{F}$, a notion of distance, known as inherent Bellman error was first proposed by [147] and used in the analysis of related ADP algorithms [150, 148, 11, 12, 119, 118, 120, 140].

Definition C.1.1 (Inherent Bellman Error)**.** Given a function class F and a chosen distribution ρ, the *inherent Bellman error* of F is defined as:

$$d_{\mathrm{F}} = d(\mathrm{F}, \mathbb{T}\mathrm{F}) = \sup_{h \in \mathrm{F}} \inf_{f \in \mathrm{F}} \|f - \mathbb{T}h\|_\rho ,$$

where $\|\cdot\|_\rho$ is the $\rho-$weighted ℓ_2 norm and $\mathbb{T}$ is the Bellman optimality operator defined in (C.1).

To analyze FQE, we will form a similar definition for the Bellman evaluation operator

Definition C.1.2 (Inherent Bellman Evaluation Error)**.** Given a function class F and a policy π, the *inherent Bellman evaluation error* of F is defined as:

$$d_{\mathrm{F}}^\pi = d(\mathrm{F}, \mathbb{T}^\pi\mathrm{F}) = \sup_{h \in \mathrm{F}} \inf_{f \in \mathrm{F}} \|f - \mathbb{T}^\pi h\|_{\rho_\pi} ,$$

where $\|\cdot\|_{\rho_\pi}$ is the ℓ_2 norm weighted by ρ_π. ρ_π is defined as the state-action distribution induced by policy π, and $\mathbb{T}^\pi$ is the Bellman operator defined in (C.2)

Concentration coefficients

Let P^π denote the operator acting on $f : \mathcal{X} \mapsto \mathbb{R}$ such that $(P^\pi f)(s,a) = \int_{\mathcal{S}} f(s', \pi(s'))P(s'|s,a)ds'$. Acting on f (e.g., approximates Q), P^π captures the transition dynamics of taking action a and following π thereafters.

The following definition and assumption are standard in the analysis of related approximate dynamic programming algorithms [118, 150, 11]. As approximate value iteration and policy iteration algorithms perform policy update, the new policy at each round will induce a different stationary state-action distribution.

One way to quantify the distribution shift is the notion of concentrability coefficient of future state-action distribution, a variant of the notion introduced by [147].

Definition C.1.3 (Concentration coefficient of state-action distribution)**.** Given data generating distribution $\mu \sim \pi_{\mathrm{D}}$, initial state distribution d_0. For $m \geq 0$, and an arbitrary sequence of stationary policies $\{\pi_m\}_{m\geq 1}$ let

$$\beta_\mu(m) = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m})}{d\mu} \right\|_\infty .$$

($\beta_\mu(m) = \infty$ if the future state distribution $d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m}$ is not absolutely continuous w.r.t. μ, i.e, $d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m}(s,a) > 0$ for some $\mu(s,a) = 0$)

Assumption C.1. $\beta_\mu = (1-\gamma)^2 \sum\limits_{m\geq 1} m\gamma^{m-1}\beta_\mu(m) < \infty$.

Combination Lock Example. An example of an MDP that violates Assumption C.1 is the "combination lock" example proposed by [109]. In this finite MDP, we have N states $\mathcal{S} = \{1, 2, \ldots, N\}$, and 2 actions: going L or R. The initial state is $s_0 = 1$. In any state x, action L takes agent back to initial state s_0, and action R advances the agent to the next state $x + 1$ in a chain fashion. Suppose that the reward is 0 everywhere except for the very last state N. One can see that for an MDP such that any behavior policy π_D that has a bounded from below probability of taking action L from any state x, i.e., $\pi_D(L|s) \geq \nu > 0$, then it takes an exponential number of trajectories to learn or evaluate a policy that always takes action R. In this setting, we can see that the concentration coefficient β_μ can be designed to be arbitrarily large.

Complexity measure of function class F

Definition C.1.4 (Random L_1 Norm Covers)**.** Let $\epsilon > 0$, let F be a set of functions $\mathcal{S} \mapsto \mathbb{R}$, let $s_1^n = (s_1, \ldots, s_n)$ be n fixed points in $\mathcal{S}$. Then a collection of functions $\mathrm{F}_\epsilon = \{f_1, \ldots, f_N\}$ is an ϵ-cover of F on s_1^n if

$$\forall f \in \mathrm{F}, \exists f' \in \mathrm{F}_\epsilon : |\frac{1}{n}\sum_{i=1}^n f(s_i) - \frac{1}{n}\sum_{i=1}^n f'(s_i)| \leq \epsilon.$$

The empirical covering number, denote by $\mathcal{N}_1(\epsilon, \mathrm{F}, s_1^n)$, is the size of the smallest ϵ-cover on s_1^n. Take $\mathcal{N}_1(\epsilon, \mathrm{F}, s_1^n) = \infty$ if no finite ϵ-cover exists.

Definition C.1.5 (Pseudo-Dimension)**.** A real-valued function class F has pseudo-dimension $\mathtt{dim}_{\mathrm{F}}$ defined as the VC dimension of the function class

induced by the sub-level set of functions of F. In other words, define function class $\mathrm{H} = \{(s, y) \mapsto \mathrm{sign}(f(s) - y) : f \in \mathrm{F}\}$, then

$$\texttt{dim}_{\mathrm{F}} = \texttt{VC-dimension}(\mathrm{H}).$$

C.2 Generalization Analysis of Fitted Q Evaluation

In this section we prove the following statement for Fitted Q Evaluation (FQE).

Theorem C.2.1 (Guarantee for FQE - General Case (theorem 5.2.1))**.** *Under Assumption C.1, for $\epsilon > 0$ & $\delta \in (0, 1)$, after K iterations of Fitted Q Evaluation (Algorithm 3), for $n = O\Big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \texttt{dim}_{\mathrm{F}} \log\frac{\overline{C}^2}{\epsilon^2} + \log \texttt{dim}_{\mathrm{F}})\Big)$, we have with probability $1 - \delta$:*

$$\Big|Q(\pi) - Q_K(\pi)\Big| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\Big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}}^{\pi} + \epsilon) + \frac{2\gamma^{K/2}\overline{C}}{(1-\gamma)^{1/2}}\Big).$$

Theorem C.2.2 (Guarantee for FQE - Bellman Realizable Case)**.** *Under Assumptions C.1-C.2, for any $\epsilon > 0, \delta \in (0, 1)$, after K iterations of Fitted Q Evaluation (Algorithm 3), when $n \geq \frac{24\cdot 214\cdot\overline{C}^4}{\epsilon^2}\Big(\log\frac{K}{\delta} + \texttt{dim}_{\mathrm{F}} \log\frac{320\overline{C}^2}{\epsilon^2} + \log(14e(\texttt{dim}_{\mathrm{F}} + 1))\Big)$, we have with probability $1 - \delta$:*

$$\Big|Q(\pi) - Q_K(\pi)\Big| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\Big(\sqrt{\beta_\mu}\epsilon + \frac{2\gamma^{K/2}\overline{C}}{(1-\gamma)^{1/2}}\Big).$$

We first focus on theorem C.2.2, analyzing FQE assuming a sufficiently rich function class F so that the Bellman evaluation update $\mathbb{T}^\pi$ is closed wrt F (thus inherent Bellman evaluation error is 0). We call this the *Bellman evaluation realizability assumption.* This assumption simplifies the presentation of our bounds.

After analyzing FQE under this Bellman realizable setting, we will turn to error bound for general, non-realizable setting in theorem C.2.1 (also theorem 5.2.1). The main difference in the non-realizable setting is the appearance of an extra term d_{F}^{π} our final bound.

Error bound for single iteration - Bellman realizable case

Assumption C.2 (Bellman evaluation realizability)**.** We consider function classes F sufficiently rich so that $\forall f, \mathbb{T}^\pi f \in \mathrm{F}$.

We begin with the following result bounding the error for a single iteration of FQE, under "training" distribution $\mu \sim \pi_D$.

Proposition C.2.3 (Error bound for single iteration)**.** *Let the functions in* F *also be bounded by* $\overline{C}$*, and let* $\texttt{dim}_{\text{F}}$ *denote the pseudo-dimension of the function class* F*. We have with probability at least* $1-\delta$*:*

$$\|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu < \epsilon,$$

when $n \geq \frac{24\cdot 214\cdot \overline{C}^4}{\epsilon^2}\left(\log\frac{1}{\delta} + \texttt{dim}_{\text{F}}\log\frac{320\overline{C}^2}{\epsilon^2} + \log(14e(\texttt{dim}_{\text{F}}+1))\right)$

Remark C.2.4. *Note from proposition C.2.3 that the dependence of sample complexity* n *here on* ϵ *is* $\widetilde{O}(\frac{1}{\epsilon^2})$*, which is better than previously known analysis for Fitted Value Iteration [150] and FittedPolicyQ (continuous version of Fitted Q Iteration [11]) dependence of* $\widetilde{O}(\frac{1}{\epsilon^4})$*. The finite sample analysis of LSTD [119] showed an* $\widetilde{O}(\frac{1}{\epsilon^2})$ *dependence using linear function approximation. Here we prove similar convergence rate for general non-linear (bounded) function approximators.*

Proof of Proposition C.2.3. Recall the training target in round k is $y_i = c_i + \gamma Q_{k-1}(s_i', \pi(s_i'))$ for $i = 1, 2, \ldots, n$, and $Q_k \in$ F is the solution to the following regression problem:

$$Q_k = \arg\min_{f\in\text{F}} \frac{1}{n}\sum_{i=1}^{n}(f(s_i,a_i) - y_i)^2.$$

Consider random variables $(s,a) \sim \mu$ and $y = c(s,a) + \gamma Q_{k-1}(s', \pi(s'))$ where $s' \sim p(\cdot|s,a)$. By this definition, $\mathbb{T}^\pi Q_{k-1}$ is the *regression function* that minimizes square loss $\min_{h:\mathbb{R}^{X\times A}\mapsto\mathbb{R}} \mathbb{E}|h(s,a)-y|^2$ out of all functions h (not necessarily in F). This is due to $(\mathbb{T}^\pi Q_{k-1})(\tilde{s},\tilde{a}) = \mathbb{E}\left[y|s=\tilde{s}, a=\tilde{a}\right]$ by definition of the Bellman operator. Consider Q_{k-1} fixed and we now want to relate the learned function Q_k over finite set of n samples with the regression function over the whole data distribution via uniform deviation bound. We use the following lemma:

Lemma C.2.5 ([76], theorem 11.4. Original version [124], theorem 3)**.** *Consider random vector* (X,Y) *and* n *i.i.d samples* (X_i, Y_i)*. Let* $m(x)$ *be the (optimal) regression function under square loss* $m(x) = \mathbb{E}[Y|X=x]$*. Assume* $|Y| \leq B$ *a.s. and* $B \leq 1$*. Let* F *be a set of function* $f : \mathbb{R}^d \mapsto \mathbb{R}$ *and let* $|f(x)| \leq B$*.*

Then for each $n \geq 1$

$$\mathbf{P}\Bigg\{\exists f \in \mathrm{F} : \mathbb{E}|f(X)-Y|^2 - \mathbb{E}|m(X)-Y|^2 - \frac{1}{n}\sum_{i=1}^{n}\Bigg(|f(X_i)-Y_i|^2 - |m(X_i)-Y_i|^2\Bigg) \geq \epsilon \cdot \Big(\alpha + \beta + \mathbb{E}|f(X)-Y|^2 - \mathbb{E}|m(X)-Y|^2\Big)\Bigg\} \leq 14 \sup_{x_1^n} \mathcal{N}_1\left(\frac{\beta\epsilon}{20B}, \mathrm{F}, x_1^n\right) \exp\left(-\frac{\epsilon^2(1-\epsilon)\alpha n}{214(1+\epsilon)B^4}\right),$$

where $\alpha, \beta > 0$ *and* $0 < \epsilon < 1/2$

To apply this lemma, first note that since $\mathbb{T}^\pi Q_{k-1}$ is the optimal regression function[1], we have:

$$\begin{aligned}
&\mathbb{E}_\mu\left[(Q_k(s,a)-y)^2\right] \\
&= \mathbb{E}_\mu\left[(Q_k(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a) + \mathbb{T}^\pi Q_{k-1}(s,a) - y)^2\right] \\
&= \mathbb{E}_\mu\left[(Q_k(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a))^2\right] + \mathbb{E}_\mu[(\mathbb{T}^\pi Q_{k-1}(s,a) - y)^2\big].
\end{aligned}$$

Thus

$$\begin{aligned}
\|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu^2 &= \mathbb{E}\left[(Q_k(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a))^2\right] \\
&= \mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right],
\end{aligned}$$

where by definition

$$\begin{aligned}
\mathbb{E}\left[(Q_k(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a))^2\right] &= \int (Q_k(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a))^2 \, \mu(dx,da) \\
&= \int (Q_k(s,a) - \mathbb{T}^\pi(s,a))^2 \mu_x(dx)\pi_{\mathrm{D}}(a|dx)
\end{aligned}$$

Next, given a fixed data set $\widetilde{D}_k \sim \mu$,

$$\mathbf{P}\Big\{\|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu^2 > \epsilon\Big\} \tag{C.3}$$

$$\begin{aligned}
&= \mathbf{P}\Big\{\mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right] > \epsilon\Big\} \\
&\leq \mathbf{P}\Bigg\{\mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right] \\
&\qquad - 2\cdot\left(\frac{1}{n}\sum_{i=1}^{n}(Q_k(s_i,a_i)-y_i)^2 - \frac{1}{n}\sum_{i=1}^{n}(\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right) > \epsilon\Bigg\}
\end{aligned} \tag{C.4}$$

[1]It is easy to see that if $m(x) = \mathbb{E}[y|s]$ is the regression function then for any function $f(x)$, we have $\mathbb{E}\left[(f(x)-m(x))(m(x)-y) = 0\right]$

$$= \mathbf{P}\Bigg\{ \mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right] - \frac{1}{n}\sum_{i=1}^{n}\left[(Q_k(s_i,a_i)-y_i)^2 - (\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right] > \frac{1}{2}(\epsilon + \mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right])\Bigg\} \quad \text{(C.5)}$$

$$\leq \mathbf{P}\Bigg\{ \exists f \in \mathrm{F} : \mathbb{E}\left[(f(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right] - \frac{1}{n}\sum_{i=1}^{n}\left[(f(s_i,a_i)-y_i)^2 - (\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right] \geq \frac{1}{2}(\frac{\epsilon}{2} + \frac{\epsilon}{2} + \mathbb{E}\left[(f(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right])\Bigg\}$$

$$\leq 14 \sup_{x_1^n} \mathcal{N}_1\left(\frac{\epsilon}{80\overline{C}}, \mathrm{F}, x_1^n\right) \cdot \exp\left(-\frac{n\epsilon}{24\cdot 214\overline{C}^4}\right). \quad \text{(C.6)}$$

Equation (C.4) uses the definition of $Q_k = \arg\min_{f\in\mathrm{F}} \frac{1}{n}\sum_{i=1}^{n}(f(s_i,a_i)-y_i)^2$ and the fact that $\mathbb{T}^\pi Q_{k-1} \in \mathrm{F}$, thus making the extra term a positive addition. Equation (C.5) is due to rearranging the terms. Equation (C.6) is an application of lemma C.2.5. We can further bound the empirical covering number by invoking the following lemma due to Haussler [87]:

Lemma C.2.6 ([87], Corollary 3)**.** *For any set X, any points $x^{1:n} \in \mathcal{X}^n$, any class* F *of functions on X taking values in $[0,\overline{C}]$ with pseudo-dimension $\mathtt{dim}_\mathrm{F} < \infty$, and any $\epsilon > 0$:*

$$\mathcal{N}_1(\epsilon, \mathrm{F}, x_1^n) \leq e(\mathtt{dim}_\mathrm{F}+1)\left(\frac{2e\overline{C}}{\epsilon}\right)^{\mathtt{dim}_\mathrm{F}}.$$

Applying lemma C.2.6 to equation (C.6), we have the inequality

$$\mathbf{P}\Big\{ \|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu^2 > \epsilon \Big\} \leq 14\cdot e\cdot(\mathtt{dim}_\mathrm{F}+1)\left(\frac{320\overline{C}^2}{\epsilon}\right)^{\mathtt{dim}_\mathrm{F}} \cdot \exp\left(-\frac{n\epsilon}{24\cdot 214\overline{C}^4}\right). \quad \text{(C.7)}$$

Thus, when $n \geq \frac{24\cdot 214\cdot\overline{C}^4}{\epsilon^2}\left(\log\frac{1}{\delta} + \mathtt{dim}_\mathrm{F}\log\frac{320\overline{C}^2}{\epsilon^2} + \log(14e(\mathtt{dim}_\mathrm{F}+1))\right)$:

$$\|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\rho < \epsilon,$$

with probability at least $1-\delta$. Notice that the dependence of sample complexity n here on ϵ is $\widetilde{O}(\frac{1}{\epsilon^2})$, which is better than previously known analyses for other approximate dynamic programming algorithms such as Fitted Value Iteration [150], FittedPolicyQ [12, 11] with dependence of $O(\frac{1}{\epsilon^4})$.

Error bound for single iteration - Bellman non-realizable case

We now give similar error bound for the general case, where Assumption C.2 does not hold. Consider the decomposition

$$\begin{aligned}
&\|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu^2 \qquad\qquad (C.8)\\
&= \mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right]\\
&= \Bigg\{\mathbb{E}\left[(Q_k(s,a)-y)^2\right] - \mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right]\\
&\qquad - 2\cdot\left(\frac{1}{n}\sum_{i=1}^n (Q_k(s_i,a_i)-y_i)^2 - \frac{1}{n}\sum_{i=1}^n (\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right)\Bigg\}\\
&\qquad + \left\{2\cdot\left(\frac{1}{n}\sum_{i=1}^n (Q_k(s_i,a_i)-y_i)^2 - \frac{1}{n}\sum_{i=1}^n (\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right)\right\}\\
&= \texttt{component_1} + \texttt{component_2}.
\end{aligned}$$

Splitting the probability of error into two separate bounds. We saw from the previous section (equation (C.7)) that

$$\mathbf{P}(\texttt{component\_1} > \epsilon/2) \le 14\cdot e\cdot(\texttt{dim}_\text{F}+1)\left(\frac{640\overline{C}^2}{\epsilon}\right)^{\texttt{dim}_\text{F}}\cdot\exp\left(-\frac{n\epsilon}{48\cdot 214\overline{C}^4}\right). \qquad (C.9)$$

We no longer have $\texttt{component\_2} \le 0$ since $\mathbb{T}^\pi Q_{k-1} \notin \text{F}$. Let

$$f^* \equiv \arg\inf_{f\in\text{F}} \|f - \mathbb{T}^\pi Q_{k-1}\|_\mu^2 .$$

Since $Q_k = \arg\min_{f\in\text{F}} \frac{1}{n}\sum_{i=1}^n (f(s_i,a_i)-y_i)^2$, we can upper-bound component_2 by

$$\texttt{component\_2} \le 2\cdot\left(\frac{1}{n}\sum_{i=1}^n (f^*(s_i,a_i)-y_i)^2 - \frac{1}{n}\sum_{i=1}^n (\mathbb{T}^\pi Q_{k-1}(s_i,a_i)-y_i)^2\right)$$

We can treat f^* as a fixed function, unlike random function Q_k, and use standard concentration inequalities to bound the empirical average from the expectation. Let random variable $z = ((s,a),y)$, $z_i = ((s_i,a_i),y_i), i = 1,\ldots,n$ and let

$$h(z) = (f^*(s,a)-y)^2 - (\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2.$$

We have $|h(z)| \le 4\overline{C}^2$. We will derive a bound for

$$\mathbf{P}\left(\frac{1}{n}\sum_{i=1}^n h(z_i) - \mathbb{E}h(z) > \frac{\epsilon}{4} + \mathbb{E}h(z)\right),$$

using Bernstein inequality[145]. First, using the relationship $h(z) = (f^*(s,a) + \mathbb{T}^\pi Q_{k-1}(s,a) - 2y)(f^*(s,a) - \mathbb{T}^\pi Q_{k-1}(s,a))$, the variance of $h(z)$ can be bounded

by a constant factor of $\mathbb{E}h(z)$, since

$$\begin{aligned}\mathbf{Var}(h(z)) &\le \mathbb{E}h(z)^2 \le 16\overline{C}^2\mathbb{E}\left[(f^*(s,a)-\mathbb{T}^\pi Q_{k-1}(s,a))^2\right] \\ &= 16\overline{C}^2\left(\mathbb{E}\left[(f^*(s,a)-y)^2\right]-\mathbb{E}\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right]\right) \quad \text{(C.10)} \\ &= 16\overline{C}^2\mathbb{E}h(z). \quad \text{(C.11)}\end{aligned}$$

Equation (C.10) stems from $\mathbb{T}^\pi Q_{k-1}$ being the optimal regression function. Now we can apply equation (C.11) and Bernstein inequality to obtain

$$\begin{aligned}&\mathbf{P}\left(\frac{1}{n}\sum_{i=1}^n h(z_i)-\mathbb{E}h(z) > \frac{\epsilon}{4}+\mathbb{E}h(z)\right) \\ &\le \mathbf{P}\left(\frac{1}{n}\sum_{i=1}^n h(z_i)-\mathbb{E}h(z) > \frac{\epsilon}{4}+\frac{\mathbf{Var}(h(z))}{16\overline{C}^2}\right) \le \dots \\ &\le \exp\left(-\frac{n\left(\frac{\epsilon}{4}+\frac{\mathbf{Var}}{16\overline{C}^2}\right)^2}{2\mathbf{Var}+2\frac{4\overline{C}^2}{3}\left(\frac{\epsilon}{4}+\frac{\mathbf{Var}}{16\overline{C}^2}\right)}\right) \le \exp\left(-\frac{n\left(\frac{\epsilon}{4}+\frac{\mathbf{Var}}{16\overline{C}^2}\right)^2}{\left(32\overline{C}^2+\frac{8\overline{C}^2}{3}\right)\left(\frac{\epsilon}{4}+\frac{\mathbf{Var}}{16\overline{C}^2}\right)}\right) \\ &= \exp\left(-\frac{n\left(\frac{\epsilon}{4}+\frac{\mathbf{Var}}{16\overline{C}^2}\right)}{32\overline{C}^2+\frac{8\overline{C}^2}{3}}\right) \le \exp\left(-\frac{1}{128+\frac{32}{3}}\cdot\frac{n\epsilon}{\overline{C}^2}\right).\end{aligned}$$

Thus

$$\mathbf{P}\left(2\cdot\left[\frac{1}{n}\sum_{i=1}^n h(z_i)-2\mathbb{E}h(z)\right] > \frac{\epsilon}{2}\right) \le \exp\left(-\frac{3}{416}\cdot\frac{n\epsilon}{\overline{C}^2}\right). \quad \text{(C.12)}$$

Now we have

$$\texttt{component\_2} \le 2\cdot\frac{1}{n}\sum_{i=1}^n h(z_i) = 2\cdot\left[\frac{1}{n}\sum_{i=1}^n h(z_i)-2\mathbb{E}h(z)\right]+4\mathbb{E}h(z).$$

Using again the fact that $\mathbb{T}^\pi Q_{k-1}$ is the optimal regression function

$$\begin{aligned}\mathbb{E}h(z) &= \mathbb{E}_D\left[(f^*(s,a)-y)^2\right]-\mathbb{E}_D\left[(\mathbb{T}^\pi Q_{k-1}(s,a)-y)^2\right] \\ &= \mathbb{E}_D\left[(f^*(s,a)-\mathbb{T}^\pi Q_{k-1}(s,a))^2\right] \\ &= \inf_{f\in\mathrm{F}}\|f-\mathbb{T}^\pi Q_{k-1}\|_\mu^2. \quad \text{(C.13)}\end{aligned}$$

Combining equations (C.9), (C.12) and (C.13), we can conclude that

$$\begin{aligned}&\mathbf{P}\Big\{\|Q_k-\mathbb{T}^\pi Q_{k-1}\|_\mu^2-4\inf_{f\in\mathrm{F}}\|f-\mathbb{T}^\pi Q_{k-1}\|_\mu^2 > \epsilon\Big\} \quad \text{(C.14)} \\ &\le 14\cdot e\cdot(\texttt{dim}_\mathrm{F}+1)\left(\frac{640\overline{C}^2}{\epsilon}\right)^{\texttt{dim}_\mathrm{F}}\cdot\exp\left(-\frac{n\epsilon}{48\cdot 214\overline{C}^4}\right)+\exp\left(-\frac{3}{416}\cdot\frac{n\epsilon}{\overline{C}^2}\right),\end{aligned}$$

implying

$$\begin{aligned}&\mathbf{P}\Big\{\|Q_k-\mathbb{T}^\pi Q_{k-1}\|_\mu-2\inf_{f\in\mathrm{F}}\|f-\mathbb{T}^\pi Q_{k-1}\|_\mu > \epsilon\Big\} \quad \text{(C.15)} \\ &\le 14\cdot e\cdot(\texttt{dim}_\mathrm{F}+1)\left(\frac{640\overline{C}^2}{\epsilon^2}\right)^{\texttt{dim}_\mathrm{F}}\cdot\exp\left(-\frac{n\epsilon^2}{48\cdot 214\overline{C}^4}\right)+\exp\left(-\frac{3}{416}\cdot\frac{n\epsilon^2}{\overline{C}^2}\right). \quad \text{(C.16)}\end{aligned}$$

We now can further upper-bound the term

$$2 \inf_{f \in \mathrm{F}} \|f - \mathbb{T}^\pi Q_{k-1}\|_\mu \leq 2 \sup_{f' \in \mathrm{F}} \inf_{f \in \mathrm{F}} \|f - \mathbb{T}^\pi f'\|_\mu = 2 d_{\mathrm{F}}^\pi,$$

(the worst-case *inherent Bellman evaluation error*), leading to the final bound for the Bellman non-realizable case.

One may wish to further remove the inherent Bellman evaluation error from our error bound. However, counter-examples exist where the inherent Bellman error cannot generally be estimated using function approximation (see section 11.6 of [191]). Fortunately, inherent Bellman error can be driven to be small by choosing rich function class F (low bias), at the expense of more samples requirement (higher variance, through higher pseudo-dimension $\mathtt{dim}_{\mathrm{F}}$).

While the bound in (C.16) looks more complicated than the Bellman realizable case in equation C.7, note that the convergence rate will still be $O(\frac{1}{n^2})$.

Bounding the error across iterations

Previous sub-sections C.2 and C.2 have analyzed the error of FQE for a single iteration in Bellman realizable and non-realizable case. We now analyze how errors from different iterations flow through the FQE algorithm. The proof borrows the idea from lemma 3 and 4 of [150] for fitted value iteration (for value function V instead of Q), with appropriate modifications for our off-policy evaluation context.

Recall that C^π, Q^π denote the true value function and action-value function, respectively, under the evaluation policy π. And $C_K = \mathbb{E}[Q_K(s, \pi(s))]$ denote the value function associated with the returned function Q_K from algorithm 3. Our goal is to bound the difference $C^\pi - C_K$ between the true value function and the estimated value of the returned function Q_K.

Let the unknown state-action distribution induced by the evaluation policy π be ρ. We first bound the loss $\|Q^\pi - Q_K\|_\rho$ under the "test-time" distribution ρ of (s, a), which differs from the state-action μ induced by data-generating policy π_{D}. We will then lift the loss bound from Q_K to C_K.

Step 1: Upper-bound the value estimation error

Let $\epsilon_{k-1} = Q_k - \mathbb{T}^\pi Q_{k-1} \in \mathcal{X}, \bar{\mathcal{C}}$. We have for every k that

$$\begin{aligned} Q^\pi - Q_k &= \mathbb{T}^\pi Q^\pi - \mathbb{T}^\pi Q_{k-1} + \epsilon_{k-1} \quad (Q^\pi \text{ is fixed point of } T^\pi) \\ &= \gamma P^\pi (Q^\pi - Q_{k-1}) + \epsilon_{k-1}. \end{aligned}$$

Thus by simple recursion

$$\begin{aligned} Q^\pi - Q_K &= \sum_{k=0}^{K-1} \gamma^{K-k-1}(P^\pi)^{K-k-1}\epsilon_k + \gamma^K (P^\pi)^K (Q^\pi - Q_0) \\ &= \frac{1-\gamma^{K+1}}{1-\gamma}\left[\sum_{k=0}^{K-1} \frac{(1-\gamma)\gamma^{K-k-1}}{1-\gamma^{K+1}}(P^\pi)^{K-k-1}\epsilon_k \right. \\ &\qquad \left. + \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}}(P^\pi)^K (Q^\pi - Q_0)\right] \\ &= \frac{1-\gamma^{K+1}}{1-\gamma}\left[\sum_{k=0}^{K-1} \alpha_k A_k \epsilon_k + \alpha_K A_K (Q^\pi - Q_0)\right], \end{aligned} \tag{C.17}$$

where for simplicity of notations, we denote

$$\begin{aligned} \alpha_k &= \frac{(1-\gamma)\gamma^{K-k-1}}{1-\gamma^{K+1}} \text{ for } k < K, \alpha_K = \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}} \\ A_k &= (P^\pi)^{K-k-1}, A_K = (P^\pi)^K. \end{aligned}$$

Note that A_k's are probability kernels and α_k's are deliberately chosen such that $\sum_k \alpha_k = 1$.

We can apply point-wise absolute value on both sides of (C.17) with $|f|$ being the short-hand notation for $|f(s,a)|$ and inequality holds point-wise. By triangle inequalities:

$$|Q^\pi - Q_K| \le \frac{1-\gamma^{K+1}}{1-\gamma}\left[\sum_{k=0}^{K-1} \alpha_k A_k |\epsilon_k| + \alpha_K A_K |Q^\pi - Q_0|\right]. \tag{C.18}$$

Step 2: Bounding $\|Q^\pi - Q_K\|_\rho$ for any unknown distribution ρ. To handle distribution shift from μ to ρ, we decompose the loss as follows:

$$\begin{aligned} &\|Q^\pi - Q_K\|_\rho^2 = \int \rho(dx, da)\left(Q^\pi(s,a) - Q_K(s,a)\right)^2 \\ &\le \left[\frac{1-\gamma^{K+1}}{1-\gamma}\right]^2 \int \left[\left(\sum_{k=0}^{K-1} \alpha_k A_k |\epsilon_k| + \alpha_K A_K |Q^\pi - Q_0|\right)(s,a)\right]^2 \text{ (from}(C.18)\text{)} \\ &\le \left[\frac{1-\gamma^{K+1}}{1-\gamma}\right]^2 \int \left[\sum_{k=0}^{K-1} \alpha_k (A_k \epsilon_k)^2 + \alpha_K (A_K (Q^* - Q_0))^2\right](s,a) \text{ (Jensen)} \\ &\le \left[\frac{1-\gamma^{K+1}}{1-\gamma}\right]^2 \int \left[\sum_{k=0}^{K-1} \alpha_k A_k \epsilon_k^2 + \alpha_K A_K (Q^* - Q_0)^2\right](s,a). \text{ (Jensen)} \end{aligned}$$

Using assumption C.1 (assumption 5.1), we can bound each term ρA_k as:

$$\rho A_k = \rho (P^\pi)^{K-k-1} \le \mu \beta_\mu (K-k-1). \text{ (definition C.1.3)}$$

Thus

$$\frac{\|Q^\pi - Q_K\|_\rho^2}{\left[\frac{1-\gamma^{K+1}}{1-\gamma}\right]^2} \le \left[\frac{1-\gamma}{1-\gamma^{K+1}} \sum_{k=0}^{K-1} \gamma^{K-k-1} \beta_\mu (K-k-1) \|\epsilon_k\|_\mu^2 + \alpha_K (2\overline{C})^2\right].$$

Assumption C.1 (stronger than necessary for proof of FQE) can be used to

upper-bound the first order concentration coefficient:

$$(1-\gamma)\sum_{m\geq 0}\gamma^m\beta_\mu(m) \leq \frac{\gamma}{1-\gamma}\left[(1-\gamma)^2\sum_{m\geq 1}m\gamma^{m-1}\beta_\mu(m)\right] = \frac{\gamma}{1-\gamma}\beta_\mu.$$

This gives the upper-bound for $\|Q^\pi - Q_K\|_\rho^2$ as

$$\begin{aligned}\|Q^\pi - Q_K\|_\rho^2 &\leq \left[\frac{1-\gamma^{K+1}}{1-\gamma}\right]^2\left[\frac{\gamma}{(1-\gamma)(1-\gamma^{K+1})}\beta_\mu\max_k\|\epsilon_k\|_\mu^2 \right. \\ &\qquad \left. + \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}}(2\overline{C})^2\right] \\ &\leq \frac{1-\gamma^{K+1}}{(1-\gamma)^2}\left[\frac{\gamma}{1-\gamma}\beta_\mu\max_k\|\epsilon_k\|_\mu^2 + (1-\gamma)\gamma^K(2\overline{C})^2\right] \\ &\leq \frac{\gamma}{(1-\gamma)^3}\beta_\mu\max_k\|\epsilon_k\|_\mu^2 + \frac{\gamma^K}{1-\gamma}(2\overline{C})^2.\end{aligned} \tag{C.19}$$

Using $a^2+b^2 \leq (a+b)^2$ for nonnegative a, b, we conclude that

$$\|Q^\pi - Q_K\|_\rho \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\left(\sqrt{\beta_\mu}\max_k\|\epsilon_k\|_\mu + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}}2\overline{C}\right). \tag{C.20}$$

Step 3: Desired Bound Now we can choose ρ to be the state-action distribution by the evaluation policy π. The error bound on the value function Q follows simply by integrating inequality (C.20) over state-action pairs induced by π. The final error across iterations can be related to individual iteration error by

$$|Q^\pi - Q_K| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\left(\sqrt{\beta_\mu}\max_k\|\epsilon_k\|_\mu + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}}2\overline{C}\right). \tag{C.21}$$

Finite-sample guarantees for Fitted Q Evaluation

Combining results from (C.7), (C.16) and (C.21), we have the final guarantees for FQE under both realizable and general cases.

Realizable Case - Proof of theorem C.2.2. From (C.7), when $n \geq N^* \equiv \frac{24\cdot 214\cdot\overline{C}^4}{\epsilon^2}\left(\log\frac{K}{\delta} + \mathtt{dim}_{\mathrm{F}}\log\frac{320\overline{C}^2}{\epsilon^2} + \log(14e(\mathtt{dim}_{\mathrm{F}}+1))\right)$, we have $\|\epsilon_k\|_\mu < \epsilon$ with probability at least $1-\delta/K$ for any $0 \leq k < K$. Thus we conclude that for any $\epsilon > 0, 0 < \delta < 1$, after K iterations of Fitted Q Evaluation, the value estimate returned by Q_K satisfies:

$$|Q^\pi - Q_K| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\left(\sqrt{\beta_\mu}\epsilon + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}}2\overline{C}\right)$$

holds with probability $1-\delta$ when $n \geq N^*$. This concludes the proof of theorem C.2.2.

Non-realizable Case - Proof of theorem C.2.1 and theorem 5.2.1.

Similarly, from (C.16) we have

$$\mathbf{P}\Big\{ \|Q_k - \mathbb{T}^\pi Q_{k-1}\|_\mu - 2 \inf_{f\in \mathrm{F}} \|f - \mathbb{T}^\pi Q_{k-1}\|_\mu > \epsilon \Big\} \tag{C.22}$$

$$\leq 14 \cdot e \cdot (\mathtt{dim}_\mathrm{F} + 1) \left(\frac{640\overline{C}^2}{\epsilon^2}\right)^{\mathtt{dim}_\mathrm{F}} \cdot \exp\left(-\frac{n\epsilon^2}{48\cdot 214\overline{C}^4}\right) + \exp\left(-\frac{3}{416}\cdot\frac{n\epsilon^2}{\overline{C}^2}\right).$$

Since $\inf_{f\in \mathrm{F}} \|f - \mathbb{T}^\pi Q_{k-1}\|_\mu \leq \sup_{h\in \mathrm{F}} \inf_{f\in \mathrm{F}} \|f - \mathbb{T}^\pi h\|_\mu = d_\mathrm{F}^\pi$ (the *inherent Bellman evaluation error*), similar arguments to the realizable case lead to the conclusion that for any $\epsilon > 0, 0 < \delta < 1$, after K iterations of FQE:

$$|Q^\pi - Q_K| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}} \left(\sqrt{\beta_\mu}(2d_\mathrm{F}^\pi + \epsilon) + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}} 2\overline{C}\right)$$

holds with probability $1-\delta$ when $n = O\Big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \mathtt{dim}_\mathrm{F}\log\frac{\overline{C}^2}{\epsilon^2} + \log \mathtt{dim}_\mathrm{F})\Big)$, thus finishes the proof of theorem C.2.1.

Note that in both cases, the $\widetilde{O}(\frac{1}{\epsilon^2})$ dependency of n is significant improvement over previous finite-sample analysis of related approximate dynamic programming algorithms [150, 12, 11]. This dependency matches that of previous analysis using linear function approximators from [118, 119] for LSTD and LSPI algorithms. Here our analysis, using similar assumptions, is applicable for general non-linear, bounded function classes, which is an improvement over convergence rate of $O(\frac{1}{n^4})$ in related approximate dynamic programming algorithms [11, 12, 150].

Appendix D

CHAPTER 4 APPENDIX

D.1 Equivalence between Regularization and Constraint Satisfaction

Formulating Different Regularized Policy Learning Problems as Constrained Policy Learning

In this section, we provide connections between regularized policy learning and our constrained formulation (OPT). Although this chapter focuses on batch policy learning, here we are agnostic between online and batch learning settings.

Entropy regularized RL. The standard reinforcement learning objective, either in online or batch setting, is to find a policy π^*_{std} that minimizes the long-term cost (equivalent to maximizing the accumulated rewards): $\pi^*_{\text{std}} = \arg\min_\pi \sum_t \mathbb{E}_{(s_t,a_t)\sim\pi}[c(s_t,a_t)] = \arg\min_\pi \mathbb{E}_{(s,a)\sim\mu_\pi}[c(s,a)]$. Maximum entropy reinforcement learning [78] augments the cost with an entropy term, such that the optimal policy maximizes its entropy at each visited state: $\pi^*_{\text{MaxEnt}} = \arg\min_\pi \mathbb{E}_{(s,a)\sim\mu_\pi}[c(s,a)] - \lambda\mathbb{H}(\pi(\cdot|s))$. As discuseed by [78], the goal is for the agent to maximize the entropy of the entire trajectory, and not greedily maximizing entropy at the current time step (i.e., Boltzmann exploration). Maximum entropy policy learning was first proposed by [232, 231] in the context of learning from expert demonstrations. Entropy regulazed RL/IL is equivalent to our problem (OPT) by simply set $C(\pi) = \mathbb{E}_{(s_t,a_t)\sim\pi}[c(s_t,a_t)]$ (standard RL objective), and $g(s,a) = \pi(a|s)\log\pi(a|s)$, and thus $G(\pi) = -\mathbb{H}(\pi) \leq \tau$.

Smooth imitation learning (& Regularized imitation learning). This is a constrained imitation learning problem studied by [122]: learning to mimic smooth behavior in continuous space from human desmonstrations. The data collected from human demonstrations is considered to be fixed and given a priori, thus the imitation learning task is also a batch policy learning problem. The proposed approach from [122] is to view policy learning as a function regularization problem: policy $\pi = (f, g)$ is a combination of functions f and h, where f belongs to some expressive function class F (e.g., decision trees, neural networks) and $h \in$ H with certifiable smoothness property (e.g., linear models). Policy learning is the solution to the functional regularization

problem $\pi = \arg\min_{f,g} \mathbb{E}_{s\sim\mu_\pi} \|f(s) - \pi_E(s)\| + \lambda \|h(s) - \pi_E(s)\|$, where π_E is the expert policy. This constrained imitation learning setting is equivalent to our problem (OPT) as follows: $C(\pi) = C((f,h)) = \mathbb{E}_{s\sim\mu_\pi} \|f(s) - \pi_E(s)\|$ and $G(\pi) = G((f,h)) = \min_{h'\in \mathrm{H}} \|h'(s) - \pi_E(s)\| \leq \tau$.

Regularizing RL with expert demonstrations / Learning from imperfect demonstrations. Efficient exploration in RL is a well-known challenge. Expert demonstrations provide a way around online exploration to reduce the sample complexity for learning. However, the label budget for expert demonstrations may be limited, resulting in a sparse coverage of the state space compared to what the online RL agent can explore [91]. Additionally, expert demonstrations may be imperfect [157]. Some recent work proposed to regularize standard RL objective with some deviation measure between the learning policy and (sparse) expert data [91, 157, 88].

For clarity we focus on the regularized RL objective for Q-learning in [91], which is defined as $J(\pi) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q)$, where $J_{DQ}(Q)$ is the standard deep Q-learning loss, $J_n(Q)$ is the n-step return loss, $J_E(Q)$ is the imitation learning loss given by $J_E(Q) = \max_{a\in \mathrm{A}} [Q(s,a) + \ell(a_E, a) - Q(s, a_E)]$, and $J_{L2}(Q)$ is an L2 regularization loss applied to the Q-network to prevent overfitting to a small expert dataset. The regularization parameters λ's are obtained by hyperparameter tuning. This approach provides a bridge between RL and IL, whose objective functions are fundamentally different (see AggreVate by [175] for an alternative approach).

We can cast this problem into (OPT) as $C(\pi) = C_{DQ}(Q) + \lambda_3 C_{L2}(Q)$ (standard RL objective), and two constraints: $g_1(\pi) = \mathbb{E}_{s\sim\mu_\pi}[\max_{a\in \mathrm{A}} Q(s,a) + \ell(a_E, a) - Q(s, a_E)]$, and $g_2(s,a) = \mathbb{E}_{s\sim\mu_\pi}[c_t + \gamma c_{t+1} + \ldots + \gamma^{n-1} c_{t+n-1} + \min'_a \gamma^n Q(s_{t+n}, a') - Q(s_t, a)]$. Here g_1 captures the loss w.r.t. expert demonstrations and g_2 reflects the n-step return constraint.

More generally, one can define the imitation learning constraint as $G(\pi) = \mathbb{E}_{s\sim\mu_\pi} \ell(\pi(s), \pi_E(s))$ for an appropriate divergence definition between $\pi(s)$ and $\pi_E(s)$ (defined at states where expert demonstrations are available).

Conservative policy improvement. Many policy search algorithms perform small policy update steps, requiring the new policy π to stay within a neighborhood of the most recent policy iterate π_k to ensure learning stability [125, 179, 146, 3]. This simply corresponds to the definition of $G(\pi) = \texttt{distance}(\pi, \pi_k) \leq$

τ, where `distance` is typically KL-divergence or total variation distance between the distribution induced by π and π_k. For KL-divergence, the single timestep cost $g(s,a) = -\pi(a|s)\log(\frac{\pi_k(a|s)}{\pi(a|s)})$.

Equivalence of Regularization and Constraint Viewpoint - Proof of Proposition 6.2.1

`Regularization` $\implies$ `Constraint`: Let $\lambda > 0$ and π^* be optimal policy in `Regularization`. Set $\tau = G(\pi^*)$. Suppose that π^* is not optimal in `Constraint`. Then $\exists \pi \in \Pi$ such that $G(\pi) \leq \tau$ and $C(\pi) < C(\pi^*)$. We then have

$$C(\pi) + \lambda^\top G(\pi) < C(\pi^*) + \lambda^\top \tau = C(\pi^*) + \lambda^\top G(\pi^*),$$

which contradicts the optimality of π^* for `Regularization` problem. Thus π^* is also the optimal solution of the `Constraint` problem.

`Constraint` $\implies$ `Regularization`: Given τ and let π^* be the corresponding optimal solution of the `Constraint` problem. The Lagrangian of `Constraint` is given by $L(\pi, \lambda) = C(\pi) + \lambda^\top G(\pi), \lambda \geq 0$. Then $\pi^* = \arg\min_{\pi\in\Pi} \max_{\lambda\geq 0} L(\pi, \lambda)$. Let

$$\lambda^* = \arg\max_{\lambda\geq 0} \min_{\pi\in\Pi} L(\pi, \lambda).$$

Slater's condition implies strong duality. By strong duality and the strong max-min property [26], we can exchange the order of maximization and minimization. Thus π^* is the optimal solution of

$$\min_{\pi\in\Pi} \quad C(\pi) + (\lambda^*)^\top (G(\pi) - \tau).$$

Removing the constaint $(\lambda^*)^\top \tau$, we have that π^* is the optimal solution of the `Regularization` problem with $\lambda = \lambda^*$. And since $\pi^* \neq \arg\min_{\pi\in\Pi} C(\pi)$, we must have $\lambda^* \geq 0$.

D.2 Convergence Proofs

Convergence of Meta-algorithm - Proof of Proposition 6.3.1

Let us evaluate the empirical primal-dual gap of the Lagrangian after T iterations:

$$\max_{\lambda} L(\widehat{\pi}_T, \lambda) = \max_{\lambda} \frac{1}{T} \sum_t L(\pi_t, \lambda) \tag{D.1}$$

$$\leq \frac{1}{T} \sum_t L(\pi_t, \lambda_t) + \frac{o(T)}{T} \tag{D.2}$$

$$\leq \frac{1}{T} \sum_t L(\pi, \lambda_t) + \frac{o(T)}{T} \quad \forall \pi \in \Pi \tag{D.3}$$

$$= L(\pi, \widehat{\lambda}_T) + \frac{o(T)}{T} \quad \forall \pi. \tag{D.4}$$

Equations (D.1) and (D.4) are due to the definition of $\widehat{\pi}_T$ and $\widehat{\lambda}_T$ and linearity of $L(\pi, \lambda)$ wrt λ and the distribution over policies in Π. Equation (D.2) is due to the no-regret property of `Online-algorithm`. Equation (D.3) is true since π_t is best response wrt λ_t. Since equation (D.4) holds for all π, we can conclude that for T sufficiently large such that $\frac{o(T)}{T} \leq \omega$, we have $\max_\lambda L(\widehat{\pi}_T, \lambda) \leq \min_\pi L(\pi, \widehat{\lambda}_T) + \omega$, which will terminate the algorithm.

Note that we always have $\max_\lambda L(\widehat{\pi}_T, \lambda) \geq L(\widehat{\pi}_T, \widehat{\lambda}_T) \geq \min_\pi L(\pi, \widehat{\lambda}_T)$. Algo 4's convergence rate depends on the regret bound of the `Online-algorithm` procedure. Multiple algorithms exist with regret scaling as $\Omega(\sqrt{T})$ (e.g., online gradient descent with regularizer, variants of online mirror descent). In that case, the algorithm will terminate after $O(\frac{1}{\omega^2})$ iterations.

Empirical Convergence Analysis of Main Algorithm - Proof of Theorem 6.4.1

By choosing normalized exponentiated gradient as the online learning subroutine, we have the following regret bound after T iterations of the main algorithm 5 (Chapter 2 of [182]) for any $\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B$:

$$\frac{1}{T} \sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda) \leq \frac{1}{T} \sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda_t) + \frac{\frac{B \log(m+1)}{\eta} + \eta \overline{G}^2 BT}{T}. \tag{D.5}$$

Denote $\omega_T = \frac{\frac{B\log(m+1)}{\eta} + \eta\overline{G}^2 BT}{T}$ to simplify notations. By the linearity of $\widehat{L}(\pi, \lambda)$ in both π and λ, we have for any λ that

$$\begin{aligned}
\widehat{L}(\widehat{\pi}_T, \lambda) \overset{\text{linearity}}{=} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda) &\overset{\text{eqn (D.5)}}{\leq} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda_t) + \omega_T \\
&\overset{\text{best response } \pi_t}{\leq} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\widehat{\pi}_T, \lambda_t) + \omega_T \\
&\overset{\text{linearity}}{=} \widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) + \omega_T.
\end{aligned}$$

Since this is true for any λ, $\max_\lambda \widehat{L}(\widehat{\pi}_T, \lambda) \leq \widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) + \omega_T$.

On the other hand, for any policy π, we also have

$$\begin{aligned}
\widehat{L}(\pi, \widehat{\lambda}_T) \overset{\text{linearity}}{=} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\pi, \lambda_t) &\overset{\text{best response } \pi_t}{\geq} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda_t) \\
&\overset{\text{eqn (D.5)}}{\geq} \frac{1}{T}\sum_{t=1}^{T} \widehat{L}(\pi_t, \widehat{\lambda}_T) - \omega_T \\
&\overset{\text{linearity}}{=} \widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) - \omega_T.
\end{aligned}$$

Thus $\min_\pi \widehat{L}(\pi, \widehat{\lambda}_T) \geq \widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) - \omega_T$, leading to

$$\max_\lambda \widehat{L}(\widehat{\pi}_T, \lambda) - \min_\pi \widehat{L}(\pi, \widehat{\lambda}_T) \leq \widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) + \omega_T - (\widehat{L}(\widehat{\pi}_T, \widehat{\lambda}_T) - \omega_T) = 2\omega_T$$

After T iterations of the main algorithm 5, therefore, the empirical primal-dual gap is bounded by

$$\max_\lambda \widehat{L}(\widehat{\pi}_T, \lambda) - \min_\pi \widehat{L}(\pi, \widehat{\lambda}_T) \leq \frac{2\frac{B\log(m+1)}{\eta} + 2\eta\overline{G}^2 BT}{T}.$$

In particular, if we want the gap to fall below a desired threshold ω, setting the online learning rate $\eta = \frac{\omega}{4\overline{G}^2 B}$ will ensure that the algorithm converges after $\frac{16B^2\overline{G}^2\log(m+1)}{\omega^2}$ iterations.

D.3 End-to-end Generalization Analysis of Main Algorithm

In this section, we prove the following full statement of theorem 6.4.3. Note that to lessen notation, we define $\overline{V} = \overline{C} + B\overline{G}$ to be the bound of value functions under considerations in algorithm 5.

Theorem D.3.1 (Generalization bound of algorithm 5)**.** *Let π^* be the optimal policy to problem OPT. Let K be the number of iterations of FQE and FQI. Let $\widehat{\pi}$ be the policy returned by our main algorithm 5, with termination threshold ω. For any $\epsilon > 0, \delta \in (0,1)$, when $n \geq \frac{24\cdot 214\cdot \overline{V}^4}{\epsilon^2}\Big(\log\frac{K(m+1)}{\delta} + \mathtt{dim}_{\mathrm{F}}\log\frac{320\overline{V}^2}{\epsilon^2} + \log(14e(\mathtt{dim}_{\mathrm{F}}+1))\Big)$, we have with probability at least $1-\delta$:*

$$C(\widehat{\pi}) \leq C(\pi^*) + \omega + \frac{(4+B)\gamma}{(1-\gamma)^3}\Big(\sqrt{C_\rho\epsilon} + 2\gamma^{K/2}\overline{V}\Big)$$

and

$$G(\widehat{\pi}) \leq \tau + 2\frac{\overline{V}+\omega}{B} + \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\Big(\sqrt{C_\rho \epsilon} + \frac{2\gamma^{K/2}\overline{V}}{(1-\gamma)^{1/2}}\Big).$$

Let $\widehat{\pi} = \frac{1}{T}\sum_t \pi_t$ be the returned policy T iterations, with corresponding dual variable $\widehat{\lambda} = \frac{1}{T}\sum_t \lambda_t$.

By the stopping condition, the empirical duality gap is less than some threshold ω, i.e., $\max_{\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B} \widehat{L}(\widehat{\pi}, \lambda) - \min_{\pi \in \Pi} \widehat{L}(\pi, \widehat{\lambda}) \leq \omega$ where $\widehat{L}(\pi, \lambda) = \widehat{C}(\pi) + \lambda^\top(\widehat{G}(\pi) - \tau)$. We first show that the returned policy approximately satisfies the constraints. The proof of theorem D.3.1 will make use of the following empirical constraint satisfaction bound:

Lemma D.3.2 (Empirical constraint satisfactions)**.** *Assume that the constraints $\widehat{G}(\pi) \leq \tau$ are feasible. Then the returned policy $\widehat{\pi}$ approximately satisfies all constraints*

$$\max_{i=1:m+1} (\widehat{g}_i(\widehat{\pi}) - \tau_i) \leq 2\frac{\overline{C}+\omega}{B}.$$

Proof. We consider $\max_{i=1:m+1} (\widehat{g}_i(\widehat{\pi}) - \tau_i) > 0$ (otherwise the lemma statement is trivially true). The termination condition implies that

$$\widehat{L}(\widehat{\pi}, \widehat{\lambda}) - \max_{\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B} \widehat{L}(\widehat{\pi}, \lambda) \geq -\omega$$

$$\implies \widehat{\lambda}^\top(\widehat{G}(\widehat{\pi}) - \widehat{\tau}) \geq \max_{\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B} \lambda^\top(\widehat{G}(\widehat{\pi}) - \widehat{\tau}) - \omega. \tag{D.6}$$

We relax the RHS of Eq (D.6) by setting $\lambda[j] = B$ for $j = \arg\max_{i=1:m+1} [\widehat{g}_i(\widehat{\pi}) - \tau_i]$ and $\lambda[i] = 0 \ \forall i \neq j$ yields:

$$B \max_{i=1:m+1} [\widehat{g}_i(\widehat{\pi}) - \tau_i] - \omega \leq \widehat{\lambda}^\top(\widehat{G}(\widehat{\pi}) - \tau). \tag{D.7}$$

Given π such that $\widehat{G}(\pi) \leq \tau$, also by the termination condition:

$$\widehat{L}(\widehat{\pi}, \widehat{\lambda}) - \widehat{L}(\pi, \widehat{\lambda}) \leq \max_{\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B} \widehat{L}(\widehat{\pi}, \lambda) - \min_{\pi \in \Pi} \widehat{L}(\pi, \widehat{\lambda}) \leq \omega.$$

Thus implies

$$\widehat{L}(\widehat{\pi}, \widehat{\lambda}) \leq \widehat{L}(\pi, \widehat{\lambda}) + \omega = \widehat{C}(\pi) + \widehat{\lambda}^\top(\widehat{G}(\pi) - \tau) \leq \widehat{C}(\pi) + \omega. \tag{D.8}$$

Combining what we have from equation (D.8) and (D.7):

$$B \max_{i=1:m+1} [\widehat{g}_i(\widehat{\pi}) - \widehat{\tau}_i] - \omega \leq \widehat{\lambda}^\top(\widehat{G}(\widehat{\pi}) - \widehat{\tau}) = \widehat{L}(\widehat{\pi}, \widehat{\lambda}) - \widehat{C}(\widehat{\pi}) \leq \widehat{C}(\pi) + \omega - \widehat{C}(\widehat{\pi}).$$

Rearranging and bounding $\widehat{C}(\pi) \leq \overline{C}$ and $\widehat{C}(\widehat{\pi}) \leq -\overline{C}$ completes the proof. $\square$

We now return to the proof of theorem D.3.1, our task is to lift empirical error to generalization bound for main objective and constraints.

Denote by ϵ_{FQE} the (generalization) error introduced by the Fitted Q Evaluation procedure (algorithm 3) and ϵ_{FQI} the (generalization) error introduced by the Fitted Q Iteration procedure (algorithm 13). For now we keep ϵ_{FQE} and ϵ_{FQI} unspecified (to be specified shortly). That is, for each $t = 1, 2, \ldots, T$, we have with probability at least $1 - \delta$:

$$C(\pi_t) + \lambda_t^\top (G(\pi_t) - \tau) \leq C(\pi^*) + \lambda_t^\top (G(\pi^*) - \tau) + \epsilon_{FQI}.$$

Since π^* satisfies the constraints, i.e., $G(\pi^*) - \tau \leq 0$ componentwise, and $\lambda_t \geq 0$, we also have with probability $1 - \delta$

$$L(\pi_t, \lambda_t) = C(\pi_t) + \lambda_t^\top (G(\pi_t) - \tau) \leq C(\pi^*) + \epsilon_{FQI}. \tag{D.9}$$

Similarly, with probability $1 - \delta$, all of the following inequalities are true

$$\widehat{C}(\pi_t) + \epsilon_{FQE} \geq C(\pi_t) \geq \widehat{C}(\pi_t) - \epsilon_{FQE} \tag{D.10}$$

$$\widehat{G}(\pi_t) + \epsilon_{FQE}\mathbf{1} \geq G(\pi_t) \geq \widehat{G}(\pi_t) - \epsilon_{FQE}\mathbf{1} \text{ (row wise for all } m \text{ constraints).} \tag{D.11}$$

Thus with probability at least $1 - \delta$:

$$\begin{aligned} L(\pi_t, \lambda_t) = C(\pi_t) + \lambda_t^\top (G(\pi_t) - \tau) &\geq \widehat{C}(\pi_t) + \lambda_t^\top (\widehat{G}(\pi_t) - \tau) - \epsilon_{FQE}(1 + \lambda_t^\top \mathbf{1}) \\ &\geq \widehat{C}(\pi_t) + \lambda_t^\top (\widehat{G}(\pi_t) - \tau) - \epsilon_{FQE}(1 + B) \\ &= \widehat{L}(\pi_t, \lambda_t) - \epsilon_{FQE}(1 + B). \end{aligned} \tag{D.12}$$

Recall that the execution of mixture policy $\widehat{\pi}$ is done by uniformly sampling one policy π_t from $\{\pi_1, \ldots, \pi_T\}$, and rolling-out with π_t. Thus from equations (D.9) and (D.12), we have $\mathbb{E}_{t \sim U[1:T]} \widehat{L}(\pi_t, \lambda_t) \leq C(\pi^*) + \epsilon_{FQI} + (1 + B)\epsilon_{FQE}$ w.p. $1 - \delta$. In other words, with probability $1 - \delta$:

$$\frac{1}{T} \sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda_t) \leq C(\pi^*) + \epsilon_{FQI} + (1 + B)\epsilon_{FQE}.$$

Due to the no-regret property of our online algorithm (EG in this case):

$$\frac{1}{T} \sum_{t=1}^{T} \widehat{L}(\pi_t, \lambda_t) \geq \max_{\lambda} \widehat{L}(\widehat{\pi}, \lambda) - \omega = \widehat{C}(\widehat{\pi}) + \max_{\lambda} \lambda^\top (\widehat{G}(\widehat{\pi}) - \tau) - \omega$$

If $\widehat{G}(\widehat{\pi}) - \tau \leq 0$ componentwise, choose $\lambda[i] = 0, i = 1, 2, \ldots, m$ and $\lambda[m+1] = B$. Otherwise, we can choose $\lambda[j] = B$ for $j = \arg\max_{i=1:m+1} [\widehat{g}_i(\widehat{\pi}) - \tau[i]]$ and $\lambda[i] = 0 \ \forall i \neq j$. We can see that $\max_{\lambda \in \mathbb{R}_+^{m+1}, \|\lambda\|_1 = B} \lambda^\top (\widehat{G}(\widehat{\pi}) - \tau) \geq 0$. Therefore:

$\widehat{C}(\widehat{\pi}) - \omega \leq C(\pi^*) + \epsilon_{FQI} + (1 + B)\epsilon_{FQE}$ with probability at least $1 - \delta$.

Combined with the first term from equation (D.10):

$$C(\widehat{\pi}) - \epsilon_{FQE} - \omega \leq C(\pi^*) + \epsilon_{FQI} + (1+B)\epsilon_{FQE}$$

or

$$C(\widehat{\pi}) \leq C(\pi^*) + \omega + \epsilon_{FQI} + (2+B)\epsilon_{FQE}. \tag{D.13}$$

We now bring in the generalization error results from our standalone analysis of FQI (appendix D.5) and FQE (Chapter 5 appendix C.2) into equation (D.13).

Specifically, when

$$n \geq \frac{24 \cdot 214 \cdot \overline{V}^4}{\epsilon^2}\left(\log\frac{K(m+1)}{\delta} + \mathtt{dim}_{\mathrm{F}}\log\frac{320\overline{V}^2}{\epsilon^2} + \log(14e(\mathtt{dim}_{\mathrm{F}}+1))\right),$$

when FQI and FQE are run with K iterations, we have the guarantee that for any $\epsilon > 0$, with probability at least $1-\delta$

$$\begin{aligned} C(\widehat{\pi}) &\leq C(\pi^*) + \omega + \underbrace{\frac{2\gamma}{(1-\gamma)^3}\left(\sqrt{C_\mu}\epsilon + 2\gamma^{K/2}\overline{V}\right)}_{\text{FQI generalization error}} \\ &\quad + \underbrace{\frac{\gamma^{1/2}(2+B)}{(1-\gamma)^{3/2}}\left(\sqrt{C_\mu}\epsilon + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}}2\overline{V}\right)}_{(2+B)\times \text{ FQE generalization error}} \\ &\leq C(\pi^*) + \omega + \frac{(4+B)\gamma}{(1-\gamma)^3}\left(\sqrt{C_\mu}\epsilon + 2\gamma^{K/2}\overline{V}\right). \end{aligned} \tag{D.14}$$

From lemma D.3.2, $\widehat{G}(\widehat{\pi}) \leq \tau + 2\frac{\overline{C}+\omega}{B} \leq \tau + 2\frac{\overline{V}+\omega}{B}$. From equation (D.11), for each t=1,2,...,T, we have $\widehat{G}(\pi_t) \geq G(\pi_t) - \epsilon_{FQE}\mathbf{1}$ with probability $1-\delta$. Thus

$$\begin{aligned} \mathbf{P}\left(\widehat{G}(\widehat{\pi}) \geq G(\widehat{\pi}) - \epsilon_{FQE}\mathbf{1}\right) &= \sum_{t=1}^{T}\mathbf{P}(\widehat{G}(\pi_t) \geq G(\pi_t) - \epsilon_{FQE}\mathbf{1}|\widehat{\pi} = \pi_t)\mathbf{P}(\widehat{\pi} = \pi_t) \\ &\geq T(1-\delta)\frac{1}{T} = 1-\delta. \end{aligned}$$

Therefore, we have the following generalization guarantee for the approximate satisfaction of all constraints:

$$G(\widehat{\pi}) \leq \tau + 2\frac{\overline{V}+\omega}{B} + \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\left(\sqrt{C_\mu}\epsilon + \frac{\gamma^{K/2}}{(1-\gamma)^{1/2}}2\overline{V}\right). \tag{D.15}$$

Inequalities (D.14) and (D.15) complete the proof of theorem D.3.1 (and theorem 6.4.3).

D.4 Preliminaries to Analysis of Fitted Q Iteration (FQI)

This section follows the same setup and structure as Chapter 5 Appendix section C.1.

We show improved convergence rate from $O(n^{-4})$ to $O(n^{-2})$, where n is the number of samples in data set D.

To be consistent with the notation of this chapter, we use the convention $C(\pi)$ as the value function that denotes long-term accumulated cost, instead of using $V(\pi)$ denoting long-term rewards in the traditional RL literature. Our notation for Q function is similar to the RL literature - the only difference is that the optimal policy minimizes $Q(x, a)$ instead of maximizing. We denote the bound on the value function as $\overline{C}$ (alternatively if the single timestep cost is bounded by $\bar{c}$, then $\overline{C} = \frac{\bar{c}}{1-\gamma}$). For simplicity, the standalone analysis of FQI concerns only with the cost objective c. Dealing with cost $c + \lambda^\top g$ offers no extra difficulty - in that case we simply augment the bound of the value function to $\overline{V} = \overline{C} + B\overline{G}$.

We begin by recalling a few ideas and definitions from Chapter 5 Appendix section C.1.

Reminder from 5 Appendix section C.1

Recall, the *Bellman optimality operator* $\mathbb{T} : \mathcal{B}(\mathcal{X}; \overline{C}) \mapsto \mathcal{B}(\mathcal{X}; \overline{C})$ as:

$$(\mathbb{T}Q)(s, a) = c(s, a) + \gamma \int_{\mathcal{S}} \min_{a' \in \mathcal{A}} Q(s', a') p(ds'|s, a). \tag{D.16}$$

The optimal value functions are defined as usual by $C^*(s) = \sup_\pi C^\pi(s)$ and $Q^*(s, a) = \sup_\pi Q^\pi(s, a) \ \ \forall (s, a) \in \mathcal{X}$.

Denote the state-action data generating distribution as μ, induced by some data-generating (behavior) policy π_{D}, that is, $(s_i, a_i) \sim \mu$ for $(s_i, a_i, s_i', c_i) \in \mathrm{D}$.

Note that data set D is formed by multiple trajectories generated by π_{D}. For each (s_i, a_i), we have $s_i' \sim p(\cdot|s_i, a_i)$ and $c_i = c(s_i, a_i)$. For any (measurable) function $f : \mathcal{X} \mapsto \mathbb{R}$, define the μ-weighted ℓ_2 norm of f as $\|f\|_\mu^2 = \int_{\mathcal{X}} f(s, a)^2 \mu(ds, da) = \int_{\mathcal{X}} f(s, a)^2 \mu_s(ds) \pi_{\mathrm{D}}(a|ds)$. Similarly for any other state-action distribution ρ, $\|f\|_\rho^2 = \int_{\mathcal{X}} f(s, a)^2 \rho(ds, da)$

Definition D.4.1 (Inherent Bellman Error)**.** Given a function class F and a chosen distribution ρ, the *inherent Bellman error* of F is defined as:

$$d_{\mathrm{F}} = d(\mathrm{F}, \mathbb{T}\mathrm{F}) = \sup_{h \in \mathrm{F}} \inf_{f \in \mathrm{F}} \|f - \mathbb{T}h\|_\rho ,$$

where $\|\cdot\|_\rho$ is the $\rho-$weighted ℓ_2 norm and $\mathbb{T}$ is the Bellman optimality operator defined in (D.16)

Definition D.4.2 (Concentration coefficient of state-action distribution)**.** Given data generating distribution $\mu \sim \pi_{\mathrm{D}}$, initial state distribution d_0. For

$m \geq 0$, and an arbitrary sequence of stationary policies $\{\pi_m\}_{m\geq 1}$ let

$$\beta_\mu(m) = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m})}{d\mu} \right\|_\infty .$$

($\beta_\mu(m) = \infty$ if the future state distribution $d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m}$ is not absolutely continuous w.r.t. μ, i.e, $d_0 P^{\pi_1} P^{\pi_2} \ldots P^{\pi_m}(s,a) > 0$ for some $\mu(s,a) = 0$)

Assumption D.1. $\beta_\mu = (1-\gamma)^2 \sum\limits_{m\geq 1} m\gamma^{m-1}\beta_\mu(m) < \infty$.

Definition D.4.3 (Random L_1 Norm Covers)**.** Let $\epsilon > 0$, let F be a set of functions $\mathcal{S} \mapsto \mathbb{R}$, let $s_1^n = (s_1, \ldots, s_n)$ be n fixed points in $\mathcal{S}$. Then a collection of functions $\mathrm{F}_\epsilon = \{f_1, \ldots, f_N\}$ is an ϵ-cover of F on s_1^n if

$$\forall f \in \mathrm{F}, \exists f' \in \mathrm{F}_\epsilon : |\frac{1}{n}\sum_{i=1}^n f(s_i) - \frac{1}{n}\sum_{i=1}^n f'(s_i)| \leq \epsilon.$$

The empirical covering number, denote by $\mathcal{N}_1(\epsilon, \mathrm{F}, s_1^n)$, is the size of the smallest ϵ-cover on s_1^n. Take $\mathcal{N}_1(\epsilon, \mathrm{F}, s_1^n) = \infty$ if no finite ϵ-cover exists.

Definition D.4.4 (Pseudo-Dimension)**.** A real-valued function class F has pseudo-dimension $\mathtt{dim}_\mathrm{F}$ defined as the VC dimension of the function class induced by the sub-level set of functions of F. In other words, define function class $\mathrm{H} = \{(x,y) \mapsto \mathrm{sign}(f(x) - y : f \in \mathrm{F}\}$, then

$$\mathtt{dim}_\mathrm{F} = \mathtt{VC\text{-}dimension}(\mathrm{H}).$$

D.5 Finite-Sample Analysis of Fitted Q Iteration (FQI)

Algorithm and Discussion

Algorithm 13 Fitted Q Iteration with Function Approximation: $\mathtt{FQI}(c)$ [56]

Require: Collected data set $\mathrm{D} = \{s_i, a_i, s_i', c_i\}_{i=1}^n$. Function class F

1: Initialize $Q_0 \in \mathrm{F}$ randomly
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Compute target $y_i = c_i + \gamma \min_a Q_{k-1}(s_i', a) \ \ \forall i$
4: Build training set $\widetilde{\mathrm{D}}_k = \{(s_i, a_i), y_i\}_{i=1}^n$
5: Solve a supervised learning problem:
$Q_k = \arg\min_{f\in\mathrm{F}} \frac{1}{n}\sum_{i=1}^n (f(s_i,a_i) - y_i)^2$

Ensure: $\pi_K(\cdot) = \arg\min_a Q_K(\cdot, a)$ (greedy policy with respect to the returned function Q_K)

The analysis of FQI (algorithm 13) follows analogously from the analysis of FQE (Chapter 5 Appendix C.2). For brevity, we skip certain detailed derivations, especially those that are largely identical to FQE's analysis.

To the best of our knowledge, a finite-sample analysis of FQI with general non-linear function approximation has not been published (Continuous FQI from [11] is in fact a Fitted Policy Iteration algorithm and is different from algo 13). In principle, one can adapt existing analysis of fitted value iteration [150] and FittedPolicyQ [12, 11] to show that under similar assumptions, among policies greedy w.r.t. functions in F, FQI will find $\epsilon-$ optimal policy using $n = \widetilde{O}(\frac{1}{\epsilon^4})$ samples. We derive an improved analysis of FQI with general non-linear function approximations, with better sample complexity of $n = \widetilde{O}(\frac{1}{\epsilon^2})$. We note that the appendix of [120] contains an analysis of LinearFQI showing similar rate to ours, albeit with linear function approximators.

In this section, we prove the following statement:

Theorem D.5.1 (Guarantee for FQI - General Case (theorem 6.4.2))**.** *Under Assumption D.1, for any $\epsilon > 0, \delta \in (0, 1)$, after K iterations of Fitted Q Iteration (algorithm 13), for $n = O\Big(\frac{\overline{C}^4}{\epsilon^2}(\log \frac{K}{\delta} + \mathtt{dim}_{\mathrm{F}} \log \frac{\overline{C}^2}{\epsilon^2} + \log \mathtt{dim}_{\mathrm{F}})\Big)$, we have with probability $1 - \delta$:*

$$C^* - C(\pi_K) \leq \frac{2\gamma}{(1-\gamma)^3}\Big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}} + \epsilon) + 2\gamma^{K/2}\overline{C}\Big),$$

where π_K is the policy greedy with respect to the returned function Q_K, and C^* is the value of the optimal policy.

The key steps to the proof follow similar scheme to the proof of FQE. We first bound the error for each iteration, and then analyze how the errors flow through the algorithm.

Single iteration error bound $\|Q_k - \mathbb{T}Q_{k-1}\|_\mu$

Here μ is the state-action distribution induced by the data-generating policy π_{D}. We begin with the decomposition:

$$\begin{aligned}
&\|Q_k - \mathbb{T}Q_{k-1}\|_\mu^2 = \mathbb{E}\left[(Q_k(s,a) - y)^2\right] - \mathbb{E}\left[(\mathbb{T}Q_{k-1}(s,a) - y)^2\right] \\
&= \Bigg\{\mathbb{E}\left[(Q_k(s,a) - y)^2\right] - \mathbb{E}\left[(\mathbb{T}Q_{k-1}(s,a) - y)^2\right] \\
&\qquad - 2 \cdot \left(\frac{1}{n}\sum_{i=1}^{n}(Q_k(s_i,a_i) - y_i)^2 - \frac{1}{n}\sum_{i=1}^{n}(\mathbb{T}Q_{k-1}(s_i,a_i) - y_i)^2\right)\Bigg\} \\
&\qquad + \left\{2 \cdot \left(\frac{1}{n}\sum_{i=1}^{n}(Q_k(s_i,a_i) - y_i)^2 - \frac{1}{n}\sum_{i=1}^{n}(\mathbb{T}Q_{k-1}(s_i,a_i) - y_i)^2\right)\right\} \\
&= \mathtt{component_1} + \mathtt{component_2}.
\end{aligned} \tag{D.17}$$

For $\mathbb{T}$ the Bellman (optimality) operator (equation D.16), $\mathbb{T}Q_{k-1}$ is the *regression function* that minimizes square loss $\min_{h:\mathbb{R}^{\mathcal{X}}\mapsto\mathbb{R}} \mathbb{E}|h(s,a)-y|^2$, with the random variables $(s,a) \sim \mu$ and $y = c(s,a) + \gamma \min_{a'} Q_{k-1}(s',a')$ where $s' \sim p(s'|s,a)$. Invoking lemma C.2.5 and following the steps similar to equations (C.4),(C.5),(C.6), and (C.7) from Chapter 5 Appendix C.2, we can bound the first component as

$$\mathbf{P}(\texttt{component\_1} > \epsilon/2) \leq 14 \cdot e \cdot (\texttt{dim}_{\mathrm{F}} + 1) \left(\frac{640\overline{C}^2}{\epsilon}\right)^{\texttt{dim}_{\mathrm{F}}} \cdot \exp\left(-\frac{n\epsilon}{48 \cdot 214\overline{C}^4}\right). \tag{D.18}$$

Let $f^* = \arg\inf_{f\in\mathrm{F}} \|f - \mathbb{T}Q_{k-1}\|_\mu^2$. Since $Q_k = \arg\min_{f\in\mathrm{F}} \frac{1}{n}\sum_{i=1}^n (f(s_i,a_i)-y_i)^2$, we can upper-bound `component_2` by

$$\texttt{component\_2} \leq 2 \cdot \left(\frac{1}{n}\sum_{i=1}^n (f^*(s_i,a_i)-y_i)^2 - \frac{1}{n}\sum_{i=1}^n (\mathbb{T}Q_{k-1}(s_i,a_i)-y_i)^2\right).$$

Let random variable $z = ((s,a),y)$, $z_i = ((s_i,a_i),y_i), i = 1,\ldots,n$ and let

$$h(z) = (f^*(s,a)-y)^2 - (\mathbb{T}Q_{k-1}(s,a)-y)^2.$$

We have $|h(z)| \leq 4\overline{C}^2$. We can derive a bound for

$$\mathbf{P}\left(\frac{1}{n}\sum_{i=1}^n h(z_i) - \mathbb{E}h(z) > \frac{\epsilon}{4} + \mathbb{E}h(z)\right),$$

using Bernstein inequality, similar to equations (C.10) and (C.11) from Chapter 5 Appendix C.2 to obtain:

$$\mathbf{P}\left(2 \cdot \left[\frac{1}{n}\sum_{i=1}^n h(z_i) - 2\mathbb{E}h(z)\right] > \frac{\epsilon}{2}\right) \leq \exp\left(-\frac{3}{416} \cdot \frac{n\epsilon}{\overline{C}^2}\right). \tag{D.19}$$

Now we have

$$\texttt{component\_2} \leq 2 \cdot \frac{1}{n}\sum_{i=1}^n h(z_i) = 2 \cdot \left[\frac{1}{n}\sum_{i=1}^n h(z_i) - 2\mathbb{E}h(z)\right] + 4\mathbb{E}h(z).$$

Since

$$\begin{aligned}
\mathbb{E}h(z) &= \mathbb{E}_{\widetilde{D}_k}\left[(f^*(s,a)-y)^2\right] - \mathbb{E}_{\widetilde{D}_k}\left[(\mathbb{T}Q_{k-1}(s,a)-y)^2\right] \\
&= \mathbb{E}_{\widetilde{D}_k}\left[(f^*(s,a) - \mathbb{T}Q_{k-1}(s,a))^2\right] \\
&= \inf_{f\in\mathrm{F}} \|f - \mathbb{T}Q_{k-1}\|_\mu^2 .
\end{aligned} \tag{D.20}$$

Combining equations (D.18), (D.19) and (D.20), we obtain that

$$\begin{aligned}
&\mathbf{P}\Big\{ \|Q_k - \mathbb{T}Q_{k-1}\|_\mu^2 - 4\inf_{f\in\mathrm{F}} \|f - \mathbb{T}Q_{k-1}\|_\mu^2 > \epsilon\Big\} \\
&\quad\leq 14 \cdot e \cdot (\texttt{dim}_{\mathrm{F}} + 1)\left(\frac{640\overline{C}^2}{\epsilon}\right)^{\texttt{dim}_{\mathrm{F}}} \cdot \exp\left(-\frac{n\epsilon}{48 \cdot 214\overline{C}^4}\right) \\
&\qquad + \exp\left(-\frac{3}{416} \cdot \frac{n\epsilon}{\overline{C}^2}\right).
\end{aligned} \tag{D.21}$$

Propagation of error bound for $\|Q^* - Q^{\pi_K}\|_\rho$

The analysis of error propagation for FQI is more involved than that of FQE, but the proof largely follows the error propagation analysis in lemma 3 and 4 of [150] in the fitted value iteration context (for V function). We include the Q function's (slighly more complicated) derivation here for completeness.

Recall that π_K is greedy wrt the learned function Q_K returned by FQI. We aim to bound the difference $C^* - C^{\pi_K}$ between the optimal value function and that π_K. For a (to-be-specified) distribution ρ of state-action pairs (different from the data distribution μ), we bound the generalization loss $\|Q^* - Q^{\pi_K}\|_\rho$

Step 1: Upper-bound the propagation error (value). Let $\epsilon_{k-1} = Q_k - \mathbb{T}Q_{k-1}$. We have that

$$\begin{aligned} Q^* - Q_k &= \mathbb{T}^{\pi^*}Q^* - \mathbb{T}^{\pi^*}Q_{k-1} + \mathbb{T}^{\pi^*}Q_{k-1} - \mathbb{T}Q_{k-1} + \epsilon_{k-1} \\ &\leq \mathbb{T}^{\pi^*}Q^* - \mathbb{T}^{\pi^*}Q_{k-1} + \epsilon_{k-1} \ \textit{(b/c } \mathbb{T}Q_{k-1} \geq \mathbb{T}^{\pi^*}Q_{k-1}) \\ &= \gamma P^{\pi^*}(Q^* - Q_{k-1}) + \epsilon_{k-1}. \end{aligned}$$

Thus by recursion $Q^* - Q_K \leq \sum_{k=0}^{K-1} \gamma^{K-k-1}(P^{\pi^*})^{K-k-1}\epsilon_k + \gamma^K(P^{\pi^*})^K(Q^* - Q_0)$

Step 2: Lower-bound the propagation error (value). Similarly

$$\begin{aligned} Q^* - Q_k &= \mathbb{T}Q^* - \mathbb{T}^{\pi_{k-1}}Q^* + \mathbb{T}^{\pi_{k-1}}Q^* - \mathbb{T}Q_{k-1} + \epsilon_{k-1} \\ &\geq \mathbb{T}^{\pi_{k-1}}Q^* - \mathbb{T}Q_{k-1} + \epsilon_{k-1} \ (\text{as } \mathbb{T}Q^* \geq \mathbb{T}^{\pi_{k-1}}Q^*) \\ &\geq \mathbb{T}^{\pi_{k-1}}Q^* - \mathbb{T}^{\pi_{k-1}}Q_{k-1} + \epsilon_{k-1} \ \textit{(b/c } \pi_{k-1} \textit{ greedy wrt } Q_{k-1}) \\ &= \gamma P^{\pi_{k-1}}(Q^* - Q_{k-1}) + \epsilon_{k-1}. \end{aligned}$$

And by recursion:

$$\begin{aligned} Q^* - Q_K \geq \sum_{k=0}^{K-1} \gamma^{K-k-1}(P^{\pi_{K-1}}P^{\pi_{K-2}}\dots P^{\pi_{k+1}})\epsilon_k \\ + \gamma^K(P^{\pi_{K-1}}P^{\pi_{K-2}}\dots P^{\pi_0})(Q^* - Q_0). \end{aligned}$$

Step 3: Upper-bound the propagation error (policy). Beginning with a decomposition of value wrt to policy π_K:

$$\begin{aligned} Q^* - Q^{\pi_K} &= \mathbb{T}^{\pi^*}Q^* - \mathbb{T}^{\pi^*}Q_K + \mathbb{T}^{\pi^*}Q_K - \mathbb{T}^{\pi_K}Q_K + \mathbb{T}^{\pi_K}Q_K - \mathbb{T}^{\pi_K}Q^{\pi_K} \\ &\overset{(a)}{\leq} (\mathbb{T}^{\pi^*}Q^* - \mathbb{T}^{\pi^*}Q_K) + (\mathbb{T}^{\pi_K}Q_K - \mathbb{T}^{\pi_K}Q^{\pi_K}) \\ &= \gamma P^{\pi^*}(Q^* - Q_K) + \gamma P^{\pi_K}(Q_K - Q^{\pi_K}) \\ &= \gamma P^{\pi^*}(Q^* - Q_K) + \gamma P^{\pi_K}(Q_K - Q^* + Q^* - Q^{\pi_K}), \end{aligned}$$

where (a) uses $\mathbb{T}^{\pi^*}Q_K \leq \mathbb{T}Q_K = \mathbb{T}^{\pi_K}Q_K$. This leads to $(I - \gamma P^{\pi_K})(Q^* - Q^{\pi_K}) \leq \gamma(P^{\pi^*} - P^{\pi_K})(Q^* - Q_K)$ The operator $(I - \gamma P^{\pi_K})$ is invertible and

$(I - \gamma P^{\pi_K})^{-1} = \sum_{m\geq 0} \gamma^m (P^{\pi_K})^m$ is monotonic. Thus

$$\begin{aligned} &Q^* - Q^{\pi_K} \\ &\leq \gamma(I - \gamma P^{\pi_K})^{-1}(P^{\pi^*} - P^{\pi_K})(Q^* - Q_K) \\ &= \gamma(I - \gamma P^{\pi_K})^{-1} P^{\pi^*}(Q^* - Q_K) - \gamma(I - \gamma P^{\pi_K})^{-1} P^{\pi_K}(Q^* - Q_K). \quad \text{(D.22)} \end{aligned}$$

Applying inequalities from Step 1 and Step 2 to the RHS of (D.22), we have

$$\begin{aligned} Q^* - Q^{\pi_K} \leq (I - \gamma P^{\pi_K})^{-1} \Bigg[\sum_{k=0}^{K-1} \gamma^{K-k} \left((P^{\pi^*})^{K-k} - P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_{k+1}} \right) \epsilon_k \\ + \gamma^{K+1} \left((P^{\pi^*})^{K+1} - (P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_0}) \right) (Q^* - Q_0) \Bigg]. \quad \text{(D.23)} \end{aligned}$$

Next we apply point-wise absolute value on RHS of (D.23), with $|\epsilon_k|$ being the short-hand notation for $|\epsilon_k(s,a)|$ point-wise. Using triangle inequalities and rewriting (D.23) in a more compact form ([150]):

$$Q^* - Q^{\pi_K} \leq \frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2} \left[\sum_{k=0}^{K-1} \alpha_k A_k |\epsilon_k| + \alpha_K A_K |Q^* - Q_0| \right],$$

where $\alpha_k = \frac{(1-\gamma)\gamma^{K-k-1}}{1-\gamma^{K+1}}$ for $k < K, \alpha_K = \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}}$ and

$$\begin{aligned} A_k &= \frac{1-\gamma}{2}(I - \gamma P^{\pi_K})^{-1} \left[(P^{\pi^*})^{K-k} + P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_{k+1}} \right] \text{ for } k < K \\ A_K &= \frac{1-\gamma}{2}(I - \gamma P^{\pi_K})^{-1} \left[(P^{\pi^*})^{K+1} + P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_0} \right]. \end{aligned}$$

Note that A_k's are probability kernels that combine the P^{π_i} terms and α_k's are chosen such that $\sum_k \alpha_k = 1$.

Step 4: Bounding $\|Q^* - Q^{\pi_K}\|_\rho^2$ for any test distribution ρ.

This step handles distribution shift from μ to ρ (similar to Step 2 from subsection C.2 of Chapter 5 Appendix C.2). Using Jensen twice:

$$\|Q^* - Q^{\pi_K}\|_\rho^2 \leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2} \right]^2 \int \left[\sum_{k=0}^{K-1} \alpha_k A_k \epsilon_k^2 + \alpha_K A_K (Q^* - Q_0)^2 \right] (s,a).$$

Using assumption D.1 (assumption 6.1), each term ρA_k is bounded as

$$\begin{aligned} \rho A_k &= \frac{1-\gamma}{2} \rho (I - \gamma P^{\pi_K})^{-1} \left[(P^{\pi^*})^{K-k} + P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_{k+1}} \right] \\ &= \frac{1-\gamma}{2} \sum_{m\geq 0} \gamma^m \rho (P^{\pi_K})^m \left[(P^{\pi^*})^{K-k} + P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_{k+1}} \right] \\ &\leq (1-\gamma) \sum_{m\geq 0} \gamma^m \beta_\mu (m + K - k) \mu. \qquad \text{(def D.4.2)} \end{aligned}$$

Thus

$$
\begin{aligned}
&\|Q^* - Q^{\pi_K}\|_\rho^2 \\
&\leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^2 \left[\frac{1}{1-\gamma^{K+1}} \times \right. \\
&\qquad \left. \sum_{k=0}^{K-1} (1-\gamma)^2 \sum_{m\geq 0} \gamma^{m+K-k-1} \beta_\mu (m+K-k) \|\epsilon_k\|_\mu^2 + \alpha_K (2\overline{C})^2 \right] \\
&\overset{(a)}{\leq} \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^2 \left[\frac{1}{1-\gamma^{K+1}} \beta_\mu \max_k \|\epsilon_k\|_\mu^2 + \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}} (2\overline{C})^2 \right] \\
&\leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^2 \left[\frac{1}{1-\gamma^{K+1}} \beta_\mu \max_k \|\epsilon_k\|_\mu^2 + \frac{\gamma^K}{1-\gamma^{K+1}} (2\overline{C})^2 \right] \\
&\leq \left[\frac{2\gamma}{(1-\gamma)^2}\right]^2 \left[\beta_\mu \max_k \|\epsilon_k\|_\mu^2 + \gamma^K (2\overline{C})^2 \right],
\end{aligned}
$$

where (a) uses (Assump. D.1). Using $a^2 + b^2 \leq (a+b)^2$ for nonnegative a, b, we thus conclude that

$$
\|Q^* - Q^{\pi_K}\|_\rho \leq \frac{2\gamma}{(1-\gamma)^2} \left(\sqrt{\beta_\mu} \max_k \|\epsilon_k\|_\mu + 2\gamma^{K/2} \overline{C} \right). \tag{D.24}
$$

Step 5: Bounding $C^* - C^{\pi_K}$ Using the performance difference lemma (lemma 6.1 of [100], which states that $C^* - C^{\pi_K} = -\frac{1}{1-\gamma} \mathbb{E}_{\substack{s\sim d_{\pi_K} \\ a\sim\pi_K}} A^*[s,a]$. We can upper-bound the performance difference of value function as

$$
\begin{aligned}
&C^* - C^{\pi_K} \qquad\qquad (D.25) \\
&= \frac{1}{1-\gamma} \mathbb{E}_{\substack{s\sim d_{\pi_K} \\ a\sim\pi_K}} [C^*(s) - Q^*(s,a)] = \frac{1}{1-\gamma} \mathbb{E}_{s\ \sim d_{\pi_K}} [C^*(s) - Q^*(s, \pi_K(s))] \\
&\overset{(a)}{\leq} \frac{1}{1-\gamma} \mathbb{E}_{s\ \sim d_{\pi_K}} [Q^*(s,\pi^*(s)) - Q_K(s,\pi^*(s)) \\
&\qquad\qquad + Q_K(s,\pi_K(s)) - Q^*(s,\pi_K(s))] \\
&\leq \frac{1}{1-\gamma} \mathbb{E}_{s\ \sim d_{\pi_K}} |Q^*(s,\pi^*(s)) - Q_K(s,\pi^*(s))| \\
&\qquad\qquad + |Q_K(s,\pi_K(s) - Q^*(s,\pi_K(s))| \\
&\overset{(b)}{\leq} \frac{1}{1-\gamma} \left(\|Q^* - Q^{\pi_K}\|_{d_{\pi_K}\times\pi^*} + \|Q^* - Q^{\pi_K}\|_{d_{\pi_K}\times\pi_K} \right) \\
&\leq \frac{2\gamma}{(1-\gamma)^3} \left(\sqrt{\beta_\mu} \max_k \|\epsilon_k\|_\mu + 2\gamma^{K/2} \overline{C} \right), \qquad\qquad (D.26)
\end{aligned}
$$

where (a) is greedy, and (b) is a 2-norm upper bound of the 1-norm. Note that inequality (D.26) follows from (D.24) by specifying $\rho = d_0 P^{\pi_K} P^{\pi^*}$ and $\rho = d_0 P^{\pi_K} P^{\pi_K}$, respectively ($d_0$ is the initial state distribution).

Finite-sample guarantees for Fitted Q Iteration

From (D.21) we have:

$$\begin{aligned}
&\mathbf{P}\Big\{ \|Q_k - \mathbb{T}Q_{k-1}\|_\mu - 2 \inf_{f\in \mathrm{F}} \|f - \mathbb{T}Q_{k-1}\|_\mu > \epsilon \Big\} \\
&\quad \leq 14 \cdot e \cdot (\mathtt{dim}_\mathrm{F} + 1) \left(\frac{640\overline{C}^2}{\epsilon^2}\right)^{\mathtt{dim}_\mathrm{F}} \cdot \exp\left(-\frac{n\epsilon^2}{48 \cdot 214 \overline{C}^4}\right) \\
&\qquad + \exp\left(-\frac{3}{416} \cdot \frac{n\epsilon^2}{\overline{C}^2}\right).
\end{aligned}$$

Note that $\inf_{f\in \mathrm{F}} \|f - \mathbb{T}Q_{k-1}\|_\mu \leq \sup_{h\in \mathrm{F}} \inf_{f\in \mathrm{F}} \|f - \mathbb{T}h\|_\mu = d_\mathrm{F}$ (the *inherent Bellman error* from equation D.16). Combining with equation (D.26), we have the conclusion that for any $\epsilon > 0, 0 < \delta < 1$, after K iterations of Fitted Q Iteration, and for π_K the greedy policy wrt Q_K:

$$C^* - C_K^\pi \leq \frac{2\gamma}{(1-\gamma)^3}\left(\sqrt{\beta_\mu}(2d_\mathrm{F} + \epsilon) + 2\gamma^{K/2}\overline{C}\right)$$

holds with probability $1 - \delta$ when $n = O\Big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \mathtt{dim}_\mathrm{F} \log\frac{\overline{C}^2}{\epsilon^2} + \log \mathtt{dim}_\mathrm{F})\Big)$.

Note that compared to the Fitted Value Iteration analysis of [150], our error includes an extra factor 2 for d_F.

Statement for the Bellman-realizable Case

To facilitate the end-to-end generalization analysis of theorem 6.4.3, we include a version of FQI analysis under Bellman-realizable assumption in this section. The theorem is a consequence of previous analysis in this section.

Assumption D.2 (Bellman evaluation realizability)**.** We consider function classes F sufficiently rich so that $\forall f, \mathbb{T}f \in \mathrm{F}$.

Theorem D.5.2 (Guarantee for FQI - Bellman-realizable Case)**.** *Under Assumption D.1 and D.2, for any $\epsilon > 0, \delta \in (0,1)$, after K iterations of Fitted Q Iteration, for $n \geq \frac{24\cdot 214\cdot \overline{C}^4}{\epsilon^2}\Big(\log\frac{K}{\delta} + \mathtt{dim}_\mathrm{F}\log\frac{320\overline{C}^2}{\epsilon^2} + \log(14e(\mathtt{dim}_\mathrm{F}+1))\Big)$, we have with probability $1-\delta$:*

$$C^* - C(\pi_K) \leq \frac{2\gamma}{(1-\gamma)^3}\Big(\sqrt{\beta_\mu}\epsilon + 2\gamma^{K/2}\overline{C}\Big),$$

where π_K is the policy greedy with respect to the returned function Q_K, and C^* is the value of the optimal policy.

D.6 Additional Instantiation of Meta-Algorithm (algorithm 4)

We provide an additional instantiation of the meta-algorithm described in this chapter, with Online Gradient Descent (OGD) [234] and Least-Squares Policy Iteration (LSPI) [115] as subroutines. Using LSPI requires a feature map φ

such that any state-action pair can be represented by k features. The value function is linear in parameters represented by φ. Policy representation is simplified to a weight vector $w \in \mathbb{R}^k$.

Similar to our main algorithm 5, OGD updates require bounded parameters λ. We thus introduce hyper-parameter B as the bound of λ in ℓ_2 norm. The gradient update is projected to the ℓ_2 ball when the norm of λ exceeds B (line 15 of algorithm 14).

Algorithm 14 Batch Learning under Constraints using Online Gradient Descent and Least-Squares Policy Iteration

Require: Dataset $\mathrm{D} = \{x_i, a_i, x'_i, c_i, g_i\}_{i=1}^n \sim \pi_{\mathrm{D}}$. Online algorithm parameters: ℓ_2 norm bound B, learning rate η

Require: Number of basis function k. Basis function φ (feature map for state-action pairs)

1: Initialize $\lambda_1 = (0, \dots, 0) \in \mathbb{R}^m$
2: **for** each round t **do**
3: Learn $w_t \leftarrow \texttt{LSPI}(c + \lambda_t^\top g)$ *// LSPI with cost $c + \lambda_t^\top g$*
4: Evaluate $\widehat{C}(w_t) \leftarrow \texttt{LSTDQ}(w_t, c)$ *// Algo 16 with π_t, cost c*
5: Evaluate $\widehat{G}(w_t) \leftarrow \texttt{LSTDQ}(w_t, g)$ *// Algo 16 with π_t,* cost g
6: $\widehat{w}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t w_{t'}$
7: $\widehat{C}(\widehat{w}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{C}(w_{t'})$, $\widehat{G}(\widehat{w}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{G}(w_{t'})$
8: $\widehat{\lambda}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \lambda_{t'}$
9: Learn $\widetilde{w} \leftarrow \texttt{LSPI}(c + \widehat{\lambda}_t^\top g)$ *// LSPI with cost $c + \widehat{\lambda}_t^\top g$*
10: Evaluate $\widehat{C}(\widetilde{w}) \leftarrow \texttt{LSTDQ}(\widetilde{w}, c), \widehat{G}(\widetilde{w}) \leftarrow \texttt{LSTDQ}(\widetilde{w}, g)$
11: $\widehat{\mathrm{L}}_{\max} = \max_{\lambda, \|\lambda\|_2 \le B} \left(\widehat{C}(\widehat{w}_t) + \lambda^\top (\widehat{G}(\widehat{w}_t) - \tau)\right)$
12: $\widehat{\mathrm{L}}_{\min} = \widehat{C}(\widetilde{w}) + \widehat{\lambda}_t^\top (\widehat{G}(\widetilde{w}) - \tau)$
13: **if** $\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min} \le \omega$ **then**
14: Return $\widehat{\pi}_t$ greedy w.r.t $\widehat{w}_t$ $\left(\text{i.e., } \widehat{\pi}_t(x) = \arg\min_{a \in \mathcal{A}} \widehat{w}_t^\top \varphi(s, a) \ \ \forall s\right)$
15: $\lambda_{t+1} = \mathcal{P}(\lambda_t - \eta(\widehat{G}(\pi_t) - \tau))$ where projection $\mathcal{P}(\lambda) = B\frac{\lambda}{\max\{B, \|\lambda\|_2\}}$

Algorithm 15 Least-Squares Policy Iteration: $\texttt{LSPI}(c)$ [115]

Require: Stopping criterion ϵ

1: Initialize $w' \leftarrow w_0$
2: **repeat**
3: $w \leftarrow w'$
4: $w' \leftarrow \texttt{LSTDQ}(w, c)$
5: **until** $\|w - w'\| \le \epsilon$

Ensure: Policy weight w $\left(\text{i.e., } \pi(s) = \arg\min_{a \in \mathcal{A}} w^\top \varphi(s, a) \ \ \forall s\right)$

Algorithm 16 LSTDQ(w, c) [115]

Initialize $\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $k \times k$ matrix
Initialize $\widetilde{b} \leftarrow \mathbf{0}$ // $k \times 1$ vector
for each $(s, a, s', c) \in \mathrm{D}$ **do**
  $a' = \arg\min_{\widetilde{a} \in \mathcal{A}} w^\top \varphi(x', \widetilde{a})$
  $\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \varphi(s, a)(\varphi(s, a) - \gamma\varphi(x', a'))^\top$
  $\widetilde{b} \leftarrow \widetilde{b} + \varphi(s, a)c$
$\widetilde{w} \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$
Ensure: $\widetilde{w}$

D.7 Additional Experimental Details

Environment Descriptions and Procedures

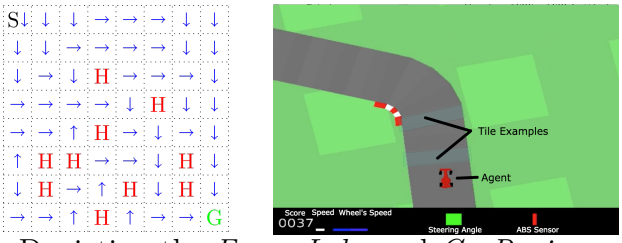


Figure D.1: Depicting the *FrozenLake* and *CarRacing* environments.

Frozen Lake. The environment is a 8x8 grid as seen in Figure D.1 (left), based on OpenAi's FrozenLake-v0. In each episode, the agent starts from S and traverse to goal G. While traversing the grid, the agent must avoid the pre-determined holes denoted by H. If the agent steps off of the grid, the agent returns to the same grid location. The episode terminates when the agent reaches the goal or falls into a hole. The arrows in Figure D.1 (left) is an example policy returned by our algorithm, showing an optimal route.

Denote $\mathcal{S}_{holes}$ as the set of all holes in the grid and $\mathcal{S}_{goal} = \{s_{goal}\}$ is a singleton set representing the goal in the grid. The contrained batch policy learning problem is:

$$\begin{aligned} \min_{\pi \in \Pi} \quad & C(\pi) = \mathbb{E}[\mathbb{I}(s' \notin \mathcal{S}_{goals})] = \mathrm{P}(s' \notin \{s_{goal}\}) \\ \text{s.t.} \quad & G(\pi) = \mathbb{E}[\mathbb{I}(s' \in \mathcal{S}_{holes})] = \mathrm{P}(s' \in \mathcal{S}_{holes}) \leq \tau. \end{aligned} \tag{D.27}$$

We collect 5000 trajectories by selecting an action randomly with probability .95 and an action from a DDQN-trained model with probability .05.Furthermore we set $B = 30$ and $\eta = 50$, the hyperparameters of our Exponentiated Gradient subroutine. We set the threshold for the constraint $\tau = .1$.

Car Racing. The environment is a racetrack as seen in Figure D.1 (right),

modified from OpenAi's CarRacing-v0. In each state, given by the raw pixels, the agent has 12 actions: $a \in A = \{(i,j,k)|i \in \{-1,0,1\}, j \in \{0,1\}, k \in \{0,.2\}\}$. The action tuple (i,j,k) cooresponds to steering angle, amount of gas applied and amount of brake applied, respectively. In each episode, the agent starts at the same point on the track and must traverse over 95% of the track, given by a discretization of 281 tiles. The agent recieves a reward of $+\frac{1000}{281}$ for each unique tile over which the agent drives. The agent receives a penalty of $-.1$ per-time step. Our collected dataset takes the form: $\mathrm{D} = \{(s_{t-6}, s_{t-3}, s_t), a_t, (s_{t-3}, s_t, s_{t+3}), c_t, g_{0,t}, g_{1,t}\}$ where s_i denotes the image at timestep i and a_t is applied 3 times between s_t and s_{t+3}. This frame-stacking option is common practice in online RL for Atari and video games. In our collected dataset D, the maximum horizon is 469 time steps.

The first constraint concerns accumulated number of brakes, a proxy for smooth driving or acceleration. The second constraint concerns how far the agent travels away from the center of the track, given by the Euclidean distance between the agent and the closest point on the center of the track. Let N_t be the number of tiles that is collected by the agent in time t. The constrained batch policy learning problem is:

$$\begin{aligned} \min_{\pi \in \Pi} \quad & \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (-\frac{1000}{281} N_t + .1)] \\ \text{s.t.} \quad & G_0(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}(a_t \in \mathcal{A}_{braking})] \leq \tau_0 \\ & G_1(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t d(u_t, v_t)] \leq \tau_1. \end{aligned} \tag{D.28}$$

We instatiate our subroutines, FQE and FQI, with multi-layered CNNs. Furthermore we set $B = 10$ and $\eta = .01$, the hyperparameters of our Exponentiated Gradient subroutine. We set the threshold for the constraint to be about 75% of the value exhibited by online RL agent trained by DDQN [211].

Additional Discussion for the Car Racing Experiment

Regularized policy learning and grid-search. We perform grid search over a range of regularization parameters λ for both Least-Squares Policy Iteration - LSPI ([115]) and Fitted Q Iteration - FQI ([56]). The results, seen from the the first and second plot of Figure D.2, show that one-shot regularized learning has difficulty learning a policy that satisfies both constraints. We augment LSPI with non-linear feature mapping from one of our best performing FQI model

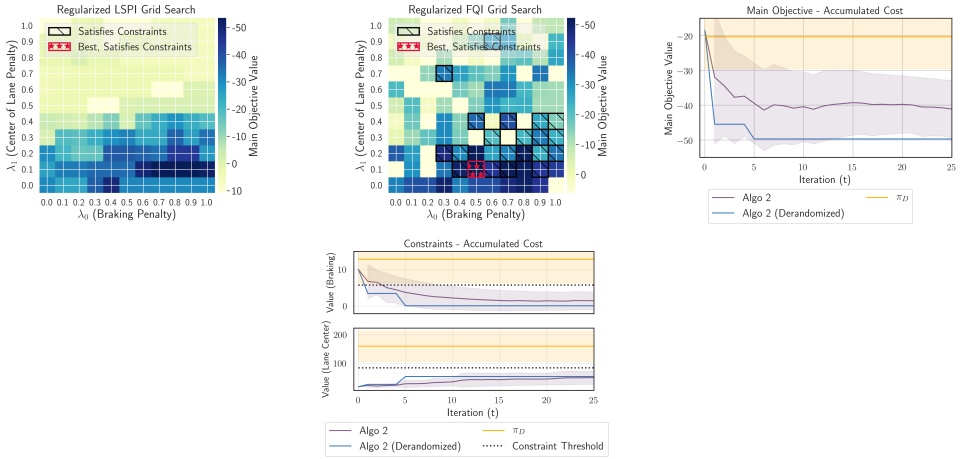


Figure D.2: (First and Second figures) Result of 2-D grid-search for one-shot, regularized policy learning for *LSPI* (left) and *FQI* (right).
(Third and Fourth figures) value range of individual policies in our mixtured policy and data generating policy π_D for main objective (left) and cost constraint (right).

(using CNNs representation). While both regularized LSPI and regularized FQI can achieve low main objective cost, the constraint cost values tend to be sensitive with the λ step. Overall for the whole grid search, about 10% of regularized policies satisfy both constraints, while none of the regularized LSPI policy satisfies both constraints.

Mixture policy and de-randomization. As our algorithm returned a mixture policy, it is natural to analyze the performance of individual policies in the mixture. The third and fourth plot from Figure D.2 show the range of performance of individual policy in our mixture (purple band). We compare individual policy return with the stochastic behavior of the data generation policy. Note that our policies satisfy constraints almost always, while the individual policy returned in the mixture also tends to outperform π_D with respect to the main objective cost.

Off-policy evaluation standalone comparison. Typically, inverse propensity scoring based methods call for stochastic behavior and evaluation policies [164, 195]. However in this domain, the evaluation policy and environment are both deterministic, with long horizon (the max horizon is D is 469). Consequently Per-Decision Importance Sampling typically evaluates the policy as 0. In general, off-policy policy evaluation in long-horizon domains is known to be challenging [134, 75]. We augment PDIS by approximating the evaluation policy with a stochastic policy, using a softmin temperature parameter. However,

PDIS still largely shows significant errors. For Doubly Robust and Weighted Doubly Robust methods, we train a model of the environment as follows:

- a 32-dimensional representation of state input is learned using variational autoencoder. Dimensionality reduction is necessary to aid accuracy, as original state dimension is $96 \times 96 \times 3$.
- an LSTM is used to learn the transition dynamics $P(z(s')|z(s), a)$, where $z(s)$ is the low-dimensional representation learned from previous step. Technically, using a recurrent neural networks is an augmentation to the dynamical modeling, as true MDPs typically do not require long-term memory.
- the model is trained separately on a different dataset, collected indendently from the dataset D used for evaluation.

The architecture of our dynamics model is inspired by recent work in model-based online policy learning [77]. However, despite our best effort, learning the dynamics model accurately proves highly challenging, as the horizon and dimensionality of this domain are much larger than popular benchmarks in the OPE literature [98, 202, 61]. The dynamics model has difficulty predicting the future state several time steps away. Thus we find that the long-horizon, model-based estimation component of DR and WDR in this high-dimensional setting is not sufficiently accurate. For future work, a thorough benchmarking of off-policy evaluation methods in high-dimensional domains would be a valuable contribution.

Appendix E

CHAPTER 5 APPENDIX

E.1 Notation and Overview

Acronym	Term
LTL	Linear Temporal Logic
MDP, LDBA	Markov Decision Process. Limit Determinisitic Buchi Automaton
AMEC	Accepting MEC
MEC/EC	Maximal EC, End Component
LCP	LTL Constrained Planning, Algo 6
`FindAMEC`	Subroutine to assist in finding AMECs
`PR`	`PlanRecurrent`. Subroutine to plan in AMECs, Algo 7
`PT`, `NB-PT`	`PlanTransient`/ `NoBlockPlanTransient`. Subroutine to plan to AMECs, Algo 8/ Algo 19
`VI`	Value Iteration Subroutine, Algo 17
AP	Atomic Proposition
Σ	Alphabet $\Sigma = 2^{AP}$
$\mathcal{S}$	State Space
$\mathcal{A}$	Action Space. $\mathcal{A}(s)$ allowable actions in state s.
$\mathcal{A}_A$	Restricted Action Space. $\mathcal{A}_A(s) \subseteq \mathcal{A}(s)$ allowable actions in state $s \in A \subseteq \mathcal{S}$.
P	Transition Function
P_π	Markov Chain induced by π in P
$\mathcal{C}$	Cost Function
$\mathcal{X}$	Product-MDP
τ	Run or trajectory in an MDP
η-greedy(π)	$= (1-\eta)\pi + \eta\texttt{Unif}(\mathcal{A}_A)$ with $0 \leq \eta \leq 1$, defined with respect to $A \subseteq \mathcal{S}$
φ	LTL Specification/Formula/Task
$\mathbb{P}[\pi \models \varphi]$	Probability that a policy satisfies the task
$\Pi, \Pi_{\max}$	Class of stochastic policies, $\pi \in \Pi$ with maximal $\mathbb{P}[\pi \models \varphi]$
β	Lower bound on minimum, nonzero transition probability
D	Dataset tracking all tuples (s, a, s') simulated
$\widehat{P}$	Empirical estimate of P from data in D
$\tilde{P}$	Optimistic dynamics returned by `VI`
$\mathcal{P}$	Plausible transition functions consistent with all the information gathered in D
$\mathcal{E}$	High probability event
$n(s,a)$	Number of samples accumulated in (s,a). Also denoted n
$\psi(n)$	Error bound on $\max_{s' \in \mathcal{S}} \lvert\widehat{P}(s,a,s') - P(s,a,s')\rvert$
$\psi^{-1}(\rho)$	Number of samples $n(s,a)$ necessary to achieve $\max_{s' \in \mathcal{S}} \lvert\widehat{P}(s,a,s') - P(s,a,s')\rvert < \rho$
ϵ_V	Cost-optimality tolerance wrt. main problem (1)
ϵ_φ	Prob-optimality tolerance wrt. main problem (1)
$\epsilon_{\texttt{PR}}$	error input into `PR`, $\epsilon_{\texttt{PR}} = \frac{\epsilon_V}{4\lambda}$
$\epsilon_{\texttt{PT}}$	error input into `PT`, $\epsilon_{\texttt{PT}} = \frac{2\epsilon_V}{9}$
$\epsilon^{\mathcal{L}}_{\texttt{PR}}$	Convergence condition for VI in `PR`, $\epsilon^{\mathcal{L}}_{\texttt{PR}} = \frac{2\epsilon_{\texttt{PR}}}{3}$
$\epsilon^{\mathcal{L}}_{\texttt{PT}}$	Convergence condition for VI in `PT`, $\epsilon^{\mathcal{L}}_{\texttt{PT}} = \frac{c_{\min}\epsilon_{\texttt{PT}}\epsilon_\varphi}{4\bar{V}}$
α	Aperiodicity Coefficient $\alpha \in (0,1)$. $P_{\alpha,\pi} = \alpha P_\pi + (1-\alpha)I$
$\mathcal{L}^\alpha_{\texttt{PR}}$	$\mathcal{L}^\alpha_{\texttt{PR}} v(s) = \min_{a \in \mathcal{A}_A(s)} \left(\mathcal{C}(s,a) + \alpha \min_{p \in \mathcal{P}(s,a)} p^T v\right) + (1-\alpha)v(s) \quad \forall s \in A$
$\mathcal{L}_{\texttt{PT}}$	$\mathcal{L}_{\texttt{PT}} v(s) = \begin{cases} \min\left\{\min_{a \in \mathcal{A}_A(s)} \left(\mathcal{C}(s,a) + \min_{p \in \mathcal{P}(s,a)} p^T v\right), \bar{V}\right\}, & s \in \mathcal{S} \setminus \cup_{i=1}^k A_i \\ \lambda g_i, & s \in A_i \end{cases}$
$d_{\texttt{PR}}$	Convergence operator for `PR`, $d_{\texttt{PR}}(v', v) = \max_{s \in A}(v'(s) - v(s)) - \min_{s \in A}(v'(s) - v(s))$
$d_{\texttt{PT}}$	Convergence operator for `PT`, $d_{\texttt{PT}}(v_{n+1}, v_n) = \lVert v_{n+1} - v_n \rVert_1$
V_T	Terminal costs. $V_T = 0$ by default
λ	Tradeoff between g_π and J_π
J_π	Transient cost, conditioned on runs satisfying φ
g_π	Gain, Average-cost, conditioned on runs satisfying φ
$\bar{V}$	Upper bound on J_π for any $\pi \in \Pi$
$\Delta(M)$	Coefficient of Ergodicity of matrix M, $\Delta(M) = \frac{1}{2}\max_{ij}\sum_k \lvert M_{ik} - M_{jk}\rvert$
$\bar{\Delta}_{A_i}$	Worst-case coefficient of ergodicty in A_i

Overview

There is a lot of notation that we will be using to get through the analysis. It is important to distinguish the following:

Table E.1: Policies and Probabilities.

Acronym	Term
π^*	Optimal policy w.r.t (LTL-OPT)
π	Policy returned by LCP (Algo 6)
π_{A_i}	Gain and Prob-optimal policy in AMEC $(A_i, \mathcal{A}_{A_i})$ in dynamics P
$\tilde{\pi}_{A_i}$	Gain and Prob-optimal policy in AMEC $(A_i, \mathcal{A}_{A_i})$ in dynamics $\tilde{P}$
π_i	A policy in states A_i of an AMEC $(A_i, \mathcal{A}_{A_i})$, ignoring what π does outside of A_i
p^π, p^*	$\mathbb{P}[\pi \models \varphi]$ and $\mathbb{P}[\pi^* \models \varphi]$. Also denoted p^π and p^{π^*}
p_i^π, $p_i^{\pi^*}$	$\mathbb{P}[\pi \text{ reaches } A_i], \mathbb{P}[\pi^* \text{ reaches } A_i] \geq 0$ denoted p_i, p_i^* (resp), $\sum_{i=1}^k p_i = p$, $\sum_{i=1}^k p_i^* = p^*$

Table E.2: Gains.

Term	Description. (Subscript i or A_i denotes "in AMEC $(A_i, \mathcal{A}_{A_i})$")
$\widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}$	Approximated gain of (greedy) optimal policy $\tilde{\pi}_{A_i}$ under optimistic dynamics $\tilde{P}$
$g^{P}_{\pi_i}$	Actual gain of policy π_i (η-greedy version of policy from `PR`) under true dynamics P
$g^{P}_{\pi_{A_i}}$	Gain of optimal policy π_{A_i} under dynamics P

The relationship between these gains is subtle. `PlanRecurrent` (Algo 7) returns $\widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}$ as the estimate for how good the best greedy policy will be in AMEC $(A_i, \mathcal{A}_{A_i})$ under dynamics $\tilde{P}$. But we don't use the greedy policy, we use the η-greedy policy π_i. With π_{A_i} being the true gain-optimal policy in dynamics P (in AMEC $(A_i, \mathcal{A}_{A_i})$), then we will find the following relations:

$$\underbrace{\widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}}_{\text{Output from PR}} \quad \underbrace{\approx}_{\frac{\epsilon^{\mathcal{L}}_{\texttt{PR}}}{2}} g^{\tilde{P}}_{\tilde{\pi}_{A_i}} \quad \underbrace{\approx}_{\text{Lem } E.2.7} \quad g^{\tilde{P}}_{\pi_i} \quad \underbrace{\approx}_{\text{Lem } E.2.8} \quad g^{P}_{\pi_i} \quad \underbrace{\approx}_{\text{Prop } E.2.5} \quad \underbrace{g^{P}_{\pi_{A_i}}}_{\text{Actual}} .$$

In general gains g are functions of state: $g(s)$. However, it is well known [166, 66] that in communicating MDPs (each state is reachable from one another by some policy) that the gain of the optimal policy (even if determinstic) is constant – independent of state. Since AMECs are communicating MDPs, then $\pi_{A_i}, \tilde{\pi}_{A_i}$ induce constant gains in $P, \tilde{P}$ respectively. Lastly, the stochastic policy π_i makes both $\tilde{P}$ and P recurrent, and so the gain is also constant. We will therefore only be considering the absolute difference between gains rather than L_∞ norms (as they coincide).

When superscripts are dropped in V, the dynamics are the true dynamics P of the product-MDP $\mathcal{X}$. Once again, the relationships between these value functions is subtle. `PlanTransient` (Algo 8) returns v as the estimate for how

Table E.3: Value Functions.

Acronym	Term
V_π	Main objective, value function $V^P_{\pi,\lambda} = J_\pi + \lambda g_\pi$
v	Approximated value of policy π from PT (Algo 7) in dynamics $\tilde{P}$
$\tilde{V}^{\tilde{P}}_\pi$	Actual value of policy π from PT in dynamics $\tilde{P}$
$\tilde{V}^{P}_\pi$	Actual value of policy π from PT in dynamics P, also denoted $\tilde{V}_\pi = p(J_\pi + \sum_{i=1}^k \frac{p_i}{p} \widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}) + (1-p)\frac{\bar{V}}{\epsilon_\varphi}$

good π (the greedy policy wrt v) will be in reaching AMECs $\{(A_i, \mathcal{A}_{A_i})\}_{i=1}^k$, but is optimistic. v is an approximation to $\tilde{V}^{\tilde{P}}_\pi$. Roughly speaking, we will find that they are all similar/related:

$$\underbrace{v}_{\text{Ouput from PT}} \quad \underbrace{\approx}_{\text{Lem } E.2.16} \quad \tilde{V}^{\tilde{P}}_\pi \quad \underbrace{\approx}_{\text{Lem } E.2.15} \quad \tilde{V}^{P}_\pi \quad \underbrace{\approx}_{\text{Prop } E.2.12/E.4.1} \quad \underbrace{\tilde{V}^{P}_{\pi^*}}_{\text{An intermediate Value Func.}} ,$$

where the last approximation has two different propositions: the first allows the simplifying assumption made in this chapter regarding paths between AMECS, and the second removes that assumption at the expense of increased computation. Finally,

$$\|\tilde{V}^P_\pi - \tilde{V}^P_{\pi^*}\| \underbrace{\approx}_{\text{Thm 7.4.1}} \|V^P_\pi - V^P_{\pi^*}\|, \tag{E.1}$$

which involves swapping $\widehat{g}^{\tilde{P}}_{\pi_{A_i}}$ in $\tilde{V}$ for $g^P_{\pi_{A_i}}$.

E.2 Analysis: Statements with Proof

Sample Complexity Guarantee

The number of samples necessary to guarantee an $(\epsilon_V, \epsilon_\varphi, \delta)$-PAC approximation to the cost-optimal and probability-optimal policy relies factors: β (lower bound on the mininum non-zero transition probability of P), $\{c_{\min}, c_{\max}\}$ (bounds on the cost function $\mathcal{C}$), $\bar{\Delta}_{A_i}$ (worst-case coefficient of ergodicity for EC $(A_i, \mathcal{A}_{A_i})$), $\bar{V}$ (upper bound on the value function), and λ (tradeoff factor). Recall that an event $\mathcal{E}$ captures the scenario where the empirical transition function $\widehat{P}$ is close to the true transition function P. $\mathcal{E}$ holds with probability at least $1-\delta$, see Lem E.2.1.

Theorem 7.4.1 (Sample Complexity)**.** *Under the event $\mathcal{E}$, Assumption 7.1 and 7.2, after*

$$n = \tilde{\mathcal{O}}\left(\frac{1}{\beta} + \frac{1}{\epsilon_V^2}\left(\frac{|\mathcal{S}|^2\bar{V}^4}{c_{\min}^2\epsilon_\varphi^4} + \lambda^2\sum_{i=1}^{k}\frac{|A_i|^2 c_{\max}^2}{(1-\bar{\Delta}_{A_i})^2}\right)\right)$$

samples[1] *are collected from each state-action pair, the policy π returned by Algorithm 6 is, with probability $1-\delta$, simultaneously ϵ_V-cost optimal and ϵ_φ-probability optimal, satisfying:*

$$(i)\ \ |\mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^* \models \varphi]| \leq \epsilon_\varphi \quad (ii)\ \ \|V_\pi - V_{\pi^*}\|_\infty < \epsilon_V. \tag{7.5}$$

Comparison To RL Literature. Before presenting the proofs, we briefly compare this guarantee with standard guarantees in model-based reinforcement learning under a generative model. It is important to note that while we show that our guarantee is a sum of 3 terms, a tighter bound would be a max over the 3 terms. To the best of our knowledge, the current state-of-the-art RL (with generative model) guarantee is $\tilde{\mathcal{O}}(\frac{1}{(1-\gamma)^3\epsilon^2})$ [6], per state-action pair. Here, $H = \frac{1}{1-\gamma}$ represents the effective horizon in discounted settings. In other words, $c_{\max}H$ is the bound on (their) $\|V\|$. In our case, for the SSP reduction, the effective horizon is $H = \frac{\|\tilde{V}\|_\infty}{c_{\min}}$, as this is the expected goal-reaching time in the worst-case (since we do not have any discounting). We estimate $\|\tilde{V}\|_\infty$ with upper bound $\frac{\bar{V}}{\epsilon_\varphi}$. Suppose we set $\epsilon = \min(\epsilon_V, \epsilon_\varphi)$. Focusing just on the center term, we have guarantee taking the form, roughly, $\frac{|\mathcal{S}|^2H^4}{\epsilon^2}$. Here, the $|\mathcal{S}|^2$ comes from a loose upper bound $\max_{s\in\mathcal{S},a\in\mathcal{A}}\|\widehat{P}(s,a,\cdot)\|_1 = |\mathcal{S}|$. In fact, as noted in [198], when the MDP is not too chaotic $\max_{s\in\mathcal{S},a\in\mathcal{A}}\|\widehat{P}(s,a,\cdot)\|_1 = \mathcal{O}(1)$.

[1]The lower bound relating to β from [131] is $\Omega(\frac{\log(2\delta)}{\log(1-\beta)})$ whereas ours is $\tilde{O}(\frac{1}{\beta})$. We conjecture that $\tilde{\Omega}(\frac{1}{\beta})$ samples are required. See Appendix Section E.3.

Further, by using careful variance-aware arguments from [198] we can decrease the dependency from H^4 to H^3. Hence, the SSP guarantee (our center term) and the standard RL guarantee are very similar. The first term $\frac{1}{\beta}$ does not appear in standard RL literature because there is no constraint verification needed, but in practice will be dominated by the other terms. The last term is also similar to the center term. $\frac{c_{\max}}{1-\Delta_{A_i}}$ can also be seen as an effective horizon, accumulating $c_{\max}$ cost until the accepting component sufficiently mixes. Here, $|A_i|^2 \leq |\mathcal{S}|^2$ and, again, comes from the loose upper bound $\max_{s\in A_i, a\in\mathcal{A}_{A_i}} \|\widehat{P}(s,a,\cdot)\|_1 = |A_i|$.

Proof *of Theorem 7.4.1.* We begin by examining the interaction of π^* with P. The Markov chain P_{π^*} has a number, say m, of recurrent classes $R_1, \ldots, R_m$, sets of states that are trapping and visited infinitely often once reached. Some of the recurrent classes R_i contain an accepting state $s \in \mathcal{S}^*$, making any trajectory entering R_i an accepting run, without loss of generality call these $R_1, \ldots, R_{m'}$ (we just relabel them). Let $\mathcal{A}_{\pi_i^*}(s) = \{a \in \mathcal{A} | \pi_i^*(s|a) > 0\}$ denote the support of actions taken by π_i^* in state $s \in R_i$. Let $\mathcal{A}_{\pi_i^*} = \{\mathcal{A}_{\pi_i^*}(s)\}_{s\in R_i}$ be the indexed action set in R_i. Then, by definition, $\{(R_i, \mathcal{A}_{\pi_i^*})\}_{i=1}^{m'}$ are accepting EC. By definition, each accepting EC $(R_i, \mathcal{A}_{\pi_i^*})$ must be contained within (or is itself) some AMEC $(A_i, \mathcal{A}_i)$.

Fix some accepting EC $(R_j, \mathcal{A}_{\pi_j^*})$. We claim, without loss of generality, $(R_j, \mathcal{A}_{\pi_j^*}) = (A_i, \mathcal{A}_{A_i})$ for some index $i \in 1, \ldots, k$. To show this, let π_{A_i} be the gain optimal, and probability-optimal policy in AMEC $(A_i, \mathcal{A}_{A_i})$: π_{A_i} is defined over all states $s \in A_i$ and actions $a \in \mathcal{A}_{A_i}$. Further, consider the modified optimal policy

$$\tilde{\pi}^*(s,a) = \begin{cases} \pi_{A_i}(s,a), & s \in A_i \\ \pi^*(s,a), & \text{otherwise} \end{cases}.$$

Because π_{A_i} is prob-optimal (ie. $\mathbb{P}[\tilde{\pi}^* \models \varphi | s_0 \in A_i] = 1$) in A_i then the probability $\mathbb{P}[\tilde{\pi}^* \models \varphi] \geq \mathbb{P}[\pi^* \models \varphi]$. Further, $J_{\tilde{\pi}^*} \leq J_\pi$ because any τ that formerly passed through $R_j \setminus A_i$ now accumulates less cost. Further, $g_{\pi_{A_i}} \leq g_{\pi_i^*}$ by definition of optimality in AMEC $(A_i, \mathcal{A}_{A_i})$. Thus, $V_{\tilde{\pi}^*} \leq V_{\pi^*}$. Of course, by definition of optimality, the opposite signs hold: $V_{\tilde{\pi}^*} \geq V_{\pi^*}$ and $\mathbb{P}[\tilde{\pi}^* \models \varphi] \leq \mathbb{P}[\pi^* \models \varphi]$. Therefore $\tilde{\pi}^*$ and π^* are indistinguishable.

Repeating the above argument for each $(R_j, \mathcal{A}_{\pi_j^*})$ means the accepting EC of π^* are AMECS and, by definition, form some subset of all of the AMECs

$\{(A_i, \mathcal{A}_{A_i}\}_{i=1}^k$. In other words, all accepting runs of π^* reach states $\cup_{i=1}^k A_i$. Furthermore, $g_{\pi^*} = \sum_{i=1}^k \frac{p_i^*}{p} g^P_{\pi_{A_i}}$ where $p_i^* \geq 0$ is the probability that π^* reaches A_i and $\sum_{i=1}^k p_i^* = p$.

Property (i) now follows as a direct consequence of Prop 7.3.3 and Prop 7.3.2. Recall by Prop 7.3.3 that

$$|\mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^k A_i\right] - \max_{\pi' \in \Pi_{\max}} \mathbb{P}\left[\pi' \text{ reaches } \cup_{i=1}^k A_i\right]| \leq \epsilon_\varphi.$$

Prop 7.3.2 implies that once a run enters some A_i, the run is accepted. Remaining runs cannot be accepted since they do not reach any AMEC, the only way to be accepted. Hence $\mathbb{P}[\pi \models \varphi] = \mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^k A_i\right]$. Since we just showed that all accepting runs of π^* reach some $(A_i, \mathcal{A}_i)$ then:

$$\begin{aligned}
0 &\leq \mathbb{P}[\pi^* \models \varphi] - \mathbb{P}[\pi \models \varphi] \\
&\leq \mathbb{P}\left[\pi^* \text{ reaches } \cup_{i=1}^k A_i\right] - \mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^k A_i\right] \\
&\leq \epsilon_\varphi.
\end{aligned}$$

To show **Property** (ii), first let us define p_i^π as the probability of π reaching AMEC $(A_i, \mathcal{A}_{A_i})$ and, by property (i), $\sum_{i=1}^k p_i^\pi = \sum_{i=1}^k p_i^* = p$. The value function given by the Bellman operator $\mathcal{L}_{\texttt{PT}}$ (Table 7.1) in Algorithm 8 takes the form

$$\tilde{V}_\pi(s) = p(J_\pi(s) + \lambda \sum_{i=1}^k \frac{p_i^\pi}{p} \widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}} + (1-p)\frac{\bar{V}}{\epsilon_\varphi}, \tag{E.2}$$

where $\widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}$ are the approximated gains for end component $(A_i, \mathcal{A}_{A_i})$ from Algorithm 7. To see this, there is probability p that $\pi \models \varphi$ and achieves (conditional) expected cost $J_\pi(s) + \lambda \sum_{i=1}^k \frac{p_i}{p} \widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}}$ and prob $1-p$ that $\pi \not\models \varphi$ where all cooresponding trajectories get stuck and accumulate $\frac{\bar{V}}{\epsilon_\varphi}$ cost. Let $\tilde{\pi}$ now represent the optimal solution to the value function $\tilde{V}_\pi$ (Algo 8). Therefore we claim:

$$\begin{aligned}
0 &\leq V_\pi - V_{\pi^*} \\
&= V_\pi - \tilde{V}_\pi + \tilde{V}_\pi - \tilde{V}_{\tilde{\pi}} + \tilde{V}_{\tilde{\pi}} - \tilde{V}_{\pi^*} + \tilde{V}_{\pi^*} - V_{\pi^*} + (1-p)\frac{\bar{V}}{\epsilon_\varphi} - (1-p)\frac{\bar{V}}{\epsilon_\varphi} \\
&\leq \underbrace{|V_\pi - \tilde{V}_\pi + (1-p)\frac{\bar{V}}{\epsilon_\varphi}|}_{(a)} + \underbrace{|\tilde{V}_\pi - \tilde{V}_{\tilde{\pi}}|}_{(b)} + \underbrace{\tilde{V}_{\tilde{\pi}} - \tilde{V}_{\pi^*}}_{(c)} + \underbrace{|\tilde{V}_{\pi^*} - V_{\pi^*} - (1-p)\frac{\bar{V}}{\epsilon_\varphi}|}_{(d)} \\
&\leq \frac{\epsilon_V}{3} + \frac{\epsilon_V}{3} + 0 + \frac{\epsilon_V}{3} \leq \epsilon_V.
\end{aligned}$$

For (a), first we note that $g_\pi = \sum_{i=1}^k \frac{p_i^\pi}{p} g^P_{\pi_i}$, by definition of conditional expec-

tation. Let $\epsilon_{\texttt{PR}} = \frac{\epsilon_V}{7\lambda}$. Hence,

$$\underbrace{|V_\pi - \tilde{V}_\pi + (1-p)\frac{\bar{V}}{\epsilon_\varphi}|}_{(a)} = |p(J_\pi(s) + \lambda \sum_{i=1}^{k} \frac{p_i^\pi}{p} g_{\pi_i}^P) - p(J_\pi(s) + \lambda \sum_{i=1}^{k} \frac{p_i^\pi}{p} \widehat{g}_{\tilde{\pi}_{A_i}}^{\tilde{P}})|$$

$$\leq \lambda \max_{i=1,\ldots,k} |g_{\pi_i}^P - \widehat{g}_{\tilde{\pi}_{A_i}}^{\tilde{P}}|$$

$$\leq \lambda \frac{4\epsilon_{\texttt{PR}}}{3}, \quad \text{Corollary } E.2.6$$

$$\leq \frac{\epsilon_V}{3}.$$

By similar argument, for (d), together with earlier argument that $g_\pi^* = \sum_{i=1}^{k} \frac{p_i^*}{p} g_{\pi_{A_i}}^P$ then we also have that:

$$\underbrace{|\tilde{V}_{\pi^*} - V_{\pi^*} - (1-p)\frac{\bar{V}}{\epsilon_\varphi}|}_{(d)} = |p(J_{\pi^*}(s) + \lambda \sum_{i=1}^{k} \frac{p_i^*}{p} \widehat{g}_{\tilde{\pi}_{A_i}}^{\tilde{P}}) - p(J_{\pi^*}(s) + \lambda \sum_{i=1}^{k} \frac{p_i^\pi}{p} g_{\pi_{A_i}}^P)|$$

$$\leq \lambda \max_{i=1,\ldots,k} |g_{\pi_{A_i}}^P - \widehat{g}_{\tilde{\pi}_{A_i}}^{\tilde{P}}|$$

$$\leq \lambda \max_{i=1,\ldots,k} |g_{\pi_{A_i}}^P - g_{\pi_i}^P| + |g_{\pi_i}^P - \widehat{g}_{\tilde{\pi}_{A_i}}^{\tilde{P}}|$$

$$\leq \lambda \frac{7\epsilon_{\texttt{PR}}}{3}, \quad \text{Prop } E.2.5 \text{ and Corollary } E.2.6$$

$$\leq \frac{\epsilon_V}{3}.$$

Further, we have $(c) \leq 0$ holds because $\tilde{\pi}$ is optimal in $\tilde{V}$ (either by assuming $\cup_{i=1}^{k} A_i$ is the correct choice of AMECS, or using Algo 19 instead of `planTransient`). In either case, $(b) \leq \frac{3\epsilon_{\texttt{PT}}}{2} \leq \frac{\epsilon_V}{3}$ by Prop E.2.12 or Prop E.4.1, where $\epsilon_{\texttt{PT}}$ is set to $\epsilon_{\texttt{PT}} = \frac{2\epsilon_V}{9}$, completing the approximation guarantee.

We now compute the number of samples, per state-action pair, required by Algorithm 6. By Prop E.2.4, we need $n = \tilde{\mathcal{O}}(\frac{1}{\beta})$ to verify the support of P. After calculating the AMECs $\{(A_i, \mathcal{A}_{A_i})\}_{i=1}^{k}$, we calculate the gain-optimal policy π_i for each AMEC. By Prop 7.3.2, we need $n = \tilde{\mathcal{O}}((\frac{|A_i| c_{\max}}{\epsilon_{\texttt{PR}}(1-\bar{\Delta}_{A_i})})^2) = \tilde{\mathcal{O}}((\frac{\lambda |A_i| c_{\max}}{\epsilon_V (1-\bar{\Delta}_{A_i})})^2)$ for each state-action pair in each end component $(A_i, \mathcal{A}_{A_i})$, since $\epsilon_{\texttt{PR}} = \frac{\epsilon_V}{7\lambda}$. Finally, for the transient policy π_0, the SSP reduction requires $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S} \setminus \cup_{i=1}^{k} A_i| \bar{V}^2}{\epsilon_{\texttt{PT}} \epsilon_\varphi^2 c_{\min}})^2) = \tilde{\mathcal{O}}((\frac{|\mathcal{S} \setminus \cup_{i=1}^{k} A_i| \bar{V}^2}{\epsilon_V \epsilon_\varphi^2 c_{\min}})^2)$ for each state-action pair outside of the AMECs, by Prop 7.3.3, since $\epsilon_{\texttt{PT}} = \frac{2\epsilon_V}{9}$. A similar sample complexity is guaranteed by using Algo 19 in place of `PlanTransient`, where $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S}| \bar{V}^2}{\epsilon_V \epsilon_\varphi^2 c_{\min}})^2)$ is required in place of $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S} \setminus \cup_{i=1}^{k} A_i| \bar{V}^2}{\epsilon_V \epsilon_\varphi^2 c_{\min}})^2)$. Adding these together yields the worst-case number of samples necessary in any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. These sample guarantees hold only when the event $\mathcal{E}$ holds, which itself holds

with probability $1-\delta$ (see Lem E.2.1).

□

Corollary 7.4.2 (Gain (Average Cost) Optimality)**.** *There exists $\lambda^* > 0$ s.t. for $\lambda > \lambda^*$, the policy π returned by Alg. 6 satisfies* (7.5)*, $g_\pi = \arg\min_{\pi' \in \Pi_{\max}} g_{\pi'}$, and is probability and gain optimal.*

Proof *of Corollary 7.4.2.* Fix some $\lambda > 0$. Let $\pi' = \arg\min_{\pi \in \Pi_{\max}} g_\pi$. Suppose $g_{\pi'} < g_\pi$ but $V_{\pi,\lambda} < V_{\pi',\lambda}$, elementwise. In other words, π is the preferred policy. Then,

$$\begin{aligned} 0 &\leq V_{\pi',\lambda} - V_{\pi,\lambda} \\ &= J_{\pi'} + \lambda g_{\pi'} - J_\pi - \lambda g_\pi \\ &\leq \max_{\tilde{\pi} \in \Pi} J_{\tilde{\pi}} + \lambda \underbrace{(g_{\pi'} - g_\pi)}_{<0}, \end{aligned}$$

since $J_{\tilde{\pi}} \geq 0$ for each $\tilde{\pi} \in \Pi$. If $\lambda > \frac{\max_{\tilde{\pi} \in \Pi} J_{\tilde{\pi}}}{g_\pi - g_{\pi'}}$ then we contradict $V_{\pi,\lambda} < V_{\pi',\lambda}$. In particular, if π' is the gain optimal policy then for any $\lambda > \lambda^* = \frac{\max_{\tilde{\pi} \in \Pi} J_{\tilde{\pi}}}{\min_{\{\pi \in \Pi | g_\pi \neq g_{\pi'}\}} g_\pi - g_{\pi'}}$ then π' is preferred to any other policy $\pi \in \Pi$.

□

High Probability Event and Sample Requirement

Definition 7.3.2 (High Probability Event)**.** A high probability event $\mathcal{E}$:

$$\mathcal{E} = \{\forall s, a, s' \in S \times A \times S, \forall n(s,a) > 1 : |(P(s,a,s') - \widehat{P}(s,a,s'))| \leq \psi_{sas'}(n) \leq \psi(n)\},$$

where $\psi_{sas'}(n) \equiv \sqrt{2\hat{P}(s,a,s')(1-\hat{P}(s,a,s')))\xi(n)} + \frac{7}{3}\xi(n)$, $\psi(n) \equiv \sqrt{\frac{1}{2}\xi(n)} + \frac{7}{3}\xi(n)$, and $\xi(n) \equiv \log(\frac{4n^2|\mathcal{S}|^2|\mathcal{A}|}{\delta})/(n-1)$.

Lemma E.2.1 (High Probability Event holds)**.** *The event $\mathcal{E}$ holds with probability at least $1-\delta$.*

***Proof** of Lemma E.2.1 .* We start with the anytime version of Theorem 4 of [141] given by Lemma 27 of [199]:

$$\mathbb{P}\left[\forall\, n \geq 1, \left|\mathbb{E}[Z] - \frac{1}{n}\sum_{i=1}^{n} Z_i\right| > \sqrt{\frac{2\hat{V}_n \log(4n^2/\delta)}{n-1}} + \frac{7\log(4n^2/\delta)}{3(n-1)}\right] \leq \delta,$$

for any $Z_i \in [0,1]$ iid. By re-setting $\delta \leftarrow \frac{\delta}{|\mathcal{S}|^2|\mathcal{A}|}$, applying union bound over all $(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and observing that $Z_i \sim P(s,a,s')$ is a Bernoulli random variable with empirical variance $\hat{V}_n = \widehat{P}(s,a,s')(1-\widehat{P}(s,a,s'))$ yields the result:

$\{\forall s, a, s' \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}, \forall n > 1 : |P(s,a,s') - \widehat{P}(s,a,s')| \leq \psi_{sas'}(n)\}$ w. prob $1-\delta$

Observing that $\psi_{sas'}(n) \leq \psi(n)$ for all $n > 1$ because $\psi_{sas'}(n)$ takes on a maximum when $\widehat{P}(s,a,s') = \frac{1}{2}$, completes the proof. □

Lemma E.2.2 (Inverting $\mathcal{E}$)**.** *Fix $(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$. Under the event $\mathcal{E}$, the number of samples $\psi^{-1}(\rho)$ required to achieve $|P(s,a,s') - \widehat{P}(s,a,s')| \leq \psi_{sas'}(n) \leq \psi(n) < \rho$ is given by:*

$$\psi^{-1}(\rho) = \lceil \frac{2}{\zeta^2}\log(\frac{16|\mathcal{S}|^2|\mathcal{A}|}{\zeta^4\delta})\rceil + 3 = \tilde{\mathcal{O}}(\frac{1}{\rho^2}),$$

where $\zeta \equiv \frac{-\frac{3}{7\sqrt{2}} + \sqrt{(\frac{3}{7\sqrt{2}})^2 + \frac{12}{7}\rho}}{2}$.

Proof of E.2.2. We have $\psi_{sas'} < \psi(n) = \frac{x}{\sqrt{2}} + \frac{7}{3}x^2 \leq \rho$ where $x^2 = \xi(n) = \frac{\log(4n^2|\mathcal{S}|^2|\mathcal{A}|\delta^{-1})}{n-1}$. Solving the quadratic inequality, we have

$$x \leq \frac{-\frac{3}{7\sqrt{2}} + \sqrt{(\frac{3}{7\sqrt{2}})^2 + \frac{12}{7}\rho}}{2} \equiv \zeta.$$

Hence, we have

$$\begin{aligned}\frac{\log(4n^2|\mathcal{S}|^2|\mathcal{A}|\delta^{-1})}{n-1} &\leq \zeta^2 \\ \implies n &\geq \frac{\log(4n^2|\mathcal{S}|^2|\mathcal{A}|\delta^{-1})}{\zeta^2} + 1 \\ &= \frac{1}{\zeta^2}\log(e^{\zeta^2}4n^2|\mathcal{S}|^2|\mathcal{A}|\delta^{-1}) \\ &= \underbrace{\frac{2}{\zeta^2}}_{c_1}\log(\underbrace{e^{\frac{\zeta^2}{2}}\sqrt{4|\mathcal{S}|^2|\mathcal{A}|\delta^{-1}}}_{c_2}n). \quad (\star)\end{aligned}$$

By Lemma E.2.3, if $n > 2c_1\log(c_1c_2)$ then $n > (\star)$. Simplifying,

$$n \geq \frac{2}{\zeta^2}\log(\frac{16|\mathcal{S}|^2|\mathcal{A}|}{\zeta^4\delta}) + 2.$$

Selecting $\psi^{-1}(\rho) = \lceil \frac{2}{\zeta^2}\log(\frac{16|\mathcal{S}|^2|\mathcal{A}|}{\zeta^4\delta})\rceil + 3$ and noting that $\zeta = \tilde{\mathcal{O}}(\rho)$ completes the proof: $n = \tilde{\mathcal{O}}(1/\rho^2)$. □

Lemma E.2.3. *(Lemma 10 of [105]) If* $\log(c_1c_2) \geq 1$ *and* $c_1, c_2 > 0$ *then*

$$N > 2c_1\log(c_1c_2) \implies N > c_1\log(c_2N).$$

`FindAMEC` proofs

Proposition E.2.4. *(Support Verification* ***`FindAMEC`****) Under the event $\mathcal{E}$ and Assumption 7.1, if $n = \varphi_{\texttt{FindAMEC}}(\beta) = \frac{5}{\beta}\log(\frac{100|\mathcal{S}|^2|\mathcal{A}|}{\beta^2\delta}) = \tilde{\mathcal{O}}(\frac{1}{\beta})$ samples are collected for each state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$ then the support of P is verified:*

$$P(s,a,s') = \begin{cases} 0, & \widehat{P}(s,a,s') = 0 \\ 1, & \widehat{P}(s,a,s') = 1 \\ \in [\beta, 1-\beta], & \textit{otherwise} \end{cases}.$$

Proof *of Prop E.2.4 .* Fix $(s,a,s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$. Suppose $\widehat{P}(s,a,s') \in \{0,1\}$ then by $\mathcal{E}$ we have

$$\frac{7\log(4n^2|\mathcal{S}|^2|\mathcal{A}|/\delta)}{3(n-1)} \leq \beta. \tag{E.3}$$

Following the second half of the proof of E.2.2 with $\zeta^2 = \frac{3\beta}{7}$, we have that if we take $n = \varphi_{\texttt{FindAMEC}}(\beta) = \frac{5}{\beta}\log(\frac{100|\mathcal{S}^2\mathcal{A}|}{\beta^2\delta}) > \frac{14}{3\beta}\log(\frac{784|\mathcal{S}^2\mathcal{A}|}{9\beta^2\delta})$ then we have

$$|P(s,a,s') - \widehat{P}(s,a,s')| < \beta. \tag{E.4}$$

Case $\widehat{P}(s,a,s') = 1$. Suppose $\widehat{P}(s,a,s') = 1$. By Eq (E.4), $P(s,a,s') > 1-\beta$. By Assumption 7.1 together with the fact that $\sum_{x\in\mathcal{S}} P(s,a,x) = 1$ then $P(s,a,x) = 0$ for any $x \neq s'$. Therefore, $P(s,a,s') = \widehat{P}(s,a,s') = 1$.

Case $\widehat{P}(s,a,s') = 0$. Suppose $\widehat{P}(s,a,s') = 0$. By Eq (E.4), $P(s,a,s') < \beta$. Hence $P(s,a,s') = \widehat{P}(s,a,s') = 0$, otherwise violating Assumption 7.1.

Case, Otherwise. If $P(s,a,s') > 1-\beta$ or $P(s,a,s') < \beta$ then by following the above arguments we'd yield similar contradictions with Assumption 7.1. Hence, $P(s,a,s') \in [\beta, 1-\beta]$. □

`PlanRecurrent` proofs

Proposition 7.3.2 (PR Convergence & Correctness, Informal)**.** *Let π_A be the gain-optimal policy in AMEC $(A, \mathcal{A})$. Algorithm 7 terminates after at most $\log_2\left(\frac{6|A|c_{\max}}{\epsilon_{PR}(1-\tilde{\Delta}_A)}\right)$ repeats, and collects at most $n = \tilde{\mathcal{O}}(\frac{|A|^2 c_{\max}^2}{\epsilon_{PR}^2(1-\tilde{\Delta}_A)^2})$ samples for each $(s,a) \in (A, \mathcal{A}_A)$. The η-greedy policy π w.r.t. v' (Alg. 7, Line 5) is gain optimal and probability optimal: $|g_\pi - g_{\pi_A}| < \epsilon_{PR}$, $\mathbb{P}[\pi \models \varphi | s_0 \in A] = 1$.*

We formalize Prop 7.3.2 as follows by adding the necessary PAC statements:

Proposition E.2.5 (PR Convergence & Correctness, Formal)**.** *Let π_A be the gain-optimal policy in AMEC $(A, \mathcal{A})$. Algorithm 7 terminates after at most $\log_2\left(\frac{6|A|c_{\max}}{\epsilon_{PR}(1-\tilde{\Delta}_A)}\right)$ repeats, and collects at most $n = \tilde{\mathcal{O}}(\frac{|A|^2 c_{\max}^2}{\epsilon_{PR}^2(1-\tilde{\Delta}_A)^2})$ samples for each $(s,a) \in (A, \mathcal{A}_A)$. Under the event $\mathcal{E}$ and Assumption 7.1 then with probability $1-\delta$, the η-greedy policy π w.r.t. v' (Alg. 7, Line 5) is gain optimal and probability optimal: $|g_\pi - g_{\pi_A}| < \epsilon_{PR}$, $\mathbb{P}[\pi \models \varphi | s_0 \in A] = 1$.*

Proof of Prop 7.3.2 & Prop E.2.5. Let $\pi_{v'}$ be the greedy policy with respect to v'. Let $g^{\tilde{P}}_{\tilde{\pi}_A}$ be the gain of the gain-optimal policy, $\tilde{\pi}_A$, in A with respect to dynamics $\tilde{P}$.

For the approximation error,

$$\begin{aligned} 0 \leq g^P_\pi - g^P_{\pi_A} &= g^P_\pi - g^{\tilde{P}}_\pi + g^{\tilde{P}}_\pi - g^{\tilde{P}}_{\pi_{v'}} + g^{\tilde{P}}_{\pi_{v'}} - g^{\tilde{P}}_* + g^{\tilde{P}}_* - g^P_{\pi_A} \\ &\leq \underbrace{|g^P_\pi - g^{\tilde{P}}_\pi|}_{(a)} + \underbrace{|g^{\tilde{P}}_\pi - g^{\tilde{P}}_{\pi_{v'}}|}_{(b)} + \underbrace{|g^{\tilde{P}}_{\pi_{v'}} - g^{\tilde{P}}_{\tilde{\pi}_A}|}_{(c)} + \underbrace{g^{\tilde{P}}_{\tilde{\pi}_A} - g^P_{\pi_A}}_{(d)} \\ &< \frac{\epsilon_{\texttt{PR}}}{3} + \frac{\epsilon_{\texttt{PR}}}{3} + \frac{\epsilon_{\texttt{PR}}}{3} + 0 = \epsilon_{\texttt{PR}}. \end{aligned}$$

We have the first inequality because π_A is gain optimal in P. By the Simulation Lemma E.2.8 we have that $(a) < \frac{\epsilon_{\texttt{PR}}}{3}$ by setting $\epsilon_{(2)} = \frac{\epsilon_{\texttt{PR}}}{3}$ in the Lemma. By the η-greedy approximation Lemma E.2.7 we have $(b) < \frac{\epsilon_{\texttt{PR}}}{3}$ by setting $\epsilon_{(1)} = \frac{\epsilon_{\texttt{PR}}}{3}$ in the Lemma. For (c), since $\pi_{v'}$ represents the approximately optimal policy in $\tilde{P}$ then, by value iteration approximation guarantees, $(c) = |g^{\tilde{P}}_{\pi_{v'}} - g^{\tilde{P}}_{\tilde{\pi}_A}| < \frac{\epsilon^{\mathcal{L}}_{\texttt{PR}}}{2} \leq \frac{\epsilon_{\texttt{PR}}}{3}$ by setting $\epsilon^{\mathcal{L}}_{\texttt{PR}} = \frac{2\epsilon_{\texttt{PR}}}{3}$ [66]. It is known that, by optimism and the aperiodicity transformation [66, 94] for the average cost Bellman operator, $g^{\tilde{P}}_{\tilde{\pi}_A} < g^P_{\pi_A}$ implying $(d) < 0$.

For the probability of satisfaction, when $s_0 \in A$, following a policy that samples every action in $\mathcal{A}_A$ with positive probability makes the Markov Chain P_π

recurrent. Thus, each $s \in A$ is visited infinitely often. In particular there is some $s^* \in A$ visited infinitely often, implying $\pi \models \varphi$.

Convergence is guaranteed by Lemma E.2.8: since ρ is halved every iteration then ρ never falls below $\frac{\epsilon_{(2)}(1-\bar{\Delta}_A)}{2|A|c_{\max}}$, which is reached after $\log_{\frac{1}{2}}(\frac{\epsilon_{\mathtt{PR}}(1-\bar{\Delta}_A)}{6|A|c_{\max}}) = \log_2(\frac{6|A|c_{\max}}{\epsilon_{\mathtt{PR}}(1-\bar{\Delta}_A)})$ iterations (since $\epsilon_{(2)} = \frac{\epsilon_{\mathtt{PR}}}{3}$). Further by Lemma E.2.8, we get the sample complexity $n = \tilde{\mathcal{O}}(\frac{|A|^2 c_{\max}^2}{\epsilon_{\mathtt{PR}}^2(1-\bar{\Delta}_A)^2})$, completing the proof. □

Corollary E.2.6. *Under the same assumptions as Prop E.2.5, in addition,* $|g_\pi^P - \widehat{g}_{\tilde{\pi}_A}^{\tilde{P}}| \leq \frac{4\epsilon_{PR}}{3}$.

Proof. Continuing the same argument as in Prop E.2.5, we have

$$0 \leq g_\pi^P - g_{\tilde{\pi}_A}^{\tilde{P}} + g_{\tilde{\pi}_A}^{\tilde{P}} - \widehat{g}_{\tilde{\pi}_A}^{\tilde{P}} \leq \epsilon_{\mathtt{PR}} + \frac{\epsilon_{\mathtt{PR}}^{\mathcal{L}}}{2} = \frac{4\epsilon_{\mathtt{PR}}}{3},$$

where we use triangle inequality and appeal to Prop E.2.5 for $|g_\pi^P - g_{\tilde{\pi}_A}^{\tilde{P}}| \leq \epsilon_{\mathtt{PR}}$ and [66] where $|g_{\tilde{\pi}_A}^{\tilde{P}} - \widehat{g}_{\pi_A}^{\tilde{P}}| \leq \frac{\epsilon_{\mathtt{PR}}^{\mathcal{L}}}{2} \leq \frac{\epsilon_{\mathtt{PR}}}{3}$ since $\epsilon_{\mathtt{PR}}^{\mathcal{L}} = \frac{2\epsilon_{\mathtt{PR}}}{3}$. □

Lemma E.2.7. *(η-greedy approximation) Let P be any dynamics. Let π be a greedy policy in AMEC $(A, \mathcal{A}_A)$ with dynamics P. With $0 \leq \eta \leq 1$, let π_η be η-greedy with respect to π. Then, for any error $\epsilon_{(1)} > 0$, there exists some threshold $\eta^* \in (0, 1]$ such that when $\eta \in (0, \eta^*]$ we have*

$$|g_\pi^P - g_{\pi_\eta}^P| \leq \epsilon_{(1)}. \tag{E.5}$$

Proof. Let $s_0, \ldots, s_{|A|-1}$ be any ordering of the states in A. The standard (non-optimistic) average cost Bellman equation with known dynamics P is given by $\mathcal{L}v(s) = g(s) + \min_{a \in \mathcal{A}_A(s)} \Big(\mathcal{C}(s,a) + P(s,a)v\Big)$ for each $s \in A$ for a unique g and v unique up to a constant translation [21]. Furthermore, since the end components are communicating sets then we know that g is a constant vector, i.e. $g = g(s) = g(s')$ for any $s, s' \in A$ [21]. Since v is unique up to translation, we can always set $v(0) = 0$ to make v unique. The evaluation equations, under policy π, is similarly, $\mathcal{L}_\pi v(s) = g_\pi + \mathbb{E}_{a \sim \pi}\Big[\mathcal{C}(s,a)] + P_\pi(s,a)v$ [21]. For more generality, instead of P_π we consider $\alpha P_\pi + (1-\alpha)I$, an aperiodicity transform

with any coefficient $\alpha \in [0,1]$. Then the $\mathcal{L}_\pi$ written as a system takes the form:

$$0_{2|A|} = \underbrace{\left[\begin{array}{c|c} \alpha P_\pi - (1-\alpha)I & -I \\ \hline C & D \end{array}\right]}_{X_\pi} \underbrace{\left[\begin{array}{c} v_0 \\ \vdots \\ v_{|A|-1} \\ \hline g_0 \\ \vdots \\ g_{|A|-1} \end{array}\right]}_{y} - \underbrace{\begin{bmatrix} \mathbb{E}_{a\sim\pi}\mathcal{C}(s_0,a) \\ \vdots \\ \mathbb{E}_{a\sim\pi}\mathcal{C}(s_{|A|-1},a) \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{b_\pi}$$

with

$$C = \begin{bmatrix} 1 & 0 & \dots \\ 0 & 0 & \dots \\ \vdots & \ddots & \end{bmatrix}, D = \begin{bmatrix} 0 & \dots & & 0 \\ 1 & -1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & -1 \end{bmatrix}.$$

This system combines $\mathcal{L}v(s) = \mathbb{E}_{a\sim\pi(s)}[\mathcal{C}(s,a)] + (\alpha P_\pi + (1-\alpha)I)v$ together with $g(s) = g(s')$ for any $s, s' \in A$ and $v(0) = 0$.

Succinctly, $X_\pi y - b_\pi = 0$. Similarly, we have $X_{\pi_\eta} y' - b_{\pi_\eta} = 0$. Let $dX = X_{\pi_\eta} - X_\pi$, $db = b_{\pi_\eta} - b_\pi$ and $dy = y' - y$ then $(X_\pi + dX)(y + dy) - (b_\pi + db) = 0$. Hence,

$$\begin{aligned} dy &= (X_\pi + dX)^{-1}(db - dXy) \\ &= (I + X_\pi^{-1} dX)^{-1} X_\pi^{-1}(db - dXy). \end{aligned}$$

We calculate $\|dX\|_\infty$:

$$\begin{aligned} \|dX\|_\infty &= \max_{s\in A} \sum_{s'\in A} |\alpha P_{\pi_\eta}(s,s') - \alpha P_\pi(s,s')| \\ &= \max_{s\in A} \sum_{s'\in A} |\alpha((1-\eta)P_\pi(s,s') + \eta P_{Unif}(s,s')) - \alpha P_\pi(s,s')| \\ &= \alpha\eta \max_{s\in A} \sum_{s'\in A} |P_{Unif}(s,s') - P_\pi(s,s')| \\ &\leq \alpha\eta 2|A|. \end{aligned}$$

By a similar argument, together with $\mathcal{C} \leq c_{\max}$, then $\|db\|_\infty \leq 2\eta c_{\max}$ Hence,

$$\begin{aligned} \|dy\|_\infty &\leq \|(I + X_\pi^{-1} dX)^{-1}\|_\infty \|X_\pi^{-1}\|_\infty (\|db\|_\infty + \|dX\|_\infty \|y\|_\infty) \\ &\leq \frac{\|X_\pi^{-1}\|_\infty}{1 - \|X_\pi^{-1}\|_\infty \|dX\|_\infty} (\|db\|_\infty + \|dX\|_\infty \|y\|_\infty) \\ &\leq \frac{\eta \|X_\pi^{-1}\|_\infty}{1 - 2\alpha|A|\eta\|X_\pi^{-1}\|_\infty} (2c_{\max} + 2\alpha|A|\|y\|_\infty). \end{aligned}$$

By selecting

$$\eta \leq \eta^* = \frac{\epsilon_{(1)}}{\|X_\pi^{-1}\|_\infty(2c_{\max} + 2\alpha|A|\|y\|_\infty) + \epsilon_{(1)}2\alpha|A|\|X_\pi^{-1}\|_\infty},$$

we get that $\|dy\|_\infty \leq \epsilon_{(1)}$ and therefore $|g_\pi^P - g_{\pi_\eta}^P| \leq \epsilon_{(1)}$, as desired. □

Lemma E.2.8. *(Simulation Lemma, Avg. Cost) Fix some $\alpha \in (0,1)$ arbitrary. Let $\tilde{P}$ be the optimistic dynamics achieving the inner minimum of the Bellman equation with respect to $\mathcal{L}^{\alpha}_{PR}$ (see Table 7.1) in the AMEC given by $(A, \mathcal{A}_A)$. Let π be the η^* stochastic policy as in Lemma E.2.7. For some error $\epsilon_{(2)} > 0$. Let $m \in \mathbb{N}$ be the smallest value such that $\Delta((\alpha\tilde{P}_\pi + (1-\alpha)I)^m) < 1$. When n is large enough that $\psi(n) \leq \frac{1}{\alpha^2}\left(\left(\frac{\epsilon_{(2)}(1-\Delta(\tilde{P}^m_{\alpha,\pi}))}{|A|c_{\max}} + 1\right)^{1/m} - 1\right)$ then*

$$|g^P_\pi - g^{\tilde{P}}_\pi| < \epsilon_{(2)}. \tag{E.6}$$

For Π_A, the set of deterministic policies in A, let $\bar{\Delta}_A = \max_{\pi\in\Pi_A} \Delta((\alpha P_{\pi^*_\eta} + (1-\alpha)I)^m)$ where $m = \max_{\pi\in\Pi_A} \min_{m\in\mathbb{N}}\{m|\Delta((\alpha P_{\pi^*_\eta} + (1-\alpha)I)^m) < 1$. Then, in particular, (E.6) *holds after $n = \tilde{\mathcal{O}}(\frac{|A|^2 c^2_{\max}}{\epsilon^2_{(2)}(1-\bar{\Delta}_A)^2})$ samples are collected for each state-action pair in $(A, \mathcal{A}_A)$.*

Proof. Consider, notationally, $P_\alpha(s,a,s') = \alpha P(s,a,s') + (1-\alpha)\mathbf{1}_{\{s=s'\}}$ be an aperiodicity transform with $\alpha \in (0,1)$. When fixed by a policy, then $P_{\alpha,\pi} = \alpha P_\pi + (1-\alpha)I$. By [166] (Prop. 8.5.8), aperiodicity transforms do not affect gain. Hence $g^P_\pi = g^{P_\alpha}_\pi$ and $g^{\tilde{P}}_\pi = g^{\tilde{P}_\alpha}_\pi$. Let x_{π,P_α} be the stationary distribution of π in P_α and $x_{\pi,\tilde{P}_\alpha}$ be the stationary distribution of π in $\tilde{P}_\alpha$. These quantities exist due to the fact that π has full support over $\mathcal{A}_A$ making both $P_\alpha, \tilde{P}_\alpha$ ergodic (finite, irreducible, recurrent, and aperiodic). Hence,

$$\begin{aligned}
|g^P_\pi - g^{\tilde{P}}_\pi| &= |g^{P_\alpha}_\pi - g^{\tilde{P}_\alpha}_\pi| \\
&= |\mathbb{E}_{s\sim x_{\pi,P_\alpha}}[\mathbb{E}_{a\sim\pi(s)}[\mathcal{C}(s,a)]] - \mathbb{E}_{s\sim x_{\pi,\tilde{P}_\alpha}}[\mathbb{E}_{a\sim\pi(s)}[\mathcal{C}(s,a)]]| \\
&= |\sum_{s\in A} \mathbb{E}_{a\sim\pi(s)}[\mathcal{C}(s,a)](x_{\pi,P_\alpha}(s) - x_{\pi,\tilde{P}_\alpha}(s))| \\
&\leq c_{\max}\|x_{\pi,P_\alpha} - x_{\pi,\tilde{P}_\alpha}\|_1.
\end{aligned}$$

To bound $\|x_{\pi,P_\alpha} - x_{\pi,\tilde{P}_\alpha}\|_1$, we appeal to classic stationary-distribution perturbation bounds [41]. First, since $\tilde{P}_{\alpha,\pi}$ is ergodic then $\exists m_0 < \infty$ such that for any $m \geq m_0$ then $\Delta(\tilde{P}^m_{\alpha,\pi}) < 1$. Then, in particular, $\|x_{\pi,P_\alpha} - x_{\pi,\tilde{P}_\alpha}\|_1 \leq \frac{\|\tilde{P}^m_{\alpha,\pi} - P^m_{\alpha,\pi}\|_\infty}{1-\Delta(\tilde{P}^m_{\alpha,\pi})}$ [181, 41]. Let $E = P_{\pi,\alpha} - \tilde{P}_{\pi,\alpha}$, and thus $\|E\|_\infty = \alpha\|P_\pi - \tilde{P}_\pi\|_\infty \leq \alpha|A|\psi(n)$. Then,

$$\begin{aligned}
\|\tilde{P}^m_{\alpha,\pi} - P^m_{\alpha,\pi}\|_\infty &= \|\tilde{P}^m_{\alpha,\pi} - (\alpha P_\pi + (1-\alpha)I)^m\|_\infty \\
&= \|\tilde{P}^m_{\alpha,\pi} - (\alpha E + \alpha\tilde{P}_\pi + (1-\alpha)I)^m\|_\infty \\
&= \|\tilde{P}^m_{\alpha,\pi} - (\alpha E + \tilde{P}_{\alpha,\pi})^m\|_\infty \\
&\leq (\alpha\|E\|_\infty + 1)^m - 1 \\
&\leq (\alpha^2|A|\psi(n) + 1)^m - 1.
\end{aligned}$$

where in the second-to-last inequality uses that $\|\tilde{P}_{\alpha,\pi}\|_\infty = 1$ and $\|AB\|_\infty \leq \|A\|_\infty \|B\|_\infty$ for matrices A, B. Putting it all together we have that

$$|g_\pi^P - g_\pi^{\tilde{P}}| \leq c_{\max} \frac{(\alpha^2 |A| \psi(n) + 1)^m - 1}{1 - \Delta(\tilde{P}_{\alpha,\pi}^m)}. \tag{E.7}$$

We therefore require that

$$\psi(n) \leq \frac{1}{\alpha^2 |A|} \left(\left(\frac{\epsilon_{(2)}(1 - \Delta(\tilde{P}_{\alpha,\pi}^m))}{c_{\max}} + 1 \right)^{1/m} - 1 \right) \tag{E.8}$$

to yield $|g_\pi^P - g_\pi^{\tilde{P}}| < \epsilon_{(2)}$. The equation (E.8) also holds with $\tilde{P}$ replaced with P, with (some other) m appropriate.

In the AMEC $(A, \mathcal{A}_A)$ then there are at most $|\Pi_A| = |A|^{|\mathcal{A}_A|}$ deterministic policies. For each policy $\pi \in \Pi_A$, there is some η_π^* satisfying Lemma $E.2.7$. Let $m = \max_{\pi \in \Pi_A} \min_{m \in \mathbb{N}} \{m | \Delta(P_{\alpha,\pi_{\eta_\pi^*}}^m) < 1\}$ and $\bar{\Delta}_A = \max_{\pi \in \Pi_A} \Delta(P_{\alpha,\pi_{\eta_\pi^*}}^m) < 1$ (recall this is guaranteed because $P_{\alpha,\pi_{\eta_\pi^*}}$ is ergodic). Then, when $\psi(n) < \frac{1}{\alpha^2|A|} \left(\left(\frac{\epsilon_{(2)}(1-\bar{\Delta}_A)}{c_{\max}} + 1 \right)^{1/m} - 1 \right)$ then $|g_\pi^P - g_\pi^{\tilde{P}}| < \epsilon_{((2)}$. By Lemma E.2.2, we have $n = \tilde{\mathcal{O}}(\frac{|A|^{\frac{2}{m}} c_{\max}^{\frac{2}{m}}}{\epsilon_{(2)}^{\frac{2}{m}} (1-\bar{\Delta}_A)^{\frac{2}{m}}}) = \tilde{\mathcal{O}}(\frac{|A|^2 c_{\max}^2}{\epsilon_{(2)}^2 (1-\bar{\Delta}_A)^2})$, since $m = 1$ achieves the maximum. □

Remark E.2.9. *We do not require knowledge of $\bar{\Delta}_A < 1$. The existence is sufficient to guarantee convergence.*

Remark E.2.10. *The function $\Delta(M)$, coefficient of ergodicity of matrix M, is a measure (and bound) of the second largest eigenvalue of M.*

Remark E.2.11. *In this chapter, we have assumed that $m = 1$ and $\alpha = 1$, for simplicity in exposition. For full rigor, m may be larger, though typically small. m can be seen as the smallest value making any column of $P_{\alpha,\pi}^m$ dense. From a computational perspective, it is efficient to compute powers of $\tilde{P}_{\alpha,\pi}$ and stop when $\tilde{P}_{\alpha,\pi}^m$ has a dense column, making $\Delta(\tilde{P}_{\alpha,\pi}) < 1$. From there, we can check if ρ (Line 6, Algo 7) satisfies the r.h.s of Eq* (E.8)*. We present the samples required by maximizing over $m \in \mathbb{N}$.*

`PlanTransient` proofs

Proposition 7.3.3 (`PlanTransient` Convergence & Correctness, Informal)**.** *Denote the cost- and prob-optimal policy as π'. After collecting at most $n = \tilde{\mathcal{O}}(\frac{|\mathcal{S}\setminus\cup_{i=1}^k A_i|^2\bar{V}^4}{c_{\min}^2\epsilon_{PT}^2\epsilon_\varphi^4})$ samples for each $(s,a) \in (\mathcal{S}\setminus\cup_{i=1}^k A_i)\times\mathcal{A}$, the greedy policy π w.r.t. v' (Alg. 8, Line 3) is both cost and probability optimal:*

$$\|\tilde{V}_\pi - \tilde{V}_{\pi'}\| < \epsilon_{PT}, \quad |\mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^k A_i\right] - \mathbb{P}\left[\pi' \text{ reaches } \cup_{i=1}^k A_i\right]| \leq \epsilon_\varphi.$$

Proposition E.2.12 (`PlanTransient` Convergence & Correctness, Formal)**.** *Let $\{A_i, g_i\}_{i=1}^k$ be the set of inputs to Algorithm 8, together with error $\epsilon_{PT} > 0$. Denote the cost- and prob-optimal policy as π'. After collecting at most $n = \tilde{\mathcal{O}}(\frac{|\mathcal{S}\setminus\cup_{i=1}^k A_i|^2\bar{V}^4}{c_{\min}^2\epsilon_{PT}^2\epsilon_\varphi^4})$ samples for each $(s,a) \in (\mathcal{S}\setminus\cup_{i=1}^k A_i)\times\mathcal{A}$, under the event $\mathcal{E}$ and Assumption 7.1 then with probability $1-\delta$, , the greedy policy π w.r.t. v' (Alg. 8, Line 3) is both cost and probability optimal:*

$$\|\tilde{V}_\pi - \tilde{V}_{\pi'}\| < \epsilon_{PT}, \quad |\mathbb{P}\left[\pi \text{ reaches } \cup_{i=1}^k A_i\right] - \mathbb{P}\left[\pi' \text{ reaches } \cup_{i=1}^k A_i\right]| \leq \epsilon_\varphi.$$

Proof of 7.3.3. Convergence follows from boundedness of $\|v\| \leq \bar{V}$, and monotone convergence and is well studied [166, 94, 199, 66].

Fix $\lambda > 0$ and drop it from the notation $V_{\pi,\lambda}^P$. Let $\tilde{V}_*^{\tilde{P}}$ be the value function for the optimal policy in $\tilde{P}$. For the approximation error, we have

$$0 \leq \tilde{V}_\pi^P - \tilde{V}_*^P = \underbrace{\tilde{V}_\pi^P - \tilde{V}_\pi^{\tilde{P}}}_{(a)} + \underbrace{\tilde{V}_\pi^{\tilde{P}} - \tilde{V}_*^P}_{(b)} < \epsilon_{\texttt{PT}}.$$

For (a) we appeal to Lemma E.2.15 and set $\epsilon_{(3)} = \epsilon_{\texttt{PT}}/2$ requiring that $\psi(n) = \frac{\epsilon_{\texttt{PT}} c_{\min}}{14|\mathcal{S}\setminus\cup_{i=1}^k A_i|\bar{V}^2(1+\frac{1}{\epsilon_\varphi})^2}$, occuring when $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S}\setminus\cup_{i=1}^k A_i|\bar{V}^2}{\epsilon_{\texttt{PT}}\epsilon_\varphi^2 c_{\min}})^2)$ samples per state-action pair have been collected. For (b), by Lemma E.2.16, by selecting $\epsilon_{\texttt{PT}}^{\mathcal{L}} = \frac{c_{\min}\epsilon_{\texttt{PT}}\epsilon_\varphi}{4\bar{V}}$ we have that

$$\begin{aligned} V_\pi^{\tilde{P}} - V_*^P &\leq (1 + \frac{2\epsilon_{\texttt{PT}}^{\mathcal{L}}}{c_{\min}})v - V_*^P \\ &= \frac{2\epsilon_{\texttt{PT}}^{\mathcal{L}} v}{c_{\min}} \\ &\leq \frac{2\bar{V}\epsilon_{\texttt{PT}}^{\mathcal{L}}}{\epsilon_\varphi c_{\min}} \leq \frac{\epsilon_{\texttt{PT}}}{2}. \end{aligned}$$

For the probability of satisfaction, by Prop E.2.13, we have that π and π^* coincide in probability of reaching the states in $\cup_{i=1}^k A_i$. □

Proposition E.2.13 (Selecting a bound on $\|v\|$)**.** *Let $\{A_i, g_i\}_{i=1}^k$ be the set of inputs to Algorithm 8. Let π' have maximal probability of reaching*

$\cup_{i=1}^{k} A_i$. *Then, with error* $\epsilon_\varphi > 0$, *bounding* $\|v\|_\infty = \|\mathcal{L}_{PT} v\|_\infty \leq \frac{\bar{V}}{\epsilon_\varphi}$ *where* $\bar{V} \geq \left(\frac{1}{\beta^{|\mathcal{S}|}}\left(\frac{1-\beta^{|\mathcal{S}|}}{1-\beta}\right) + \lambda\right) c_{\max}$ *guarantees that* π *returned by Algorithm 8 is near probability optimal:*

$$|\mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi' \models \varphi]| < \epsilon_\varphi.$$

Proof of E.2.13. Suppose $\bar{V} \geq J_\pi + \lambda c_{\max}$ for any $\pi \in \Pi$. Let $\frac{\bar{V}}{\epsilon_\varphi}$ be chosen as upper bound on $\|v\| = \|\mathcal{L}_{\mathtt{PT}} v\|$. Denote $\mathbb{P}[\pi \models \varphi]$ as p, and $\mathbb{P}[\pi' \models \varphi]$ as p^*. Suppose, for contradiction, $p^* - p > \epsilon_\varphi$, yet π is returned by the Algorithm. This would imply that $\tilde{V}_\pi \leq \tilde{V}_{\pi'}$.

Hence,

$$\begin{aligned} 0 \leq \tilde{V}_{\pi'} - \tilde{V}_\pi &\leq \underbrace{p^*(J_{\pi'} + \lambda \sum_{i=1}^{k} \frac{p_i^*}{p^*} \widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}})}_{\leq J_\pi + \lambda c_{\max}} - \underbrace{p(J_\pi + \lambda \sum_{i=1}^{k} \frac{p_i}{p} \widehat{g}^{\tilde{P}}_{\tilde{\pi}_{A_i}})}_{\geq 0} + \underbrace{(p - p^*)}_{< -\epsilon_\varphi} \frac{\bar{V}}{\epsilon_\varphi} \\ &< J_\pi + \lambda c_{\max} - \bar{V} \\ &\leq 0. \end{aligned}$$

Hence, we have a contradiction. Thus, $|p^* - p| \leq \epsilon_\varphi$ if $\bar{V} \geq J_\pi + \lambda c_{\max}$ for any $\pi \in \Pi$. In fact, since the solution to $\mathcal{L}_{\mathtt{PT}}$ is deterministic, it suffices to consider only deterministic Π.

We will now bound $J_\pi = \mathbb{E}_{\tau \sim \mathrm{T}_\pi}\left[\sum_{t=0}^{\kappa_\pi} \mathcal{C}(s_t, \pi(s_t)) \middle| \tau \models \varphi\right] \leq c_{\max} \mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau | \tau \models \varphi]$, as this is the only unknown quantity. Here $\mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau | \tau \models \varphi]$ is the expected number of steps it takes π to leave the transient states. This means that a worst-case bound would be a policy that remains in the transient states as long as possible.

We construct the worst-case scenario and give a justification, a formal proof follows from induction. Suppose the starting state is s_0. If π induces a prob-1 transition back to s_0 then s_0 is recurrent, and so κ_τ would be small. Instead, π induces a prob $1 - \beta$ transition to s_0 and a prob β transition to s_1. Notice that the transition to s_1 must be at least probability β due to Assumption 7.1. Again, if s_1 gave all of its probability to s_1 or s_0 then a MEC would form and strictly decrease κ_τ. This process repeats until we reach state $s_{|\mathcal{S}|-1}$, which has to have a self-loop. If it does not, then, again a large MEC would form and decrease κ_τ. Of course, this is the well known chain graph, with easily computable expected hitting time: $\mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau] \leq \frac{1}{\beta^{|\mathcal{S}|}} \frac{1-\beta^{|\mathcal{S}|}}{1-\beta}$. By making $s_{|\mathcal{S}|-1}$ the accepting state, then $\mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau | \tau \models \varphi] = \mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau] = \frac{1}{\beta^{|\mathcal{S}|}} \frac{1-\beta^{|\mathcal{S}|}}{1-\beta}$ achieves

the bound. Any other choice of accepting states would strictly decrease κ_τ. Hence, we can select

$$\bar{V} \geq \left(\frac{1}{\beta^{|\mathcal{S}|}} \left(\frac{1-\beta^{|\mathcal{S}|}}{1-\beta} \right) + \lambda \right) c_{\max} \geq J_\pi + \lambda c_{\max},$$

completing the proof. □

Remark E.2.14. *It may also be possible to empirically estimate J_π rather than take the bound from Prop E.2.13, considering that we have the structure of P through $\widehat{P}$. We give the high level idea. We know all of the AMECs and rejecting EC, so we have all the transient states (denoted T). Then for some policy π and $P' \in \mathcal{P}$, submatrix $Q_\pi(s,s') = P'_\pi(s,s')$ for $s,s' \in T$ represents the transitions in the transient states. It is well known that $\mathbb{E}_{\tau \sim \mathrm{T}_\pi}[\kappa_\tau] = \|(I-Q)^{-1}\|_\infty$. Taking the max over all $\pi \in \Pi$, $P' \in \mathcal{P}$, and finally multiplying by $c_{\max}$ gives a bound on J_π.*

Lemma E.2.15. *(Simulation Lemma, Transient Cost [198]) Consider an MDP $(\mathcal{S}, \mathcal{A}, \ldots)$. For any two transition functions $P', P'' \in \mathcal{P}$, policy π, and error $\epsilon_{(3)} > 0$ then*

$$\|\tilde{V}_\pi^{P''}\|_\infty = \|\tilde{V}_\pi^{P'}\|_\infty \leq (1+\frac{1}{\epsilon_\varphi})\bar{V}, \|\tilde{V}_\pi^{P'} - \tilde{V}_\pi^{P''}\|_\infty \leq \frac{7|\mathcal{S}|\bar{V}^2(1+\frac{1}{\epsilon_\varphi})^2\psi(n)}{c_{\min}} \leq \epsilon_{(3)},$$

occurring after $n = \tilde{\mathcal{O}}(\frac{|\mathcal{S}|^2\bar{V}^4}{\epsilon_{(3)}^2 \epsilon_\varphi^4 c_{\min}^2})$ samples from each state-action pair in $\mathcal{S} \times \mathcal{A}$.

Proof. Direct consequence of the definition of $\bar{V}$ from Prop E.2.13, application of Lemma 2 from [198] and Lemma E.2.2. □

Lemma E.2.16. *(EVI Bound, [198]) Suppose v is returned by `VI` with accuracy $\epsilon_{PT}^{\mathcal{L}}$ with Bellman equation $\mathcal{L}_{PT}$ (See Table 7.1). Suppose π is greedy with respect to v. If $\epsilon_{PT}^{\mathcal{L}} \leq \frac{c_{\min}}{2}$ then, element-wise,*

$$v \leq \tilde{V}_{\pi^*}^{P}, \quad v \leq \tilde{V}_\pi^{\tilde{P}} \leq (1 + \frac{2\epsilon_{PT}^{\mathcal{L}}}{c_{\min}})v.$$

E.3 Conjecture on Sample Complexity

As we have proven in Theorem 7.4.1, the optimal policy creates a set of AMECs which coincide with $(A_i, \mathcal{A}_i)_{i=1}^k$. For any potential AMEC, we need to guarantee probabilistic closure. For each state-action pair $(s, a) \in A_i \times \mathcal{A}_{A_i}$ we have to sample enough times to guarantee that we have "collected" all of the possible unique transitions (s, a, s'). Indeed, this is similar to the famous coupon collection problem, where we want to know how much time it will take to collect all unique transitions (s, a, s'). Suppose there are m unique tuples each with probability $\beta = \frac{1}{m}$.

We can use a Chebyshev-based lower bound:

$$\mathbb{P}\left[N > m \log m - \log(\frac{1}{\delta})m\right] \geq \delta.$$

Simplifying, we get that $\mathbb{P}\left[N > m \log(\frac{m}{\delta})\right] \geq \delta$. Thus, the number of transitions needed is

$$N = \Omega(m \log(\frac{m}{\delta})) = \Omega(\frac{1}{\beta} \log(\frac{1}{\beta\delta})) = \tilde{\Omega}(\frac{1}{\beta}).$$

Further, [222] show that indeed $N \geq \frac{\beta}{\log \beta}$.

E.4 Additional Algorithms

In this section we discuss the additional subroutines used in this chapter. We discuss the case where selecting $\cup_{i=1}^{k} A_i$ as the terminal states for SSP in Algo 6 can fail and an alternative solution.

Value Iteration

Our version of Value Iteration `VI` (Algo 17) is a two-in-one version, due to the similarity of Relative VI (used in `PlanRecurrent`) and SSP (used in `PlanTransient`). The general idea is that you apply the Bellman Operator $\mathcal{L}$ onto your current iterate v_n repeatedly until $d(v_{n+1}, v_n)$ exceeds ϵ. When we wish to find the gain, then V_T (terminal states) is empty, and we use shifting by the first value of $v_n(0)$ for stability [21]. In other words, we subtract $v_n(0)$ from every value of v_n. On the other hand, if a set of terminal costs is provided then these represent the set of states that we want to reach through SSP and the value $v_n(s) = V_T(s)$ is known and must be kept fixed throughout applications of $\mathcal{L}$. The only difference in our application of $\mathcal{L}$ over standard Bellman operators is that $\mathcal{L}$ is optimistic and has an interior minimization over $\min_{p \in \mathcal{P}(s,a)} p^T v_n$ (See Table 7.1). To solve this, minimization we use a modified version from [94] given in Algo 18. The idea of Algo 18 is simple: put all the mass of $\tilde{P}$ onto the lowest possible values of v_n while still being consistent with $\widehat{P}$. This is efficient as it requires an ordering over v and then a single pass over the states $\tilde{S}$. The calculated probability $p(\tilde{s}_l)$ (see Algo 18) are what we call the optimistic dynamics $\tilde{P}(s, a, \tilde{s}_l)$.

Algorithm 17 Value Iteration (`VI`)

Require: Optimistic Bellman Operator $\mathcal{L}$, Error Measure d, accuracy $\epsilon > 0$, V_T terminal values (optional)

Set $n = 0, v_0 = 0_S, v_1 = \mathcal{L}v_0$
repeat
  $n \xleftarrow{+} 1$
  if V_T is empty **then**
    Shift $v_n \leftarrow v_n - v_n(0)\mathbf{1}$ {Relative Value Iteration}
  else
    $v_n(s) \leftarrow V_T(s)$ for $s \in V_T$ {SSP}
  Apply operator $v_{n+1}, \tilde{P} \leftarrow \mathcal{L}v_n$ {Bellman Backup}
until $d(v_{n+1}, v_n) > \epsilon$
return $v_{n+1}, v_n, \tilde{P}$

Algorithm 18 `InnerMin` (for `PT`/`PR`)

Require: A set of states $\tilde{\mathcal{S}}$, current estimate from VI v_n, estimates $\widehat{P}(s,a,\cdot)$ for a specific (s,a) pair with $s \in \tilde{\mathcal{S}}$, errors $\psi(n)$, lower bound β (See Assumption 7.1)

1: Sort $\tilde{\mathcal{S}} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_m\}$ according to $v_n(\tilde{s}_1) \leq v_n(\tilde{s}_2) \leq \ldots \leq v_n(\tilde{s}_m)$, where v_n is the current

2: Set

$$p(\tilde{s}_1) = \begin{cases} \min(1-\beta, \widehat{P}(s,a,\tilde{s}_1) + \psi(n)), & \widehat{P}(s,a,\tilde{s}_1) \notin \{0,1\} \\ 1, & \widehat{P}(s,a,\tilde{s}_1) = 1 \\ 0, & \widehat{P}(s,a,\tilde{s}_1) = 0 \end{cases}$$

3: For remaining $j > 1$, set $p(\tilde{s}_j) = \widehat{P}(s,a,\tilde{s}_j)$

4: Set $l \leftarrow m$

5: **while** $\sum_{\tilde{s}_j \in \tilde{\mathcal{S}}} p(\tilde{s}_j) > 1$ **do**

6: Reset

$$p(\tilde{s}_l) = \begin{cases} \max(\beta, 1 - \sum_{\tilde{s}_j \neq \tilde{s}_l} p(\tilde{s}_j)), & \widehat{P}(s,a,\tilde{s}_l) \notin \{0,1\} \\ 1, & \widehat{P}(s,a,\tilde{s}_l) = 1 \\ 0, & \widehat{P}(s,a,\tilde{s}_l) = 0 \end{cases}$$

7: Decrement $l \leftarrow l - 1$

8: Set $\tilde{P}(s,a,\tilde{s}) = p(\tilde{s})$ for each $\tilde{s} \in \tilde{S}$

9: **return** $\tilde{P}(s,a,\tilde{s})$

Modified Algorithm handling Blocking Failure in Algorithm 6

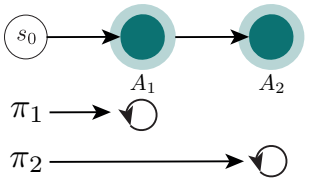


Figure E.1: *Blocking Issue. If A_1 is included in the terminal AMECs (the states we want to reach) then once it is reached π_{A_1} is instantiated and A_1 becomes recurrent, implying only π_1 is considered. However, even though it may be the case that $J_{\pi_1} < J_{\pi_2}$, we may still have $V_{\pi_1} > V_{\pi_2}$. This example demonstrates the necessity to pick the terminal AMECs properly, rather than just the union of all AMECs found, to avoid blocking.*

One of the failure modes of Algorithm 6 is in its selection of which AMECs are the necessary AMECs to reach. In fact, by selecting unnecessary AMECs, the SSP procedure fails to treat some AMECs as transient states when in fact, maybe, lower cost could have been achieved if they were. One way to see this is to consider a single directional chain of AMECS (See Figure E.1). In the figure, two policies can be considered: (1) π_1 that reaches for A_1 and then starts π_{A_1} when A_1 is reached, and (2) π_2 that reaches for A_2 and then starts π_{A_2} when A_2

is reached. It may be the case that $V_{\pi_2} < V_{\pi_1}$ despite $J_{\pi_2} > J_{\pi_1}$, since it requires a longer cost path to reach the desired AMEC. Despite this observation, when A_1 is selected as terminal states in the subroutine `PlanTransient` (Algo 8), we disallow consideration of π_2 at all. As explained in the proof of Theorem 7.4.1, whatever AMECs are induced by π^* coincide with $AMEC = \{A_i, \mathcal{A}_{A_i}\}_{i=1}^k$. Let $\Omega = 2^{AMEC} \setminus \varnothing$, all non-empty subsets of AMECs (possible targets). Since all accepting trajectories of π^* land in an AMEC, then another way of looking at π^* is:

$$\pi^* = \min_{\omega \in \Omega} \min_{\pi \in \tilde{\Pi}(\omega)} V_\pi,$$

where $\tilde{\Pi}(\omega) = \{\pi \in \Pi_{\max} | \pi(s,a) = \pi_{A_i}(s,a) \text{ for } s \in A_i \in \omega, a \in \mathcal{A}_{A_i}(s)\}$, which is a policy class where the only degrees of freedom are outside of ω. In other words, $\pi \in \tilde{\Pi}(\omega)$ is followed until the trajectory hits $A_i \in \omega$ and then π_{A_i} is followed thereafter.

We will reconcile this failure mode of `PlanTransient` through a modified, nonblocking, subroutine `NoBlockPlanTransient` (Algo 19).

Algorithm 19 `NoBlockPlanTransient` (`NB-PT`)

Require: States & gains: $\{(A_i, g_i)\}_{i=1}^k$, err. $\epsilon_{\texttt{PT}} > 0$
1: Set $v(s) = \infty$ for each $s \in \mathcal{S}$.
2: Sample $\varphi_{\texttt{PT}}$ times $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$
3: **for** $\omega \in 2^{\{A_i\}_{i=1}^k} \setminus \varnothing$ **do**
4: Set $V_T(s) = \lambda g_i$ for $s \in A_i \subseteq \omega$
5: $v'_\omega, v_\omega, \tilde{P} \leftarrow \texttt{VI}(\mathcal{L}_{\texttt{PT}}, d_{\texttt{PT}}, \epsilon^{\mathcal{L}}_{\texttt{PT}}, V_T)$
6: **if** $\mathbb{E}_{s \sim d_0}[v'_\omega(s)] < \mathbb{E}_{s \sim d_0}[v(s)]$ **then**
7: Set $v = v'_\omega$
8: Set $\pi \leftarrow$greedy policy w.r.t v
9: **return** π

The proof of correctness follows from the fact that v'_ω closely tracks V_π where π is greedy wrt v'_ω. Then, selecting the smallest V_π coincides with V_{π^*}.

Proposition E.4.1 (Correctness and Convergence of `NoBlockPlanTransient`)**.** *After collecting at most* $n = \tilde{\mathcal{O}}(\frac{|\mathcal{S}|^2 \bar{V}^4}{c^2_{\min} \epsilon^2_{PT} \epsilon^4_\varphi})$ *samples for each* $(s,a) \in \mathcal{S} \times \mathcal{A}$*, under the event* $\mathcal{E}$ *and Assumption 7.1 then with probability* $1 - \delta$*, , the greedy policy* π *w.r.t.* v' *(Alg. 8, Line 3) is both cost and probability optimal:*

$$\|\tilde{V}_\pi - \tilde{V}_{\pi^*}\| < \frac{3\epsilon_{PT}}{2}, \quad |\mathbb{P}[\pi \models \varphi] - \mathbb{P}[\pi^* \models \varphi]| \leq \epsilon_\varphi.$$

Proof. Suppose $v_\omega < v'_\omega$ for any $\omega' \in \Omega$, with $\omega \in \Omega$. Fix some ω'. Denote the greedy policies $\pi_{v_\omega}, \pi_{v_{\omega'}}$ wrt $v_\omega, v_{\omega'}$. Suppose $\tilde{V}_{\pi_{v_{\omega'}}} < \tilde{V}_{\pi_{v_\omega}}$. Then an error was

made and

$$\begin{aligned}
0 &\leq \tilde{V}^P_{\pi_{v_\omega}} - \tilde{V}^P_{\pi_{v_{\omega'}}} \\
&\leq \tilde{V}^P_{\pi_{v_\omega}} - \tilde{V}^{\tilde{P}}_{\pi_{v_\omega}} + \tilde{V}^{\tilde{P}}_{\pi_{v_\omega}} - v_\omega + v_\omega - v_{\omega'} + v_{\omega'} - \tilde{V}^{\tilde{P}}_{\pi_{v_{\omega'}}} + \tilde{V}^{\tilde{P}}_{\pi_{v_{\omega'}}} - \tilde{V}^P_{\pi_{v_{\omega'}}} \\
&\leq \frac{\epsilon_{\texttt{PT}}}{2} + \frac{\epsilon_{\texttt{PT}}}{2} + 0 + 0 + \frac{\epsilon_{\texttt{PT}}}{2} \\
&\leq \frac{3\epsilon_{\texttt{PT}}}{2},
\end{aligned}$$

where the second line comes from grouping each pair of elements from the first line and applying the bounds found in proof of Proposition E.2.12.

On the other hand, suppose $p+\epsilon_\varphi = \mathbb{P}\left[\pi_{v_\omega} \models \varphi\right] + \epsilon_\varphi < \mathbb{P}\left[\pi_{v_{\omega'}} \models \varphi\right] = p'$. The same proof as in Prop E.2.13 applies to show that the probability of satisfaction remains close:

$$\begin{aligned}
0 &\leq \tilde{V}_{\pi_{v_{\omega'}}} - \tilde{V}_{\pi_{v_\omega}} \\
&\leq \underbrace{p'(J_{\pi_{v_{\omega'}}} + \lambda \sum_{i=1}^{k} \frac{p'_i}{p'} \widehat{g}_{\tilde{\pi}_{A_i}})}_{\leq J_\pi + \lambda c_{\max}} - \underbrace{p(J_{\pi_{v_\omega}} + \lambda \sum_{i=1}^{k} \frac{p_i}{p} \widehat{g}_{\tilde{\pi}_{A_i}})}_{\geq 0} + \underbrace{(p-p')}_{<-\epsilon_\varphi} \frac{\bar{V}}{\epsilon_\varphi} \\
&< J_\pi + \lambda c_{\max} - \bar{V} \\
&\leq 0,
\end{aligned}$$

showing that $|p - p'| < \epsilon_\varphi$.

In particular, since the choice of ω' was arbitrary, it holds for ω' achieving $\omega' = \min_{\omega\in\Omega} \min_{\pi\in\tilde{\Pi}(\omega)} V_\pi$. Therefore the previous bounds all hold for with p' replaced with p^* and $\tilde{V}^P_{\pi_{v_{\omega'}}}$ replaced with $\tilde{V}^P_{\pi^*}$.

It is clear we can think of this non-blocking subroutine as checking the different inputs to Algo 8, which requires $\varphi_{\texttt{PT}}(\omega) = \frac{\epsilon_{\texttt{PT}} c_{\min}}{14|\mathcal{S}\setminus\omega|\bar{V}^2(1+\frac{1}{\epsilon_\varphi})^2}$, occuring when $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S}\setminus\omega|\bar{V}^2}{\epsilon_{\texttt{PT}}\epsilon_\varphi^2 c_{\min}})^2)$ samples per state-action pair have been collected. Taking the maximum over $\omega \in \Omega$, we have $n = \tilde{\mathcal{O}}((\frac{|\mathcal{S}|\bar{V}^2}{\epsilon_{\texttt{PT}}\epsilon_\varphi^2 c_{\min}})^2)$ samples required for each state-action pair in $\mathcal{S} \times \mathcal{A}$. □

Remark E.4.2. *Recall $\mathcal{S}^*$ is set of accepting states in Product-MDP $\mathcal{X}$. This subroutine appears to have an exponential runtime in $|\mathcal{S}^*|$; Ω is at most $2^{|\mathcal{S}^*|}$, which is not related to the typical PAC parameters. In general, Ω is modestly small.*

Remark E.4.3. *While the runtime scales poorly with $|\mathcal{S}^*|$, the sample complexity remains PAC.*

Remark E.4.4. *We believe it is possible to bring the runtime of the subroutine to be polynomial in $|\mathcal{S}^*|$ by leveraging the MEC quotient structure (see [16]), but leave that for future work.*

E.5 Experiments

Environments and Details

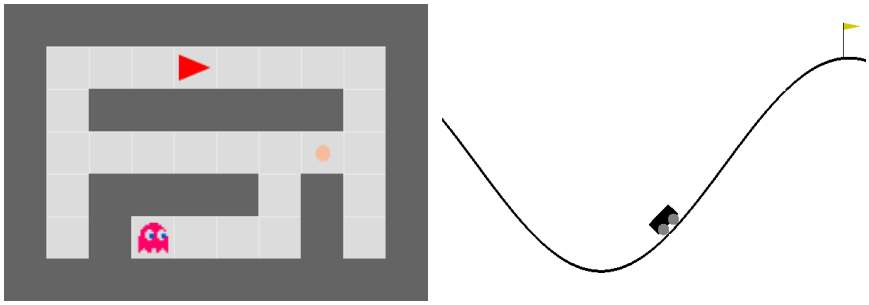


Figure E.2: *Environment Illustrations. (Left) Pacman. φ is for the agent, the red triangle, to eventually collect the food, given by the yellow dot, and always avoid the ghost, the red semicircle with eyes. (Right) Mountain Car (MC). φ is to eventually reach the flag.*

Pacman. This environment (pictured in Fig E.2 Left) is a 5x8 gridworld. The starting positions of the agent (red triangle), food (yellow circle), and ghost (red semicircle with eyes), are as illustrated in Fig E.2. The agent has 4 cardinal directions at each state in addition to a "do nothing" action. The LTL specification is to eventually reach the food and to forever avoid the ghost "F(food) & G(!ghost)", where the food state is labelled "food" and the ghost state is labelled "ghost". Once the food is picked up, it is gone. The ghost chases the agent (following the shortest path) with probability .4 and chooses a random action with probability .6. Though this is an infinite horizon problem, as there is no terminal state, we allow a maximum horizon of $H = 100$ in our experiments. We track how long the agent has avoided the ghost and whether the agent has picked up the food. To simplify verification, we say the agent has satisfied the spec if the food has been picked up and the ghost has been avoided for all H timesteps. The cost function is defined as 1 everywhere.

For the shaped LCRL baseline, we use progression through the LDBA as a "reward": if the agent progresses to a new state in the automaton then the cost of that transition is .1 instead of 1. The authors of LCRL used similar ideas in their code as well. However, we must note that progression-based cost shaping eliminates any guarantee of LTL satisfaction. An agent is incentivized to find cycles in the LDBA rather than find an accepting state. In the case when no such cycles exist, then this form of cost shaping can work.

Mountain Car This domain (pictured in Fig E.2 Right) is a discretization

of the Mountain Car domain from OpenAI [29], with state-space given by tuple (position, velocity) and cost of 1. We discretize the position space into 32 equal size bins and the velocity into 32 geometrically-spaced bins, allowing more granularity around low velocity than high velocity, making 32^2 bins (states) in the MDP. The starting state is the standard MC starting state, but then placed in the appropriate bin. A bin can be converted back to (pos,vel), for purposes of sampling from P, by uniformly selecting from the valid positions/velocities implied by the bin. The agent has 3 actions: accelerate left, do nothing, accelerate right. The specification is to eventually reach the goal state "F(goal)", the standard task, where any bin with position beyond the flag position is labelled "goal".

For the shaped LCRL baseline, we use a cost function of $c = .1$ if the change in position is positive and the agent accelerated right, likewise if the change is negative and the agent accelerated left, otherwise $c = 1$. This cost function should incentivize the agent to seek actions which make the car go faster. Unlike the previous experiment, here cost-shaping has no effect on the guarantee of LTL satisfaction.

Safe Delivery This domain (pictured in Fig 7.1 Right) is a 4-state MDP: (0) start state, (1) sniffed packet, (2) stolen packet (3) delivered packet. In each state, the agent has two actions, A and B. The transition function P in the MDP is given by $P(0, A, 1) = 1$, $P(0, B, 2) = .5$, $P(0, B, 3) = .5$, $P(1, A, 3) = 1$, $P(1, B, 3) = 1$, $P(2, A, 2) = 1$, $P(2, B, 2) = 1$, $P(3, A, 3) = 1$, $P(3, B, 3) = 1$. In other words, choosing action A in the initial state immediately leads to a sniffed packet, which subsequently leads to the packet being delivered by any action. Alternatively, choosing action B in the initial state has a $50 - 50$ chance of having the packet stolen or immediately delivered, regardless of action. Once, stolen, it remains stolen. Once delivered, the packet remains delivered, regardless of action. The states are labelled as $L(0) = L(3) =$"safe". The specification is to always stay in safe states: "G(safe)". Let all the costs be 1. The Product-MDP can be seen in Figure 7.2 Right.

The probability-optimal and cost-optimal policy is then choosing B is state 0 and then arbitrarily afterward. The maximum probability of satisfying the policy is 50% because 50% of the time the packet gets stolen. Though this is an infinite horizon problem, as there is no terminal state, we allow a maximum horizon of $H = 100$ in our experiments. Thus, the average number of timesteps

should be $.5 * H = 50$.

Similarly to Pacman, for the shaped LCRL baseline, we use progression through the LDBA as a "reward": if the agent progresses to a new state in the automaton then the cost of that transition is .5 instead of 1.

Infinite Loop This environment (pictured in Fig 7.1 Left) is a 2x5 gridworld. The agent starts in the bottom right corner. The agent has 4 cardinal directions at each state in addition to a "do nothing" action. We consider two specifications:

φ_1: The LTL specification is to perpetually visit the office (in the top right corner) followed by the coffee room (top left corner): "GF(o & XFc)", where the office is labelled o and the coffee room is labelled c. The Product MDP is illustrated in Figure 7.2 Center.

φ_2: We require the agent to

$$\text{"G((c -> XXXXXo) \& (o ->XXXXXc)) \& Xo"}, \tag{E.9}$$

meaning to get to first get to office in 1 step, then repeatedly reach the coffee room in 5 steps followed by the office in 5 steps.

Similarly to Pacman, for the shaped LCRL baseline, we use progression through the LDBA as a "reward": if the agent progresses to a new state in the automaton then the cost of that transition is .5 instead of 1.

Hyperparameters

We use the following hyperparameters for our experiments. Each set of hyperparameters was run with 20 seeds, with the exception of Safe Delivery which was run with 40 seeds.

Table E.4: Hyperparameters.

Param(s)	Infinite Loop φ_1	Infinite Loop φ_2	Safe Delivery	Pacman	MC
$\bar{V}$	50	50	10	100	150
$c_{\min}$	1	1	1	1	1
$c_{\max}$	1	1	1	1	1
φ	GF(o & XFc)	See φ_2 in (E.9)	G(!unsafe)	F(food0) & G!ghost	Fgoal
ϵ	3	3	3	3	10
δ	.1	.1	.1	.1	.1
LCRL Params	Infinite Loop	Infinite Loop 2	Safe Delivery	Pacman	MC
Max Traj len.	100	100	100	100	200
γ	.99	.99	.99	.99	.95
Learning rate	.95	.95	.95	.95	.9

Additonal Results

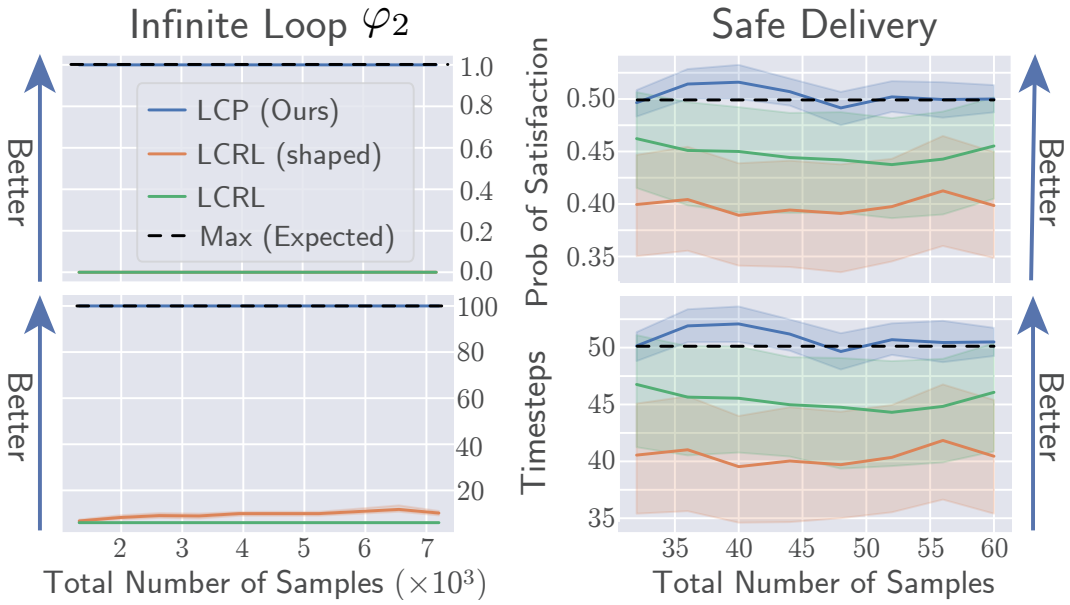


Figure E.3: *Additional Results. (Left Column) Infinite Loop 2. φ is a specific trajectory that needs to be followed: first get to the office in 1 timestep and then the coffee room in 5 and then back to the office in 5, over and over. (Right) Safe Delivery (Right Column). φ is to always be safe.*

In this section we examine additional results for the experiments we ran.

For the Infinite Loop environment under φ_2, we see (Figure E.3 Left Column) that our method is able to follow the trajectory specified by φ_2 even in low sample regimes. The learning signal for LCRL is very poor as the episode terminates extremely quickly if the agent does not get to the next location that it needs to be in within the allotted time. The shaped LCRL only does marginally better, but still struggles to satisfy the LTL with any probability.

For the Safe Delivery environment, we see (Figure E.3 Left Right) that our method picks out the probability-optimal policy. LCRL is nearly optimal. The sparsity of this problem is significantly less as the feedback for spec satisfaction verification comes after a single timestep. Interestingly, cost shaping in Safe Delivery performs worse than straight LCRL. This isn't surprising since, as noted, the verification feedback comes after a single timestep and is more important than any cost-shaping. However, cost-shaping muddles the feedback making shaped LCRL perform worse. We speculate that with only a few hundred or thousand more samples, both LCRL and shaped LCRL would reach the optimal policy. Recall that LCRL and shaped LCRL are not the same as

Q-learning, as they operate in the product-MDP rather than the underlying MDP. Thus, these observations are still consistent with our Motivation section (Section 7.1), insisting that Q-learning would have trouble in this environment.

Policies

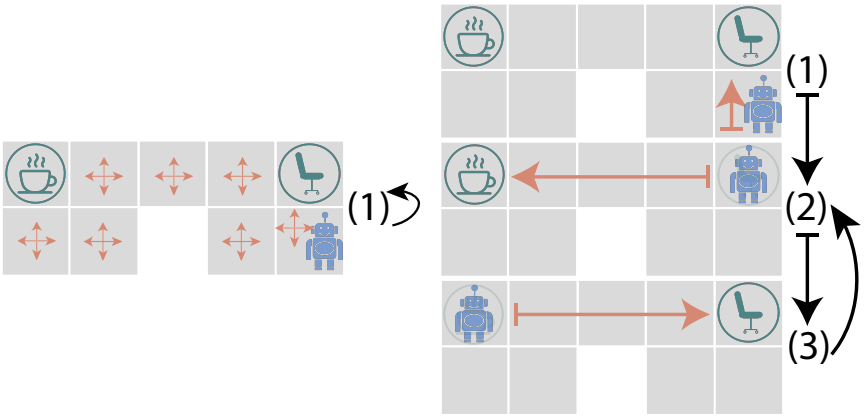


Figure E.4: *Types of policies for different φ. (Left) Infinite Loop φ_1. φ_1 is to go perpetually walk between the office and the coffee room (Right) Infinite Loop φ_2. φ_2 is to get to the office in 1 time step then perpetually, take 5 timesteps to get to the coffee room and 5 steps back to the office.*

In this section we examine the policies induced by different specificity in specifications. In particular, we consider the Infinite Loop environment with two different specifications φ_1, φ_2, see Section E.5, E.5 for a description. For φ_1, we only require that the agent "eventually" navigate between the office and coffee room. The agent is incentivized to stay in place (create a cost-1 cycle) for as long as possible and very infrequently take a random action. Of course, eventually taking random actions will loop the agent between the office and coffee room. This behavior is illustrated in Figure E.4 Left, where the agent is always in LDBA state 1 and takes random actions with low probability and does nothing with high probability. It takes exponential time for the agent to make a single loop between the office and coffee room.

On the other hand, we may want the agent to move quickly. In this case, we can be more specific and use specification φ_2. The behavior for an agent satisfying φ_2 is illustrated in Figure E.4 Right. The agent gets to LDBA state 2 by first reaching the office in a single time step. Then the agent loops between LDBA states 2 and 3 by reaching the coffee room and office, repeatedly, within the allotted time. If the agent does not reach the office or chair within the allotted

time, there is a fourth LDBA state (unpictured) which is a sink denoting failure of the spec. In essence, the LDBA has created the options, or hierarchy, of solving the problem, as noted in Section 7.1. It takes 10 timesteps for the agent to make a single loop between the office and coffee room.

Notice that the high level description of the task is unchanged, but the details of how the task is accomplished is much more specific in φ_2 rather than φ_1. This demonstrates that writing LTL task specifications is flexible, but requires thought about "how" the task should be accomplished.

Appendix F

CHAPTER 6 APPENDIX

F.1 Experiments

Environment Details

The environment and experiment details are summarized in Table F.1.

Table F.1: Environment Details.

Environment	Experiment	$\mathcal{S}^{\mathcal{M}}$	$\mathcal{A}^{\mathcal{M}}$	Dynamics	LTL Formula
Minecraft	Q-learning	Discrete	Discrete	Deterministic	$GF(y \,\&\, XF(b)) \,\&\, (G\neg r)$
Pacman	Q-learning	Discrete	Discrete	Stochastic	$F(\text{food}) \,\&\, (G\neg\text{ghost})$
Flatworld 1	Q-learning	$\mathbb{R}^2$	Discrete	Deterministic	FGy
Flatworld 2	Q-learning	$\mathbb{R}^2$	Discrete	Deterministic	$GF(y \,\&\, XF(r)) \,\&\, (G\neg b)$
Flatworld 3	PPO	$\mathbb{R}^2$	$[0,1]^2$	Deterministic	FGy
Flatworld 4	PPO	$\mathbb{R}^2$	$[0,1]^2$	Deterministic	$GF(y \,\&\, XF(r)) \,\&\, (G\neg b)$
Carlo	PPO	$\mathbb{R}^5$	$[-1,1]^2$	Deterministic	$GF(\text{zone1} \,\&\, XF(\text{zone2})) \,\&\, (G\neg\text{crash})$

Experiment Setup

Each experiment is run with 10 random seeds. Results from Figure 8.2 are from an average over the seeds.

Q-learning experiments. Let k be the greatest number of jump transitions available in some LDBA state $k = \max_{b\in\mathcal{S}^{\mathcal{B}}} |\mathcal{A}^{\mathcal{B}}(b)|$. Let $m = \max_{s\in\mathcal{S}^{\mathcal{M}}} |\mathcal{A}^{\mathcal{M}}(s)|$. The neural network $Q_\theta(s)$ takes as input $s \in \mathcal{S}^{\mathcal{M}}$ and outputs $\mathbb{R}^{(m+k)\times|\mathcal{S}^{\mathcal{B}}|}$ a $(m+k)$-dim vector for each $b \in \mathcal{S}^{\mathcal{B}}$. For our purposes, we consider $Q_\theta(s,b)$ to be the single $(m+k)$-dim vector cooresponding to the particular current state of the LDBA b.

When $\mathcal{S}^{\mathcal{M}}$ is discrete then we parametrize $Q_\theta(s,b)$ as a table. Otherwise, $Q_\theta(s,b)$ is parameterized by 3 linear layers with hidden dimension 128 with intermediary ReLU activations and no final activation. After masking for how many jump transitions exist in b, we can select $\arg\max_{i\in[0,\ldots,|\mathcal{A}^{\mathcal{B}}(b)|]} Q_\theta(s,b)_i$ the highest Q-value with probability $1-\eta$ and uniform with η probability. Here, η is initialized to η_0 and decays linearly (or exponentially) at some specified frequency (see Table F.2).

At each episode (after a rollout of length T), we perform K gradient steps with

different batches of size given in Table F.3. We use Adam optimizer [107] with a learning rate also specified by the table.

When in a continuous state space, we implement DDQN [86] (rather than DQN) with a target network that gets updated at some frequency specified by Table F.3.

Table F.2: Hyperparameters for Q-learning experiments (Discrete Action Space).

	η		η Decay								
Experiment	η_0	Min η	Type	Rate	Freq	Batch size	K (batches)	LR	Target update	T	γ
Minecraft	.3	0	Exp	.9	100	128	20	-	-	100	.99
Pacman	.4	0	Linear	.05	400	512	200	-	-	100	.999
Flatworld 1	.8	.15	Exp	.9	100	128	5	.001	15	20	.95
Flatworld 2	.8	.15	Exp	.9	100	128	5	.001	15	50	.95

PPO experiments. Let k be the greatest number of jump transitions available in some LDBA state $k = \max_{b \in \mathcal{S}^{\mathcal{B}}} |\mathcal{A}^{\mathcal{B}}(b)|$. The neural network $f_\theta(s)$ takes as input $s \in \mathcal{S}^{\mathcal{M}}$ and outputs $\mathbb{R}^{(k+2)\times|\mathcal{S}^{\mathcal{B}}|}$ is a $(k+2)$-dim vector for each $b \in \mathcal{S}^{\mathcal{B}}$. For our purposes, we consider $f_\theta(s, b)$ to be the single $(k+2)$-dim vector cooresponding to the particular current state of the LDBA b.

$f_\theta(s, b)$ is parameterized by 3 linear layers with hidden dimension 64 with intermediary ReLU activations. The first dimension corresponds to sampling a Gaussian action $a \sim \mathcal{N}(f_\theta(s, b)[0], \mathrm{diag}(\sigma^2))$ where σ is initialized to σ_0 (see Table F.3) and decays exponentially (at a rate given in the table) every 10 episodes. The remaining $k+1$ dimensions (after proper masking to account for the size of $|\mathcal{A}^{\mathcal{B}}(b)|$ and softmax) represent the probability $p = [p_a, p_{\epsilon_0}, \ldots, p_{\epsilon_k}]$ of taking either the MDP action a or a some jump transition ϵ_i. We sample from a Categorical(p) variable to select whether to return $a \sim \mathcal{N}(\mathrm{Tanh}(f_\theta(s, b)[0]), \mathrm{diag}(\sigma^2))$ or $a = \epsilon_i$ for some i. The density can be calculated by multiplying p_a by the Gaussian density when a is selected, and p_{ϵ_i} otherwise.

For the critic, we have a parametrized network $f_\varphi(s, b) \to \mathbb{R}$ of 3 linear layers with hidden dimension 64 with intermediary Tanh activations and no final activation.

At each episode (after a rollout of length T), we perform 5 gradient steps with different batches of size given in Table F.3. The importance sampling term

in PPO is clipped to $1 \pm .4$. The critic learning rate is .01. We use Adam optimizer [107] for both the actor and critic.

Table F.3: Hyperparameters for PPO experiments (Continuous Action Space).

Experiment	σ_0	σ Decay Rate	Min σ	Batch size	LR Actor	T
Flatworld 3	1.8	.98	.3	128	.001	20
Flatworld 4	1.8	.99	.1	128	.001	50
Carlo	.5	.999	.3	16	.0001	500

F.2 Constructing feasible trajectories for policy gradient during rollout

Algorithm 20 `LCER` for Policy Gradient (Option 2)

Require: Dataset D. Trajectory τ of length T.
1: Set $\mathcal{T}_0 \leftarrow \{(s_0, b) | b \in \mathcal{B}\}$
2: **for** $(s_t, a_t, s_{t+1}) \in \tau$ **do**
3: Form $\mathcal{T}_t$ according to Eq (F.1)
4: Set $D \leftarrow D \cup \mathcal{T}_T$
5: **return** D

Suppose we wanted to generate feasible trajectories in realtime while the policy is being rolled out. That is, we have a partial trajectory of the form $\tau_t = (s_0, b_0, a_0, \ldots, s_t, b_t)$ generated by running π in P. Let $a_t = a \in \mathcal{A}$ be the t-th action taken by π and $s_{t+1} = s' \in \mathcal{M}$ be the next observed state observed in the MDP.

Let $\mathcal{T}_t$ be the current set of feasible (partial) trajectories at timestep t. Elements $\tau_k = (s_0, b_0, a_0, \ldots, s_k, b_k) \in \mathcal{T}_t$ denote k-step (partial) trajectory, not necessarily part of the trajectory observed during the course of a rollout of π. Here, $k \geq t$. Then, for each $\tau_k \in \mathcal{T}_t$, one of 4 cases holds:

Case 1. Action a is not a jump transition (ie. $a \in \mathcal{A}^{\mathcal{M}}(s_k)$) and there are no jump transitions available in b_k ($\mathcal{A}^{\mathcal{B}}(b_k) = \varnothing$). Then we can form the concatenation: $\tau_{k+1} = \tau_k \cup (a, s', b_{k+1})$ where $b_{k+1} = P^{\mathcal{B}}(b_k, L^{\mathcal{M}}(s'))$. We set $\mathcal{T}_\epsilon = \varnothing$.

Case 2. Action a is a jump transition and is currently feasible in b_k (ie. $a \in \mathcal{A}^{\mathcal{M}}(b_k)$). Then we can form the concatenation $\tau_{k+1} = \tau_k \cup (a, s', b_{k+1})$ where $b_{k+1} = P^{\mathcal{B}}(b_k, a)$. We set $\mathcal{T}_\epsilon = \varnothing$.

Case 3. Action a is a not a jump transition (ie. $a \in \mathcal{A}^{\mathcal{M}}(s_k)$), but there is at least one feasible jump transition in b_k (ie. $\mathcal{A}^{\mathcal{B}}(b_k) \neq \varnothing$). Then, in addition to forming τ_{k+1} from Case 1, we have all the possible jumps:

$$\mathcal{T}_\epsilon = \{\tau_k \cup (\epsilon, s_k, b_{k+1}, a, s', b_{k+2}) | \forall \epsilon \in \mathcal{A}^{\mathcal{B}}(b_k), b_{k+1} = P^{\mathcal{B}}(b_k, \epsilon), \\ b_{k+2} = P^{\mathcal{B}}(b_{k+1}, a_t)\}.$$

Case 4. Action a is a jump transition is infeasible in b_k (ie. $a \notin \mathcal{A}^{\mathcal{B}}(b_k)$). In this case, we just pass this trajectory. Setting $\tau_{k+1} = \tau_k$ and $\mathcal{T}_\epsilon = \varnothing$.

At the end of iterating over each element of $\tau_k \in \mathcal{T}_t$ and forming τ_{k+1} and $\mathcal{T}_\epsilon$,

we can update our current set of feasible trajectories:

$$\mathcal{T}_{t+1} = \cup_{\tau_k \in \mathcal{T}_t} \Big((\mathcal{T}_t \setminus \{\tau_k\}) \cup \{\tau_{k+1}\} \cup \mathcal{T}_\epsilon \Big). \tag{F.1}$$

To put this process simply, we are swapping out τ_k for τ_{k+1} and also adding in any jump transitions if they are available. The algorithm can be seen in Algo 20.