Towards Generalization and Efficiency in Reinforcement Learning

Wen Sun

CMU-RI-TR-19-37

April 25th, 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

Abstract

Different from classic Supervised Learning, Reinforcement Learning (RL), is fundamentally *interactive* : an autonomous agent must learn how to behave in an unknown, uncertain, and possibly hostile environment, by actively interacting with the environment to collect useful feedback to improve its sequential decision making ability. The RL agent will also intervene in the environment: the agent makes decisions which in turn affects further evolution of the environment.

Because of its generality– most machine learning problems can be viewed as special cases– RL is hard. As there is no direct supervision, one central challenge in RL is how to explore an unknown environment and collect useful feedback *efficiently*. In recent RL success stories (e.g., super-human performance on video games [Mnih et al., 2015]), we notice that most of them rely on *random exploration strategies*, such as ϵ-greedy. Similarly, policy gradient method such as REINFORCE [Williams, 1992], perform exploration by injecting randomness into action space and hope the randomness can lead to a good sequence of actions that achieves high total reward. The theoretical RL literature has developed more sophisticated algorithms for efficient exploration (e.g., [Azar et al., 2017]), however, the sample complexity of these near-optimal algorithms *has* to scale exponentially with respect to key parameters of underlying systems such as dimensions of state and action space. Such exponential dependence prohibits a direct application of these theoretically elegant RL algorithms to large-scale applications. In summary, without any further assumptions, RL is hard, both in practice and in theory.

In this thesis, we attempt to gain purchase on the RL problem by introducing additional assumptions and sources of information.The first contribution of this thesis comes from improving RL sample complexity via imitation learning. Via leveraging expert's demonstrations, imitation learning significantly simplifies the tasks of exploration. We consider two settings in this thesis: interactive imitation learning setting where an expert is available to query during training time, and the setting of imitation learning from observation alone, where we only have a set of demonstrations that consist of observations of the expert's states (no expert actions are recorded). We study in both theory and in practice how one can imitate experts to reduce sample complexity compared to a pure RL approach. The second contribution comes from model-free Reinforcement Learning. Specifically, we study policy evaluation by building a general reduction from policy evaluation to no-regret online learning which is an active research area that has well-established theoretical foundation. Such a reduction creates a new family of algorithms for provably correct policy evaluation under very weak assumptions on the generating process. We then provide a through theoretical study and empirical study of two model-free exploration strategies: exploration in action space and exploration in parameter space. The third contribution of this work comes from model-based Reinforcement Learning. We provide the first exponential sample complexity separation between model-based RL and general model-free RL approaches. We then provide PAC model-based RL algorithm that can achieve sample efficiency simultaneously for many interesting MDPs such as tabular MDPs, Factored MDPs, Lipschitz continuous MDPs, low rank MDPs, and Linear Quadratic Control. We also provide a more practical model-based RL framework, called Dual Policy Iteration (DPI), via integrating optimal control, model learning, and imitation learning together. Furthermore, we show a general convergence analysis that extends the existing approximate policy iteration theories to DPI. DPI generalizes and provides the first theoretical foundation for recent successful practical RL algorithms such as ExIt and AlphaGo Zero [Anthony et al., 2017, Silver et al., 2017], and provides a theoretical sound and practically efficient way of unifying model-based and model-free RL approaches.

To my wife, Li Zhu, and my parents, Jianhua Shen and Mengjun Sun

Acknowledgements

First and foremost, I am greatly thankful to my advisor, Drew Bagnell, who, throughout my five years at Carnegie Mellon University, has pointed me to interesting research problems, provided me insightful discussions, and gave me freedom to pursuit my own research ideas. Drew himself is an inspiring person. As the CTO of Aurora, he comes to CMU every Wednesday to spend time with us to discuss research ideas, read research papers, and give advice on our research projects. His desire to learn about new papers, to hunt for new research directions, and to understand new research problems, have been motivating me: if a busy CTO still has the time and the enthusiasm for research, there will be no excuse for me! Drew is a great advisor, a great researcher and a great role model! For that, I am forever grateful.

I would like to thank my thesis committee members: Geoffrey Gordon, Martial Hebert, and Byron Boots, who have provided extremely useful discussion. In addition, many thanks to Geoff Gordon and Byron Boots, who have been providing guidance on my research at various stages of its development. After Drew took his sabbatical leave during my first year at CMU, it was Geoff and Bryon who immediately provided help and welcomed me to collaborate with them and their students. Our collaboration has been fruitful and I hope I will have the chance to work with and learn from Geoff and Byron in the future.

I am also grateful to Ron Alterovitz, my former advisor at the University of North Carolina at Chapel Hill. Ron has contributed greatly to my interests for research in robotics, Artificial Intelligence, and machine learning. I also would like to thank the robotics lab at UNC-CH, where I started to develop my research interests in AI and started to learn how to conduct research. I would like to thank Jur van den Berg, Sachin Patil, and Luis Torres, for their help and guidance.

Throughout my five years at CMU, I had the chance to collaborate with great colleagues who have provided invaluable help on my research. Many thanks to Arun Venkatraman, for his collaboration on the project of Predictive State Inference Machines, and the related follow up works. I also would like to thank other LairLab members and friends, Jiaji Zhou, Hanzhang Hu, Zhao Song, Debadeepta Dey, Allie Del Giorno, Anirudh Vemula, Shervin Javdani, Kevin Waugh, Matt Barnes, Sanjiban Choudhury, Abhijeet Tallavajhula, Sibi Venkatesan, Venkataramanan Rajagopalan, Tejas Sudharshan Mathai, Wenhao Luo, Peiyun Hu, Nick Rhinehart, Kumar Shaurya Shankar, Leonid Keselman, Akshara Rai, Ahmed Hefny, and Zita Marinho for many insightful discussions on research ideas.

Many thanks to Ashish Kapoor, Debadeepta Dey, Alina Beygelzimer, John Langford, Akshay Krishnamurthy, Alekh Agarwal, and Nan Jiang, for providing me internship opportunities and mentoring me. Working with them has been a pleasure and a great privilege, and I would love to continue the collaboration with them in my future career.

Many thanks to Kris Kitani, David Held, Chris Atkenson, and Sidd Srinivasa for helping me and giving me suggestions for my academic job search. I would like to thank Kris for going through my job talk slides one by one. I would like to thank Chris for listening to my practice talk and providing valuable comments. I would like to thank David for sharing his academic job search experience which made me better prepared for interview questions. I would like to thank Sidd for writing a recommandation letter for me and for his encouragement.

I also would like to thank Suzanne Muth for making things here easier for me, for many cups of coffee, and for letting me play with her rabbits.

To my parents, Jianhua Shen and Mengjun Sun, who provided me the support and the education that made me who I am today. For that, I am forever indebted to them.

Last but not least, I would like to thank my wife, Dr. Li Zhu, for her unconditional love and for teaching me statistics in her spare time (e.g., A/B test, Lasso, Group Lasso, MCMC, and so on). My journey through graduate school would not have been as much of a pleasure without her!

List of Figures

List of Tables

Contents

Chapter 1

Introduction

A fundamental challenge in Artificial Intelligence (AI), robotics, and language processing is sequential prediction: to reason, plan, and make a sequence of predictions or decisions to minimize accumulated cost, achieve a long-term goal, or optimize for a loss acquired only after many predictions. Reinforcement Learning (RL), as a general framework for learning from experience to make predictions and decisions, is often considered as one of the perfect tools for solving such a challenge in AI. Recently, equipped with the advancement from Deep Learning literature, we have dramatically advanced the state-of-the-art of RL on a number of applications including simulated high-dimensional robotics control, video games, and board games [Mnih et al., 2015, Schulman et al., 2015a, Silver et al., 2017, 2016].

Classic supervised machine learning considers settings where training examples come all at once and a learner makes predictions that have not effect on future examples it will receive, i.e., the examples the learner received are generated from a fixed unknown distribution, identically and independently, without being intervened by the learner. Reinforcement Learning (RL), on the other hand, is *interactive*, where an autonomous agent must learn how to behave in an unknown, uncertain, and possibly hostile environment. By interacting with the environment, the agent receives sensory feedback that describes the current state of the environment, and reward signals that correspond to achieving some specific task. There is no human in the loop that tries to provide direct supervision to the agent. The goal in RL is to learn an optimal strategy (i.e., a policy) that maximizes the total reward.

Because of its generality—RL is a powerful framework that summarizes many special machine learning algorithms and applications—RL is hard. Without any prior knowledge about the environment where an RL agent is operating, the RL agent needs to explore: it must explore unknown part of the environment or try actions that have not been tested before to discover if there are high-reward regions or better behavior strategies. The exploration strategies an RL agent leverages largely determine the sample complexity: the amount of interaction with the environment before the agent can learn a near-optimal policy. Most of the existing successful practical RL algorithms typical deploy the simplest ever exploration strategy—random exploration. For instance, in Q-learning, we often explore by ϵ-greedy: with probability ϵ, we randomly pick an action; otherwise we follow suggestions from the latest Q function. For non-zero ϵ, while ϵ-greedy guarantees asymptotic optimality (i.e., with infinite amount of training time and interaction with the environment, we will discover the optimal policy), it requires huge amount of interactions with the environment. Similarly, policy gradient based method such as REINFORCE [Williams, 1992], relies on stochastic policy to introduce randomness in the action space, such that by randomly trying different actions, we hope we can discover a sequence of behaviors that lead to higher total reward, and then adapt the policy in a way such that the likelihood of generating such a good sequence of behaviors is increased. Such purely randomized exploration strategies often can work when one does not care the cost of samples (e.g., simulation), but inevitably leads to poor sample efficiency. This is one of the reasons why today's most successful RL stories are still living in ideal playgrounds: Atari video games, board games, and robotics simulations.

RL is even hard in simplified worlds (e.g., discrete Markov Decision Processes). Looking into the theoretical RL literature where researchers have been focusing on developing provably sample efficient RL algorithms for discrete MDPs, we notice that in the worst case, the sample complexity of any RL algorithms, have to suffer an exponential dependency on the key parameters of the underlying MDPs such as dimensions of state and action spaces. Such exponential dependency potentially will prohibits a direct application of these mathematically beautiful RL algorithms on real-world large-scale problems. What is even worse is that such exponential dependency is not improvable, i.e., there exists a set of MDPs where no RL algorithm can break such dependency!

Hence, in general, without any further assumptions, RL is hard. Our main focus in this thesis is to improve the efficiency of RL, by introducing additional assumptions and sources of information.

1.1 Main Contributions

This thesis included the following major three contributions.

1.1.1 Imitation Learning

We consider the imitation learning setting where we have access to expert during training (but not in test time) and also the reward signals during training [Chang et al., 2015b, Choudhury et al., 2017a,b, Ross and Bagnell, 2014, Sun et al., 2017c], and (2) the setting of imitation learning from observations alone, where we only have a set pre-collected expert demonstrations that consists of observations of the expert's states without expert's actions being recorded. For interactive imitation learning with reward signal setting, we argue that by leveraging an expert and the reward signals—combining reinforcement and imitation, not only we can learn a good policy that is in par with the expert (hence less sensitive to local optimality) with up to exponentially less samples than any RL approaches (hence more sample efficient), we can also learn a policy that outperforms the expert, with reasonable assumptions. We provide detailed analysis of IL versus RL in sample complexity, and a set of practical IL algorithms that can leverage modern deep networks in Chapter 2.

For the second setting, we developed an algorithm *Forward Adversarial Imitation Learning (FAIL)*, that learns a sequence of time-dependent policies forward in time using Integral Probability Metrics. We show both in theory and in practice that FAIL achieves excellent sample efficiency.

1.1.2 Model-Free Reinforcement Learning

We study model-free Reinforcement Learning from two directions: (1) Online Policy Evaluation and (2) model-free exploration. We generalize policy evaluation [Baird, 1995, Sutton, 1988] by providing a reduction from policy evaluation to no-regret online learning. As no-regret online learning is an extremely active research area, such reduction creates a family of new algorithms for provably correct policy evaluation under very weak assumptions on the generating process. Also any new, faster no-regret online learning algorithm can be immediately plugged into the reduction framework and leads to a new, faster policy evaluation algorithm. As policy evaluation plays an important role in almost all RL algorithms (e.g., actor-critic [Konda and Tsitsiklis, 2000]), a faster and more general policy evaluation approach can directly benefit these RL algorithms.

For model-free exploration, we provide a through theoretical and empirical study of two exploration strategies: exploration in action space and exploration in parameter space. Both exploration strategies are commonly used in RL literature. Policy Gradient methods, such as REINFORCE [Williams, 1992], is the representative of the strategy of exploration in action space. The representatives of exploration in parameter spaces are derivative-free (or zeroth-order) optimization techniques that directly approximate the policy gradient using ideas of finite differencing [Bagnell and Schneider, 2001, Heidrich-Meisner and Igel, 2008, Mania et al., 2018, Mannor et al., 2003, Salimans et al., 2017, Sehnke et al., 2010, Tesch et al., 2011]. Although policy gradient methods leverage *more* information, notably the Jacobian of the action with respect to policy, black-box policy search methods have at times demonstrated better empirical performance (see the discussion in [Kober et al., 2013, Mania et al., 2018]). These perhaps surprising results motivate us to analyze: *In what situations should we expect parameter space policy search methods to outperform action space methods?*

1.1.3 Model-Based Reinforcement Learning

We study model-based RL from both theoretical and practical perspectives. Theoretically, we provide the first exponential sample complexity separation between model-based RL and model-free RL. Specially, we first provide a general definition of model-free RL, which is general enough to capture common model-free methods such as Q-learning, fitted value iteration, delayed Q-learning and so on. Under this definition of model-free RL, we show that there exists a family of MDPs (including common MDPs such as Factored MDPs), where there exists a model-based RL method can learn in sample complexity scales polynomially with respect to all relevant parameters, while for any model-free RL algorithm that falls into our definition, it has to spend sample complexity that scales at least exponentially with respect to the horizon of the problem. This result partially resolves a long debate between model-based RL and model-free RL from a theoretical perspective.

We also identify a large family of MDPs that could be solved by a new model-based RL algorithm we proposed. Specifically, we identity a new complexity measure for MDPs: *witness model rank*, which can be shown to capture specific complexities that were previously used to study model-based RL algorithms for problems such as Factored MDPs (e.g., number of parameters in the underlying conditional probability tables), Lipschitz continuous MDPs (covering number of the underlying state space), tabular MDPs (number of unique states), and Linear Quadratic Regulator (dimension of the underlying state space). We propose a model-based RL algorithm whose sample complexity scales quadratically with respect to the underlying Witness Model Rank, and has no explicitly polynomial dependency on the total number of unique states in the MDPs. Hence, we show that MDPs with low Witness Model Rank is PAC learnable.

We then provide a more practical model-based RL framework called Dual Policy Iteration (DPI). DPI improves sample efficiency of RL via model-based optimal control by using a reduction of RL to IL. DPI is a general framework that provides a principle way of unifying model-free RL and model-based RL approaches, and is a framework where we maintain and alternatively update two policies: a fast reactive policy that aims to provide generalization, and locally optimal policies that are computed by model-based optimal control approaches with learned local dynamics. We can use any stochastic optimal control algorithms (e.g., [Sun et al., 2016b, Todorov and Li, 2005]) to compute policies that are locally optimal around the current reactive, non-linear policy. The reactive policy then treats the locally optimal policy as an intermediate "expert" and leverage the expert's direct supervision to improve itself (i.e., policy iteration is reduced to simple supervised learning). We present the detailed convergence analysis and a practical algorithm in Chapter 7.

1.2 Background

We provide background knowledge in this section. We give a brief introduction to Markov Decision Process, Concentration of measure, Offline Supervised Learning, and no-regret online learning here. Much more detailed definitions of these concepts will appear in the following chapters again. The purpose of this section is to give a brief introduction of the models and statistical tools that we will use throughout the entire thesis. But each chapter is self-contained and has more detailed definitions with variants that are tailored to specific problems considered in the chapter.

1.2.1 Fundamentals of Markov Decision Process

Finite Horizon MDPs

Finite-horizon Markov Decision Process (MDP) [Bellman, 1957] is defined as $(\mathcal{S}, \mathcal{A}, P, R, \rho_0, H)$. Here, $\mathcal{S}$ is a set of S states and $\mathcal{A}$ is a set of A actions; at time step t, P is the transition dynamics such that for any $s_t \in \mathcal{S}, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$, $P(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} from state s_t by taking action a_t at step t; R is the reward function such that a reward r_t at step t is computed as $R(s_t, a_t)$. We denote ρ_0 as the initial distribution of states, and $H \in \mathbb{N}^+$ as the finite horizon of the MDP.

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps from a state to a distribution over actions. Given π, we define state-action Q function $Q_t^\pi(s, a)$ as follows:

$$Q_t^\pi(s,a) \triangleq R(s,a) + \mathbb{E}_{s' \sim P_{s,a}}\left[V_{t+1}^\pi(s')\right],$$

where $V_t^\pi(s)$—the value function, is defined as:

$$V_t^\pi(s) \triangleq \mathbb{E}\left[\sum_{i=t}^{H} r(s_i, a_i)\Big| a_i \sim \pi_{s_i}, s_{i+1} \sim P_{s_i, a_i}, s_t = s\right].$$

The objective function $J(\pi)$ is defined as:

$$J(\pi) \triangleq \mathbb{E}_{s_1 \sim \rho_0} V_1^\pi(s_1),$$

and the goal is to find a policy, from a given policy set Π, that maximizes the objective function.

If the reward function and the transition dynamics is known, then one can compute the optimal policy using Dynamic Programming (DP). Define $Q^\star_H(s,a) \triangleq r(s,a)$, we perform DP as follows:

$$V^\star_t(s) \triangleq \max_{a\in\mathcal{A}} Q^\star_t(s,a), \quad Q^\star_{t-1}(s,a) \triangleq r(s,a) + \mathbb{E}_{s'\sim P_{s,a}}[V^\star_t(s')], \forall t \in [2,H].$$

With $Q^\star_t$, the optimal policy is defined as $\pi_t(s) \triangleq \arg\max_{a\in\mathcal{A}} Q^\star_t(s,a)$. The above procedure is also called Value Iteration.

Closed-forms of $Q^\star$, $V^\star$, and $\pi^\star$ can be obtained for special cases including Tabular MDPs where $Q^\star_t$ can be represented by a table with size $|\mathcal{S}| \times |\mathcal{A}|$, and Linear Quadratic Regulator, where $Q^\star_t(s,a)$ is a quadratic polynomial with respect to s and a.

Discounted Infinite Horizon MDPs

A discounted infinite horizon MDP is defined as $(\mathcal{S},\mathcal{A},P,R,\gamma)$, where $\gamma \in (0,1)$ is a discount factor. For discounted finite horizon MDPs, the definitions of state-action Q functions, value functions, and optimal policies become time-independent. For any state s and policy π, let us define $V^\pi(s)$ as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} r(s_t,a_t) | a_t \sim \pi_{s_t}, s_{t+1} \sim P_{s_t,a_t}, s_1 = s\right],$$

and define $Q^\pi(s,a)$ using Bellman equation:

$$Q^\pi(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V^\pi(s')].$$

Value Iteration One way to solve the discounted infinite horizon MDP is *Value Iteration*, which we explains how and why it works below.

Let us define the operator $\Gamma^\star : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ of Bellman Update as follows:

$$(\Gamma^\star V)(s) \triangleq \max_a \left(r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V(s')]\right).$$

One can show that starting from any $V \in \mathbb{R}^{|\mathcal{S}|}$, value iteration updates V by applying Bellman update: $V = \Gamma^\star V$. We can show that $(\Gamma^\star)^n V \to V^\star$ as $n \to \infty$, which can be proved by showing the $\Gamma^\star$ is a contraction map in infinite norm:

$$\begin{aligned}
|\Gamma^\star V(s) - V^\star(s)| &= \left|\max_a (r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V(s')]) - V^\star(s)\right| \\
&\leq \max_a \left| r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V(s')] - (r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V^\star(s')])\right| \\
&= \max_a \gamma \left| \mathop{\mathbb{E}}_{s'\sim P_{s,a}}[V(s') - V^\star(s')]\right| \leq \gamma \|V - V^\star\|_\infty,
\end{aligned}$$

where the first inequality uses fact that $|\max_x f(s) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$ for any functions f, g, and uses Bellman equation to expand $V^\star(s)$. Since the above inequality holds for any $s \in \mathcal{S}$, we conclude that $\|\Gamma^\star V - V^\star\|_\infty \leq \gamma \|V - V^\star\|_\infty$. The above derivation basically proves that Value Iteration converges to the optimal solution.

Policy Evaluation The problem of policy evaluation is to compute V^π or Q^π given any policy π. Starting from any initialization $V_0 \in \mathbb{R}^{|\mathcal{S}|}$, we perform the following update:

$$V_{t+1}(s) = \left(r(s, \pi(s)) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,\pi(s)}} [V_t(s')] \right) \triangleq (\Gamma^\pi V_t)(s). \tag{1.1}$$

We show below that the V_t from the above iterative procedure converges to V^π:

$$|V_{t+1}(s) - V^\pi(s)| = \left| r(s, \pi(s)) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,\pi(s)}} [V_t(s')] - r(s, \pi(s)) - \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,\pi(s)}} V^\pi(s') \right|$$
$$= \gamma \left| \mathop{\mathbb{E}}_{s' \sim P_{s,\pi(s)}} [V_t(s') - V^\pi(s')] \right| \leq \gamma \|V_t - V^\pi\|_\infty.$$

Hence, we conclude that $\|V_t - V^\star\|_\infty \leq \gamma^t \|V_0 - V^\pi\|_\infty$, and $(\Gamma^\pi)^\infty V_0 = V^\pi$.

Policy Iteration Another approach to solve discounted infinite horizon MDP is *Policy Iteration*. Starting from any policy $\pi_0 : \mathcal{S} \to \Delta(\mathcal{A})$, policy iteration updates policy as follows:

$$\pi_{t+1}(s) = \arg\max_{a \in \mathcal{A}} r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,a}} [V^{\pi_t}(s')], \forall s \in \mathcal{S},$$

where V^{π_t} is the value function of π_t, which could be computed by policy evaluation methods such as Eq. 1.1.

To show policy iteration converges to the optimal solution, we first need to show that policy iteration improves every iteration in the following sense:

$$V^{\pi_{t+1}}(s) \geq V^{\pi_t}(s), \forall s \in \mathcal{S}. \tag{1.2}$$

We prove the above inequality as follows. For any $s \in \mathcal{S}$,

$$V^{\pi_t}(s) \leq \max_a \left(r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,a}} [V^{\pi_t}(s')] \right)$$
$$= r(s, \pi_{t+1}(s)) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,\pi_{t+1}(s)}} [V^{\pi_t}(s)] \triangleq \left(\Gamma^{\pi_{t+1}} V^{\pi_t}\right)(s).$$

For any two vectors $V, V' \in \mathbb{R}^{|\mathcal{S}|}$ such that $V \leq V'$ entry-wise, we must have $\Gamma^{\pi_{t+1}} V \leq \Gamma^{\pi_{t+1}} V'$ entry-wise. Since we show that $V^{\pi_t} \leq \Gamma^{\pi_{t+1}} V^{\pi_t}$ entry-wise, we must have $\Gamma^{\pi_{t+1}} V^{\pi_t} \leq (\Gamma^{\pi_{t+1}})^2 V^{\pi_t}$, which implies that $V^{\pi_t} \leq (\Gamma^{\pi_{t+1}})^2 V^{\pi_t}$. Repeat the above procedure many times, we must have $V^{\pi_t} \leq (\Gamma^{\pi_{t+1}})^\infty V^{\pi_t} = V^{\pi_{t+1}}$, where the final equality is the consequence of the contraction property of the policy evaluation procedure shown in Eq. 1.1. Hence we prove Eq. 1.2.

Now we can show that Policy Iteration is a contraction map in the following sense:

$$V^\star(s) - V^{\pi_{t+1}}(s) = \max_a \left(r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,a}} V^\star(s') \right) - \max_a \left(r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,a}} V^{\pi_t}(s') \right)$$
$$\leq \max_a \gamma \left(\mathop{\mathbb{E}}_{s' \sim P_{s,a}} (V^\star(s') - V^{\pi_t}(s')) \right) \leq \gamma \|V^\star - V^{\pi_t}\|_\infty.$$

Note that the above inequality holds for any $s \in \mathcal{S}$ and $V^\star \geq V^\pi$ for any π entry-wise, we conclude that $\|V^\star - V^{\pi_{t+1}}\|_\infty \leq \gamma \|V^\star - V^{\pi_t}\|_\infty$, which implies that $\|V^\star - V^{\pi_t}\|_\infty \leq \gamma^t \|V^\star - V^{\pi_0}\|_\infty$.

Remark Note that the contraction result from Value Iteration is different from the one in Policy Iteration. Value iteration shows that the constructed value function V is approaching to $V^\star$ (V there does not have to correspond to any policy), while in Policy Iteration, it is the current policy's value function V^π approaching to $V^\star$.

Linear Programming Lastly, one can also solve discounted infinite horizon MDP using Linear Programming (LP). Define $\mu : \mathcal{S} \to (0, 1]$. The optimal solution of the following LP equal to $V^\star$:

$$\begin{aligned} &\min_{V \in \mathbb{R}^{|\mathcal{S}|}} \sum_s V(s)\mu(s), \\ &\text{s.t., } \forall s \in \mathcal{S}, a \in \mathcal{A}, V(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s'). \end{aligned}$$

One can also show that $V^\star$ is the unique solution of the above LP, as long as $\mu(s) > 0$ for any $s \in \mathcal{S}$.

One can show that the dual of the above LP is:

$$\begin{aligned} &\max_{\lambda \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}, \lambda_{x,a} \geq 0} \sum_{x,a} \lambda_{s,a} r(s,a), \\ &\text{s.t.,} \forall s \in \mathcal{S}, \mu(s) + \gamma \sum_{s',a'} \lambda_{s',a'} P(s|s',a') = \sum_{a'} \lambda_{s,a'}, \end{aligned}$$

where the interpretation of $\lambda_{s,a}$ is the $\lambda_{s,a}$ is the (unnormalized) state action visitation of the optimal policy with μ as the initial state distribution (here we assume $\mu(s) \geq 0, \sum_s \mu(s) = 1$). The optimal policy induced from $\lambda_{s,a}$ is $\pi(a|s) = \lambda_{s,a} / \sum_a \lambda_{s,a}$.

Remark The procedures of Value Iteration, Policy Iteration, Policy Evaluation, and Linear Programming that we present above all assume that the transition kernel P and the reward function r are given, and are only computationally efficient when the number of states and actions are relative small.

1.2.2 Concentration Inequalities

We will use three concentration inequalities throughout the thesis: Azuma-Hoeffding's inequality for Martingale difference sequences, Hoeffding's inequality and Bernstein's inequality for bounded independent random variables.

Theorem 1 (Azuma-Hoeffding). *Assume $\{x_t\}_{t=0}$ is a Martingale (i.e., $\mathbb{E}[|x_t|] < \infty, \mathbb{E}[x_t|x_{0,\dots,t-1}] = x_{t-1}$ for all t), and for any t, $|x_t - x_{t-1}| \leq c \in \mathbb{R}^+$. then we have:*

$$Pr(|x_n - x_0| \geq t) \leq 2\exp(-t^2/(2Nc^2)).$$

Denote $\{y_t\}_{t=0}$ as $y_t = x_{t+1} - x_t$. The sequence $\{x_t\}_t$ being Martingale implies the $\{y_t\}_t$ is a Martingale difference sequence, i.e., $\mathbb{E}[|y_t|] < \infty$, and $\mathbb{E}[y_t|y_{0,..,t-1}] = 0$.

Hoeffding's inequality deals with independent random variables (note they do not have to be identically distributed. Independence is enough here).

Theorem 2 (Hoeffding's inequality)**.** *Assume $\{x_t\}_{t=1}^N$ are independent random variables with mean zero, and for all t, $x_t \in [a,b]$ with $a, b \in \mathbb{R}$, then, denote $\bar{x} = \sum_{i=0}^N x_i/N$ we have:*

$$Pr(|\bar{x}| \geq t) \leq 2\exp(-2nt^2/(b-a)^2).$$

Bernstein's inequality not only considers the boundness of random variables, but also considers the variance of the random variables. When the variance of random variables is small, then Bernstein's inequality will be sharper than Hoeffding's inequality.

Theorem 3 (Bernstein's inequality)**.** *Assume $\{x_t\}_{t=1}^N$ are independent random variables with mean zero, $|x_i| \leq c$ for all i, and $\sigma^2 = (1/N)\sum_{i=1}^N Var(x_i)$, then denote $\bar{x} = (1/N)\sum_{i=1}^N x_i$, we have:*

$$Pr(|\bar{x}| \geq t) \leq 2\exp(-2nt^2/(2\sigma^2 + 2ct/3)).$$

We refer readers to Wainwright [2019] for a detailed introduction of concentration of measure. Below we provide a proof for Hoeffding's inequality, as the proof strategy is widely used in proving variants of concentration inequalities. For simplicity, we only prove it for sub-Gaussian random variables. One can show the random variables with bounded values and Gaussian random variables are special cases of Sub-Gaussian.

Definition 1 (sub-Gaussian)**.** *A random variable x is a $\sigma-$sub-Gaussian if $\mathbb{E}[\exp(\lambda(x-\mu))] \leq \exp(\lambda^2\sigma^2/2)$, with $\mu = \mathbb{E}[x]$ (or $Pr(|x-\mu| \geq t) \leq 2\exp(-t^2/(2\sigma^2))$ for $t > 0$).*

Proof of Hoeffding's inequality. We will prove one side of Hoeffding's inequality under the assumption that x_i is zero mean, σ-sub-Gaussian random variable.

$$Pr\left(\sum_{t=1}^N x_t \geq \epsilon\right) = Pr\left(\exp(\lambda\sum_{t=1}^N x_t) \geq \exp(\lambda\epsilon)\right) \leq \frac{\mathbb{E}\left(\exp(\lambda\sum_{t=1}^N x_t)\right)}{\exp(\lambda\epsilon)}$$
$$= \exp(-\lambda\epsilon)\prod_{t=1}^N \mathbb{E}[\exp(\lambda x_t)] \leq \exp(-\lambda\epsilon)\prod_{t=1}^N \exp(\lambda^2\sigma^2/2) = \exp(-\lambda\epsilon + \lambda^2\sigma^2 N/2).$$

where the first inequality uses Markov inequality, and the second equality uses the independence of random variables, the second inequality uses the definition of σ sub-Gaussian random variable.

Now set $\lambda = \epsilon/(\sigma^2 N)$, and substitute it back to the above inequality, we have:

$$Pr(\sum_{t=1}^N x_t \geq \epsilon) \leq \exp\left(-\epsilon^2/(2\sigma^2 N)\right).$$

One can show that for random variable $x \in [a,b]$, x is a sub-Gaussian with $\sigma = (b-a)/2$. Substitute $\sigma = (b-a)/2$ into the above inequality, we finish the proof. □

We will also use *Union Bound* extensively throughout the thesis. Consider a countable set of events $A_1, ..., A_N$. Union Bound states that:

$$Pr(\cup_{i=1}^N A_i) \leq \sum_{i=1}^N Pr(A_i),$$

which can be proved by induction.

1.2.3 Offline Learning

We consider offline supervised learning setting here. In offline supervised learning setting, we have a dataset with finitely many training samples $\{x_i, y_i\}_{i=1}^N$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. For example, for binary classification task, $x_i \in \mathbb{R}^d$ represents the feature and $y_i \in \{-1, 1\}$ represents the label. We also have a hypothesis class $\mathcal{F}$, where $f \in \mathcal{F} : \mathcal{X} \to \mathbb{R}$, and a loss function $\ell : \mathcal{F} \times \mathbb{X} \times \mathcal{Y} \to [0,1]$. Again, in binary classification task, $\mathcal{F}$ could be all linear predictors with bounded norm $\mathcal{F} \triangleq \{\theta : \|\theta\|_2 \leq c\}$, and $\ell(\theta, x, y) = \max\{0, -\theta^\top xy\}$—namely ℓ is a hinge loss.

We further assume that training data is i.i.d sampled from some unknown but fixed distribution, i.e., $(x_i, y_i) \sim \mathcal{D}$. Empirical Risk Minimization (ERM) finds a hypothesis by minimizing the loss averaged over the training data points:

$$\hat{f} = \arg\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^N \ell(f, x_i, y_i). \tag{1.3}$$

We are interested in the performance of $\hat{f}$ under the true distribution $\mathcal{D}$: $\mathbb{E}_{x,y\sim\mathcal{D}}[\ell(\hat{f}, x, y)]$, which we call the *risk* of $\hat{f}$.

We are interested in comparing the performance of $\hat{f}$ to the performance of the best possible hypothesis $f^\star = \arg\min_{f\in\mathcal{F}} \mathbb{E}_{\mathcal{D}}[\ell(f, x, y)]$. We can relate the risk of $f^\star$ to the risk of $\hat{f}$ using the complexity of the hypothesis class $\mathcal{F}$. In this thesis, for analysis simplicity, we will assume the hypothesis class $\mathcal{F}$ is discrete but could be exponentially large. We use $\log(|\mathcal{F}|)$ to represent the complexity of $\mathcal{F}$. Note that the analysis in this thesis that contains $\log(|\mathcal{F}|)$ uses classic concentration inequalities and uniform convergence analysis. Hence, it is standard to replace $\log(|\mathcal{F}|)$ by statistical complexities such as VC-dimeonsion, Rademacher complexity, and Covering dimension (e.g., [Shalev-Shwartz and Ben-David, 2014]).

Theorem 4. *Fix $\delta \in (0,1)$. Assume $\|\ell\|_\infty \leq C \in \mathbb{R}^+$, with probability at least $1-\delta$, the solution $\hat{f}$ from ERM (Eq. 1.3) has the following property:*

$$\mathbb{E}_{\mathcal{D}}[\ell(\hat{f}, x, y)] \leq \min_{f^\star\in\mathcal{F}} \mathbb{E}_{\mathcal{D}}[\ell(f^\star, x, y)] + O\left(\sqrt{\frac{C\log(|\mathcal{F}|/\delta)}{N}}\right). \tag{1.4}$$

Proof. Fix a $f \in \mathcal{F}$. Let us use Hoeffding's inequality to relate $(1/N)\sum_{i=1}^N \ell(f, x_i, y_i)$ to $\mathbb{E}_{\mathcal{D}}[\ell(f, x, y)]$.

By Hoeffding's inequality, we know that:

$$Pr\left(\left|(1/N)\sum_i \ell(f, x_i, y_i) - \mathbb{E}[\ell(f, x, y)]\right| \geq t\right) \leq 2\exp(-Nt^2/2C^2),$$

where we use the assumption that $\|\ell\|_\infty \leq C$. Set $2\exp(-Nt^2/2C^2) = \delta$ and solve for t, we have $t = \sqrt{\frac{2C^2\log(2/\delta)}{N}}$. Namely with probability at least $1-\delta$, we have:

$$\left|(1/N)\sum_i \ell(f, x_i, y_i) - \mathbb{E}[\ell(f, x, y)]\right| \leq \sqrt{\frac{2C^2\log(2/\delta)}{N}}.$$

Now apply Union Bound over $\mathcal{F}$, we have that with probability at least $1-\delta$, we have that for all $f \in \mathcal{F}$:

$$\left|(1/N)\sum_{i=1}^{N}\ell(f,x_i,y_i) - \mathbb{E}[\ell(f,x,y)]\right| \leq \sqrt{\frac{2C^2\log(2|\mathcal{F}|/\delta)}{N}}.$$

Note that the above inequality holds for a particular function $\hat{f}$ and $f^\star = \arg\min_{f\in\mathcal{F}}\mathbb{E}[\ell(f,x,y)]$ as well. Hence, we have:

$$\begin{aligned}\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\hat{f},x,y)] &\leq (1/N)\sum_{i=1}^{N}\ell(\hat{f},x_i,y_i) + \sqrt{\frac{2C^2\log(2|\mathcal{F}|/\delta)}{N}} \\ &\leq (1/N)\sum_{i=1}^{N}\ell(f^\star,x_i,y_i) + \sqrt{\frac{2C^2\log(2|\mathcal{F}|/\delta)}{N}} \\ &\leq \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\ell(f^\star,x,y)] + 2\sqrt{\frac{2C^2\log(2|\mathcal{F}|/\delta)}{N}}.\end{aligned}$$

□

The above theorem essentially shows a tradeoff between the performance and the richness of the hypothesis class $\mathcal{F}$. If we increase the complexity of $\mathcal{F}$, then $\min_{f^\star\in\mathcal{F}}[\ell(f^\star,x,y)]$ will decrease (e.g., richer function approximator can fit data better), but at the cost of increasing the finite sample error $\sqrt{\frac{C\log(|\mathcal{F}|/\delta)}{N}}$. On the other hand, one can restrict the power of the hypothesis class to reduce the finite sample error, but at the risk of increasing the minimum risk. In Chapter 6, we will use the above derivation extensively. We refer readers to Shalev-Shwartz and Ben-David [2014] for more general results with continuous hypothesis classes.

1.2.4 Online Learning

In online learning (e.g., [Shalev-Shwartz et al., 2012]), there are two components, a learner and an environment. Every round t, the learner has to propose a decision $x_t \in \mathcal{X}$ first based on all the information she has received so far from the first round, and then the environment reveals a loss function $\ell_t : \mathcal{X} \to [0,1]$, and the learner suffers loss $\ell_t(x_t)$. The learner's performance is quantified by Regret:

$$\text{Regret}_T \triangleq \sum_{t=1}^{T}\ell_t(x_t) - \min_{x\in\mathcal{X}}\sum_{t=1}^{T}\ell_t(x).$$

We call the learner no-regret if and only if Regret grows sublinearly $o(T)$. In other words, when $T\to\infty$, we have $\text{Regret}/T \to 0$. Namely, in a long run, the learner is doing as well as the best decision in hindsight (i.e., $\arg\max_{x\in\mathcal{X}}\sum_{t=1}^{T}\ell_t(x)$).

When $\mathcal{X}$ and $\ell_t(\cdot)$ are both convex, then there exist many online learning algorithms that can achieve the no-regret property. We refer readers to Shalev-Shwartz et al. [2012] for detailed introduction of many no-regret online learning algorithms. Below, we explain, perhaps, the simplest no-regret online learning algorithm—Online Gradient Descent (OGD).

OGD operates as follows. At round $t+1$, given the decision x_t the learner made at round t, the learner proposes a new decision x_{t+1} as follows:

$$x_{t+1} = \Pi_{\mathcal{X}}\left(x_t - \eta_t \nabla_x \ell_t(x)|_{x=x_t}\right), \tag{1.5}$$

where the projection operator $\Pi_{\mathcal{X}}(x)$ is defined as $\arg\min_{x' \in \mathcal{X}} \|x - x'\|_2$ (here we simply focus on Euclidean Space). Assuming T is known (this assumption can be removed), and settting $\eta_t = 1/\sqrt{T}$, we can show that OGD's regret grows at most $O(\sqrt{T})$ [Zinkevich, 2003].

Note that OGD does not assume any statistical assumption on the generation of the loss sequence ℓ_t. Indeed, the loss functions could be generated in an arbitrary manner. This gives us a tool to analyze machine learning algorithms that are beyond the traditional i.i.d setting.

Except OGD, we will use Follow the Regularized Leader (FTRL) in this thesis, which we explain now. Let us define $R : \mathcal{X} \to \mathbb{R}$ as the α-strongly convex regularization function ($\alpha \in \mathbb{R}^+$).[1] FTRL operates as follows:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{t} \ell_i(x) + (1/\eta_t) R(x).$$

The regularization forces x_{t+1} to stay close to x_t (i.e., imagine the case where $\eta_t \to 0$). When ℓ_i is convex and R is strongly convex, with proper setup of the learning rate η_t, FTRL has a regret rate $O(\sqrt{T})$ as well [Shalev-Shwartz et al., 2012].

The regret rate can be improved to $O(\log T)$ if the loss functions are strongly convex as well [Hazan et al., 2007].

When working with distributions, i.e., $\mathcal{X} \triangleq \Delta(K)$ where $K \in \mathbb{N}^+$, and each $x \in \mathcal{X}$ is a distribution such that $x[i] \geq 0$ and $\sum_{i=1}^{K} x[i] = 1$, we will use another no-regret online learning algorithm, called Exponential Gradient Descent (or Weighted Majority [Littlestone and Warmuth, 1994]). Consider the loss function $\ell_t(x) \triangleq \mathbb{E}_{i\sim x}[c_t(i)]$, where $c : [K] \to [0,1]$, Exponential Gradient Descent works as follows:

$$x_{t+1}[i] \propto x_t[i] \exp\left(-\eta_t c_t[i]\right), \forall i \in [K].$$

One can show that exponential gradient descent has regret rate $O(\sqrt{(\ln K)T})$. Note that the $\ln(K)$ dependency here allows exponential gradient descent to scale to high-dimension setting. If one uses OGD here, then one will get $\sqrt{K}$ rather than $\ln(K)$ in the final regret rate.

1.3 Organization

The remainder of the thesis is organized as follows.

Part I: Imitation Learning consists of two chapters, where Chapter 2 focuses on interactive imitation learning with access to reward signal setting, and Chapter 3 proposes the setting of imitation learning from observations alone and proposes the algorithm FAIL—-the first algorithm that achieves provable sample efficiency for this setting.

[1]A function $f : \mathcal{X} \to \mathbb{R}$ is α-strongly convex if and only if $f(x) \geq f(x') + \nabla f(x')^\top (x - x') + \frac{\alpha}{2}\|x - x'\|_2^2$ for any pair $x, x' \in \mathcal{X}$.

Part II: Model-Free Reinforcement Learning consists of two chapters. In Chapter 4, we study online policy evaluation, where the underlying stochastic process could be non-Markovian or even adversarially. We propose reduction framework that reduces online policy evaluation to no-regret and stable online learning. In Chapter 5, we study two classic simple exploration strategies: exploration in action space and exploration in parameter space, both of which are commonly used in practice.

Part III: model-Based Reinforcement Leraning has two chapters. In the first part (Chapter 6) we study model-based RL versus Model-Free RL under the general framework called Contextual Decision Processes—a larger sequential decision making frameworks that captures models such as MDPs, POMDPs and PSRs. Besides the exponential separation results between Model-Based RL and Model-Free RL, we design a new model-based RL algorithm that can achieves sample efficiency for a large family of MDPs. In Chapter 7, we propose Dual Policy Iteration, a framework that integrates model-based optimal control, model learning, and imitation learning together. We analyze the convergence property of DPI and its empirical efficacy on a set of simulated robotics continuous control tasks.

Part IV summarizes the contributions of this thesis and discusses open problems and future directions.

Part V contains the detailed proofs and experiments setup.

1.4 Bibliographical Remarks

The thesis only contains works for which this author was one of the primary contributors.

Chapter 2 is based on the joint work with Arun Venkatraman, Byron Boots, Geoff Gordon and Drew Bangell [Sun et al., 2017c]. Chapter 3 is based on the ongoing work with Byron Boots and Drew Bagnell. Chapter 4 is based on the joint work with Drew Bagnell [Sun and Bagnell, 2015]. Chapter 5 is based on the joint work with Anirudh Vemula and Drew Bagnell [Vemula et al., 2019]. Chapter 6 is based on the joint work with Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford [Sun et al., 2018d]. Chapter 7 is based on the joint work with Geoff Gordon, Byron Boots and Drew Bagnell [Sun et al., 2018c].

1.5 Excluded Research

Except the research works presented in this thesis, I excluded a large portion of my Ph.D work for the purpose of keeping this thesis succinct. Below we list the excluded work.

1. System Identification: Predictive State Inference Machines [Sun et al., 2016c] and its extensions to online filtering, smoothing, and control [Hefny et al., 2018, Sun et al., 2016a, Venkatraman et al., 2016a,b, 2017].

2. Safety-Aware sequential decision making [Sun et al., 2017a,b].

3. Gradient boosting on stochastic data streams [Hu et al., 2017].

4. Sketching for Kronecker product regression [Diao et al., 2017].
5. Learnable memory data structures [Sun et al., 2018b].
6. Temporal Difference learning [Sun and Bagnell, 2016].
7. Efficient model-free Reinforcement Learning in metric spaces [Song and Sun, 2019].

Part I

Imitation Learning

Chapter 2

Interactive and Differentiable Imitation Learning for Sequential Prediction

Recently, researchers have demonstrated state-of-the-art performance on sequential prediction problems using deep neural networks and Reinforcement Learning (RL). For some of these problems, oracles that can demonstrate good performance may be available during training, but are not used by plain RL methods. To take advantage of this extra information, in this chapter, we propose *AggreVaTeD*, an extension of the Imitation Learning (IL) approach of Ross and Bagnell [2014]. AggreVaTeD allows us to use expressive differentiable policy representations such as deep networks, while leveraging training-time oracles to achieve faster and more accurate solutions with less training data. Specifically, we present two gradient procedures that can learn neural network policies for several problems, including a sequential prediction task and several high-dimensional robotics control problems. We also provide a comprehensive theoretical study of IL that demonstrates that we can expect up to exponentially-lower sample complexity for learning with AggreVaTeD than with plain RL algorithms. Our results and theory indicate that IL (and AggreVaTeD in particular) can be a more effective strategy for sequential prediction than plain RL.

The contributions in this chapter were first presented in the joint works with Arun Venkatraman, Byron Boots, Geoff Gordon, and Drew Bagnell [Sun et al., 2017c, 2018a].

2.1 Introduction

Although conventional supervised learning of deep models has been pivotal in advancing performance in sequential prediction problems, researchers are beginning to utilize Reinforcement Learning (RL) methods to achieve even higher performance [Bahdanau et al., 2016, Li et al., 2016, Ranzato et al., 2015]. In sequential prediction tasks, future predictions often depend on the history of previous predictions; thus, a poor prediction early in the sequence can lead to high loss (cost) for future predictions. Viewing the predictor as a *policy* π, deep RL algorithms are able to reason about the future accumulated cost in sequential prediction problems. These approaches have dramatically advanced the state-of-the-art on a number of problems including high-dimensional robotics control tasks and video and board games [Schulman et al., 2015a, Silver et al., 2016].

In contrast with general reinforcement learning methods, *imitation learning* and related sequential prediction algorithms such as SEARN [Daumé III et al., 2009], DaD [Venkatraman et al., 2015], AggreVaTe [Ross and Bagnell, 2014], PSIM [Sun et al., 2016c], and LOLS [Chang et al., 2015b] reduce the sequential prediction problems to supervised learning by leveraging a (near) optimal *cost-to-go oracle*[1] that can be queried for the next (near)-best prediction at any point during training. Specifically, these methods assume access to an oracle that provides an optimal or near-optimal action and the future accumulated loss Q^*, the so-called cost-to-go. For robotics control problems, this oracle may be a human expert guiding the robot during the training phase [Abbeel and Ng, 2004] or the policy from an optimal MDP solver [Choudhury et al., 2017b, Ross et al., 2011] that is either too slow to use at test time or leverages information unavailable at test time. For sequential prediction problems, an oracle can be constructed by optimization (e.g., beam search) or by a clairvoyant greedy algorithm [Chang et al., 2015a, Daumé III et al., 2009, Rhinehart et al., 2015, Ross et al., 2013] that, given the training data's ground truth, is near-optimal on the task-specific performance metric (e.g., cumulative reward, IoU, Unlabeled Attachment Score, BLEU).

We stress that the oracle is only required to be available during training. Therefore, the goal of IL is to learn a policy $\hat{\pi}$ with the help of the oracle (π^*, Q^*) during the training session, such that $\hat{\pi}$ achieves similar or better performance at test time when the oracle is unavailable. In contrast to IL, reinforcement learning methods often initialize with a random policy π_0 or cost-to-go estimate Q_0 that may be far from optimal. The optimal policy (or cost-to-go) must be found by exploring, often with random actions.

A classic family of IL methods is to collect data from running the demonstrator or oracle and train a regressor or classifier via supervised learning. These methods [Abbeel and Ng, 2004, Ratliff et al., 2006, Syed et al., 2008, Ziebart et al., 2008] learn either a policy $\hat{\pi}^*$ or $\hat{Q}^*$ from a fixed-size dataset pre-collected from the oracle. Unfortunately, these methods exhibit a pernicious problem: they require the training and test data to be sampled from the same distribution, despite the fact they explicitly change the sample policy during training. As a result, policies learned by these methods can fail spectacularly [Ross and Bagnell, 2010b]. *Interactive* approaches to IL such as SEARN [Daumé III et al., 2009], DAgger [Ross et al., 2011], and AggreVaTe [Ross and Bagnell, 2014] interleave learning and testing to overcome the data mismatch issue and, as a result, work well in practical applications. Furthermore, these interactive approaches can provide strong theoretical guarantees between training time loss and test time performance through a reduction to no-regret online learning.

[1]Expert, demonstrator, and oracle are used interchangeably.

In this work, we introduce *AggreVaTeD*, a *differentiable* version of AggreVaTe (Aggregate Values to Imitate [Ross and Bagnell, 2014]) which allows us to train policies with efficient gradient update procedures. AggreVaTeD extends and scales interactive IL for use in sequential prediction and challenging continuous robot control tasks. We provide two gradient update procedures: a regular gradient update developed from Online Gradient Descent (OGD) [Zinkevich, 2003] and a natural gradient update [Bagnell and Schneider, 2003, Kakade, 2002], which is closely related to Weighted Majority (WM) [Littlestone and Warmuth, 1994], a popular no-regret algorithm that enjoys an almost dimension-free property [Bubeck et al., 2015].[2]

AggreVaTeD leverages the oracle to learn rich polices that can be represented by complicated non-linear function approximators. Our experiments with deep neural networks on various robotics control simulators and on a dependency parsing sequential prediction task show that AggreVaTeD can achieve expert-level performance and even *super-expert* performance when the oracle is sub-optimal, a result rarely achieved by non-interactive IL approaches. The differentiable nature of AggreVaTeD additionally allows us to employ Recurrent Neural Network policies, e.g., Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997], to handle partially observable settings (e.g., observe only partial robot state). Empirical results demonstrate that by leveraging an oracle, IL can learn much faster than RL.

In addition to providing a set of practical algorithms, we develop a comprehensive theoretical study of IL on discrete MDPs. We construct an MDP that demonstrates exponentially better sample efficiency for IL than any RL algorithm. For general discrete MDPs, we provide a regret upper bound for AggreVaTeD with WM, which shows IL can learn dramatically faster than RL. We provide a regret lower bound for any IL algorithm, which demonstrates that AggreVaTeD with WM is near-optimal.

To summarize the contributions of this chapter: (1) AggreVaTeD allows us to handle continuous action spaces and employ recurrent neural network policies for Partially Observable Markov Decision Processes (POMDPs); (2) understanding IL from a perspective that is related to policy gradient allows us to leverage advances from the well-studied RL policy gradient literature (e.g., gradient variance reduction techniques, efficient natural gradient computation); (3) we provide a new sample complexity study of IL and compare to RL, showing that we can expect up to exponentially lower sample complexity. Our experimental and theoretical results support the proposition:

> Imitation Learning is a more effective strategy than Reinforcement Learning for sequential prediction with near-optimal cost-to-go oracles.

2.2 Preliminaries

A Markov Decision Process consists of a set of states, actions (that come from a policy), cost (loss), and a model that transitions states given actions. Interestingly, most sequential prediction problems can be framed in terms of MDPs Daumé III et al. [2009]. The actions are the learner's (e.g., RNN's) predictions. The state is then the result of all the predictions made so far (e.g., the dependency tree constructed so far or the words translated so far). The cumulative cost is the performance metric such as (negative) UAS, received at the end

[2]i.e., the regret bound depends on poly-log of the dimension of parameter space.

(horizon) or after the final prediction. For robotics control problems, the robot's configuration is the state, the controls (e.g., joint torques) are the actions, and the cost is related to achieving a task (e.g., distance walked).

Formally, a finite-horizon Markov Decision Process (MDP) is defined as $(\mathcal{S}, \mathcal{A}, P, C, \rho_0, H)$. Here, $\mathcal{S}$ is a set of S states and $\mathcal{A}$ is a set of A actions; at time step t, P_t is the transition dynamics such that for any $s_t \in \mathcal{S}, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$, $P_t(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} from state s_t by taking action a_t at step t; C is the cost distribution such that a cost c_t at step t is sampled from $C_t(\cdot|s_t, a_t)$. Finally, we denote $\bar{c}_t(s_t, a_t)$ as the expected cost, ρ_0 as the initial distribution of states, and $H \in \mathbb{N}^+$ as the finite horizon (max length) of the MDP.

We define a stochastic policy π such that for any state $s \in \mathcal{S}$, $\pi(\cdot|s) \in \Delta(A)$, where $\Delta(A)$ is a A-dimensional simplex, conditioned on state s. $\pi(a|s) \in [0, 1]$ outputs the probability of taking action a at state s. The distribution of trajectories $\tau = (s_1, a_1, \ldots, a_{H-1}, s_H)$ is determined by π and the MDP, and is defined as

$$\rho_\pi(\tau) = \rho_0(s_1) \prod_{t=2}^{H} \pi(a_{t-1}|s_{t-1}) P_{t-1}(s_t|s_{t-1}, a_{t-1}).$$

The distribution of the states at time step t, induced by running the policy π until t, is defined $\forall s_t$:

$$d_t^\pi(s_t) = \sum_{\{s_i, a_i\}_{i \le t-1}} \rho_0(s_1) \prod_{i=1}^{t-1} \pi(a_i|s_i) P_i(s_{i+1}|s_i, a_i).$$

Note that the summation above can be replaced by an integral if the state or action space is continuous. The average state distribution $\bar{d}^\pi(s) = \sum_{t=1}^{H} d_t^\pi(s)/H$.

The expected average cost of a policy π can be defined with respect to ρ_π or $\{d_t^\pi\}$:

$$\mu(\pi) = \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=1}^{H} \bar{c}_t(s_t, a_t) \right] = \sum_{t=1}^{H} \mathop{\mathbb{E}}_{s \sim d_t^\pi(s), a \sim \pi(a|s)} \left[\bar{c}_t(s, a) \right].$$

We define the state-action value $Q_t^\pi(s, a)$ (i.e., cost-to-go) for policy π at time step t as:

$$Q_t^\pi(s_t, a_t) = \bar{c}_t(s_t, a_t) + \mathop{\mathbb{E}}_{s \sim P_t(\cdot|s_t, a_t), a \sim \pi(\cdot|s)} Q_{t+1}^\pi(s, a),$$

where the expectation is taken over the randomness of the policy π and the MDP.

We define π^* as the expert policy (e.g., human demonstrators, search algorithms equipped with ground-truth) and $Q_t^*(s, a)$ as the expert's cost-to-go oracle. We emphasize that π^* may not be optimal, i.e., $\pi^* \notin \arg\min_\pi \mu(\pi)$. Throughout the paper, we assume $Q_t^*(s, a)$ is known or can be estimated without bias (e.g., by rolling out π^*: starting from state s, applying action a, and then following π^* for $H - t$ steps).

When π is represented by a function approximator, we use the notation π_θ to represent the policy parametrized by $\theta \in \mathbb{R}^d$: $\pi(\cdot|s; \theta)$. In this work we specifically consider optimizing policies in which the parameter dimension d may be large. We also consider the partially observable setting in our experiments, where the policy $\pi(\cdot|o_1, a_1, ..., o_t; \theta)$ is defined over the whole history of observations and actions (o_t is generated from the hidden state s_t). We use both LSTM and Gated Recurrent Unit (GRU) Chung et al. [2014] based policies where the RNN's hidden states provide a compressed feature of the history. To our best knowledge, this is the first time RNNs are employed in an IL framework to handle partially observable environments.

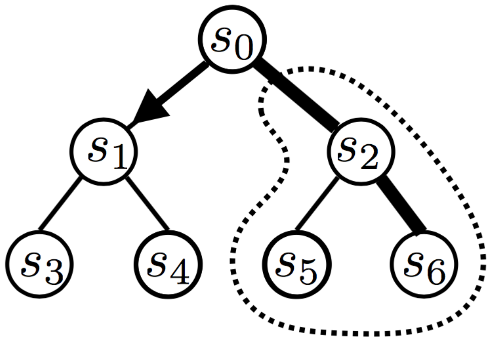


Figure 2.1: The binary tree structure MDP $\tilde{\mathcal{M}}$.

2.3 Why Imitation Learning: Regret Analysis for IL with Perfect Oracle

How much faster can IL learn a good policy than RL? In this section we quantify the gap on discrete MDPs when IL can (1) query for an optimal Q^* or (2) query for a noisy but unbiased estimate of Q^*. To measure the speed of learning, we look at the *cumulative* regret of the entire learning process, defined as $R_N = \sum_{n=1}^{N}(J(\pi_n) - J(\pi^*))$. A smaller regret rate indicates faster learning. Throughout this section, we assume the expert π^* is optimal. We consider finite-horizon, episodic IL and RL algorithms.

2.3.1 Exponential Gap

We consider an MDP $\mathcal{M}$ shown in Fig. 2.1 which is a depth-K binary tree-structure with $S = 2^K - 1$ states and two actions a_l, a_r: go-left and go-right. The transition is deterministic and the initial state s_0 (root) is fixed. The cost for each non-leaf state is zero; the cost for each leaf is i.i.d sampled from a given distribution (possibly different distributions per leaf). Below we show that for $\mathcal{M}$, IL can be exponentially more sample efficient than RL.

Theorem 5. *For $\mathcal{M}$, the regret R_N of any finite-horizon, episodic RL algorithm is at least:*

$$\mathbb{E}[R_N] \geq \Omega(\sqrt{SN}). \tag{2.1}$$

The expectation is with respect to random generation of cost and internal randomness of the algorithm. However, for the same MDP $\mathcal{M}$, with the access to Q^*, we show IL can learn exponentially faster:

Theorem 6. *For the MDP $\mathcal{M}$, AggreVaTe [Ross and Bagnell, 2014] with FTL can achieve the following regret bound:*

$$R_N \leq O(\ln(S)). \tag{2.2}$$

Fig. 2.1 illustrates the intuition behind the theorem. Assume during the first episode, the initial policy π_1 picks the rightmost trajectory (bold black) to explore. We query from the cost-to-go oracle Q^* at s_0 for a_l and a_r, and learn that $Q^*(s_0, a_l) < Q^*(s_0, a_r)$. This immediately tells us that the optimal policy will go left (black arrow) at s_0. Hence the algorithm does not have to explore the right sub-tree (dotted circle).

Next we consider a more difficult setting where one can only query for a noisy but unbiased estimate of Q^* (e.g., by rolling out π^* finite number of times). The above halving argument will not apply since deterministically eliminating nodes based on noisy estimates might permanently remove good trajectories. However, IL can still achieve a poly-log regret with respect to S, even in the noisy setting:

Theorem 7. *With only access to unbiased estimate of Q^*, for the MDP $\mathcal{M}$, AggreVaTe [Ross and Bagnell, 2014] with WM can achieve the following regret with probability at least $1-\delta$:*

$$R_N \leq O\Big(\ln(S)(\sqrt{\ln(S)N} + \sqrt{\ln(2/\delta)N})\Big). \tag{2.3}$$

The detailed proofs of the above three theorems can be found in Appendix 9.1.5, 9.1.6, 9.1.7. In summary, for MDP $\mathcal{M}$, IL is exponentially faster than RL.

2.3.2 Polynomial Gap and Near-Optimality

We next quantify the gap in general discrete MDPs and also show that AggreVaTeD is near-optimal. We consider the harder case where we can only access an unbiased estimate of Q_t^*, for any t and state-action pair. The policy π is represented as a set of probability vectors $\pi^{s,t} \in \Delta(A)$, for all $s \in \mathcal{S}$ and $t \in [H]$: $\pi = \{\pi^{s,t}\}_{s\in\mathcal{S},t\in[H]}$.

Theorem 8. *With access to unbiased estimates of Q_t^*, AggreVaTeD with WM achieves the regret upper bound:*

$$R_N \leq O\big(HQ^e_{\max}\sqrt{S\ln(A)N}\big). \tag{2.4}$$

Here $Q^e_{\max}$ is the maximum cost-to-go of the expert.[3] The total regret shown in Eq. 2.4 allows us to compare IL algorithms to RL algorithms. For example, the Upper Confidence Bound (UCB) based, near-optimal optimistic RL algorithms from [Jaksch et al., 2010], specifically designed for efficient exploration, admit regret $\tilde{O}(HS\sqrt{HAN})$, leading to a gap of approximately $\sqrt{HAS}$ compared to the regret bound of imitation learning shown in Eq. 2.4.

We also provide a lower bound on R_N for the $H=1$ case which shows the dependencies on N, A, S are tight:

Theorem 9. *There exists an MDP (H=1) such that, with only access to unbiased estimates of Q^*, any finite-horizon episodic imitation learning algorithm must have:*

$$\mathbb{E}[R_N] \geq \Omega(\sqrt{S\ln(A)N}). \tag{2.5}$$

The proofs of the above two theorems regarding general MDPs can be found in Appendix 9.1.8 and 9.1.9. In summary for discrete MDPs, one can expect at least a polynomial gap and a possible exponential gap between IL and RL.

[3]Here we assume $Q^e_{\max}$ is a constant compared to H. If $Q^e_{\max} = \Theta(H)$, then the expert is no better than a random policy of which the cost-to-go is around $\Theta(H)$.

2.4 Differentiable Imitation Learning

Policy based imitation learning aims to learn a policy $\hat{\pi}$ that approaches the performance of the expert π^* at test time when π^* is no longer available. In order to learn rich policies such as LSTMs or deep networks [Schulman et al., 2015a], we derive a method related to *policy gradients* for imitation learning and sequential prediction. To do this, we leverage the reduction of IL and sequential prediction to online learning as shown in [Ross and Bagnell, 2014] to learn policies represented by expressive differentiable function approximators.

The fundamental idea in Ross and Bagnell [2014] is to use a no-regret online learner to update policies using the following loss function at each episode n:

$$\ell_n(\pi) = \frac{1}{H}\sum_{t=1}^{H} \mathbb{E}_{s_t\sim d_t^{\pi_n}}\Big[\mathbb{E}_{a\sim\pi(\cdot|s_t)}[Q_t^*(s_t,a)]\Big]. \tag{2.6}$$

The loss function intuitively encourages the learner to find a policy that minimize the expert's cost-to-go *under the state distribution resulting from the current learned policy* π_n. Specifically, Ross and Bagnell [2014] suggest an algorithm named *AggreVaTe* (Aggregate Values to Imitate) that uses Follow-the-Leader (FTL) [Shalev-Shwartz et al., 2012] to update policies: $\pi_{n+1} = \arg\min_{\pi\in\Pi}\sum_{i=1}^{n}\ell_n(\pi)$, where Π is a pre-defined convex policy set. When $\ell_n(\pi)$ is strongly convex with respect to π and $\pi^* \in \Pi$, after N iterations AggreVaTe with FTL can find a policy $\hat{\pi}$ with:

$$\mu(\hat{\pi}) \leq \mu(\pi^*) - \epsilon_N + O(\ln(N)/N), \tag{2.7}$$

where $\epsilon_N = [\sum_{n=1}^{N}\ell_n(\pi^*) - \min_\pi\sum_{n=1}^{N}\ell_n(\pi)]/N$. Note that $\epsilon_N \geq 0$ and the above inequality indicates that $\hat{\pi}$ can outperform π^* when π^* is not (locally) optimal (i.e., $\epsilon_n > 0$). Our experimental results support this observation.

A simple implementation of AggreVaTe that aggregates the values (as the name suggests) will require an exact solution to a batch optimization procedure in each episode. When π is represented by large, non-linear function approximators, the $\arg\min$ procedure generally takes more and more computation time as n increases. Hence an efficient incremental update procedure is necessary for the method to scale.

To derive an incremental update procedure, we can take one of two routes. The first route, suggested already by [Ross and Bagnell, 2014], is to update our policy with an incremental no-regret algorithm such as weighted majority [Littlestone and Warmuth, 1994], instead of with a batch algorithm like FTRL. Unfortunately, for rich policy classes such as deep networks, no-regret learning algorithms may not be available (e.g., a deep network policy is non-convex with respect to its parameters). So instead we propose a novel second route: we directly differentiate Eq. 2.6, yielding an update related to policy gradient methods. We work out the details below, including a novel update rule for IL based on natural gradients.

Interestingly, the two routes described above yield almost identical algorithms if our policy class is simple enough: e.g., for a tabular policy, AggreVaTe with weighted majority yields the natural gradient version of AggreVaTeD described below. And, the two routes yield complementary theoretical guarantees: the first route yields a regret bound for simple-enough policy classes, while the second route yields convergence to a local optimum for extremely flexible policy classes.

2.4.1 Online Gradient Descent

For discrete actions, the gradient of $\ell_n(\pi_\theta)$ (Eq. 2.6) with respect to the parameters θ of the policy is

$$\nabla_\theta \ell_n(\theta) = \frac{1}{H}\sum_{t=1}^{H} \mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}}} \sum_a \nabla_\theta \pi(a|s_t;\theta) Q_t^*(s_t,a). \tag{2.8}$$

For continuous action spaces, we cannot simply replace the summation by integration since in practice it is hard to evaluate $Q_t^*(s,a)$ for infinitely many a, so, instead, we use importance weighting to re-formulate ℓ_n (Eq. 2.6) as

$$\begin{aligned}\ell_n(\pi_\theta) &= \frac{1}{H}\sum_{t=1}^{H} \mathop{\mathbb{E}}_{s\sim d_t^{\pi_{\theta_n}}, a\sim\pi(\cdot|s;\theta_n)} \frac{\pi(a|s;\theta)}{\pi(a|s;\theta_n)} Q_t^*(s,a) \\ &= \frac{1}{H}\mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_{\theta_n}}} \sum_{t=1}^{H} \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)} Q_t^*(s_t,a_t). \end{aligned}\tag{2.9}$$

With this reformulation, the gradient with respect to θ is

$$\begin{aligned}\nabla_\theta \ell_n(\theta) &= \frac{1}{H}\mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_{\theta_n}}} \sum_{t=1}^{H} \frac{\nabla_\theta\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)} Q_t^*(s_t,a_t) \\ &= \frac{1}{H}\mathbb{E}_{\tau\sim\rho_{\pi_{\theta_n}}} \sum_{t=1}^{H} \nabla_\theta \ln(\pi(a_t|s_t;\theta_n)) Q_t^*(s_t,a_t). \end{aligned}\tag{2.10}$$

The above gradient computation enables a very efficient update procedure with online gradient descent: $\theta_{n+1} = \theta_n - \eta_n \nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$, where η_n is the learning rate.

2.4.2 Policy Updates with Natural Gradient Descent

We derive a natural gradient update procedure for imitation learning inspired by the success of natural gradient descent in RL [Bagnell and Schneider, 2003, Kakade, 2002, Schulman et al., 2015a]. Following [Bagnell and Schneider, 2003], we define the Fisher information matrix $I(\theta_n)$ using trajectory likelihood:

$$I(\theta_n) = \frac{1}{H^2}\mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_{\theta_n}}} \nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))^T, \tag{2.11}$$

where $\nabla_\theta \log(\rho_{\pi_\tau}(\tau))$ is the gradient of the log likelihood of the trajectory τ which can be computed as $\sum_{t=1}^H \nabla_\theta \log(\pi_\theta(a_t|s_t))$. Note that this representation is equivalent to the original Fisher information matrix proposed by [Kakade, 2002]. Now, we can use Fisher information matrix together with the IL gradient derived in the previous section (Eq. 2.102.8) to compute the natural gradient as $I(\theta_n)^{-1}\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$, which yields a natural gradient update: $\theta_{n+1} = \theta_n - \mu_n I(\theta_n)^{-1}\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$.

Interesting, as we mentioned before, when the given MDP is discrete and the policy class is in a tabular representation, AggreVaTe with Weighted Majority [Littlestone and Warmuth, 1994] yields an extremely similar update procedure as AggreVaTeD with natural gradient. As Weighted Majority can speed up online learning (i.e., almost dimension free [Bubeck et al., 2015]) and AggreVaTe with Weighted Majority enjoys strong theoretical guarantees on the performance of the learned policy [Ross and Bagnell, 2014], this similarity provides an intuitive explanation why we can expect AggreVaTeD with natural gradient to speed up IL and learn a high quality policy.

2.5 Sample-based Practical Algorithms

In the previous section, we derived a regular gradient update procedure and a natural gradient update procedure for IL. Note that all of the computations of gradients and Fisher information matrices assumed it was possible to exactly compute expectations including $\mathbb{E}_{s\sim d^{\pi}}$ and $\mathbb{E}_{a\sim\pi(a|s)}$. In this section, we provide practical algorithms where we approximate the gradients and Fisher information matrices using finite samples collected during policy execution.

2.5.1 Gradient Estimation and Variance Reduction

We consider an episodic framework where given a policy π_n at episode n, we roll out π_n K times to collect K trajectories $\{\tau_i^n\}$, for $i\in[K]$, $\tau_i^n=\{s_1^{i,n},a_1^{i,n},...\}$. For gradient $\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$ we can compute an unbiased estimate using $\{\tau_i^n\}_{i\in[K]}$:

$$\tilde{\nabla}_{\theta_n}=\frac{1}{HK}\sum_{i=1}^{K}\sum_{t=1}^{H}\sum_{a}\nabla_{\theta_n}\pi_{\theta_n}(a|s_t^{i,n})Q_t^*(s_t^{i,n},a), \tag{2.12}$$

$$\tilde{\nabla}_{\theta_n}=\frac{1}{HK}\sum_{i=1}^{K}\sum_{t=1}^{H}\nabla_{\theta_n}\ln(\pi_{\theta_n}(a_t^{i,n}|s_t^{i,n}))Q_t^*(s_t^{i,n},a_t^{i,n}). \tag{2.13}$$

for discrete and continuous setting respectively.

When we can compute $V_t^*(s)$, we can replace $Q_t^*(s_t^{i,n},a)$ by the state-action advantage function $A_t^*(s_t^{i,n},a)=Q_t^*(s_t^{i,n},a)-V_t^*(s_t^{i,n})$, which leads to the following unbiased and variance-reduced gradient estimation for continuous action setting [Greensmith et al., 2004]:

$$\tilde{\nabla}_{\theta_n}=\frac{1}{HK}\sum_{i=1}^{K}\sum_{t=1}^{H}\nabla_{\theta_n}\ln(\pi_{\theta_n}(a_t^{i,n}|s_t^{i,n}))A_t^*(s_t^{i,n},a_t^{i,n}), \tag{2.14}$$

In fact, we can use any baselines to reduce the variance by replacing $Q_t^*(s_t,a_t)$ by $Q_t^*(s_t,a_t)-b(s_t)$, where $b_(s_t):\mathcal{S}\to\mathbb{R}$ is a action-independent function. Ideally $b(s_t)$ should be some function approximator that approximates $V^*(s_t)$. In our experiments, we test linear function approximator $b(s)=w^Ts$, which is online learned using π^*'s roll-out data.

The Fisher information matrix (Eq. 9.4) is approximated as:

$$\begin{aligned}\tilde{I}(\theta_n)&=\frac{1}{H^2K}\sum_{i=1}^{K}\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))^T\\&=S_nS_n^T,\end{aligned} \tag{2.15}$$

where, for notation simplicity, we denote S_n as a $d\times K$ matrix where the i's th column is $\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))/(H\sqrt{K})$. Namely the Fisher information matrix is represented by a sum of

Algorithm 1 AggreVaTeD (Differentiable AggreVaTe)

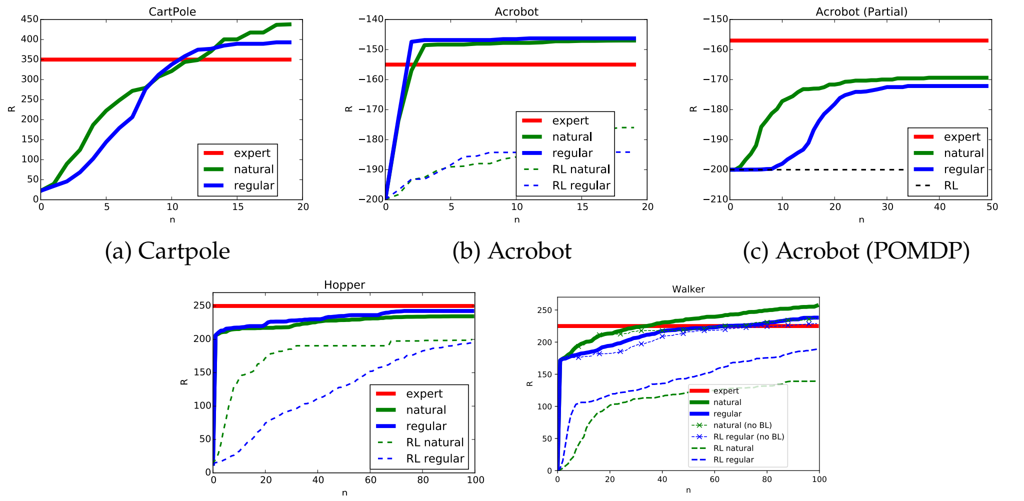
```
1: Input: The given MDP and expert $\pi^*$. Learning rate $\{\eta_n\}$. Schedule rate $\{\alpha_i\}$, $\alpha_n \to 0, n \to \infty$.
2: Initialize policy $\pi_{\theta_1}$ (either random or supervised learning).
3: for n = 1 to N do
4:     Mixing policies: $\hat{\pi}_n = \alpha_n \pi^* + (1-\alpha_n)\pi_{\theta_n}$.
5:     Starting from $\rho_0$, roll out by executing $\hat{\pi}_n$ on the given MDP to generate $K$ trajectories $\{\tau_i^n\}$.
6:     Using $Q^*$ and $\{\tau_i^n\}_i$, compute the descent direction $\delta_{\theta_n}$ (Eq. 2.12, Eq. 2.13, Eq. 2.14, or CG).
7:     Update: $\theta_{n+1} = \theta_n - \eta_n \delta_{\theta_n}$.
8: end for
9: Return: the best hypothesis $\hat{\pi} \in \{\pi_n\}_n$ on validation.
```

K rank-one matrices. For large policies represented by neural networks, $K \ll d$, and hence $\tilde{I}(\theta_n)$ a low rank matrix. One can find the descent direction δ_{θ_n} by solving the linear system $S_n S_n^T \delta_{\theta_n} = \tilde{\nabla}_{\theta_n}$ for δ_{θ_n} using Conjugate Gradient (CG) with a fixed number of iterations, which is equivalent to solving the above linear systems using Partial Least Squares [Phatak and de Hoog, 2002]. This approach is used in TRPO [Schulman et al., 2015a]. The difference is that our representation of the Fisher matrix is in the form of $S_n S_n^T$ and in CG we never need to explicitly compute or store $S_n S_n^T$ which requires d^2 space and time. Instead, we only compute and store S_n ($O(Kd)$) and the total computational time is still $O(K^2 d)$. The learning-rate for natural gradient descent can be chosen as $\eta_n = \sqrt{\delta_{KL}/(\tilde{\nabla}_{\theta_n}^T \delta_{\theta_n})}$, such that $KL(\rho_{\pi_{\theta_{n+1}}(\tau)} \| \rho_{\pi_{\theta_n}(\tau)}) \approx \delta_{KL} \in \mathbb{R}^+$

2.5.2 Differentiable Imitation Learning: AggreVaTeD

Summarizing the above discussion, we present the differentiable imitation learning framework *AggreVaTeD*, in Alg. 1. At every iteration n, the roll out policy $\hat{\pi}_n$ is a mix of the expert policy π^* and the current policy π_{θ_n}, with mixing rate α ($\alpha_n \to 0, n \to \infty$): at every step, with probability α $\hat{\pi}_n$ picks π^* and picks π_{θ_n} otherwise. This mixing strategy with the decay rate was first introduced in [Ross et al., 2011] for IL, and later on was used in sequence prediction [Bengio et al., 2015]. In Line 6, one can either choose Eq. 2.13 or the corresponding variance reduced estimation Eq. 2.14 to perform regular gradient descent, and choose CG to perform natural gradient descent. AggreVaTeD is extremely simple: we do not need to perform any data aggregation (i.e., we do not need to store all $\{\tau_i\}_i$ from all previous iterations); the computational complexity of each policy update scales in $O(d)$.

When we use non-linear function approximators to represent the polices, the analysis of AggreVaTe from [Ross and Bagnell, 2014] will not hold, since the loss function $\ell_n(\theta)$ is not convex with respect to parameters θ. Nevertheless, as we will show in experiments, in practice AggreVaTeD is still able to learn a policy that is competitive with, and sometimes superior to, the oracle's performance.



(a) Cartpole (b) Acrobot (c) Acrobot (POMDP)

(d) Hopper (e) Walker

Figure 2.2: Performance (cumulative reward R on y-axis) versus number of episodes (n on x-axis) of AggreVaTeD (blue and green), experts (red), and RL algorithms (dotted) on different robotics simulators.

2.6 Experiment

We evaluate our algorithms on robotics simulations from OpenAI Gym [Brockman et al., 2016a] and on Handwritten Algebra Dependency Parsing [Duyck and Gordon, 2015]. We report reward instead of cost, since OpenAI Gym by default uses reward and dependency parsing aims to maximize UAS score. As our approach only promises there exists a policy among all of the learned polices that can perform as well as the expert, we report the performance of the best policy so far: $\max\{\mu(\pi_1), ..., \mu(\pi_i)\}$. For regular gradient descent, we use ADAM [Kingma and Ba, 2014] which is a first-order no-regret algorithm, and for natural gradient, we use CG to compute the descent direction. For RL we use REINFORCE [Williams, 1992] and Truncated Natural Policy Gradient (TNPG) [Duan et al., 2016].

2.6.1 Robotics Simulations

We consider CartPole Balancing, Acrobot Swing-up, Hopper and Walker. For generating an expert, similar to previous work [Ho and Ermon, 2016], we used a Deep Q-Network (DQN) to generate Q^* for CartPole and Acrobot (e.g., to simulate the settings where Q^* is available), while using the publicly available TRPO implementation to generate π^* for Hopper and Walker to simulate the settings where one has to estimate Q^* by Monte-Carlo roll outs π^*.

Discrete Action Setting We use a one-layer (16 hidden units) neural network with ReLu activation functions to represent the policy π for the Cart-pole and Acrobot benchmarks. The value function Q^* is obtained from the DQN [Mnih et al., 2015] and represented by a multi-layer fully connected neural network. The policy π_{θ_1} is initialized with common ReLu

Arc-Eager	AggreVaTeD (LSTMs)	AggreVaTeD (NN)	SL-RL (LSTMs)	SL-RL(NN)	RL (LSTMs)	RL (NN)	DAgger	SL (LSTMs)	SL (NN)	Random
Regular	**0.924**$\pm$0.10	0.851$\pm$0.10	0.826$\pm$ 0.09	0.386$\pm$0.1	0.256$\pm$0.07	0.227$\pm$0.06	0.832$\pm$0.02	0.813$\pm$0.1	0.325$\pm$0.2	$\sim$0.150
Natural	0.915$\pm$0.10	0.800$\pm$0.10	0.824$\pm$0.10	0.345$\pm$0.1	0.237$\pm$0.07	0.241$\pm$0.07				

Table 2.1: Performance (UAS) of different approaches on handwritten algebra dependency parsing. *SL* stands for supervised learning using expert's samples: maximizing the likelihood of expert's actions under the sequences generated by expert itself. *SL-RL* means RL with initialization using SL. *Random* stands for the initial performances of random policies (LSTMs and NN). The performance of DAgger with Kernel SVM is from Duyck and Gordon [2015].

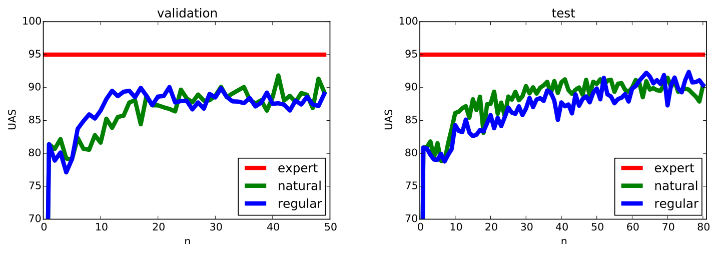
neural network initialization techniques. For the scheduling rate $\{\alpha_i\}$, we set all $\alpha_i = 0$: namely we did not roll-in using the expert's actions during training. We set the number of roll outs $K = 50$ and horizon $H = 500$ for CartPole and $H = 200$ for Acrobot.

Fig. 7.1b and 2.2b shows the performance averaged over 10 random trials of AggreVaTeD with regular gradient descent and natural gradient descent. Note that AggreVaTeD outperforms the experts' performance significantly: Natural gradient surpasses the expert by 5.8% in Acrobot and $\mathbf{25}\%$ in Cart-pole. Also, for Acrobot swing-up, at horizon $H = 200$, with high probability a randomly initialized neural network policy won't be able to collect any reward signals. Hence the improvement rates of REINFORCE and TNPG are slow. In fact, we observed that for a short horizon such as $H = 200$, REINFORCE and Truncated Natural Gradient often even fail to improve the policy at all (failed 6 times among 10 trials). On the contrary, AggreVaTeD does not suffer from the delayed reward signal issue, since the expert will collect reward signals much faster than a randomly initialized policy.

Fig. 2.2c shows the performance of AggreVaTeD with an LSTM policy (32 hidden states) in a partially observed setting where the expert has access to full states but the learner has access to partial observations (link positions). RL algorithms did not achieve any improvement while AggreVaTeD still achieved 92% of the expert's performance.

Continuous Action Setting We test our approaches on two robotics simulators with continuous actions: (1) the 2-d Walker and (2) the Hopper from the MuJoCo physics simulator. Following the neural network settings described in Schulman et al. [2015a], the expert policy π^* is obtained from TRPO with one hidden layer (64 hidden states), which is the same structure that we use to represent our policies π_θ. We set $K = 50$ and $H = 100$. We initialize π_{θ_1} by collecting K expert demonstrations and then maximize the likelihood of these demonstrations (i.e., supervised learning). We use a linear baseline $b(s) = w^T s$ for RL and IL.

Fig. 2.2e and 7.1f show the performance averaged over 5 random trials. Note that AggreVaTeD outperforms the expert in the Walker by 13.7% while achieving 97% of the expert's performance in the Hopper problem. After 100 iterations, we see that by leveraging the help from experts, AggreVaTeD can achieve much faster improvement rate than the corresponding RL algorithms (though eventually we can expect RL to catch up). In Walker, we also tested AggreVaTeD without linear baseline, which still outperforms the expert but performed slightly worse than AggreVaTeD with baseline as expected.



(a) Validation (b) Test

Figure 2.3: UAS (y-axis) versus number of iterations (n on x-axis) of AggreVaTeD with LSTM policy (blue and green), experts (red) on validation set and test set for Arc-Eager Parsing.

2.6.2 Dependency Parsing For Handwritten Linear Algebra

We consider a sequential prediction problem: transition-based dependency parsing for handwritten algebra with raw image data Duyck and Gordon [2015]. The parsing task for algebra is similar to the classic dependency parsing for natural language Chang et al. [2015a] where the problem is modelled in the IL setting and the state-of-the-art is achieved by AggreVaTe with FTRL (using Data Aggregation). The additional challenge here is that the inputs are handwritten algebra symbols in raw images. We directly learn to predict parse trees from low level image features (Histogram of Gradient features (HoG)). During training, the expert is constructed using the ground-truth dependencies in training data. The full state s during parsing consists of three data structures: Stack, Buffer and Arcs, which store raw images of the algebraic symbols. Since the sizes of stack, buffer and arcs change during parsing, a common approach is to featurize the state s by taking the features of the latest three symbols from stack, buffer and arcs (e.g., Chang et al. [2015a]). Hence the problem falls into the *partially observable* setting, where the feature o is extracted from state s and only contains partial information about s. The dataset consists of 400 sets of handwritten algebra equations. We use 80% for training, 10% for validation, and 10% for testing. Note that different from robotics simulators where at every episode one can get fresh data from the simulators, the dataset is fixed and sample efficiency is critical.

The RNN policy follows the design from Sutskever et al. [2014]. It consists of two LSTMs. Given a sequence of algebra symbols τ, the first LSTM processes one symbol at a time and at the end outputs its hidden states and memory (i.e., a summary of τ). The second LSTM initializes its own hidden states and memory using the outputs of the first LSTM. At every parsing step t, the second LSTM takes the current partial observation o_t (o_t consists of features of the most recent item from stack, buffer and arcs) as input, and uses its internal hidden state and memory to compute the action distribution $\pi(\cdot|o_1, ..., o_t, \tau)$ conditioned on history. We also tested reactive policies constructed as fully connected ReLu neural networks (NN) (one-layer with 1000 hidden states) that directly maps from observation o_t to action a, where o_t uses the most three recent items. We use variance reduced gradient estimations, which give better performance in practice. The performance is summarised in Table 2.1. Due to the partial observability of the problem, AggreVaTeD with a LSTM policy achieves significantly better UAS scores compared to the NN reactive policy and DAgger with a Kernelized SVM [Duyck and Gordon, 2015]. Also AggreVaTeD with a LSTM policy achieves 97% of optimal expert's performance. Fig. 2.3 shows the improvement rate of regular gradient and natural gradient on both validation set and test set. Overall we observe that both methods have similar performance. Natural gradient achieves a better UAS score in validation and converges slightly faster on the test set but also achieves a lower UAS score on test set.

2.7 Conclusion

In this chapter, we introduced AggreVaTeD, a differentiable imitation learning algorithm which trains neural network policies for sequential prediction tasks such as continuous robot control and dependency parsing on raw image data. We showed that in theory and in practice IL can learn much faster than RL with access to optimal cost-to-go oracles. The IL learned policies were able to achieve expert and sometimes super-expert levels of performance in both fully observable and partially observable settings. The theoretical and experimental results suggest that IL is significantly more effective than RL for sequential prediction with near optimal cost-to-go oracles.

For practical implementation, one can interpret AggreVaTeD as an actor-critic algorithm, where the actor is the policy that is being learned, and the critic is the expert's cost-to-go Q-function. Hence, existing efficient framewors of actor-critic RL methods can be used here by replacing the critic—the Q function of the current learned policy, by the expert's critic—the expert's Q-function.

Chapter 3

Imitation Learning from Observation Alone

In this chapter, we study Imitation Learning (IL) from Observations alone (ILfO) in large-scale MDPs. While most IL algorithms rely on an expert to directly provide actions to the learner, in this setting the expert only supplies sequences of observations. We design a new model-free algorithm for ILfO, *Forward Adversarial Imitation Learning* (Fail), which learns a sequence of time-dependent policies by minimizing an Integral Probability Metric between the observation distributions of the expert policy and the learner. Fail is the *first* probably efficient algorithm in ILfO setting, which learns a near-optimal policy with a number of samples that is polynomial in all relevant parameters but independent of the number of unique observations. The resulting theory extends the domain of provably sample efficient learning algorithms beyond existing results, which typically only consider tabular reinforcement learning settings or settings that require access to a near-optimal reset distribution. We also demonstrate the efficacy of Fail on multiple OpenAI Gym control tasks. Our implementation of Fail can be found in supplementary materials with scripts to reproduce all experimental results.

This is a joint work with Anirudh Vemula, Byron Boots, and Drew Bagnell [Sun et al., 2019].

3.1 Introduction

Imitation Learning (IL) is a sample efficient approach to policy optimization [Ross and Bagnell, 2014, Ross et al., 2011, Sun et al., 2017c] that has been extensively used in real applications, including Natural Language Processing [Chang et al., 2015a,b, Daumé III et al., 2009], game playing [Hester et al., 2017, Silver et al., 2016], and robotics control tasks [Pan et al., 2018]. Most previous IL work considers settings where an expert can directly provide action signals to the learner. In these settings, a general strategy is to directly learn a policy that maps from state to action, via supervised learning approaches (e.g., DAgger [Ross et al., 2011], AggreVaTe [Ross and Bagnell, 2014], Behaviour Cloning [Syed and Schapire, 2010]). Another popular strategy is to learn a policy by minimizing some divergence between the policy's state-action distribution and the expert's state-action distribution. Popular divergences include Forward KL (i.e., Behaviour Cloning), Jensen Shannon Divergence (e.g., GAIL [Ho and Ermon, 2016]).

Here, we consider a more challenging IL setting, where experts' demonstrations consist only of observations, no action or reward signals are available to the learner, and no reset is allowed (e.g., a robot learns a task by just watching an expert performing the task). We call this setting *Imitation Learning from Observations alone* (ILFO). Under this setting, without access to expert actions, approaches like DAgger, AggreVaTe, GAIL, and Behaviour Cloning by definition cannot work. Although recently several model-based approaches, which learn an inverse model that predicts the actions taken by an expert [Edwards et al., 2018, Torabi et al., 2018] based on successive observations, have been proposed, these approaches can suffer from covariate shift Ross et al. [2011]. While we wish to train a predictor that can infer an expert's actions accurately *under the expert's observation distribution*, we do not have access to actions generated by the expert conditioned on the expert's observation (See Section 3.6 for a more detailed discussion). An alternative strategy is to handcraft cost functions that use some distance metric to penalize deviation from the experts' trajectories (e.g., Liu et al. [2018], Peng et al. [2018]), which is then optimized by Reinforcement Learning (RL). These methods typically involve hand-designed cost functions that sometimes require prior task-specific knowledge Peng et al. [2018]. The quality of the learned policy is therefor completely dependent on the hand-designed costs which could be widely different from the true cost of the underlying MDP. Ideally, we would like to learn a policy that minimizes the unknown true cost function of the underlying MDP.

In this work, we explicitly consider learning near-optimal policies in a sample and computationally efficient manner. Specifically, we focus on large-scale MDPs where the number of unique observations is extremely large (e.g., high-dimensional observations such as raw-pixel images). Such large-scale MDP settings immediately exclude most existing sample efficient RL algorithms, which are often designed for small tabular MDPs, whose sample complexities have a polynomial dependency on the number of observations and hence cannot scale well. To solve large-scale MDPs, we need to design algorithms that leverage function approximation for generalization. Specifically, we are interested in algorithms with the following three properties: (1) *near-optimal performance guarantees*, i.e., we want to search for a policy whose performance is close to the expert's in terms of the expected total cost of the underlying MDP (and not a hand-designed cost function); (2) *sample efficiency*, we require sample complexity that scales polynomially with respect to all relevant parameters (e.g., horizon, number of actions, statistical complexity of function approximators) except the cardinality of the observation space—hence excluding PAC RL algorithms designed for small tabular MDPs; (3) *computational efficiency*: we rely on the notion of oracle-efficiency

Agarwal et al. [2014] and require the number of efficient oracle calls to scale polynomially—thereby excluding recently proposed algorithms for Contextual Decision Processes which are not computationally efficient (Jiang et al. [2016], Sun et al. [2018d])). To the best of our knowledge, the desiderata above requires designing new algorithms.

With access to experts' trajectories of observations we introduce a model-free algorithm, called Forward Adversarial Imitation Learning (Fail), that decomposes ILfO into H independent two-player min-max games, where H is the horizon length. We aim to learn a sequence of time-dependent policies from $h = 1$ to H, where at any time step h, the policy π_h is learned such that the generated observation distribution at time step $h+1$, conditioned on $\pi_1, \dots, \pi_{h-1}$ being fixed, matches the expert's observation distribution at time step $h + 1$, in terms of an Integral Probability Metric (IPM) [Müller et al., 1997]. IPM is a family of divergences that can be understood as using a set of discriminators to distinguish two distributions (e.g., Wasserstein distance is one such special instance). We analyze the sample complexity of Fail and show that Fail can learn a near-optimal policy in sample complexity that does not explicitly depend on the cardinality of observation space, but rather only depends on the complexity measure of the policy class and the discriminator class. Hence Fail satisfies the above mentioned three properties. To the best of our knowledge, Fail is the first provably efficient algorithm in ILfO setting. In addition to the algorithmic contribution, we also demonstrate that learning under ILfO can be exponentially more sample efficient than pure RL.We also study Fail under two specific settings: (1) Lipschitz continuous MDPs, and (2) Interactive ILfO settings where one can query the expert at any time during training. Finally, we demonstrate the efficacy of Fail on multiple OpenAI Gym control tasks.

3.2 Preliminaries

We consider an episodic finite horizon Decision Process that consists of $\{\mathcal{X}_h\}_{h=1}^{H}, \mathcal{A}, c, H, \rho, P$, where $\mathcal{X}_h$ for $h \in [H]$ is a time-dependent observation space,[1] $\mathcal{A}$ is a discrete action space, H is the horizon, the cost function $c : \mathcal{X} \to \mathbb{R}$, $\rho \in \Delta(\mathcal{X}_1)$ is the initial observation distribution, and P is the transition model i.e., $P : \mathcal{X}_h \times \mathcal{A} \to \Delta(\mathcal{X}_{h+1})$ for $h \in [H-1]$. Note that here we assume the cost function only depends on observations. We assume that $\mathcal{X}_h$ for all $h \in [H]$ is discrete, but $|\mathcal{X}_h|$ is extremely large and hence any sample complexity that has polynomial dependency on $|\mathcal{X}_h|$ should be considered as sample inefficient, i.e., one cannot afford to visit every unique observation. We assume that the total cost is bounded, i.e., for any sequence of costs, $\sum_{h=1}^{H} c_h \leq 1$ (e.g., zero-one loss at the end of each episode). For a time-dependent policy $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_H\}$ with $\pi_h : \mathcal{X}_h \to \Delta(\mathcal{A})$, the value function $V_h^{\boldsymbol{\pi}} : \mathcal{X}_h \to [0, 1]$ is defined as:

$$V_h^{\boldsymbol{\pi}}(x_h) = \mathbb{E}\left[\sum_{i=h}^{H} c(x_i) | a_i \sim \pi_i(\cdot|x_i), x_{i+1} \sim P_{x_i, a_i}\right],$$

and state-action function $Q_h^{\boldsymbol{\pi}}(x_h, a_h)$ is defined as $Q_h^{\boldsymbol{\pi}}(x_h, a_h) = c(x_h) + \mathbb{E}_{x_{h+1} \sim P_{x_h, a_h}}\left[V_{h+1}^{\boldsymbol{\pi}}(x_{h+1})\right]$. We denote $\mu_h^{\boldsymbol{\pi}}$ as the distribution over $\mathcal{X}_h$ at time step h following $\boldsymbol{\pi}$. Given H policy classes

[1]we use the term observation throughout the paper instead of the term state as one would normally use in defining MDPs, for the purpose of sharply distinguishing our setting from tabular MDPs where $\mathcal{X}$ has very small number of states.

$\{\Pi_1, \ldots, \Pi_H\}$, the goal is to learn a $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_H\}$ with $\pi_h \in \Pi_h$, which minimizes the expected total cost:

$$J(\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{h=1}^{H} c(x_h) | a_h \sim \pi_h(\cdot|x_h), x_{h+1} \sim P(\cdot|x_h, a_h)\right].$$

Denote $\mathcal{F}_h \subseteq \{f : \mathcal{X}_h \to \mathbb{R}\}$ for $h \in [H]$. We define a Bellman Operator Γ_h associated with the expert $\pi_h^\star$ at time step h as $\Gamma_h : \mathcal{F}_{h+1} \to \{f : \mathcal{X}_h \to \mathbb{R}\}$where for any $x_h \in \mathcal{X}_h, f \in \mathcal{F}_{h+1}$,

$$\left(\Gamma_h f\right)(x_h) \triangleq \mathop{\mathbb{E}}_{a \sim \pi_h^\star(\cdot|x_h), x_{h+1} \sim P_{x_h, a_h}}\left[f(x_{h+1})\right].$$

Notation For a function $f : \mathcal{X} \to \mathbb{R}$, we denote $\|f\|_\infty = \sup_{x \in \mathcal{X}} |f(x)|$, $\|f\|_L$ as the Lipschitz constant: $\|f\|_L = \sup_{x_1, x_2, x_1 \neq x_2} (f(x_1) - f(x_2))/d(x_1, x_2)$, with $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ being the metric in space $\mathcal{X}$.[2] We consider a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ defined with a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to [0, 1]$ such that $\mathcal{H}$ is the span of $\{k(x, \cdot) : x \in \mathcal{X}\}$, and we have $f(x) = \langle f, k(x, \cdot)\rangle$, with $\langle k(x_1, \cdot), k(x_2, \cdot)\rangle \triangleq k(x_1, x_2)$. For any $f \in \mathcal{H}$, we define $\|f\|_{\mathcal{H}}^2 = \langle f, f\rangle$. We denote $U(\mathcal{A})$ as a uniform distribution over action set $\mathcal{A}$. For $N \in \mathbb{N}^+$, we denote $[N] \triangleq \{1, 2, \ldots, N\}$.

3.2.1 Integral Probability Metrics (IPM)

Integral Probability Metrics (IPM) [Müller, 1997] is a family of distance measures on distributions: given two distributions P_1 and P_2 over $\mathcal{X}$, and a function class $\mathcal{F}$ containing real-value functions $f : \mathcal{X} \to \mathbb{R}$ and symmetric (e.g., $\forall f \in \mathcal{F}$, we have $-f \in \mathcal{F}$), IPM is defined as:

$$\sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim P_1}[f(x)] - \mathbb{E}_{x \sim P_2}[f(x)]\right). \tag{3.1}$$

By choosing different class of functions $\mathcal{F}$, various popular distances can be obtained. For instance, IPM with $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$ recovers Total Variation distance, IPM with $\mathcal{F} = \{f : \|f\|_L \leq 1\}$ recovers Wasserstein distance, and IPM with $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$ with RKHS $\mathcal{H}$ reveals maximum mean discrepancy (MMD).

3.2.2 Assumptions

We first assume access to a Cost-Sensitive oracle and and a Linear Programming oracle.

Cost-Sensitive Oracle The Cost-Sensitive (CS) oracle takes a class of classifiers $\Pi \triangleq \{\pi : \mathcal{X} \to \Delta(\mathcal{A})\}$, a dataset consisting of pairs of feature x and cost vector $c \in \mathbb{R}^K$, i.e., $\mathcal{D} = \{x_i, c_i\}_{i=1}^N$, as inputs, and outputs a classifier that minimizes the average expected classification cost: $\sum_{i=1}^N \pi(\cdot|x_i)^\top c_i / N$.

Efficient cost sensitive classifiers exist (e.g., Beygelzimer et al. [2005]) and are widely used in sequential decision making tasks Agarwal et al. [2014], Chang et al. [2015b].

[2]A metric d (e.g., Euclidean distance) satisfies the following conditions: $d(x, y) \geq 0$, $d(x, y) = 0$ iff $x = y$, $d(x, y) = d(y, x)$ and d satisfies triangle inequality.

Linear Programming Oracle A Linear Programming (LP) oracle takes a class of functions $\mathcal{G}$ as inputs, optimizes a linear functional with respect to $g \in \mathcal{G}$: $\min_{g\in\mathcal{G}} \sum_{i=1}^{N} \alpha_i g(x_i)$.

When $\mathcal{G}$ is in RKHS with bounded norm, the linear functional becomes $\max_{g:\|g\|\leq c}\langle g, \sum_{i=1}^{n} \alpha_i \phi(x_i)\rangle$, from which one can obtain the closed-form solution. Another example is when $\mathcal{G}$ consists of all functions with bounded Lipschitiz constant, i.e., $\mathcal{G} = \{g : \|g\|_L \leq c\}$ for $c \in \mathbb{R}^+$, Sriperumbudur et al. [2012] showed that $\max_{g\in\mathcal{G}} \sum_{i=1}^{n} \alpha g(x_i)$ can be solved by Linear Programming with n many constraints, one for each pair (α_i, x_i). In Appendix 9.2.5, we provide a new reduction to Linear Programming for $\mathcal{G} = \{g : \|g\|_L \leq c_1, \|g\|_\infty \leq c_2\}$ for $c_1, c_2 \in \mathbb{R}^+$.

The second assumption is related to the richness of the function class. We simply consider a time-dependent policy class Π_h and $\mathcal{F}_h$ for $h \in [H]$, and we assume realizability:

Assumption 1 (Realizability and Capacity of Function Class)**.** *We assume Π_h and $\mathcal{F}_h$ contains $\pi_h^\star$ and $V_h^\star$, i.e., $\pi_h^\star \in \Pi_h$ and $V_h^\star \in \mathcal{F}_h$, $\forall h \in [H]$. Further assume that for all h, $\mathcal{F}_h$ is symmetric, and Π_h and $\mathcal{F}_h$ is finite in size.*

Note that we assume Π_h and $\mathcal{F}_h$ to be discrete (but could be extremely large) for analysis simplicity. As we will show later, our bound scales only logarithmically with respect to the size of function class. Extension to the continuous case using classic complexity measures such as VC-dimension and Rademacher complexity is straightforward.

3.3 Algorithm

Our algorithm, Forward Adversarial Imitation Learning (Fail), aims to learn a sequence of policies $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_H\}$ such that its value $J(\boldsymbol{\pi})$ is close to $J(\boldsymbol{\pi}^\star)$. Note that $J(\boldsymbol{\pi}) \approx J(\boldsymbol{\pi}^\star)$ does not necessarily mean that the state distribution of $\boldsymbol{\pi}$ is close to $\boldsymbol{\pi}^\star$. Fail learns a sequence of policies with this property in a forward training manner. The algorithm learns π_i starting from $i = 1$. When learning π_i, the algorithm fixes $\{\pi_1, \ldots, \pi_{i-1}\}$, and solves a min-max game to compute π_i; it then proceeds to time step $i + 1$. At a high level, Fail decomposes a sequential learning problem into H-many independent two-player min-max games, where each game can be solved efficiently via no-regret online learning. Below, we first consider how to learn π_h conditioned on $\{\pi_1, \ldots, \pi_{h-1}\}$ being fixed. We then present Fail by chaining all H many two-player min-max games together.

3.3.1 Learning a Single Step Policy via a Two-Player Min-Max Game

Throughout this section, we assume $\{\pi_1, \ldots, \pi_{h-1}\}$ are learned already and fixed. The sequence of policies $\{\pi_1, \ldots, \pi_{h-1}\}$ for $h \geq 2$ determines a distribution $\nu_h \in \Delta(\mathcal{X}_h)$ over observation space $\mathcal{X}_h$ at time step h. Also expert policy $\boldsymbol{\pi}^\star$ naturally induces a sequence of observation distributions $\mu_h^\star \in \Delta(\mathcal{X}_h)$ for $h \in [H]$. The problem we consider in this section is to learn a policy $\pi_h \in \Pi_h$, such that the resulting observation distribution from $\{\pi_1, \ldots, \pi_{h-1}, \pi_h\}$ at time step $h+1$ is close to the expert's observation distribution $\mu_{h+1}^\star$ at time step $h + 1$.

We consider the following IPM minimization problem. Given the distribution $\nu_h \in \Delta(\mathcal{X}_h)$, and a policy $\pi \in \Pi_h$, via the Markov property, we have the observation distribution at time step $h + 1$ conditioned on v_h and π as $\nu_{h+1}(x) \triangleq \sum_{x_h,a_h} \nu_h(x_h)\pi(a_h|x_h)P(x|x_h, a_h)$ for any $x \in \mathcal{X}_{h+1}$. Recall that the expert observation distribution at time step $h+1$ is denoted as $\mu_{h+1}^\star$. The IPM with $\mathcal{F}_{h+1}$ between ν_{h+1} and $\mu_{h+1}^\star$ is defined as:

$$d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu_{h+1}^\star)$$

Algorithm 2 Min-Max Game $(\mathcal{D}^\star, \mathcal{D}, \Pi, \mathcal{F}, T)$

```
1: Initialize $\pi^0 \in \Pi$
2: for $n = 1$ to $T$ do
3:     $f^n = \arg\max_{f \in \mathcal{F}_{h+1}} u(\pi^n, f)$ (LP Oracle)
4:     $u^n = u(\pi^n, f^n)$
5:     $\pi^{n+1} = \arg\min_{\pi \in \Pi_h} \sum_{t=1}^{n} u(\pi, f^t) + \phi(\pi)$ (Regularized CS Oracle)
6: end for
7: Output: $\pi^{n^\star}$ with $n^\star = \arg\min_{n \in [T]} u^n$
```

$$\triangleq \max_{f \in \mathcal{F}_{h+1}} \left(\mathbb{E}_{x \sim \nu_{h+1}} [f(x)] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [f(x)] \right)$$

Note that $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$ is parameterized by π, and our goal is to minimize $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$ with respect to π over Π_h. However, $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$ is not measurable directly as we do not have access to $\mu^\star_{h+1}$ but only samples from $\mu^\star_{h+1}$. To estimate $d_{\mathcal{F}_{h+1}}$, we draw a dataset $\mathcal{D} = \left\{(x_h^i, a_h^i, x_{h+1}^i)\right\}_{i=1}^{N}$ such that $x_h^i \sim \nu_h$, $a_h^i \sim U(\mathcal{A})$, $x_{h+1}^i \sim P(\cdot|x_h^i, a_h^i)$, together with observation set resulting from expert $\mathcal{D}^\star = \{\tilde{x}_{h+1}^i\}_{i=1}^{N'} \overset{iid}{\sim} \mu^\star_{h+1}$, we form the following empirical estimation of $d_{\mathcal{F}_{h+1}}$ for any π, via importance weighting:

$$\widehat{d}_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1}) \triangleq \max_{f \in \mathcal{F}_{h+1}} \left(\frac{1}{N} \sum_{i=1}^{N} \frac{\pi(a_h^i|x_h^i)}{1/K} f(x_{h+1}^i) - \frac{1}{N'} \sum_{i=1}^{N'} f(\tilde{x}_{h+1}^i) \right), \tag{3.2}$$

where the importance weight $K\pi(a_h^i|x_h^i)$ is used to account for the fact that we draw actions uniformly from $\mathcal{A}$ but want to evaluate π. Though due to the max operator, $\widehat{d}_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$ is not an unbiased estimate of $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$, in Appendix 9.2.1, we show that $\widehat{d}_{\mathcal{F}_{h+1}}$ indeed is a good approximation of $d_{\mathcal{F}_{h+1}}$ via an application of the standard Bernstein's inequality and a union bound over $\mathcal{F}_{h+1}$. Hence we can approximately minimize $\widehat{d}_{\mathcal{F}_{h+1}}$ with respect to π: $\min_{\pi \in \Pi} \widehat{d}_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$, resulting in a two-player min-max game. Intuitively, we can think of π as a generator, such that, conditioned on ν_h, it generates next-step samples x_{h+1} that are similar to the expert samples from $\mu^\star_{h+1}$, via fooling discriminators $\mathcal{F}_{h+1}$.

Note that the above formulation is a two-player game, with the utility function for π and f defined as:

$$u(\pi, f) \triangleq \sum_{i=1}^{N} K\pi(a_h^i|x_h^i) f(x_{h+1}^i)/N - \sum_{i=1}^{N'} f(\tilde{x}_{h+1}^i)/N'. \tag{3.3}$$

Algorithm 2 solves the minmax game $\min_\pi \max_f u(\pi, f)$ using no-regret online update on both f and π. At iteration n, player f plays the best-response via $f_n = \arg\min_f u(\pi^n, f)$ (Line 3) and player π plays the Follow-the-Regularized Leader (FTRL) [Shalev-Shwartz et al., 2012] as $\pi^{n+1} = \sum_{t=1}^{n} u(\pi, f^t) + \phi(\pi)$ with ϕ being convex regularization (Line 5). Note that other no-regret online learning algorithms (e.g., replacing FTRL by incremental online learning algorithms like OGD [Zinkevich, 2003]) can also be used to approximately optimize the above min-max formulation. After the end of Algorithm 2, we output a policy π among all computed policies $\{\pi^i\}_{i=1}^{T}$ such that $\widehat{d}_{\mathcal{F}_{h+1}}(\pi|\nu_n, \mu^\star_{n+1})$ is minimized (Line 7).

Regarding the computation efficiency of Algorithm 2, the best response computation on f in Line 3 can be computed by a call to the LP Oracle, while FTRL on π can be implemented by a call to the regularized CS Oracle. Regarding the statistical performance, we have the following theorem:

Theorem 10. *Given $\epsilon \in (0,1], \delta \in (0,1]$, set $T = \Theta\left(\frac{4K^2}{\epsilon^2}\right)$, $N = N' = \Theta\left(\frac{K \log(|\Pi_h||\mathcal{F}_{h+1}|/\delta)}{\epsilon^2}\right)$, Algorithm 2 outputs π such that with probability at least $1-\delta$,*

$$\left| d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1}) - \min_{\pi \in \Pi_h} d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1}) \right| \leq O(\epsilon).$$

The proof of the above theorem is included in Appendix 9.2.1, which combines standard min-max theorem and uniform convergence analysis. The above theorem essentially shows that Algorithm 2 successfully finds a policy π whose resulting IPM is close to the smallest possible IPM one could achieve if one had access to $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu^\star_{h+1})$ directly, up to ϵ error. Intuitively, from Theorem 10, we can see that if v_h—the observation distribution resulting from fixed policies $\{\pi_1, \ldots, \pi_{h-1}\}$, is similar to $\mu^\star_h$, then we guarantee to learn a policy π, such that the new sequence of policies $\{\pi_1, \ldots, \pi_{h-1}, \pi\}$ will generates a new distribution v_{h+1} that is close to $\mu^\star_{h+1}$, in terms of IPM with $\mathcal{F}_{h+1}$. The algorithm introduced below is based on this intuition.

3.3.2 Forward Adversarial Imitation Learning

Theorem 10 indicates that conditioned on $\{\pi_1, \ldots, \pi_{h-1}\}$ being fixed, Algorithm 2 finds a policy $\pi \in \Pi_h$ such that it approximately minimizes the divergence—measured under IPM with $\mathcal{F}_{h+1}$, between the observation distribution ν_{h+1} resulting from $\{\pi_1, \ldots, \pi_{h-1}, \pi\}$, and the corresponding distribution $\mu^\star_{h+1}$ from expert.

Algorithm 3 FAIL($\{\Pi_h\}_h$, $\{\mathcal{F}_h\}_h$, ϵ, n, n', T)

```
1:  Set $\boldsymbol{\pi} = \emptyset$
2:  for $h = 1$ to $H-1$ do
3:      Extract expert's data at $h+1$: $\tilde{\mathcal{D}} = \{\tilde{x}^i_{h+1}\}_{i=1}^{n'}$
4:      $\mathcal{D} = \emptyset$
5:      for $i = 1$ to $n$ do
6:          Reset $x_1^{(i)} \sim \rho$
7:          Execute $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{h-1}\}$ to generate state $x^i_h$
8:          Execute $a^i_h \sim U(\mathcal{A})$ to generate $x^i_{h+1}$ and add $(x^i_h, a^i_h, x^i_{h+1})$ to $\mathcal{D}$
9:      end for
10:     Set $\pi_h$ to be the return of Algorithm 2 with inputs $\left(\{\tilde{x}^i_{h+1}\}_{i=1}^{n'}, \mathcal{D}, \Pi_h, \mathcal{F}_{h+1}, T\right)$
11:     Append $\pi_h$ to $\boldsymbol{\pi}$
12: end for
```

With Algorithm 2 as the building block, we now introduce our model-free algorithm—Forward Adversarial Imitation Learning (FAIL) in Algorithm 3. Algorithm 3 integrates Algorithm 2 into the Forward Training framework [Ross and Bagnell, 2010a], by decomposing the sequential learning problem into H many independent distribution matching problems where each one is solved using Algorithm 2 independently. Every time step h, FAIL assumes that $\pi_1, \ldots, \pi_{h-1}$ have been correctly learned in the sense that the resulting observation distribution ν_h from $\{\pi_1, \ldots, \pi_{h-1}\}$ is close to $\mu_h^\star$ from expert. Therefore, Fail is only required to focus on learning π_h correctly conditioned on $\{\pi_1, \ldots, \pi_{h-1}\}$ being fixed, such that v_{h+1} is close to $\mu_{h+1}^\star$, in terms of the IPM with $\mathcal{F}_{h+1}$. Intuitively, when one has a strong class of discriminators, and the two-player game in each time step is solved near optimally, then by induction from $h = 1$ to H, Fail should be able to learn a sequence of policies such that ν_h is close to $\mu_h^\star$ for all $h \in [H]$ (note for the base case in the induction, we simply have $\nu_1 = \mu_1^\star = \rho$). The analysis of Fail below is based on this intuition.

3.3.3 Analysis of Algorithm 3

The performance of Fail crucially depends on the capacity of the discriminators. Intuitively, discriminators that are too strong cause overfitting (unless one has extremely large number of samples). Conversely, discriminators that are too weak will not be able to distinguish ν_h from $\mu_h^\star$. This dilemma was studied in the Generative Adversarial Network (GAN) literature already by Arora et al. [2017]. Below we study this tradeoff explicitly in IL.

To quantify the power of discriminator class $\mathcal{F}_h$ for all h, we use *inherent Bellman Error* (iBE) with respect to $\boldsymbol{\pi}^\star$:

$$\epsilon_{\mathrm{be}} = \max_h \left(\max_{g \in \mathcal{F}_{h+1}} \min_{f \in \mathcal{F}_h} \|f - \Gamma_h g\|_\infty \right). \tag{3.4}$$

The Inherent Bellman Error is commonly used in approximate value iteration literature [Lazaric et al., 2016, Munos, 2005, Munos and Szepesvári, 2008] and policy evaluation literature [Sutton, 1988]. It measures the worst possible projection error when projecting $\Gamma_h g$ to function space $\mathcal{F}_h$. Intuitively increasing the capacity of $\mathcal{F}_h$ reduces ϵ_{be}.

Using a restricted function class $\mathcal{F}$ potentially introduces ϵ_{be}, hence one may tend to set $\mathcal{F}_h$ to be infinitely powerful discriminator class such as function class consisting of all bounded functions $\{f : \|f\|_\infty \leq c\}$ (recall IPM becomes total variation in this case). However, using $\mathcal{F}_h \triangleq \{f : \|f\|_\infty \leq c\}$ makes efficient learning impossible. The following proposition excludes the possibility of sample efficiency with discriminator class being $\{f : \|f\|_\infty \leq c\}$.

Theorem 11 (Infinite Capacity $\mathcal{F}$ does not generalize)**.** *There exists a MDP with $H = 2$, a policy set $\Pi = \{\boldsymbol{\pi}, \boldsymbol{\pi}'\}$, an expert policy $\boldsymbol{\pi}^\star$ with $\boldsymbol{\pi} = \boldsymbol{\pi}^\star$ (i.e., Π is realizable), such that for datasets $\mathcal{D}^\star = \{\tilde{x}_2^i\}_{i=1}^K$ with $\tilde{x}_2^i \sim \mu_2^\star$, $\mathcal{D} = \{x_2^i\}_{i=1}^K$ with $x_2^i \sim \mu_2^{\boldsymbol{\pi}}$, and $\mathcal{D}' = \{x_2'^{(i)}\}_{i=1}^K$ with $x_2'^{(i)} \sim \mu_2^{\boldsymbol{\pi}'}$, as long as $K = O(\log(|\mathcal{X}|))$, we must have:*

$$\lim_{|\mathcal{X}| \to \infty} \mathrm{P}(\mathcal{D}^\star \cap \mathcal{D} = \emptyset) = 1, \quad \lim_{|\mathcal{X}| \to \infty} \mathrm{P}(\mathcal{D}^\star \cap \mathcal{D}' = \emptyset) = 1.$$

In other words, denote $\hat{\mathcal{D}}$ as the empirical distribution of a dataset $\mathcal{D}$ by assigning probability $1/|\mathcal{D}|$ to any sample in $\mathcal{D}$, we have:

$$\lim_{|\mathcal{X}| \to \infty} \|\hat{\mathcal{D}}^\star - \hat{\mathcal{D}}\|_1 = 1, \lim_{|\mathcal{X}| \to \infty} \|\hat{\mathcal{D}}^\star - \hat{\mathcal{D}}'\|_1 = 1.$$

The above theorem shows by just looking at the samples generated from $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$, and comparing them to the samples generated from the expert policy $\boldsymbol{\pi}^\star$ using $\{f : \|f\|_\infty \leq c\}$

(IPM becomes Total variation here), we cannot distinguish $\boldsymbol{\pi}$ from $\boldsymbol{\pi}'$, as they both look similar to $\boldsymbol{\pi}^\star$, i.e., none of the three datasets overlap with each other, resulting the total variation distances between the corresponding empirical distributions become constants, *unless* the sample size scales $\Omega(\mathrm{poly}(|\mathcal{X}|))$.

Theorem 11 suggests that one should explicitly regularize discriminator class so that it has finite capacity (e.g., bounded VC or Rademacher Complexity). The eestricted discriminator class $\mathcal{F}$ has been widely used in practical applications as well such as density ratio estimation [Nguyen et al., 2010] and learning generative models (i.e., Wasserstain GANs [Arjovsky et al., 2017]).

Denote $|\Pi| = \max_h |\Pi_h|$ and $|\mathcal{F}| = \max_h |\mathcal{F}_h|$. The following theorem shows that the learned time-dependent policies $\boldsymbol{\pi}$'s performance is close to the expert's performance:

Theorem 12 (Sample Complexity of Fail). *Under Assumption 2, for any $\epsilon, \delta \in (0,1]$, set $T = \Theta(\frac{K}{\epsilon^2})$, $n = n' = \Theta(\frac{K\log(|\Pi||\mathcal{F}|H/\delta)}{\epsilon^2})$, with probability at least $1-\delta$,* Fail *(Algorithm 3) outputs $\boldsymbol{\pi}$, such that,*

$$J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \leq H^2\epsilon'_{\mathrm{be}} + H^2\epsilon,$$

by using $\tilde{O}\left(\frac{HK}{\epsilon^2}\log\left(\frac{|\Pi||\mathcal{F}|}{\delta}\right)\right)$ [3] many trajectories with an average inherent Bellman Error ϵ'_{be}:

$$\epsilon'_{\mathrm{be}} \triangleq \max_h \max_{g\in\mathcal{F}_{h+1}} \min_{h\in\mathcal{F}_h} \mathbb{E}_{x\sim(\mu_h^{\boldsymbol{\pi}}+\mu_h^\star)/2}[|f(x) - (\Gamma_h g)(x)|].$$

Note that the average inherent Bellman error ϵ'_{be} defined above is averaged over the state distribution of the learned policy $\boldsymbol{\pi}$ and the state distribution of the expert, which is guaranteed to be smaller than the classic inherent Bellman error used in RL literature (i.e., (3.4)) which uses infinity norm over $\mathcal{X}$. The proof of Theorem 12 is included in Appendix 9.2.3. Regarding computational complexity of Algorithm 3, we can see that it requires polynomial number of calls (with respect to parameters $H, K, 1/\epsilon$) to the efficient oracles (Regularized CS oracle and LP oracle).

Since our analysis only uses uniform convergence analysis and standard concentration inequalities, extension to continuous Π and $\mathcal{F}$ with complexity measure such as VC-dimension, Rademacher complexity, and covering number is standard. We give an example in Section 3.5.1.

3.4 The Gap Between ILfO and RL

To quantify the gap between RL and ILfO, below we present an exponential separation between ILfO and RL in terms of sample complexity to learn a near-optimal policy. We assume that the expert policy is the optimal policy of the underlying MDPs.

Proposition 1 (Exponential Separation Between RL and ILfO). *Fix $H \in \mathbb{N}^+$ and $\epsilon \in (0, \sqrt{1/8})$. There exists a family of MDP with deterministic dynamics, with horizon H, $2^H - 1$ many states, two different actions, such that for any RL algorithm, the probability of outputting a policy $\hat{\boldsymbol{\pi}}$ with $J(\hat{\boldsymbol{\pi}}) \leq J(\boldsymbol{\pi}^\star) + \epsilon$ after collecting T trajectories is at most $2/3$ for all $T \leq O(2^H/\epsilon^2)$. On the other hand, for the same MDP, given one trajectory of observations $\{\tilde{x}_h\}_{h=1}^H$ from the expert policy $\boldsymbol{\pi}^\star$, there exists an algorithm that deterministically outputs $\boldsymbol{\pi}^\star$ after collecting $O(H)$ trajectories.*

[3]In $\tilde{O}$, we drop log terms that does not dependent on $|\Pi|$ or $|\mathcal{F}|$. In Θ we drop constants that do not depend on $H, K, |\mathcal{X}|, |\Pi|, |\mathcal{F}|, 1/\epsilon, 1/\delta$. Details can be found in Appendix.

Proposition 1 shows having access to expert's trajectories of observations allows us to efficiently solve some MDPs that are otherwise provably intractable for any RL algorithm (i.e., requiring exponentially many trajectories to find a near optimal policy). This kind of exponential gap previously was studied in the interactive imitation learning setting where the expert also provides action signals Sun et al. [2017c] and one can query the expert's action at any time step during training. To the best of our knowledge, this is the *first exponential gap* in terms of sample efficiency between ILFO and RL.

3.5 Cases Where Global Optimal Policies are Achievable

In this section, we study two settings where inherent Bellman Error is zero even under restricted discriminator class : (1) Lipschitz Continuous MDPs (e.g., Kakade et al. [2003]) and (2) Interactive Imitation Learning from Observation where expert is available to query during training.

3.5.1 Lipschitz Continuous MDPs

A MDP is Lipschitz continuous if cost functions, dynamics and $\pi_h^\star$ are Lipschitz continuous in metric space $(\mathcal{X}, d)$:

$$\|P(\cdot|x,a) - P(\cdot|x',a)\|_1 \leq L_P d(x,x'),$$
$$\|\pi_h^\star(\cdot|x) - \pi_h^\star(\cdot|x')\|_1 \leq L_\pi d(x,x'),$$

for the know metric d and Lipschitz constants L_P, L_π. In this section for the simplicity of analysis, we simply assume $c(x_h) = 0$ for all $h \in [H-1]$, but $c(x_H) \in [0,1]$ (i.e., sparse reward setting where one only receive a cost at the last step of an episode, e.g., win the game or not). Our results can be easily extended to general non-sparse reward setting. Under this condition, the Bellman operator with respect to $\boldsymbol{\pi}^\star$ is Lipschitz continuous in the metric space $(\mathcal{X}, d)$: $|\Gamma_h f(x_1) - \Gamma_h f(x_2)| \leq (\|f\|_\infty (L_P + L_\pi)) d(x_1, x_2)$, where we applied Holder's inequality. Hence, we can design the function class $\mathcal{F}_h$ for all $h \in [H]$ as follows:

$$\mathcal{F}_h = \{f : \|f\|_L \leq (L_P + L_\pi), \|f\|_\infty \leq 1\}, \tag{3.5}$$

which will give us $\epsilon_{\text{be}} = 0$ and $V_h^\star \in \mathcal{F}_h$ due to the assumption on the cost function. Namely $\mathcal{F}_h$ is the class of functions with bounded Lipschitz constant and bounded value. This class of functions is widely used in practice for learning generative models (e.g., Wasserstain GAN). Note that this setting was also studied in Munos and Szepesvári [2008] for value iteration RL methods.

Denote $L \triangleq L_P + L_\pi$. For $\mathcal{F} = \{f : \|f\|_L \leq L, \|\mathcal{F}\|_\infty \leq 1\}$ we show that we can evaluate the empirical IPM $\sup_{f\in\mathcal{F}} \left(\sum_{i=1}^N f(x_i)/N - \sum_{i=1}^{N'} f(x_i')/N'\right)$ by reducing it to Linear Programming, of which the details are deferred to Appendix 9.2.5.

Regarding the generalization ability, note that our function class $\mathcal{F}$ is a subset of all functions with bounded Lipschitz constant, i.e., $\mathcal{F} \subset \{f : \|f\|_L \leq L\}$. The Rademacher complexity for bounded Lipschitz function class grows in the order of $O(N^{-1/\text{cov}(\mathcal{X})})$ (e.g., see [Sriperumbudur et al., 2012]), with $\text{cov}(\mathcal{X})$ being the covering dimension of the metric space $(\mathcal{X}, d)$.[4] Extending Theorem 12 to Lipschitz continuous MDPs, we have the following corollary.

[4]Covering dimension is defined as $\text{cov}(\mathcal{X}) \triangleq \inf_{d>0}\{N_\epsilon(\mathcal{X}) \leq \epsilon^{-d}, \forall \epsilon > 0\}$, where $N_\epsilon(\mathcal{X})$ is the covering number of metric space $(\mathcal{X}, \mathcal{D})$.

Corollary 1 (Sample Complexity of Fail for Lipschitz Continuous MDPs)**.** *With the above set up on Lipschitz continuous MDP and $\mathcal{F}_h$ for $h \in [H]$ ((3.5)), given $\epsilon, \delta \in (0,1]$, set $T = \Theta(\frac{K}{\epsilon^2})$, $n = n' = \tilde{\Theta}(\frac{K(LK)^{\mathrm{cov}(\mathcal{X})}\log(|\Pi|/\delta)}{\epsilon^{2+\mathrm{cov}(\mathcal{X})}})$, then with probability at least $1-\delta$, Fail (Algorithm 3) outputs a policy with $J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \leq O(H^2\epsilon)$ using at most $\tilde{O}\left(\frac{HK(KL)^{\mathrm{cov}(\mathcal{X})}}{\epsilon^{(2+\mathrm{cov}(\mathcal{X}))}}\log\left(\frac{|\Pi|}{\delta\epsilon}\right)\right)$ many trajectories.*

The proof of the above corollary is deferred to Appendix 9.2.6 which uses a standard covering number argument over $\mathcal{F}_h$ with norm $\|\cdot\|_\infty$. Note that we get rid of iBE here and hence as the number of sample approaches to infinity, Fail finds the global optimality. Though the bound has an exponential dependency on the covering dimension, note that the covering dimension $\mathrm{cov}(\mathcal{X})$ is completely dependent on the underlying metric space $(\mathcal{X}, d)$ and could be much smaller than the real dimension of $\mathcal{X}$. Note that the above theorem also serves an example regarding how we can extend Theorem 12 to settings where $\mathcal{F}$ contains infinitely many functions but with bounded statistical complexity (similar techniques can be used for Π as well).

3.5.2 Interactive Imitation Learning from Observations

We can avoid iBE in an interactive learning setting, where we assume that we can query expert during training. But different from previous interactive imitation learning setting such as AggreVaTe, LOLS [Chang et al., 2015b, Ross and Bagnell, 2014], and DAgger [Ross et al., 2011], here we do not assume that expert provides actions nor cost signals. Given any observation x, we simply ask expert to take over for just one step, and observe the observation at the next step, i.e., $x' \sim P(\cdot|x, a)$ with $a \sim \pi^\star(\cdot|x)$. Note that compared to the non-interactive setting, interactive setting assumes a much stronger access to expert. In this setting, we can use arbitrary class of discriminators with bounded complexity. Due to space limit, we defer the detailed description of the interactive version of Fail (Algorithm 10) to Appendix 9.2.7. The following theorem states that we can avoid iBE:

Theorem 13. *Under Assumption 2 and the exists of an interactive expert, for any $\epsilon \in (0,1]$ and $\delta \in (0,1]$, set $T = \Theta(\frac{K}{\epsilon^2})$, $n = \Theta(\frac{K\log(|\Pi||\mathcal{F}|H)/\delta}{\epsilon^2})$, with probability at least $1-\delta$, Algorithm 10) outputs a policy $\boldsymbol{\pi}$ such that:*

$$J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \leq H\epsilon,$$

by using at most $\tilde{O}\left(\frac{HK}{\epsilon^2}\log\left(\frac{|\Pi||\mathcal{F}|}{\delta}\right)\right)$many trajectories.

Compare to the non-interactive setting, we get rid of iBE, at the cost of a much stronger expert.

3.6 Discussion on Related Work

Some previous works use the idea of learning an inverse model to predict actions (or latent causes) [Edwards et al., 2018, Nair et al., 2017, Torabi et al., 2018] from two successive observations and then use the learned inverse model to generate actions using expert observation demonstrations. With the inferred actions, it reduces the problem to normal imitation learning. We note here that learning an inverse model is ill-defined. Specifically, simply by the Bayes rule, the inverse model $P(a|x_h, x_{h+1})$—the probability of action

a was executed at x_h such that the system generated x_{h+1}, is equivalent to $P(a|x_h, x_{h+1}) \propto P(x_{h+1}|x_h, a)P(a|x_h)$, i.e., an inverse model $P(a|x_h, x_{h+1})$ is explicitly dependent on the action generation policy $P(a|x_h)$. Unlike $P(x_{h+1}|x_h, a)$, without the policy $P(a|x_h)$, the inverse model is ill-defined by itself alone. This means that if one wants to learn an inverse model that predicts expert actions along the trajectory of observations generated by the expert, one would need to learn an inverse model, denoted as $P^\star(a|x_h, x_{h+1})$, such that $P^\star(a|x_h, x_{h+1}) \propto P(x_{h+1}|x_h, a)\pi^\star_h(a|x_h)$, which indicates that one needs to collect actions from $\pi^\star_h$. Hence replying on learning such an inverse model $P^\star(a|x, x')$ will not provide any performance guarantees, unless we have actions collected from $\pi^\star$.

3.7 Simulation

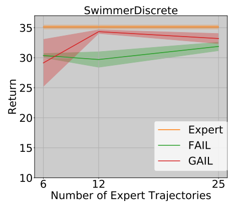
We test Fail on three simulations from openAI Gym [Brockman et al., 2016a]: Swimmer, Reacher, and the Fetch Robot Reach task (FetchReach). For Swimmer we set H to be 100 while for Reacher and FetchReach, H is 50 in default. The Swimmer task has dense reward (i.e.,., reward at every time step). For reacher, we try both dense reward and sparse reward (i.e., success if it reaches to the goal within a threshold). FetchReach is a sparse reward task. As our algorithm is presented for discrete action space, for all three tasks, we discrete the action space via discretizing each dimension into 5 numbers and applying categorical distribution independently for each dimension.[5]

For baseline, we modify GAIL [Ho and Ermon, 2016], a model-free IL algorithm, based on the implementation from OpenAI Baselines, to make it work for ILfO. We delete the input of actions to discriminators in GAIL to make it work for ILfO. Hence the modified version can be understood as using RL (the implementation from OpenAI uses TRPO Schulman et al. [2015a]) methods to minimize the divergence between the learner's average state distribution and the expert's average state distribution.

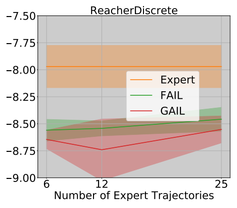
For Fail implementation, we use MMD as a special IPM, where we use RBF kernel and set the width using the common median trick (e.g.,Fukumizu et al. [2009]) without any future tuning. All policies are parameterized by one-layer neural networks. Instead of using FTRL, we use ADAM as an incremental no-regret learner, with all default parameters (e.g., learning rate) [Kingma and Ba, 2014]. The total number of iteration T in Algorithm 2 is set to 1000 without any future tuning. During experiments, we stick to default hyper-parameters for the purpose of best reflecting the algorithmic contribution of Fail. All the results below are averaged over ten random trials with seeds randomly generated between $[1, 1e6]$. The experts are trained via a reinforcement learning algorithm (TRPO Schulman et al. [2015a]) with multiple millions of samples till convergence.

Figure 3.1 shows the comparison of expert, Fail, and GAIL on two dense reward tasks with different number of expert demonstrations, under fixed total number of training samples (one million). We report mean and standard error in Figure 3.1. We observe GAIL outperforms Fail in Swimmer on some configurations, while Fail outperforms GAIL on Reacher (Dense reward) for all configuration.

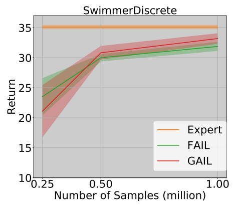
[5] i.e., $\pi(a|x) = \prod_{i=1}^{d} \pi_i(a[i]|x)$, with $a[i]$ stands for the i-th dimension. Note that common implementation for continuous control often assumes such factorization across action dimensions as the covariance matrix of the Gaussian distribution is often diagonal. Recent work [Tang and Agrawal, 2018] showed that for Open AI gym continuous control tasks, discretizing action space can capture the multi-mode distributions, often results better performance, and naturally guarantees to satisfy control input constraints. We leave the extension of Fail to continuous control for future work.



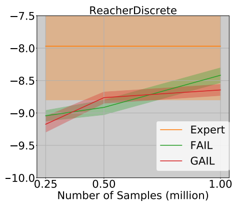
(a) Swimmer (1m)



(b) Reacher (1m)

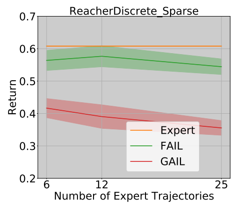


(c) swimmer (25)

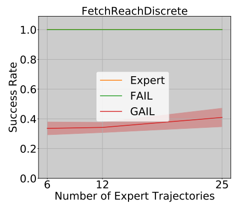


(d) Reacher (25)

Figure 3.1: Performance of expert, Fail, and GAIL (without actions) on two dense reward tasks (Reacher and Hopper). For (a) and (b), we fix the number of training samples while varying the number of expert demonstrations (6, 12, 25). For (c) and (d), we fix the number of expert trajectories, while varying the training samples.



(a) Reacher Sparse (1m)



(b) FetchReach (1m)

Figure 3.2: Performance of expert, Fail, and GAIL (without actions) on two sparse control tasks (Reacher Sparse and Fetch-Reach). We fix the number of training samples while varying the number of expert demonstrations (6, 12, 25).

Figure 3.2 shows the comparison of expert, Fail, and GAIL on two sparse reward settings. We observe that Fail significantly outperforms GAIL on these two sparse reward tasks. For sparse reward, note that the what really matters is to reach the target at the end, Fail achieves that by matching expert's state distribution and learner's state distribution one by one at every time step till the end, while GAIL (without actions) loses the sense of ordering by focusing on the average state distributions.

The above experiments also indicates that Fail in general can work well for shorter horizon (e.g., $H = 50$ for Reacher and Fetch), while shows much less improvement over GAIL on longer horizon task. We think this is because of the nature of Fail which has to learn a sequence of time-dependent policies along the entire horizon H. Long horizon requires larger number of samples. While method like GAIL learns a single stationary policy with all training data, and hence is less sensitive to horizon increase. We leave extending Fail to learning a single stationary policy as a future work.

3.8 Conclusion

In this chapter, we study Imitation Learning from Observation setting and propose an algorithm, Forward Adversarial Imitation Learning (FAIL), that achieves sample efficiency and computational efficiency. FAIL decomposes the sequential learning tasks into independent two-player min-max games of which is solved via general no-regret online learning. In addition to the algorithmic contribution, we present the first exponential gap in terms of sample complexity between ILFO and RL, demonstrating the potential benefit from expert?s observations. A key observation is that one should explicitly regularize the class of discriminators to achieve sample efficiency and design discriminators to decrease the inherent Bellman Error. Experimentally, while GAIL can be used to solve the ILFO problem by removing action inputs to the discriminators, FAIL works just as well in problems with dense reward. Our analysis of FAIL provides the first strong theoretical guarantee for solving ILFO, and FAIL significantly outperforms GAIL on sparse reward MDPs, which are common in practice

Part II

Model-Free Reinforcement Learning

Chapter 4

Policy Evaluation: Online Bellman Residual Algorithms with Predictive Error Guarantees

In this chapter, we establish a connection between optimizing the Bellman Residual and worst case long-term predictive error. In the online learning framework, learning takes place over a sequence of trials with the goal of predicting a future discounted sum of rewards. Our analysis shows that, together with a stability assumption, *any* no-regret online learning algorithm that minimizes Bellman error ensures small prediction error. No statistical assumptions are made on the sequence of observations, which could be non-Markovian or even adversarial. Moreover, the analysis is independent of the particular form of function approximation and the particular (stable) no-regret approach taken. Our approach thus establishes a broad new family of provably sound algorithms for Bellman Residual-based learning and provides a generalization of previous worst-case result for minimizing predictive error. We investigate the potential advantages of some of this family both theoretically and empirically on benchmark problems.

The contributions in this chapter were first presented in the joint work with Drew Bangell [Sun and Bagnell, 2015].

4.1 Introduction

Reinforcement learning (RL) is an online paradigm for optimal sequential decision making where an agent interacts with an environment, takes actions, receives rewards and tries to maximize its *long-term reward*, a discounted sum of all the rewards that will be received from now on. An important part of RL is policy evaluation, the problem of evaluating the expected long-term rewards of a fixed policy. *Temporal Difference* (TD) is a famous family of algorithms for policy evaluation. In practice, we are typically interested in complex problem domains (e.g., continuous state space RL) and function approximations (e.g., linear functions) are used for policy evaluation. However, it has been observed that when combined with function approximation, TD may diverge and lead to poor prediction. The *Residual Gradient* (RG) was proposed [Baird, 1995] to address these concerns. RG attempts to minimize the *Bellman Error* (BE) (see definition in Sec. 4.2), typically with linear function approximation, using stochastic gradient descent. Since then comparison between the family of TD algorithms and RG has received tremendous attention, although most of the analyses heavily rely on certain stochastic assumptions of the environment such as that the sequence of observations are Markovian or from a static Markov Decision Process (MDP). For instance Schoknecht and Merke [2003] showed that TD converges provably faster than RG if the value functions are presented by tabular form. Scherrer [2010] shows that Bellman Residual minimization enjoys a guaranteed performance while TD does not in general when states are sampled from arbitrary distributions that may not correspond to trajectories taken by the system. Experimentally, they also show that TD converges faster but may generate poor prediction when it is close to divergence.

Schapire and Warmuth [1996] and Li [2008] provided worst-case analysis of long-term predictive error for variants of the linear TD and RG under a non-probabilistic online learning setting. Their results rely on an elegant spectral analysis of a matrix that is related to **specific** update rules of the TD and RG algorithms under linear function approximation. Unfortunately, this approach makes it more difficult to extend their worst-case (assumption free) analysis to broader families of algorithms and representations that target the Bellman and Temporal Difference errors.

Following Schapire and Warmuth [1996] and Li [2008]'s online learning framework, we present a simple, general connection between long-term predictive error and no-regret online learning that attempts to minimize BE. The central idea is that methods such as RG should be fundamentally understood as online algorithms as opposed to standard gradient methods, and that one cannot simultaneously make consistent predictions in the sense of BE while doing a poor job in terms of long-run predictions. Similar to Schapire and Warmuth [1996] and Li [2008], our analysis does not rely on any statistical assumptions about the underlying system. This allows us to analyze more difficult scenarios such as Markov Decision Process with transition probabilities changing over time or even with each transition chosen entirely adversarial. Our analysis generalizes to a broader class of functions to approximate the value function. Previous work from Robards et al. [2011] and Engel et al. [2005] explored the possibility of using non-linear function approximation, but to our knowledge no further analysis on the soundness with respect to prediction error are known.

Our analysis of the connection between online long-term reward prediction and no-regret online learning provides a unifying view of the relationship between prediction errors and BE and consequently suggests a broad new family of algorithms. Specifically, we present and analyze concrete examples of how to apply several well-known no-regret online algorithms such as *Online Gradient Descent* (OGD) from Zinkevich [2003], *Online Newton Step*

(ONS) from Hazan et al. [2006] and *Online Frank Wolf* (OFW) from Hazan and Kale [2012] to online prediction of long-term rewards. Particularly, our analysis generalizes the `RG` algorithm from Baird [1995] in the following three aspects: (1) `RG` is a specific example of our family of algorithms that runs OGD on a sequence of `BE` loss functions, (2) `RG` can be naturally combined with more general function approximation such as functions in *Reproducing Kernel Hilbert Space* (RKHS), and (3) applying our analysis to `RG` provides asymptotically tighter bounds on the average prediction error of long-term rewards than that provided in Li [2008]. We also find that ONS, which has no-regret rate of $O(\log T/T)$, has a faster convergence of the average prediction error of long-term rewards. With OFW, we are able to achieve sparse predictors under some conditions. We analyze these algorithms in detail in Sec. 4.4.

We emphasize that stability of online algorithms is essential for our results– the no-regret property can be shown by example to be insufficient to achieve low predictive error. We hence introduce the definition of *Online Stability* condition in Sec. **??**, which intuitively measures the difference between two successive predictors. Our online stability condition is general enough such that most popular no-regret online algorithms naturally satisfy this condition and hence this condition does not severely limit the scope of no-regret online algorithms. Our analysis shows that the combination of the no-regret property and online stability is sufficient to promise small predictive error on the long-term rewards.

4.2 Preliminaries

We consider the sequential online learning model presented in Li [2008], Schapire and Warmuth [1996] where no statistical assumptions about the sequence of observations are made. The sequence of the observations forms a connected stream of states which can either be Markovian as typically assumed in RL problem settings or even adversarial. We define the observation at time step t as $\mathbf{x}_t \in \mathbb{R}^n$, which usually represents the features of the environment at t. Throughout the paper, we assume that feature vector $\mathbf{x}$ is bounded as $\|\mathbf{x}\|_2 \leq X, X \in \mathbb{R}^+$. The corresponding reward at step t is defined as $r_t \in \mathbb{R}$, where we assume that reward is always bounded $|r| \leq R \in \mathbb{R}^+$. Given a sequence of observations $\{\mathbf{x}_t\}$ and a sequence of rewards $\{r_t\}$, the long-term reward at t is defined as $v_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_s$, where $\gamma \in [0,1)$ is a discount factor. Given a function space $\mathcal{F}$ the learner chooses a predictor f at each time step from $\mathcal{F}$ for predicting long-term rewards. Throughout this paper, we assume that any prediction made by a predictor f at a state $\mathbf{x}$ is upper bounded as $|f(\mathbf{x})| \leq P \in \mathbb{R}^+$, for any $f \in \mathcal{F}$ and $\mathbf{x}$.

At time step $t = 0$, the learner receives $\mathbf{x}_0$, initializes a predictor $f_0 \in \mathcal{F}$ and makes prediction of v_0 as $f_0(\mathbf{x}_0)$. Rounds of learning then proceeds as follows: the learner makes a prediction of v_t at step t as $f_t(\mathbf{x}_t)$; the learner then observes a reward r_t and the next state $\mathbf{x}_{t+1}$; the learner updates its predictor to f_{t+1}. This interaction repeats and is terminated after T steps. Throughout this paper, we call this problem setting as *online prediction of long-term reward*.

We define the *signed Bellman Error* at step t for predictor f_t as $b_t = f_t(\mathbf{x}_t) - r_t - \gamma f_t(\mathbf{x}_{t+1})$, which measures effectively how self consistent f_t is in its predictions between time step t and $t+1$. For any $f^* \in \mathcal{F}$, we define the corresponding signed Bellman Error as $b_t^* = f^*(\mathbf{x}_t) - r_t - \gamma f^*(\mathbf{x}_{t+1})$. We denote the *Bellman Error* (`BE`) as the square of the signed Bellman error b_t^2.

The *Signed Prediction Error* of long-term reward at t for f_t is defined as $e_t = f_t(\mathbf{x}_t) - v_t$

and $e_t^* = f^*(\mathbf{x}_t) - v_t$ for f^* accordingly. We will typically be interested in bounding the *Prediction Error* (PE) e_t^2 of a given algorithm in terms of the best possible PE. To lighten notation in the following sections, all sums over time indices implicitly run from 0 to $T-1$ unless explicitly noted otherwise.

4.2.1 NO-REGRET ONLINE LEARNING

Under our online setting, we will define loss functional l_t at step t as the traditional Bellman Error (BE):

$$l_t(f) = (f(\mathbf{x}_t) - r_t - \gamma f(\mathbf{x}_{t+1}))^2. \tag{4.1}$$

Note that $l_t(f_t) = b_t^2$.

Following the setting of online prediction of long-term reward, the learner computes predictor f_t at time step t and then receives the loss function l_t and the loss $l_t(f_t)$ (after the learner receives r_t and $\mathbf{x}_{t+1}$). We say that the online algorithm is no-regret with respect to BE if:

$$\lim_{T\to\infty} \frac{1}{T}\sum l_t(f_t) - \frac{1}{T}\sum l_t(f^*) \leq 0, \tag{4.2}$$

for any predictor $f^* \in \mathcal{F}$, including the best predictor that minimizes $\sum l_t(f)$ in hindsight.

The sequence of predictors f_t being no-regret intuitively means that the predictors are giving nearly as consistent predictions over time as is possible in that function class. One might wish that the sequence of predictors being no-regret is a sufficient condition for small prediction error. More formally, one might expect that if Eq. 4.2 holds for the sequence of predictors $\{f_t\}$, $\sum e_t^2$ can be upper bounded:

$$\lim_{T\to\infty} \frac{1}{T}\sum e_t^2 \leq C\frac{1}{T}\sum e_t^{*2}, \quad \forall f^* \in \mathcal{F}, \tag{4.3}$$

where $C \in \mathbb{R}^+$ is a constant. Schapire and Warmuth [1996] showed such a conclusion (Eq. 4.3) for TD and later on Li [2008] proved such a conclusion for RG, both under the assumption that $f(\mathbf{x})$ is linear.

Unfortunately, however, simply being no-regret (Eq. 4.2) is **not** a sufficient condition for upper bounding prediction error ($\sum e_t^2$) as in the form of Eq. 4.3 for general function approximation form:

Theorem 14. *There exists a sequence of $\{f_t\}$ that is no-regret with respect to the loss functions $\{l_t(f)\}$, but no $C \in \mathbb{R}^+$ exists that makes Eq. 4.3 hold.*

We prove Theorem 14 by providing an example in Appendix (see *Supplementary Material*) which is no-regret on $\{l_t(f_t)\}$ (Eq. 4.2 holds) but Eq. 4.3 does not hold.

4.2.2 ONLINE STABILITY

The counter example that supports Theorem 14 presents a sequence of unstable predictors $\{f_t\}$ where two successive predictors f_t and f_{t+1} vary wildly when predicting the long-term reward of $\mathbf{x}_{t+1}$. Such behavior is rather unusual for typical no-regret online learning algorithms. This suggests introducing a notion of *Online Stability* which we defined as:

Definition 2 (Online Stability). *For the generated sequence of predictors f_t, we say the algorithm is online stable if:*

$$\lim_{T\to\infty} \frac{1}{T} \sum (f_t(\mathbf{x}_{t+1}) - f_{t+1}(\mathbf{x}_{t+1}))^2 = 0. \tag{4.4}$$

Intuitively, the online stability means that on average the difference between successive predictors is eventually small. That is, the difference between $f_t(\mathbf{x}_{t+1})$ and $f_{t+1}(\mathbf{x}_{t+1})$ is small on average. Online stability is a general condition and does not severely limit the scope of the online learning algorithms. For instance, when f is linear, the definition of stability of online learning in [Saha et al., 2012] (see Eq. 3 in Saha et al. [2012]) and [Ross and Bagnell, 2011] implies our form of online stability. We also show in the following section that many popular no-regret online learning algorithms including OGD, ONS and OWF, satisfy our online stability condition.

We show in next section that the sequence of predictors $\{f_t\}$ being no-regret with respect to the loss functions $\{l_t(f_t)\}$ **and satisfying the online stability condition** is sufficient for deriving an upper bound for prediction error as shown in Eq. 4.3.

4.3 Online Learning For Long-term Reward Prediction

In this section, we combine the no-regret condition on loss functions $\{l_t(f)\}$ and the online stability condition together to provide a worst-case analysis of sum of `PE` $\sum e_t^2$, which builds a connection between the `PE` of long-term rewards, regret and online stability.

More formally, our worst-case analysis shows that if the online algorithm running on the sequence of loss $\{l_t(f)\}$ is no-regret and the generated sequence of predictors $\{f_t\}$ satisfies the online stability condition, predictor error can be upper bounded in the form of Eq. 4.3. The analysis does not place any probabilistic assumption on the sequence of observations $\{\mathbf{x}_t\}$ or any assumption on the form of predictors $f \in \mathcal{F}$ (e.g., $f(\mathbf{x})$ does not have to be linear).

We start by first providing two important lemmas below:

Lemma 1. *Let us define $d_t = f_t(\mathbf{x}_t) - r_t - \gamma f_{t+1}(\mathbf{x}_{t+1})$. We have:*

$$\sum d_t^2 \geq (1-\gamma)^2 \sum e_t^2 + (\gamma^2 - \gamma)(e_T^2 - e_0^2). \tag{4.5}$$

Note that the difference between d_t and b_t is that d_t uses $f_{t+1}(\mathbf{x}_{t+1})$ to estimate the long-term reward at step $t+1$ while b_t uses $f_t(\mathbf{x}_{t+1})$.

Proof. Schapire and Warmuth [1996] implicitly showed that $d_t = (f_t(\mathbf{x}) - v_t + v_t - (r_t + \gamma f_{t+1}(\mathbf{x}_{t+1}))) = (e_t - \gamma e_{t+1})$. Squaring both sides and summing over from $t = 0$ to $t = T-1$, we get:

$$\begin{aligned}
\sum d_t^2 &= \sum (e_t - \gamma e_{t+1})^2 \\
&= \sum e_t^2 + \gamma^2 \sum e_{t+1}^2 - 2\gamma \sum e_t e_{t+1} \\
&\geq \sum e_t^2 + \gamma^2 \sum e_{t+1}^2 - \gamma \sum e_t^2 - \gamma \sum e_{t+1}^2 \\
&= (1-\gamma)^2 \sum e_t^2 + (\gamma^2 - \gamma)(e_T^2 - e_0^2).
\end{aligned} \tag{4.6}$$

The first inequality is obtained by applying Young's inequality to $2e_te_{t+1}$ to get $2e_te_{t+1} \le e_t^2 + e_{t+1}^2$. $\square$ $\square$

Lemma 2. *For any $f^* \in \mathcal{F}$, the prediction error $\sum e_t^{*2}$ upper bounds the* BE $\sum b_t^{*2}$ *as follows:*

$$\sum b_t^{*2} \le (1+\gamma)^2 \sum e_t^{*2} + (\gamma+\gamma^2)(e_0^{*2} - e_T^{*2}). \tag{4.7}$$

The proof of Lemma 2 is similar to the one for Lemma 1. We present the proof in Appendix.

Now let us define a measure of the change in predictors between the steps of the online algorithm as $\epsilon_t = f_t(\mathbf{x}_{t+1}) - f_{t+1}(\mathbf{x}_{t+1})$, which is closely related to the online stability condition. The b_t and d_t are then closely related with each other by ϵ_t:

$$\begin{aligned} d_t &= f_t(\mathbf{x}_t) - r_t - \gamma f_{t+1}(\mathbf{x}_{t+1}) - \gamma f_t(\mathbf{x}_{t+1}) + \gamma f_t(\mathbf{x}_{t+1}) \\ &= b_t + \gamma\epsilon_t. \end{aligned}$$

Squaring both sides, we get:

$$\begin{aligned} d_t^2 &= b_t^2 + 2b_t\gamma\epsilon_t + \gamma^2\epsilon^2 \le b_t^2 + b_t^2 + \gamma^2\epsilon_t^2 + \gamma^2\epsilon_t^2 \\ &= 2b_t^2 + 2\gamma^2\epsilon_t^2, \end{aligned} \tag{4.8}$$

where the first inequality is coming from applying Young's inequality to $2b_t\gamma\epsilon_t$ to get $2b_t\gamma\epsilon_t \le b_t^2 + \gamma^2\epsilon_t^2$. We are now ready to state the following main theorem of this paper:

Theorem 15. *Assume a sequence of predictors $\{f_t\}$ is generated by running some online algorithm on the sequence of loss functions $\{l_t\}$. For any predictor $f^* \in \mathcal{F}$, the sum of prediction errors $\sum e_t^2$ can be upper bounded as:*

$$\begin{aligned} (1-\gamma)^2 \sum e_t^2 \le & 2\sum (b_t^2 - b_t^{*2}) + 2\gamma^2 \sum \epsilon_t^2 \\ & + 2(1+\gamma)^2 \sum e_t^{*2} + M, \end{aligned} \tag{4.9}$$

where

$$M = 2(\gamma+\gamma^2)(e_0^{*2} - e_T^{*2}) - (\gamma^2 - \gamma)(e_T^2 - e_0^2).$$

By running a no-regret and online stable algorithm on the loss functions $\{l_t(f)\}$, as $T \to \infty$, the average prediction error is then asymptotically upper bounded by a constant factor of the best possible prediction error in the function class:

$$\lim_{T\to\infty} : \frac{\sum e_t^2}{T} \le \frac{2(1+\gamma)^2}{(1-\gamma)^2} \frac{\sum e_t^{*2}}{T}. \tag{4.10}$$

Proof. Combining Lemma. 1 and Lemma. 2, we have:

$$\begin{aligned} & \sum d_t^2 - 2\sum b_t^{*2} \\ & \ge (1-\gamma)^2 \sum e_t^2 + (\gamma^2 - \gamma)(e_T^2 - e_0^2) \\ & - 2(1+\gamma)^2 \sum e_t^{*2} \end{aligned}$$

$$- 2(\gamma + \gamma^2)(e_0^{*2} - e_T^{*2}). \tag{4.11}$$

Subtracting $2b_t^{*2}$ on both sides of Eq. 4.8, and then summing over from $t = 1$ to $T - 1$, we have:

$$\sum d_t^2 - \sum 2b_t^{*2} \leq 2\sum(b_t^2 - b_t^{*2}) + 2\gamma^2 \sum \epsilon_t^2.$$

Combining the above two inequalities together, we have:

$$\begin{aligned} &2\sum(b_t^2 - b_t^{*2}) + 2\gamma^2 \sum \epsilon_t^2 \\ &\geq (1-\gamma)^2 \sum e_t^2 + (\gamma^2 - \gamma)(e_T^2 - e_0^2) \\ &- 2(1+\gamma)^2 \sum e_t^{*2} - 2(\gamma + \gamma^2)(e_0^{*2} - e_T^{*2}). \end{aligned} \tag{4.12}$$

Rearrange inequality (4.12) and define $M = 2(\gamma+\gamma^2)(e_0^{*2} - e_T^{*2}) - (\gamma^2 - \gamma)(e_T^2 - e_0^2)$, we obtain inequality (4.9).

Assume that the $\bar{f} = \arg\min_{f\in\mathcal{F}} \sum l_t(f)$, then if the online algorithm is no-regret, we have

$$\begin{aligned} &\frac{1}{T}\sum b_t^2 - b_t^{*2} = \frac{1}{T}\sum l_t(f_t) - l_t(f^*) \\ &\leq \frac{1}{T}\sum l_t(f_t) - l_t(\bar{f}) \\ &= \frac{1}{T}\,\text{Regret} \leq 0, \quad T \to \infty. \end{aligned} \tag{4.13}$$

If the online algorithm satisfies the stability condition (Eq. 4.4), we have: $\frac{1}{T}\sum \epsilon_t^2 = 0$ when $T \to \infty$.

Also, since we assume $|f(\mathbf{x})| \leq P$ and $|r| \leq R$, we can see M must be upper bounded by some constant. Hence, we must have $\frac{M}{T} = 0$, as $T \to \infty$.

Under the conditions that the online algorithm is no-regret and satisfies online stability, we get Eq. 4.10 by dividing both sides of Eq. 4.9 by T and taking T to infinity. $\square$

$\square$

Note that in Theorem 15, Eq. 4.9 holds for any $f^* \in \mathcal{F}$, including the f^* that minimizes the prediction error. But note that the one that minimizes prediction error, `PE`, does not necessarily optimize the `BE`, which may lead to an improvement of the bound in Eq. 4.10 in practice. To see this, note that we showed in the proof that for a no-regret algorithm:

$$\frac{1}{T}\sum b_t^2 - b_t^{*2} \leq \frac{1}{T}\,\text{Regret} \leq 0, \text{ as } T \to \infty. \tag{4.14}$$

Hence the limit of $(1/T)\sum(b_t^2 - b_t^{*2})$ may be negative for some $f^* \in \mathcal{F}$, which could lead to a potential decrease in the upper bound of $(1/T)\sum e_t^2$ in Eq. 4.10 and give us a tighter bound in practice.

When $e_t^* = 0, \forall t$, from Theorem 15, it is easy to see that no-regret rate of $(1/T)\sum(b_t^2 - b_t^{*2})$ and the online stability rate of $(1/T)\sum \epsilon_t^2$ together determine the rate of the convergence of $(1/T)\sum e_t^2$.

When $T \to \infty$ and $\gamma \to 1$ (specifically when $\gamma \geq (1/\sqrt{2})$), our upper bound analysis in Eq. 4.10 is asymptotically tighter than the upper bound in Li [2008] (Eq. 12) provided for `RG`, which is a special case of our approach as we demonstrate in the following section. As we will additionally show, a large number of popular no-regret online algorithms also satisfy the online stability condition, broadening the family of algorithms that can be used to learn predictors of long-term rewards.

4.4 Algorithms

Our analysis in Sec. 4.3 provides a reduction from the online prediction of long-term reward to the no-regret online learning setting on a sequence of loss functions $\{l_t(f)\}$ defined in Sec. **??**, which enables us to develop a new set of algorithms. In this section, we give concrete examples of new Bellman Residual algorithms based on well-known no-regret online learning procedures such as Online Gradient Descent (OGD), Online Newton Step (ONS) and the Online variant of Frank Wolfe (OFW). The choice of algorithm depends on the size and sparsity level of features and the available computational budget. For instance, OGD generalizes `RG` and has $O(n)$ computational complexity at every update step which makes it suitable for applications where sampling observations is cheap (e.g., RL for video games). ONS provides a logarithmic no-regret rate and could lead to faster convergence in practice, making it potentially suitable for applications where obtaining samples of observations is expensive (e.g., RL for a physical robot). Finally, OFW introduces sparsity and can be applied to problems where the feature dimension is larger than the number of samples.

Although the analysis in Sec. 4.3 does not place any assumption on predictors $f \in \mathcal{F}$, in practice to achieve the no-regret property on the loss functions $\{l_t(f)\}$, additional assumptions of loss functions (e.g., convexity) are needed. Since we discuss concrete no-regret online algorithms in this section, for $\mathcal{F}$ we focus on vector spaces equipped with inner product. Specifically, we focus on two vector spaces: (1) *Reproducing Kernel Hilbert Spaces* (RKHS) where $f = \sum \alpha_i K(\mathbf{x}_i, \cdot) \in \mathcal{F}$, for some kernel $K(\mathbf{x}, \cdot)$ and (2) spaces consisting of linear functions $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, $\mathbf{w} \in \mathcal{W}$[1]. We now summarize the assumptions that we will use in section:

1. We assume $\mathcal{F}$ (or $\mathcal{W}$) is convex and bounded in a sense that the diameter of $\mathcal{F}$ (or $\mathcal{W}$) is upper bounded as $\max_{f_1,f_2\in\mathcal{F}} \|f_1 - f_2\| \leq D \in \mathbb{R}^+$, where the norm $\|f\|$ is defined by the inner product associated with the function space: $\|f\|^2 = \langle f, f\rangle$;
2. We assume that $\|\mathbf{x}_t\|_2 \leq X$, $\forall t$, $|K(\mathbf{x}_1, \mathbf{x}_2)| \leq K, \forall \mathbf{x}_1, \mathbf{x}_2$, $\|f\| \leq F$, $\forall f \in \mathcal{F}$, and $\|\mathbf{w}\|_2 \leq W, \forall \mathbf{w} \in \mathcal{W}$, where $K \in \mathbb{R}^+, F \in \mathbb{R}^+, W \in \mathbb{R}^+$.

Any prediction $f(\mathbf{x})$ is always bounded, since for RKHS, $f(\mathbf{x})$ is bounded as $|f(\mathbf{x})| \leq \|f\|\|K(\mathbf{x},\cdot)\| \leq F\sqrt{K}$, and for $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, we also have $|f(\mathbf{x})| \leq \|\mathbf{w}\|_2\|\mathbf{x}\|_2 \leq WX$. For notation simplicity, we then simply assume that $f(\mathbf{x})$ is always bounded as $|f(\mathbf{x})| \leq P, P \in \mathbb{R}^+$.

Lemma 3. *With the above assumptions, for any pair of $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, for RKHS, the loss functional $l_t(f)$ is convex and Lipschitz continuous with respect to the norm defined by the inner product $\langle\cdot,\cdot\rangle_K$; for $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, the loss function $l_t(\mathbf{w})$ is convex and Lipischitz continuous with respect to either L_1 norm $\|\cdot\|_1$ or L_2 norm $\|\cdot\|_2$.*

We present the proof of the above lemma in Appendix.

[1]Linear function space is a special case of RKHS. We discuss linear function separately since some online algorithms discussed here only work for linear functions

4.4.1 Gradient-based Approaches

Before diving into the detailed examples of mirror descent and gradient based approaches, we first introduce an important lemma about the stability of one particular online algorithm: *Follow the Regularized Leader* (FTRL). It is well known that gradient-based and mirror descent approaches can be understood in the framework of FTRL. In particular, we only focus on the case where the loss functions are convex and L-Lipschitz continuous with a regularization that is strongly-convex. We refer reader to Shalev-Shwartz [2011] for detailed definitions of convex functions, Lipschitz continuous, and strong convexity.

The update rule of FTRL at step t can be summarized as:

$$f_{t+1} = \arg\min_{f \in \mathcal{F}} \sum_{i=0}^{t} l_i(f) + \frac{1}{\mu} R(f). \tag{4.15}$$

Lemma 4. *For FRTL with convex and L-Lipschitz continuous loss functions $l_t(f)$ and strongly convex regularization function $R(f)$ (with respect to $\|f\|$), we have:*

$$\sum \|f_t - f_{t+1}\| \leq LT\mu. \tag{4.16}$$

Setting $\mu = \frac{1}{\sqrt{T}}$ to achieve no-regret property, then we have:

$$\lim_{T \to \infty} \frac{1}{T} \sum \|f_t - f_{t+1}\| = 0. \tag{4.17}$$

Similar proofs has been shown in [Ross and Bagnell, 2011] and [Saha et al., 2012]. For completeness, we present the proof of the above lemma in Appendix following our notation and problem setting.

Gradient Descent on `BE`

Gradient descent approaches can be understood in the FTRL framework where the convex loss functions in FTRL are replaced by a linear approximation:

$$f_{t+1} = \arg\min_{f \in \mathcal{F}} \sum_{i=0}^{t} \langle g_i, f \rangle + \frac{1}{\mu_t} R(f), \tag{4.18}$$

where $g_t \in \partial l_t(f_t)$ is a sub-gradient of l_t at f_t and $R(f)$ is a strongly convex regularizer.

We first consider the special case where $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ is linear. Note that our loss function is $l_t(\mathbf{w}) = (\mathbf{w}^T\mathbf{x}_t - r_t - \gamma\mathbf{w}^T\mathbf{x}_{t+1})^2$ and its gradient g_t at $\mathbf{w}_t$ is $g_t = (\mathbf{w}_t^T\mathbf{x}_t - r_t - \gamma\mathbf{w}_t^T\mathbf{x}_{t+1})(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})$. Setting the regularization $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, which is 1-strongly convex, we obtain the `RG` algorithm in Baird [1995], where the update step at t is:

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \mu_t(\mathbf{w}_t^T(\mathbf{x}_t - \gamma\mathbf{x}_{t+1}) - r_t)(\mathbf{x}_t - \gamma\mathbf{x}_{t+1}),$$

Note that the linear loss function is convex and Lipschitz continuous ($\|g_t\|_2$ is bounded based on our assumptions that $\|\mathbf{x}\|_2$, $|r|$ and $\|\mathbf{w}\|_2$ are all bounded). Online Gradient Descent (OGD) is no-regret [Zinkevich, 2003] with $\mu_t = 1/\sqrt{T}$ and from Lemma 4, we have:

$$\frac{1}{T} \sum (\mathbf{w}_t^T\mathbf{x}_{t+1} - \mathbf{w}_{t+1}^T\mathbf{x}_{t+1})^2$$

$$
\begin{aligned}
&\leq \frac{1}{T}\sum \|\mathbf{x}_{t+1}\|_2^2\|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2 \\
&\leq X^2\frac{1}{T}\sum \|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2 = 0, \;\; T \to \infty, \qquad (4.19)
\end{aligned}
$$

which exactly satisfies the online stability condition. Hence, RG enjoys the guarantee of our main theorem 15.

Exponentiated Gradient Descent on BE

When we set the regularization $R(\mathbf{w}) = \sum_i \mathbf{w}^i(\log(\mathbf{w}^i) - 1)$, where for vector $\mathbf{w}$, $\mathbf{w}^i$ is the i-th component of $\mathbf{w}$, we generalize RG to *Exponentiated Gradient* (EG) descent as Precup and Sutton [1997] did for TD.

Since we assumed that $\|\mathbf{w}\|_2 \leq W$, then $\|\mathbf{w}\|_1 \leq W'$ for $W' \in \mathbb{R}^+$. Then the regularization $R(\mathbf{w})$ becomes $(1/W')$-strongly-convex and the loss function $l_t(\mathbf{w})$ is Lipschitz continuous with respect to L_1 norm $\|\cdot\|_1$. Solving Eq.4.18, we obtain the update step at t as:

$$
\mathbf{w}_{t+1}^i = \mathbf{w}_t^i \exp\big(-\mu_t(\mathbf{w}_t^T(\mathbf{x}_t - \gamma\mathbf{x}_{t+1}) - r_t)(\mathbf{x}_t^i - \gamma\mathbf{x}_{t+1}^i)\big).
$$

Similar to RG, using Lemma 4 (the norm in Lemma 4 becomes L_1 norm) we can show that EG satisfies our online stability condition:

$$
\begin{aligned}
&\frac{1}{T}\sum(\mathbf{w}_t^T\mathbf{x}_{t+1} - \mathbf{w}_{t+1}^T\mathbf{x}_{t+1})^2 \\
&\leq \frac{1}{T}X^2\sum \|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2 \\
&\leq \frac{1}{T}X^2\sum \|\mathbf{w}_t - \mathbf{w}_{t+1}\|_1^2 = 0, \;\; T \to \infty, \qquad (4.20)
\end{aligned}
$$

and is also no-regret on $\{l_t(\mathbf{w})\}$ when $\mu_t = 1/\sqrt{T}$. Hence, EG descent approach enjoys our main theorem 15.

Gradient Descent in RKHS

Now we consider functions $f(\mathbf{x})$ that belongs to RKHS $\mathcal{H}_K$. For the special case where $R(f) = \frac{1}{2}\langle f, f\rangle_K$, we obtain an RG-style update based on functional gradient descent [Scholkopf and Smola, 2001]:

$$
f_{t+1} := f_t - \mu_t\Big((f_t(\mathbf{x}_t) - r_t - \gamma f_t(\mathbf{x}_{t+1}))(K(\mathbf{x}_t,\cdot) - \gamma K(\mathbf{x}_{t+1},\cdot))\Big). \qquad (4.21)
$$

Similarly, it is straightforward to show that gradient descent in RKHS satisfies our stability condition and no-regret condition when $\mu_t = 1/\sqrt{T}$. Hence, gradient descent in RKHS also enjoys the predictive error guarantees derived in Theorem 15.

4.4.2 Implicit Online Learning

Recently Tamar et al. [2014] has demonstrated implicit online learning for temporal difference method. We consider the same approach for Bellman Residual minimization by applying implicit online learning from Kulis et al. [2010] to the loss functions $\{l_t(f)\}$ and thus provide worst-case guarantees. Specifically at iteration t, f_{t+1} is computed implicitly as:

$$\begin{aligned} f_{t+1} &= \arg\min_{f\in\mathcal{F}} \mathcal{D}_R(f, f_t) + \mu_t l_t(f) \\ &= \arg\min_{f\in\mathcal{F}} \mathcal{D}_R(f, f_t) + \mu_t (f(x_t) - \gamma f(x_{t+1}) - r_t)^2, \end{aligned} \tag{4.22}$$

where $\mathcal{D}_R$ is a Bregman divergence. Saha et al. [2012] show that when $\mu_t = 1/\sqrt{t}$, the generating function $R(f)$ is positive and strongly-convex, and the loss function $l_t(f)$ is convex and Lipschitz continuous, implicit online learning is shown to be no-regret and also satisfies Eq. 4.17.

Implicit Online Gradient Descent

Particularly, we first consider $f = \sum \alpha_i K(\mathbf{x}_i, \cdot)$ in RKHS. Setting $R(f) = \frac{1}{2}\langle f, f\rangle_k$, we have $\mathcal{D}_R(f, f_t) = \frac{1}{2}\|f - f_t\|^2$. Then solving Eq. 4.22, we obtain the following update rule:

$$f_{t+1} := f_t - \frac{\mu_t}{1 + \mu_t\|K(\mathbf{x}_t, \cdot) - \gamma K(\mathbf{x}_{t+1}, \cdot)\|^2}(f_t(\mathbf{x}_t) - \gamma f_t(\mathbf{x}_{t+1}) - r_t)(K(\mathbf{x}_t, \cdot) - \gamma K(\mathbf{x}_{t+1}, \cdot)).$$

When considering linear function $f(x) = \mathbf{w}^T\mathbf{x}$ and $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, we obtain a similar update step:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\mu_t}{1 + \mu_t\|\mathbf{x}_t - \gamma\mathbf{x}_{t+1}\|^2}(\mathbf{w}_t^T(\mathbf{x}_t - \gamma\mathbf{x}_{t+1}) - r_t)(\mathbf{x}_t - \gamma\mathbf{x}_{t+1}).$$

As we will show in the experiments, compared to RG (OGD on BE), implicit OGD on BE is less sensitive to the choice of step-size, which enables us to set large step-size to achieve faster convergence for $(1/T)\sum e_t^2$. This phenomenon is also observed by Tamar et al. [2014] when they compare implicit temporal difference to the original TD algorithm [Sutton and Barto, 1998a].

4.4.3 Online Newton Step

We also analyze an online second-order method: the Online Newton Step (ONS) [Hazan et al., 2006] for online prediction of long-term reward. For ONS, we only focus on linear function approximation $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. We slightly adapt the ONS for our loss function $l_t(\mathbf{w})$. We first present the following lemma:

Lemma 5. *For loss function $l_t(\mathbf{w}) = (\mathbf{w}^T\mathbf{x}_t - r_t - \gamma\mathbf{w}^T\mathbf{x}_{t+1})^2$, there exists a $\lambda \in \mathbb{R}^+$, such that for all $\mathbf{w}$ and $\mathbf{w}'$:*

$$l_t(\mathbf{w}) \geq l_t(\mathbf{w}') + \nabla l_t(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}') + \frac{\lambda}{2}(\mathbf{w} - \mathbf{w}')^T\nabla l_t(\mathbf{w}')\nabla l_t(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}').$$

We present the proof of the above lemma in appendix.

With λ, then the iterative update rule for ONS is:

$$\mathbf{w}_t = \Pi_{\mathcal{W}}^{A_{t-1}}\left(\mathbf{w}_{t-1} - \frac{1}{\lambda}A_{t-1}^{-1}\nabla l_{t-1}(\mathbf{w}_{t-1})\right), \tag{4.23}$$

where $A_t = \sum_{i=0}^{t} \nabla l_t(\mathbf{w}_t)\nabla l_t(\mathbf{w}_t)^T + \epsilon I_n$, $\epsilon \in R^+$, and $\Pi_{\mathcal{W}}^{A_t}$ is a projection to $\mathcal{W}$ with the norm induced by A_t: $\Pi_{\mathcal{W}}^{A_t}(\mathbf{y}) = \arg\min_{\mathbf{w}\in\mathcal{W}}(\mathbf{w}-\mathbf{y})^T A_t(\mathbf{w}-\mathbf{y})$, which makes this projection operator $\Pi_{\mathcal{W}}^{A_t}$ not trivial and equal to solving a convex program usually.

Since $\|\mathbf{x}\|_2 \leq X$, $\|\mathbf{w}\|_2 \leq W$ and $|r| \leq R$, we have $\|\nabla l_t(\mathbf{w})\|_2 \leq G$, for $G \in \mathbb{R}^+$. The following lemma shows that ONS satisfies the our online stability condition:

Lemma 6. *The sequence $\{\mathbf{w}_t\}$ generated by ONS satisfies the online stability condition:*

$$\frac{1}{T}\sum(\mathbf{w}_t^T\mathbf{x}_{t+1} - \mathbf{w}_{t+1}^T\mathbf{x}_{t+1})^2 \leq \frac{1}{T}\frac{X^2}{G^2\lambda^2}n\log(T+1) = 0, \;\; T \to \infty. \tag{4.24}$$

The proof borrows ideas from Hazan et al. [2006] and is presented in the Appendix. Note that the convergence rate of $\frac{1}{T}\sum(f_t(\mathbf{x}_{t+1}) - f_{t+1}(\mathbf{x}_{t+1}))^2$ is $O(\log T/T)$, which is the same as the no-regret rate of ONS.

4.4.4 Projection Free Online Learning

We analyze the Online Frank Wolfe (OFW) [Hazan and Kale, 2012] for online prediction of long-term reward. Previously introduced methods, including OGD, EG, OGD in RKHS, Implicit OGD, and implicit OGD in RKHS, usually need a projection operation in each update step if the newly updated predictor is out of its pre-defined convex set $\mathcal{F}$, although in many cases, projection operations are simple[2]. OFW is a projection-free online method and every step involves solving a (typicaly very simple) linear programming problem. Again, we restrict our analysis to linear functions $f = \mathbf{w}^T\mathbf{x}$ for OFW. Without loss of generality, we assume $l_t(\mathbf{w})$ is L-Lipschitz continuous with some $L \in \mathbb{R}^+$ (Lemma 3).

Applying the adversarial variant of OFW to the sequence of loss functions $\{l_t(f)\}$, we have the following iterative update step:

$$\begin{aligned}\mathbf{v}_t &= \arg\min_{\mathbf{w}\in\mathcal{W}} \nabla F_t(\mathbf{w}_t)^T\mathbf{w},\\ \mathbf{w}_{t+1} &= (1 - t^{-\alpha})\mathbf{w}_t + t^{-a}\mathbf{v}_t,\end{aligned} \tag{4.25}$$

where $\alpha = \frac{1}{4}$ and $F_t(\mathbf{w})$ is computed as:

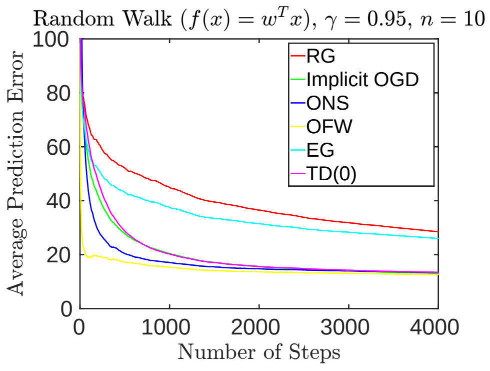
$$\begin{aligned}F_t(\mathbf{w}) &= \frac{1}{t+1}\sum_{i=0}^{t}\hat{l}_i(\mathbf{w})\\ &= \frac{1}{t+1}\sum_{i=0}^{t}(\nabla l_i(\mathbf{w}_i)^T\mathbf{w} + \sigma_i\|\mathbf{w} - \mathbf{w}_0\|_2^2)\\ &= \frac{1}{t+1}\sum_{i=0}^{t}\Big((\mathbf{w}_i^T\mathbf{x}_i - r_i - \gamma\mathbf{w}_i^T\mathbf{x}_{i+1})(\mathbf{x}_i - \gamma\mathbf{x}_{i+1})^T\mathbf{w} + \sigma_i\|\mathbf{w} - \mathbf{w}_0\|_2^2\Big),\end{aligned}$$

where $\sigma_t = (L/D)t^{-1/4}$, $\mathbf{w}_0$ is the initialization.

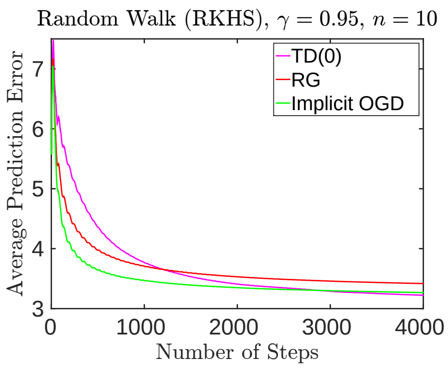
The online stability of OFW can be shown easily:

$$\frac{1}{T}\sum(\mathbf{w}_t^T\mathbf{x}_{t+1} - \mathbf{w}_{t+1}^T\mathbf{x}_{t+1})^2 \leq \frac{1}{T}X^2\sum\|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2$$

[2]Usually, when $\mathcal{W}$ is defined as $\{\mathbf{w} : \|\mathbf{w}\|_2 \leq W\}$, an L_2 projection to such $\mathcal{W}$ is easy and can be implemented in $O(n)$. The same for L_2 projection in RKHS when $\mathcal{F} = \{f : \|f\| \leq F\}$.



(a) Linear



(b) RKHS

Figure 4.1: Convergence of prediction error for Random Walk with linear function approximation (left) and RKHS (right).

$$\leq \frac{1}{T} X^2 \sum t^{-2\alpha} \|\mathbf{w}_t - \mathbf{v}_t\|_2^2 \leq \frac{1}{T} X^2 D^2 \sum t^{-2\alpha} = 0, \;\; T \to \infty.$$

The first inequality comes from Cauchy-Schwartz inequality and the assumption that $\|\mathbf{x}\|_2 \leq X$. The last equality follows from the fact that $2\alpha > 0$, and $\frac{1}{T} \sum_{t=1}^{T} t^{-\xi} = 0$, when $\xi > 0$ and $T \to \infty$.

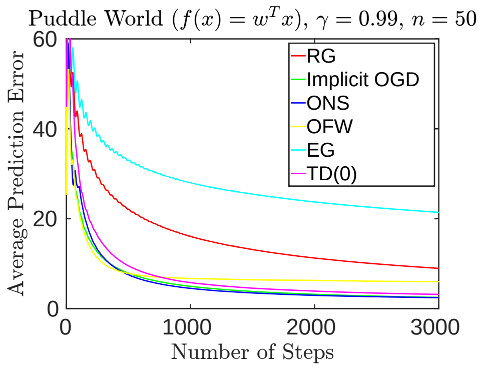
Note that the output of the OFW $\mathbf{w}_t$ is sparse when $\mathcal{W}$ is defined as $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq W'\}$ for some $W' \in \mathbb{R}^+$ and $\mathbf{w}_0$ is initialized to the origin or any corner point of $\mathcal{W}$. This is because the output $\mathbf{v}_t$ of Eq. 4.25 will be always one of the corner points of $\mathcal{W}$ and $\mathbf{w}_t$ hence is a linear combination of corner points $\{\mathbf{w}_0, \mathbf{v}_0 ..., \mathbf{v}_{t-1}\}$, leading to the fact that $\mathbf{w}_t$ can have at most $(t+1)$ non-zero entries. Also, if $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq W'\}$, Eq. 4.25 can be implemented in $O(n)$ as follows: (1) find the entry i in $\nabla F_t(\mathbf{w}_t)$ that has the maximum absolute value; (2) set $\mathbf{v}_t^i$ (the i'th entry) to be $-sign(\nabla F_t(\mathbf{w}_t)^i) W'$, and all other entries in $\mathbf{v}_t$ to zero.

When $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq W\}$, we cannot achieve sparsity, but Eq. 4.25 can still be implemented in $O(n)$ by setting $\mathbf{v}_t = \frac{-\nabla F_t(\mathbf{w}_t)}{\|\nabla F_t(\mathbf{w}_t)\|_2} W$.

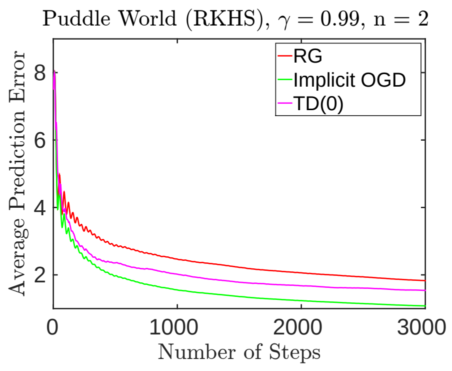
4.5 Experiment

We applied these algorithms to three simulated policy evaluation problems: (1) the Random Walk problem with a ring chain, which is a variant of the Hall problem introduced by Baird [1995], (2) PuddleWorld adopted from Sutton and Barto [1998a] and (3) Helicopter Hover using the simulator from Coates et al. [2008]. We also compare the above algorithms to standard TD(0) [Sutton and Barto, 1998a].

Random Walk The state space of the Random Walk task has $N = 50$ states. The states are linked together as a ring without any terminal states. At time step t, from any state on the ring, the transition probability of moving clockwise is $0.5/\sqrt{t}$ and the probability of moving counterclockwise is $(1 - 0.5/\sqrt{t})$. Note that the transition probability changes over

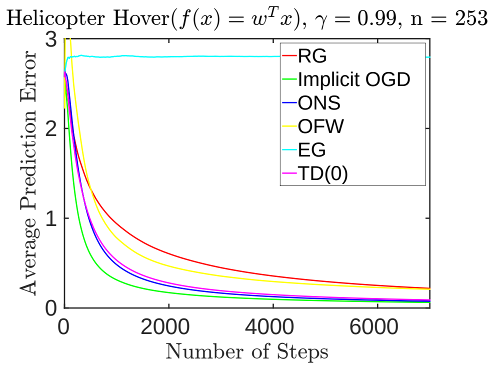


(a) Linear



(b) RKHS

Figure 4.2: Convergence of prediction error for Puddle World with linear function approximation (left) and RKHS (right).



(a) Linear

Figure 4.3: Convergence of prediction error for Helicopter Hover with linear function approximation

time. We randomly generate a feature vector $\mathbf{x} \in \mathbb{R}^{10}$ and assign it to a state. All rewards are uniformly sampled from $[-3, 3]$. In this problem, we tested both RKHS (Fig. 4.1b) and linear function approximation (Fig. 4.1a). For RKHS, we use a RBF Kernel with bandwidth 0.2, qualitatively chosen for the best performance in terms of prediction error.

PuddleWorld In the PuddleWorld scenario, the state space is a unit square with "puddles" and the agent's state is represented by its x and y coordinates. In each episode, the agent's starting state is uniformly sampled in the region $[0, 0.2] \times [0, 0.2]$. The policy at each time step selects to go north or east with probability 0.5 each. For each step, the reward is -1 if the agent does not step in a puddle, and the reward decreases quadratically as the agent steps into the puddles. The terminal region is defined as $x + y \geq 1.9$ (upper right corner of the square), which has reward 0. For linear function approximation, we used 50 RBF features ($\mathbf{x} \in \mathbb{R}^{50}$) of bandwidth 0.2, whose centers were uniformly distributed in the state space (Fig. 4.2a). For RKHS, we again use a RBF kernel bandwidth width 0.2 (Fig. 4.2b).

Helicopter Hover The helicopter simulator has a continuous 21-dimensional state space and a continuous 4-dimensional control. The reward is equal to the negative of the quadratic deviation to the targeted hover state. To generate sequence of states, we apply an LQR controller using linearized dynamics around the target hover state. We additionally corrupt the dynamics simulation with noise sampled from a Gaussian distribution. We used a degree-two polynomial feature that maps an original 21-dimensional state to a feature vector $\mathbf{x} \in \mathbb{R}^{253}$ (Fig. 4.3a) and attempt to predict the long-term cost-to-go.

Analysis of results We fixed $\mathtt{TD}(0)$'s step-size but a wide range of step-sizes were tried, and the best choice in terms of prediction error was used for $\mathtt{TD}(0)$. For $\mathtt{RG}$, implicit OGD, and EG, we set the step-size to $c/\sqrt{t}$, where c is a constant. We also tried a range of c and chose the one that leads to the best performance. For all algorithms, we provided the same random initialization. All the results are computed by averaging over 100 random trials. As we can see from Fig. 4.3, ONS and implicit OGD give good convergence speed in general. Implicit OGD performed well with both RKHS and linear function approximation. Throughout the experiments, we found that implicit OGD was able to use a larger c to speed up convergence while still maintaining good stability. Surprisingly, our experimental results clearly show that our approaches have the possibility to achieve smaller prediction error than $\mathtt{TD}(0)$ (e.g., Fig. 4.2b, bottom). This runs counter to the fact that the upper bound of prediction error provided by our analysis in Sec. 4.3 is looser than the upper bound of prediction error of $\mathtt{TD}(0)$ from both Li [2008] and Schapire and Warmuth [1996]. Though our analysis is more general, further investigation is needed to tighten the worst-case bounds on our approach.

Chapter 5

Policy Optimization: Contrasting Exploration in Action and Parameter Space

Black-box optimizers that explore in parameter space have often been shown to outperform more sophisticated action space exploration methods developed specifically for the reinforcement learning problem. In this chapter, we examine these black-box methods closely to identify situations in which they are worse than action space exploration methods and those in which they are superior. Through simple theoretical analyses, we prove that complexity of exploration in parameter space depends on the dimensionality of parameter space, while complexity of exploration in action space depends on both the dimensionality of action space and horizon length. This is also demonstrated empirically by comparing simple exploration methods on several model problems, including Contextual Bandit, Linear Regression and Reinforcement Learning in continuous control.

The contributions in this chapter were first introduced in the joint work with Anirudh Vemula and Drew Bagnell [Vemula et al., 2019].

5.1 Introduction

Model-free policy search is a general approach to learn parameterized policies from sampled trajectories in the environment without learning a model of the underlying dynamics. These methods update the parameters such that trajectories with higher returns (or total reward) are more likely to be obtained when following the updated policy [Kober et al., 2013]. The simplicity of these approaches have made them popular in Reinforcement Learning (RL).

Policy gradient methods, such as REINFORCE [Williams, 1992] and its extensions [Bagnell et al., 2004, Kakade, 2002, Schulman et al., 2015a, Silver et al., 2014], compute an estimate of a direction of improvement from sampled trajectories collected by executing a stochastic policy. In other words, these methods rely on randomized exploration in action space. These methods then leverage the Jacobian of the policy to update its parameters to increase the probability of good action sequences accordingly. Such a gradient estimation algorithm can be considered a combination of a *zeroth-order* approach and a *first-order* approach: (1) it never exploits the slope of the reward function or dynamics, with respect to actions, but rather relies only on random exploration in action space to discover potentially good sequences of actions; (2) however, it *exploits* the first order information of the parameterized policy for updating the policy's parameters. Note that the chance of finding a sequence of actions resulting in high total reward decreases (as much as exponentially [Kakade and Langford, 2002]) as the horizon length increases and thus policy gradient methods often exhibit high variance and a resulting large sample complexity [Peters and Schaal, 2008, Zhao et al., 2011].

Black-box policy search methods, on the other hand, seek to directly optimize the total reward in the space of parameters by employing , e.g., finite-difference-like methods to compute estimates of the gradient with respect to policy parameters [Bagnell and Schneider, 2001, Heidrich-Meisner and Igel, 2008, Mania et al., 2018, Mannor et al., 2003, Salimans et al., 2017, Sehnke et al., 2010, Tesch et al., 2011]. Intuitively, these methods rely on exploration in parameter space: by searching in the parameter space, these methods may discover an improvement direction. Note that these methods are fully zeroth-order, *i.e.*, they exploit no first-order information of the parameterized policy, the reward, or the dynamics. Although policy gradient methods leverage *more* information, notably the Jacobian of the action with respect to policy, black-box policy search methods have at times demonstrated better empirical performance (see the discussion in [Kober et al., 2013, Mania et al., 2018]). These perhaps surprising results motivate us to analyze: *In what situations should we expect parameter space policy search methods to outperform action space methods?*

To do so, we leverage prior work in zeroth-order optimization methods. In the convex setting, [Agarwal et al., 2010, Flaxman et al., 2005, Nesterov and Spokoiny, 2017] showed that one can construct gradient estimates using zeroth order oracles and derived upper bounds on the number of samples needed. But for most RL tasks, the return as a function of parameters, or action sequence, is highly non-convex [Sutton and Barto, 1998b]. Hence we focus on the non-convex setting and analyze convergence to stationary points. Ghadimi and Lan [2013], Nesterov and Spokoiny [2017] studied zeroth order non-convex optimization by providing upper bounds on the number of samples needed to close in on a stationary point. Computing lower bounds in zeroth order non-convex optimization is still an open problem [Carmon et al., 2017a,b].

In our work, we extend the analysis proposed in [Ghadimi and Lan, 2013] to the policy search setting and analyze the sample complexity of parameter and action space exploration methods in policy search. We begin with a degenerate, one-step control problem of online linear regression with partial feedback, [Flaxman et al., 2005], where the objective is to learn the parameters of the linear regressor without access to the true scalar regression targets. We show that for parameter space exploration methods, to achieve ϵ-optimality, requires $O(b^2/\epsilon^4)$ samples, where b is the input feature dimensionality. By contrast, an action space exploration method requires $O(1/\epsilon^4)$ many samples with a sample complexity *independent* of input feature dimensionality b. This is tested empirically on two simple tasks: Bandit Multi-class learning on MNIST with policies parameterized by convolutional neural networks which can be seen as a Contextual Bandit problem with rich observations, and Online Linear Regression with partial information. The results demonstrate action space exploration methods outperform parameter space methods when the parameter dimensionality is substantially larger than action dimensionality.

We present similar analysis for the multi-step control problem of model-free policy search in reinforcement learning, [Kober et al., 2013], by considering the objective of reaching ϵ-close to a stationary point in the sense that $\|\nabla J(\theta)\|_2^2 \leq \epsilon$ for the non-convex objective $J(\theta)$. Our results show that, under certain assumptions, parameter space exploration methods need $\mathcal{O}(\frac{d^2}{\epsilon^3})$ samples to reach ϵ close to a stationary point, where d is the policy parameter dimensionality. On the other hand, action space exploration methods need $\mathcal{O}(\frac{p^2 H^4}{\epsilon^4})$ samples to achieve the same objective, where p is the action dimensionality and H is the horizon length of the task. This shows that action space exploration methods have a dependence on the horizon length H while parameter space exploration methods depend only on parameter space dimensionality d. Ongoing work by Tu and Recht [2018] demonstrated through asymptotic lower bounds that the dependence of sample complexity of action space exploration methods on horizon H is unavoidable in the LQR setting. This is tested empirically on popular RL benchmarks from OpenAI gym [Brockman et al., 2016b], and the results show that as horizon length increases, parameter space methods outperform action space exploration methods. This matches the intuition and results presented in recent works like [Bagnell and Schneider, 2001, Mania et al., 2018, Salimans et al., 2017, Szita and Lörincz, 2006, Tesch et al., 2011] that show parameter space black-box policy search methods outperforming state-of-the-art action space methods for tasks with long horizon lengths.

In summary, our analysis and experimental results suggests that the complexity of exploration in action space depends on both the dimensionality of action space and *horizon*, while the complexity of exploration in parameter space solely depends on dimensionality of parameter space, providing a natural way to trade-off between these approaches.

5.2 Preliminaries

5.2.1 Multi-step Control: Reinforcement Learning

We consider the problem setting of model-free policy search with the goal of minimizing sum of costs (or maximizing sum of rewards) over a fixed, finite horizon H. In reinforcement learning (RL), this is typically formulated using Markov Decision Processes (MDP) [Sutton and Barto, 1998b]. Denote the state space of the MDP as $\mathcal{S} \subset \mathbb{R}^b$, action space as $\mathcal{A} \subset \mathbb{R}^p$, transition probabilities as $\mathbb{P}_{sa} = \mathbb{P}(\cdot|s,a)$ (which is the distribution of next state after executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$), an initial state distribution μ, and a cost function $c(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Note that the cost can be interpreted as negative of the reward. In addition to this, we assume a restricted class of deterministic, stationary policies Π parameterized by $\theta \in \mathbb{R}^d$ where each $\pi(\theta, \cdot) \in \Pi$ is differentiable at all θ and is a mapping from $\mathcal{S}$ to $\mathcal{A}$, i.e. $\pi(\theta, \cdot) : \mathcal{S} \to \mathcal{A}$. The distribution of states at timestep t induced by running the policy $\pi(\theta, \cdot)$ until and including t, is defined $\forall s_t : d^t_{\pi_\theta}(s_t) = \sum_{\{s_i\}_{i \leq t-1}} \mu(s_0) \prod_{i=0}^{t-1} \mathbb{P}(s_{i+1}|s_i, a_i = \pi(\theta, s_i))$, where by definition $d^0_{\pi_\theta}(s) = \mu(s)$ for any π. We define the value function $V^t_{\pi_\theta}(s)$ for $t \leq H - 1$ as

$$V^t_{\pi_\theta}(s) = \mathbb{E}[\sum_{i=t}^{H} c(s_i, \pi(\theta, s_i))|s_t = s]$$

and state-action value function $Q^t_{\pi_\theta}(s, a)$ as

$$Q^t_{\pi_\theta}(s, a) = c(s, a) + \mathbb{E}_{s' \sim \mathbb{P}_{sa}}[V^{t+1}_{\pi_\theta}(s')]$$

Throughout this work, we assume the total cost is upper bounded by a constant, i.e., $\sup_{c_1, \ldots, c_T} \sum_t c_t \leq \mathcal{Q} \in \mathbb{R}^+$, to prevent confounding due to just a change in the scale of total costs. We have then that $Q^t_{\pi_\theta}$ is upper bounded by a constant $\mathcal{Q}$ for all t and θ.

We seek to minimize the performance objective given by $J(\theta) = \mathbb{E}_{s \sim \mu}[V^0_{\pi_\theta}(s)]$. Given this objective, the optimization problem can be formulated as:

$$\min_\theta J(\theta) \tag{5.1}$$

The goal is to find parameters θ^* that minimize the expected sum of costs $J(\theta)$, given no access to the underlying dynamics of the environment other than samples from the distribution $\mathbb{P}_{sa}$ by executing the policy $\pi(\theta, \cdot)$. However, the objective $J(\theta)$ can be highly nonconvex and finding a global minima could be intractable. Thus, in this work, we hope to find a stationary point θ^* of the objective $J(\theta)$, *i.e.* a point where $\nabla_\theta J(\theta) \approx 0$.

5.2.2 One-Step Control: Online Linear Regression with Partial Information

The online linear regression problem is defined as follows: We denote $\mathcal{S} \subset \mathbb{R}^b$ as the feature space, and $\Theta \subset \mathbb{R}^d = \mathbb{R}^b$ as the linear policy parameter space where each $\theta \in \Theta$ represents a policy $\pi(\theta, s) = \theta^\top s$. Online linear regression operates in an adversarial online learning fashion: every round i, nature presents a feature vector $s_i \in \mathcal{S}$, the learner makes a decision by choosing a policy $\theta_i \in \Theta$ and predicts the scalar action $\hat{a}_i = \theta_i^\top s_i$; nature then reveals the loss $(\hat{a}_i - a_i)^2 \in \mathbb{R}^+$, which is just a scalar, to the learner, where a_i is ground truth selected by nature and is never revealed to the learner. We do not place any statistical assumption on the nature's process of generating feature vector s_i and ground truth a_i, which could be completely adversarial. Other than the adversarial aspect of the problem, note that the above setup is a special setting of RL with horizon $H = 1$, linear policy $\theta^\top s_i$, one-dimension action space, and a cost function $c_i(\theta) = (\theta^\top s_i - a_i)^2$. In this setting, we consider the *regret* with respect to the optimal solution in hindsight,

$$\text{Regret} = \sum_{i=1}^{T} c_i(\theta_i) - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star) \tag{5.2}$$

5.3 Online Linear Regression with Partial Information

5.3.1 Exploration in Parameter Space

We can apply a zeroth-order online gradient descent algorithm for the sequence of loss functions $\{c_i\}_{i=1}^{T}$, which is summarized in Algorithm 4. The main idea is to add random noise u, sampled from a unit sphere in b-dim space $\mathbb{S}_b$, to the parameter θ, and querying loss at $\theta + \delta u$ for some $\delta > 0$. Using the received loss $c_i(\theta + \delta u)$, one can form an estimation of $\nabla_\theta c_i(\theta)$ as $\frac{c_i b}{\delta} u$ [Flaxman et al., 2005].

Algorithm 4 Random Search in Parameter Space (BGD Flaxman et al. [2005])

```
1: **Input:** $\alpha \in \mathbb{R}^+$, $\delta \in \mathbb{R}^+$.
2: Learner initializes $\theta_1 \in \Theta$.
3: **for** $i = 1$ to $T$ **do**
4:     Learner samples $u \sim \mathbb{S}_b$.
5:     Learner chooses predictor $\theta'_i = \theta_i + \delta u$.
6:     Learner only receives loss signal $c_i(\theta'_i)$.
7:     Learner update: $\theta'_{i+1} = \theta_i - \alpha \frac{c_i b}{\delta} u$.
8:     Projection $\theta_{i+1} = \arg\min_{\theta \in \Theta} \|\theta'_{i+1} - \theta\|_2^2$.
9: **end for**
```

5.3.2 Exploration in Action Space

The *key difference* between exploration in action space and exploration in parameter space is that we are going to leverage our knowledge of the policy $\pi(\theta, s) = \theta^\top s$. Since we design the policy class, we can compute its *Jacobian* with respect to its parameters θ without interaction with the environment. The Jacobian of the policy gives us a locally linear relationship between a small change in parameter and the resulting change in policy's action space. The main idea then in this approach is to explore with randomization in action space, and then leverage the Jacobian of the policy to update the parameters θ accordingly so that the policy's output moves towards better actions. Intuitively, we expect that random exploration in action space will result in smaller regret, as in our setting the action space is just 1-dimensional, while the parameter space is b-dimensional. The approach is summarized in Algorithm 5. Denote $\ell_i = (\hat{a}_i - a_i)^2$ and $\hat{a}_i = \pi(\theta_i, s_i) = \theta_i^\top s_i$. The main idea is that we can compute $\nabla_\theta c_i(\theta_i)$ via a chain rule as $\nabla_\theta c_i(\theta_i) = \frac{\partial \ell_i}{\partial \hat{a}_i} \nabla_\theta \pi(\theta_i, s_i)$. Note that $\nabla_\theta \pi(s_i, \theta_i) = \nabla_\theta \theta_i^\top s_i = s_i$ is the Jacobian of the policy to which we have full access. We then use zeroth order approximation method to approximate $\partial \ell_i / \partial \hat{a}_i$ at $\hat{a}_i = \pi(\theta_i, s_i)$.

5.3.3 Analysis

We analyze the regret of the exploration in parameter space algorithm (Alg. 4) and the exploration in action space algorithm (Alg. 5) in this section. For analysis, we assume that Θ is bounded, i.e., $\sup_{\theta \in \Theta} \|\theta\|_2 \leq C_\theta \in \mathbb{R}^+$, $\mathcal{S}$ is bounded, i.e., $\sup_{s \in \mathcal{S}} \|s\|_2 \leq C_s \in \mathbb{R}^+$, and the ground truth a_i is bounded, i.e., $|a_i| \leq C_a$ for any i. Under the above assumptions, we can make sure that the loss is bounded as well, $(\theta^\top s - a)^2 \leq C \in \mathbb{R}^+$. The loss function is also Lipschitz continuous with Lipschitz constant $L \leq (C_\theta C_s + C_a) C_s$. We call these constants C_s, C_θ, and C_a as problem dependent constants, which are independent of feature

Algorithm 5 Random Search in Action Space

```
1: Input: $\alpha \in \mathbb{R}^+$, $\delta \in \mathbb{R}^+$.
2: Learner initializes $\theta_1 \in \Theta$.
3: for $i = 1$ to $T$ do
4:     Learner receives feature $s_i$.
5:     Learner samples $e$ uniformly from $\{-1, 1\}$.
6:     Learner makes a prediction $\hat{a}_i = \theta_i^\top s_i + \delta e$
7:     Learner only receives loss signal $c_i = (\hat{a}_i - a_i)^2$.
8:     Learner update: $\theta'_{i+1} = \theta_i - \alpha \frac{c_i e}{\delta} s_i$.
9:     Projection $\theta_{i+1} = \arg\min_{\theta \in \Theta} \|\theta'_{i+1} - \theta\|_2^2$.
10: end for
```

dimension b and number of rounds T. In regret bounds, we absorb problem dependent constants into $\mathcal{O}$ notations, but the bounds will be explicit in b and T. The theorem below presents the average regret analysis for these methods,

Theorem 16. *After T rounds, with $\alpha = \frac{C_\theta \delta}{b(C^2+C_s^2)\sqrt{T}}$ and $\delta = T^{-0.25}\sqrt{\frac{C_\theta b(C^2+C_s^2)}{2L}}$, Alg. 4 incurs average regret:*

$$\frac{1}{T}(\mathbb{E}[\sum_{i=1}^{T} c_i(\theta_i)] - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star)) \leq \mathcal{O}(\sqrt{b}T^{-\frac{1}{4}}), \tag{5.3}$$

and with $\alpha = \frac{C_\theta \delta}{(C^2+1)C_s\sqrt{T}}$ and $\delta = T^{-0.25}\sqrt{\frac{C_\theta (C^2+1)C_s}{2C}}$, Alg. 5 incurs average regret:

$$\frac{1}{T}(\mathbb{E}[\sum_{i=1}^{T} c_i(\theta_i)] - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star)) \leq \mathcal{O}(T^{-\frac{1}{4}}), \tag{5.4}$$

for any $\theta \in \Theta$.

The above regret analysis essentially shows that exploration in action space delivers a regret bound that is independent of parameter space dimension b, while the regret of the exploration in parameter space algorithm will have explicit polynomial dependency on feature dimension b. Converting the regret bounds to sample complexity bounds, we have that for any $\epsilon \in (0, 1)$, to achieve ϵ-average regret, Alg. 4 needs $\mathcal{O}(\frac{b^2}{\epsilon^4})$ many rounds, while Alg. 5 requires $\mathcal{O}(1/\epsilon^4)$ many rounds.

Note that in general if we have a multivariate regression problem, i.e., $a \in \mathbb{R}^p$, regret of Algorithm 5 will depend on $\sqrt{p}$ as well. But from our extreme case with $p = 1$, we clearly demonstrate the sharp advantage of exploration in action space: *when the action space's dimension is smaller than the dimension of parameter space*, we should prefer the strategy of exploration in action space.

5.4 Reinforcement Learning

In this section, we study exploration in parameter space versus exploration in action space for multi-step control problem of model-free policy search in RL. As explained in Section 5.2, we are interested in rates of convergence to a stationary point of $J(\theta)$.

5.4.1 Exploration in Parameter Space

The objective defined in Section 5.2.1 can be optimized directly over the space of parameters $\mathbb{R}^d$. Since we do not use first-order (or gradient) information about the objective, this is equivalent to derivative-free (or zeroth-order) optimization with noisy function evaluations. More specifically, for a parameter vector θ, we can execute the corresponding policy $\pi(\theta, \cdot)$ in the environment, to obtain a noisy estimate of $J(\theta)$. This noisy function evaluation can be used to construct a gradient estimate and an iterative stochastic gradient descent approach can be used to optimize the objective. An algorithm that closely follows the ones proposed in [Agarwal et al., 2010, Mania et al., 2018] and optimizes over the space of parameters is shown in Algorithm 6. Since we are working in episodic RL setting, we can use a two-point estimate to form a gradient estimation (Line 7 & 8 in Alg. 6), which in general will reduce the variance of gradient estimation [Agarwal et al., 2010], compared to one-point estimates. We will analyze the finite rate of convergence of Algorithm 6 to a stationary point

Algorithm 6 Policy Search in Parameter Space

```
1: Input: Learning rate $\alpha \in \mathbb{R}^+$, standard deviation of exploration noise $\delta \in \mathbb{R}$
2: Initialize parameters $\theta_1 \in \mathbb{R}^d$
3: for $i = 1$ to $T$ do
4:     Sample $u \sim \mathbb{S}_d$ , a $d$-dimensional unit sphere
5:     Construct parameters $\theta_i + \delta u$, $\theta_i - \delta u$
6:     Execute policies $\pi(\theta_i + \delta u, \cdot), \pi(\theta_i - \delta u, \cdot)$
7:     Obtain noisy estimates of the objective $J_i^+ = J(\theta_i + \delta u) + \eta_i^+$ and $J_i^- = J(\theta_i - \delta u) + \eta_i^-$
       where $\eta_i^+, \eta_i^-$ are zero mean random i.i.d noise
8:     Compute gradient estimate $g_i = \frac{d(J_i^+ - J_i^-)}{2\delta} u$
9:     Update $\theta_{i+1} = \theta_i - \alpha g_i$
10: end for
```

of the non-convex objective $J(\theta)$. First, we will lay out the assumptions and then present the convergence analysis.

Assumptions and Analysis To analyze convergence to stationary point of a nonconvex objective, we make several assumptions about the objective. Firstly, we assume that $J(\theta)$ is differentiable with respect to θ over the entire domain. We also assume that $J(\theta)$ is G-lipschitz and L-smooth, i.e. for all $\theta_1, \theta_2 \in \mathbb{R}^d$, we have $|J(\theta_1) - J(\theta_2)| \leq G\|\theta_1 - \theta_2\|$ and $\|\nabla_\theta J(\theta_1) - \nabla_\theta J(\theta_2)\| \leq L\|\theta_1 - \theta_2\|$. Note that these assumptions are similar to the assumptions made in other zeroth-order analysis works, [Agarwal et al., 2010, Duchi et al., 2015, Flaxman et al., 2005, Ghadimi and Lan, 2013, Nesterov and Spokoiny, 2017, Shamir, 2013].

Our analysis is along the lines of works like [Ghadimi and Lan, 2013, Nesterov and Spokoiny, 2017] that also analyze the convergence to stationary points in zeroth order nonconvex optimization. The general strategy is to first construct a smoothed version of the objective $J(\theta)$, denoted as $\hat{J}(\theta) = \mathbb{E}_{v \sim \mathbb{B}_d}[J(\theta + \delta v)]$, where $\mathbb{B}_d$ is the d-dimensional unit ball. We can then show that Algorithm 6 is essentially running SGD on the objective function $\hat{J}(\theta)$, which allows us to apply standard SGD analysis on $\hat{J}(\theta)$. Lastly we link the stationary point of the smoothed objective $\hat{J}(\theta)$ to that of the objective $J(\theta)$ using the assumptions on $J(\theta)$.

Theorem 17. *Consider running Algorithm 6 for T steps where the true objective $J(\theta)$ satisfies the assumptions stated above. Then we have,*

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(\mathcal{Q}^{\frac{1}{2}}dT^{\frac{-1}{2}} + \mathcal{Q}^{\frac{1}{3}}d^{\frac{2}{3}}T^{\frac{-1}{3}}\sigma) \tag{5.5}$$

where $J(\theta) \leq \mathcal{Q}$ for all $\theta \in \Theta$ and σ^2 is the variance of the random noise η in Algorithm 6.

The above theorem gives us a convergence rate to a stationary point of policy search in parameter space. The role of variance of i.i.d noise in the noisy evaluations of the true objective is very important. Consider the case where there is little stochasticity in the environment dynamics, i.e. $\sigma \to 0$, then the first term in Equation 5.5 becomes dominant and we only need at most $\mathcal{O}(\frac{d^2\mathcal{Q}}{\epsilon^2})$ samples to reach a point θ where $\mathbb{E}\|\nabla_\theta J(\theta)\|_2^2 \leq \epsilon$. However, if there is a lot of stochasticity in the environment dynamics then the second term is dominant and we need at most $\mathcal{O}(\frac{d^2\mathcal{Q}\sigma^3}{\epsilon^3})$ samples. It is interesting to observe the direct impact that the stochasticity of environment dynamics has on convergence rate of policy search, which is also experimentally demonstrated in Sec. 5.5.2. Note that the convergence rate has no dependency on horizon length H because of the regularity assumption we used on total reward: J is always bounded by a constant $\mathcal{Q}$ that is independent of H. However, as we will see later, even under the regularity assumption convergence rate of action space exploration methods have an explicit dependence on H which will prove to be the primary reason why black-box parameter space policy search methods in [Mania et al., 2018] have been so effective when compared to action space methods.

5.4.2 Exploration in Action Space

Another way to optimize the objective defined in Section 5.2.1 is to optimize over the space of actions $\mathcal{A}$. From [Silver et al., 2014], we know that for $J(\theta) = \mathbb{E}_{s\sim\mu}[V^0_{\pi_\theta}(s)]$ we can express the gradient as

$$\nabla_\theta J(\theta) = \sum_{t=0}^{H-1}\mathbb{E}_{s_t\sim d^t_{\pi_\theta}}\left[\nabla_\theta\pi(\theta,s_t)\nabla_a Q^t_{\pi_\theta}(s_t,\pi(\theta,s_t))\right] \tag{5.6}$$

Observe that the first term in the above gradient $\nabla_\theta\pi(\theta,s)$ is the Jacobian of the policy, the local linear relationship between a small change in policy parameters θ and a small change in its output, i.e., actions. The second term $\nabla_a Q(s,a)$ is actually the improvement direction at state action pair (s,a), i.e., conditioned on state s, if we move action a an infinitesimally small step along the negative gradient $-\nabla_a Q(s,a)$, we decrease the cost-to-go $Q(s,a)$. Eqn 5.6 then leverages policy's Jacobian to transfer the improvement direction in action space to an improvement direction in parameter space.

We can compute Jacobian $\nabla_\theta\pi(\theta,s)$ exactly as we have knowledge of the policy function, i.e, we can leverage the first-order information of the parameterized policy. The second term $\nabla_a Q^t_{\pi_\theta}(s,\pi(\theta,s_t))$, however, is unknown as it depends on the dynamics and cost functions and needs to be estimated by interacting with the environment. We could employ a similar algorithm as Algorithm 6, shown in Algorithm 7, to obtain an estimate of the gradient $\nabla_a Q^t_{\pi_\theta}(s,\pi(\theta,s_t))$, i.e., a zeroth order estimation of $\nabla_a Q^t_{\pi_\theta}$, computed as $\frac{p\tilde{Q}_i}{\delta}u$, where $\tilde{Q}_i$ is an unbiased estimate of $Q^t_{\pi_{\theta_i}}(s_t,\pi(\theta_i,s_t)+\delta u)$, with $u\sim\mathbb{S}_p$ (Line 7 & 9 in Alg. 7).

Another important difference from Algorithm 6 is the fact that we use a one-point estimate for the gradient g_i in Algorithm 7. We cannot employ the idea of two-point estimate in random exploration in action space to reduce the variance of the estimate of $\nabla_a Q^t_{\pi_\theta}(s_t, a)$. This is due to the fact that environment is stochastic, and we cannot guarantee that we will reach the same state s_t at any two independent roll-ins with π_θ at time step t. Similar to Section 5.4.1, we will analyze the rate of convergence of Algorithm 7 to a stationary point of the objective $J(\theta)$. The following section will lay out the assumptions and present the convergence analysis.

Algorithm 7 Policy Search in Action Space

```
1: Input: Learning rate $\alpha \in \mathbb{R}^+$, standard deviation of exploration noise $\delta \in \mathbb{R}$, Horizon
   length $H$, Initial state distribution $\mu$
2: Initialize parameters $\theta_1 \in \mathbb{R}^d$
3: for $i = 1$ to $T$ do
4:     Sample $u \sim \mathbb{S}_p$ , a $p$-dimensional unit sphere
5:     Sample uniformly $t \in \{0, \cdots, H-1\}$
6:     Execute policy $\pi(\theta_i, \cdot)$ until $t-1$ steps
7:     Execute perturbed action $a_t = \pi(\theta_i, s_t) + \delta u$ at timestep $t$ and continue with policy
       $\pi(\theta_i, \cdot)$ until timestep $H-1$ to obtain an estimate $\tilde{Q}_i = Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u) + \tilde{\eta}_i$
       where $\tilde{\eta}_i$ is zero mean random noise
8:     Compute policy Jacobian $\Psi_i = \nabla_\theta \pi(\theta_i, s_t)$
9:     Compute gradient estimate $g_i = H\Psi_i \frac{p\tilde{Q}_i}{\delta} u$
10:    Update $\theta_{i+1} = \theta_i - \alpha g_i$
11: end for
```

Assumptions and Analysis The assumptions for policy search in action space are similar to the assumptions in Section 5.4.1. We assume that $J(\theta)$ is differentiable with respect to θ over the entire domain. We also assume that $J(\theta)$ is G-lipschitz and L-smooth. In addition to these assumptions, we will assume that the policy function $\pi(\theta, s)$ is K-lipschitz in θ and the state-action value function $Q^t_{\pi_\theta}(s, a)$ is W-lipschitz and U-smooth in a. Finally, we assume that the state-action value function $Q(s, a)$ is differentiable with respect to a over the entire domain. Note that the Lipschitz assumptions above on $J(\theta)$, $Q^t_{\pi_\theta}(s, a)$, and $\pi(\theta, s)$ are also used in the analysis of Deterministic policy gradient [Silver et al., 2014]. We need extra smoothness assumption to study the convergence of our algorithms.

Note that the gradient estimate g_i used in Algorithm 7 is a biased estimate of $\nabla_\theta J(\theta)$. We can show this by considering

$$\mathbb{E}_i[g_i] = \mathbb{E}_t \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}} \left[H \nabla_\theta \pi(\theta_i, s_t) \mathbb{E}_{u \sim \mathbb{S}_p} \left[\frac{p\tilde{Q}_i}{\delta} u \right] \right]$$

where $\mathbb{E}_i$ denotes expectation with respect to the randomness at iteration i. From [Flaxman et al., 2005], we have that $\mathbb{E}[\frac{p\tilde{Q}_i}{\delta} u] = \nabla_a \mathbb{E}_{v \sim \mathbb{B}_p}[Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v)]$ so we can rewrite the above equation as

$$\mathbb{E}[g_i] = \sum_{t=0}^{H-1} \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}} \mathbb{E}_{v \sim \mathbb{B}_p} \left[\nabla_\theta \pi(\theta_i, s_t) \nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v) \right]$$

Comparing the above expression with equation 5.6, we can see that g_i is not an unbiased estimate of the gradient $\nabla_\theta J(\theta)$. We can also explicitly upper bound the variance of g_i by $\mathbb{E}_i\|g_i\|_2^2$. Note that in the limit when $\delta \to 0$, g_i becomes an unbiased estimate of $\nabla_\theta J(\theta)$, but the variance will approach to infinity. In our analysis, we explicitly tune δ to balance the bias and variance.

Theorem 18. *Consider running Algorithm 7 for T steps where the objective $J(\theta)$ satisfies the assumptions stated above. Then, we have*

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \le \mathcal{O}(T^{-\frac{1}{4}}Hp^{\frac{1}{2}}(\mathcal{Q}^3+\sigma^2\mathcal{Q})^{\frac{1}{4}}) \tag{5.7}$$

where $J(\theta) \le \mathcal{Q}$ for all $\theta \in \Theta$ and σ^2 is the variance of the random noise $\tilde{\eta}$ in Algorithm 7.

The above theorem gives us a convergence rate to a stationary point of $J(\theta)$ for policy search in action space. This means that to reach a point θ where $\mathbb{E}\|\nabla_\theta J(\theta)\|_2^2 \le \epsilon$, policy search in action space needs at most $\mathcal{O}\left(\frac{p^2H^4}{\epsilon^4}(\mathcal{Q}^3+\sigma^2\mathcal{Q})\right)$ samples. Interestingly, the convergence rate has a dependence on the horizon length H, unlike policy search in parameter space. Also, observe that the convergence rate has no dependence on the parameter dimensionality d as we have complete knowledge of the Jacobian of policy, and we have a dependence on stochasticity of the environment σ that slows down the convergence as the stochasticity increases, similar to policy search in parameter space.

5.5 Experiments

Given the analysis presented in the previous sections, we test the convergence properties of parameter and action space policy search approaches across several experiments: Contextual Bandit with rich observations, Linear Regression, RL benchmark tasks and Linear Quadratic Regulator (LQR). We use Augmented Random Search (ARS), from [Mania et al., 2018], as the policy search in parameter space method in our experiments as it has been empirically shown to be effective in RL tasks. For policy search in action space, we use either REINFORCE [Williams, 1992], or ExAct (Exploration in Action Space), the method described by Algorithm 7. In all the plots shown, solid lines represent the mean estimate over 10 random seeds and shaded regions correspond to ± 1 standard error. The code for all our experiments can be found here[1][2].

5.5.1 One-Step Control

In these sets of experiments, we test the convergence rate of policy search methods for one time-step prediction. The objective is to minimize the instantaneous cost incurred. The motivation behind such experiments is that we want to understand the dependence of different policy search methods on parametric dimensionality d without the effect of horizon length H.

[1] `https://github.com/LAIRLAB/contrasting_exploration_rl`

[2] `https://github.com/LAIRLAB/ARS-experiments`

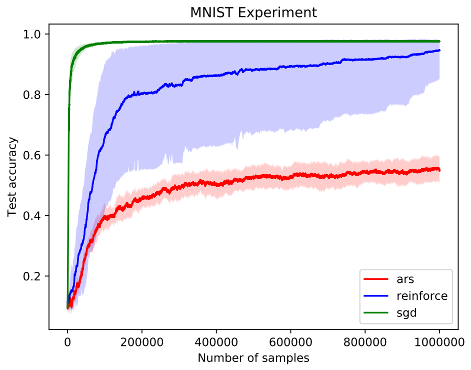


Figure 5.1: Mean test accuracy with standard error for different approaches against number of samples

MNIST as a Contextual Bandit Our first set of experiments is the MNIST digit recognition task [LeCun et al., 1998]. To formulate the task in an RL framework, we consider a sequential decision making problem where at each time-step the agent is given the features of the image and needs to predict one of ten actions (corresponding to digits). A reward of $+1$ is given for predicting the correct digit, and a reward of -1 for an incorrect prediction. With this reduction, the problem is essentially a Contextual Bandit Problem [Agarwal et al., 2014]. We use a standard LeNet-style convolutional architecture, [LeCun et al., 1998], with $d = 21840$ trainable parameters. Figure 5.1 shows the learning curves for SGD under standard full-information supervised learning setting with cross entropy loss, REINFORCE and ARS. We can observe that in this setting where the parameter space dimensionality d significantly exceeds the action space dimensionality $p = 1$, policy search in action space outperforms parameter space methods.

Linear Regression with Partial Information These set of experiments are designed to understand how the sample complexity of different policy search methods vary as the parametric complexity is varied. More specifically, from our analysis in Section 5.3, we know that sample complexity of parameter space methods have a dependence on d, the parametric complexity, whereas action space methods have no dependence on d. We test this hypothesis in this experiment using artificial data with varying input dimensionality and output scalar values.

Figure 5.2 shows the learning curves for standard full-information supervised learning approaches with full access to the square loss (SGD & Newton), REINFORCE, natural REINFORCE [Kakade, 2002], and ARS as we increase the input dimensionality, and hence parametric dimensionality d. Note that we have not included natural REINFORCE and Newton method in Figure **??** as extensive hyperparameter search for these methods is computationally expensive in such high dimensionality settings. The learning curves in Figure 5.2 match our expectations, and show that action space policy search methods do not degrade as parametric dimensionality increases whereas parameter space methods do. Moreover, action space methods lie between the curves of supervised learning and parameter space methods as they take advantage of the Jacobian of the policy and learn more quickly than parameter space methods.

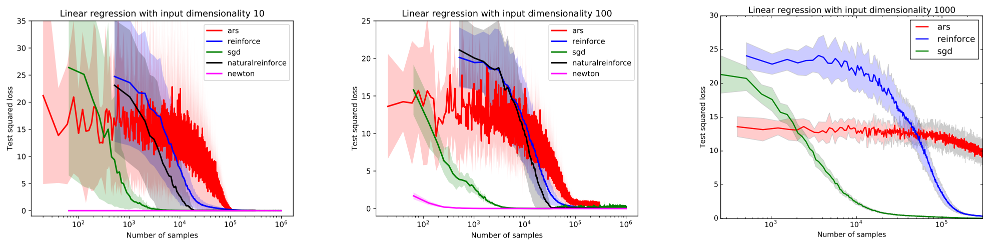
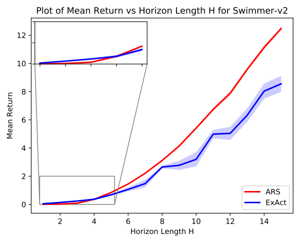
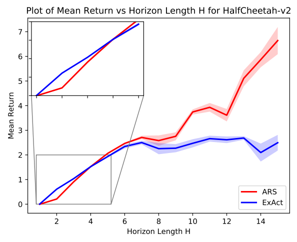


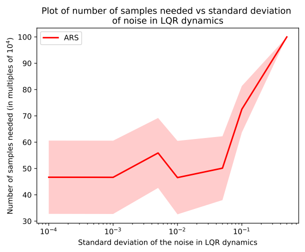
Figure 5.2: Linear Regression Experiments with varying input dimensionality



(a) Swimmer



(b) HalfCheetah



(c) LQR

Figure 5.3: Multi-step Control. Figures 5.3a and 5.3b show performance of different methods as horizon length varies. Figure 5.3c shows number of samples needed to reach close to a stationary point as noise in dynamics varies

5.5.2 Multi-Step Control

The above experiments provide insights on the dependence of policy search methods on parametric dimensionality d. We now shift our focus on to the dependence on horizon length H. In this set of experiments, we extend the time horizon and test the convergence rate of policy search methods for multi-step control. The objective is to minimize the sum of costs incurred over a horizon H, i.e. $J(\theta) = \mathbb{E}[\sum_{t=1}^{T} c(s_t, a_t)]$. According to our analysis, we expect action space policy search methods to have a dependence on the horizon length H.

We test ARS and ExAct on two popular continuous control simulated benchmark tasks in OpenAI gym [Brockman et al., 2016b]: Swimmer and HalfCheetah. We chose these two environments as they allow you to vary the horizon length H without terminating the task early. For both tasks, we use linear policies as they have been shown to be very effective in [Mania et al., 2018, Rajeswaran et al., 2017]. Swimmer has an observation space dimensionality of $d = 8$ and a continuous action space of dimensionality $p = 2$. Similarly, for HalfCheetah $d = 17$ and $p = 6$. Figures 7.1e and 7.1g show the performance of both approaches in terms of the mean return $J(\theta)$ (expected sum of rewards) they obtain as the horizon length H varies. Note that both approaches are given access to the same number of samples $10^4 \times H$ from the environments for each horizon length H. In the regime of short horizon lengths, action space methods are better than parameter space methods as they do not have a dependence on parametric complexity d. However, as the horizon length increases, parameter space methods start outperforming action space methods handily as they do not have an explicit dependence on the horizon length, as pointed out by our analysis. We have observed the same trend of parameter space methods handily outperforming action space methods as far as $H = 200$ and expect this trend to continue beyond. This empirical insight combined with our analysis presented in Sections 5.4.1, 5.4.2 explains why ARS, a simple parameter space search method, outperformed state-of-the-art actor critic action space search methods in [Mania et al., 2018] on OpenAI gym benchmarks where the horizon length H is typically as high as 1000.

Effect of environment stochasticity In this final set of experiments, we set out to understand the effect of stochasticity in environment dynamics on the performance of policy search methods. As our analysis in Sections 5.4.1 and 5.4.2 points out, the stochasticity of the environment plays an important role in controlling the variance of our gradient estimates in zeroth order optimization procedures. To empirically observe this, we use a stochastic LQR environment where we have access to the true cost function c and hence, can compute the gradient $\nabla_\theta J(\theta)$ exactly. Given access to such information, we vary the standard deviation σ of the noise in LQR dynamics and observe the number of samples needed for ARS to reach θ such that $\|\nabla_\theta J(\theta)\|_2^2 \leq 0.05$. Figure 5.3c presents the number of samples needed to reach close to a stationary point of $J(\theta)$ as the standard deviation of noise in LQR dynamics varies. Note that we limit the maximum number of samples to 10^6 for each run. The results match our expectations from the analysis, where we observed that as the stochasticity of the environment increases, convergence rate of policy search methods slows down.

5.6 Conclusion

Parameter space exploration via black-box optimization methods have often been shown to outperform sophisticated action space exploration approaches for the reinforcement learning problem. Our work highlights the major difference between parameter and action space exploration methods: the latter leverages Jacobian of the parameterized policy. This allows sample complexity of action space exploration methods to be independent of parameter space dimensionality and only dependent on the dimensionality of action space and horizon length. For domains where the action space dimensionality and horizon length are small and the dimensionality of parameter space is large, we conclude that exploration in action space should be preferred. On the other hand, for long horizon control problems with low dimensional policy parameterization, exploration in parameter space will outperform exploration in action space.

Part III

Model-Based Reinforcement Learning

Chapter 6

Model-based RL in Contextual Decision Processes: PAC bounds and Exponential Improvements over Model-free Approaches

In this chapter, we study the sample complexity of model-based reinforcement learning in general contextual decision processes. We design new algorithms for RL with a generic model class and analyze their statistical properties. Our algorithms have sample complexity governed by a new structural parameter called the *witness rank*, which we show to be small in several settings of interest, including factored MDPs and reactive POMDPs. We also show that the witness rank of a problem is never larger than the recently proposed *Bellman rank* parameter governing the sample complexity of the model-free algorithm Olive [Jiang et al., 2017], the only other provably sample efficient algorithm at this level of generality. Focusing on the special case of factored MDPs, we prove an exponential lower bound for a general class of model-free approaches, including Olive, which when combined with our algorithmic results demonstrates exponential separation between model-based and model-free RL in some rich-observation settings.

The contributions in this chapter were first introduced in the joint work with Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford [Sun et al., 2018d].

6.1 Introduction

Reinforcement learning algorithms can be broadly categorized as model-based or model-free methods. Methods in the former family explicitly model the environment dynamics and then use planning techniques to find a near-optimal policy. In contrast, the latter family models much less, typically only an optimal policy and its value. Algorithms from both families have seen substantial empirical success, but we lack a rigorous understanding of the tradeoffs between them, making algorithm selection difficult for practitioners. This paper provides a new understanding of these tradeoffs, via a comparative analysis between model-based and model-free methods in general RL settings.

Conventional wisdom and intuition suggests that model-based methods are more sample-efficient than model-free methods, since they leverage more supervision. This argument is supported by classical control-theoretic settings like the linear quadratic regulator, where state-of-the-art model-based methods have better dimension dependence than contemporary model-free ones [Tu and Recht, 2018]. On the other hand, since models typically have more degrees of freedom (e.g., parameters) and can waste effort on unimportant elements of the environment, one might suspect that model-free methods have better statistical properties. Indeed, recent work in tabular Markov Decision Processes (MDPs) suggest that there is almost no sample-efficiency gap between the two families [Jin et al., 2018]. Even worse, in the rich environments of interest, where function approximation is essential, the only algorithms with sample complexity guarantees are model-free [Jiang et al., 2017]. In such environments, which of these competing perspectives applies?

To answer this question, we study model-based RL in episodic *contextual decision processes* (CDPs) where a high-dimensional observation is used for decision making. We assume that the algorithm is given access to a class $\mathcal{M}$ of models and the true environment is representable by the class. We compare such algorithms with their model-free counterparts which are provided a class of candidate value functions that realizes the optimal value function. Under such assumptions, we posit:

> *Model-based methods rely on stronger function-approximation capabilities but can be exponentially more sample efficient than their model-free counterparts.*

Our contributions provide evidence for this thesis and can be summarized as follows:

1. We show that there exist MDPs where (1) all model-free methods, given a value function class satisfying the above realizability condition, incur exponential sample complexity (in horizon); and (2) there exist model-based methods that, given a model class containing the true model, obtain a polynomial sample complexity. In fact, these MDPs belong to the well-studied *factored MDPs* [Kearns and Koller, 1999], which we use as a running example throughout the paper.

2. We design a new model-based algorithm for general CDPs and show that the algorithm has sample complexity governed by a new structural parameter, the *witness rank*. We further show that many concrete settings, including tabular and low rank MDPs, reactive POMDPs, and reactive PSRs have a small witness rank. This algorithm is the first provably-efficient model-based algorithm that does not rely on tabular representations or highly structured control-theoretic settings.

3. We compare our algorithm and the witness rank with the model-free algorithm OLIVE [Jiang et al., 2017] and the Bellman rank, the only other algorithm and structural parameter

at this level of generality. We show that the witness rank is never larger, but can be exponentially smaller than the Bellman rank, e.g., in factored MDPs. Specifically, our algorithm has polynomial sample complexity in factored MDPs, an exponential gain over OLIVE and any other realizability-based model-free algorithm.

The caveat in our thesis is that model-based methods rely on strong realizability assumptions. In the rich environments we study, where function approximation is essential, some form of realizability is necessary (see Proposition 1 in Krishnamurthy et al. [2016]), but our model-based assumption (See Assumption 2) is strictly stronger than prior value-based analogs [Antos et al., 2008, Krishnamurthy et al., 2016]. On the other hand, our results precisely quantify the tradeoffs between model-based and model-free approaches, which may guide future empirical efforts.

6.2 Preliminaries

We study *Contextual Decision Processes* (CDPs), a general sequential decision making setting where the agent optimizes long-term reward by learning a policy that maps from rich observations (e.g., raw-pixel images) to actions. The term CDP was proposed by Krishnamurthy et al. [2016] and extended by Jiang et al. [2017], with CDPs capturing broad classes of RL problems allowing rich observation spaces including (Partially Observable) MDPs and Predictive State Representations. Please see the above references for further background.

Notation. We use $[N] \triangleq \{1, \ldots, N\}$. For a finite set S, $\Delta(S)$ is the set of distributions over S, and $U(S)$ is the uniform distribution over S. For a function $f : S \to \mathbb{R}$, $\|f\|_\infty$ denotes $\sup_{s\in S} |f(s)|$.

6.2.1 Basic Definitions

Let $H \in \mathbb{N}$ denote a time horizon and let $\mathcal{X}$ be a large context space of unbounded size, partitioned into subsets $\mathcal{X}_1, \ldots, \mathcal{X}_{H+1}$. A finite horizon episodic CDP is a tuple $(\mathcal{X}, \mathcal{A}, R, P)$ consisting of a (partitioned) context space $\mathcal{X}$, an action space $\mathcal{A}$, a transition operator $P : \{\perp\} \cup (\mathcal{X} \times \mathcal{A}) \to \Delta(\mathcal{X})$, and a reward function $R : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{R})$ with $\mathcal{R} \subseteq [0, 1]$.[1] We assume a *layered Markovian* structure, so that for any $h \in [H]$, $x_h \in \mathcal{X}_h$ and $a \in \mathcal{A}$, the future context and the reward distributions are characterized by x_h, a and moreover $P_{x_h,a} \triangleq P(x_h, a) \in \Delta(\mathcal{X}_{h+1})$. We use $P_0 \triangleq P(\perp) \in \Delta(\mathcal{X}_1)$ to denote the initial context distribution, and we assume $|\mathcal{A}| = K$ throughout.[2] Note that the layering of contexts allows us to implicitly model the level h as part of the context.

A policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ maps each context to a distribution over actions. By executing this policy in the CDP for $h - 1$ steps, we naturally induce a distribution over $\mathcal{X}_h$, and we use $\mathbb{E}_{x_h\sim\pi}[\cdot]$ to denote the expectation with respect to this distribution. A policy π has associated value and action-value functions $V^\pi : \mathcal{X} \to \mathbb{R}^+$ and $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^+$, defined as

$$\forall h \in [H], x \in \mathcal{X}_h : \; V^\pi(x) \triangleq \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=h}^{H} r_t \,\middle|\, x_h = x\right], \; Q^\pi(x, a) \triangleq \mathop{\mathbb{E}}_{r\sim R(x,a)}[r] + \mathop{\mathbb{E}}_{x'\sim P_{x,a}}[V^\pi(x')],$$

[1]We assume Markov transitions w.l.o.g., since context may encode history.

[2]Partitioning the context space by time allows us to capture more general time-dependent dynamics, reward, and policy.

Here, the expectation is over randomness in the environment and the policy, with actions sampled by π. Note that there is no need to index V and Q by the level h since it is encoded in the context. The value of a policy π is $v^\pi \triangleq \mathbb{E}_{x_1 \sim P_0}[V^\pi(x_1)]$ and the goal is to find a policy π that maximizes v^π.

For regularity, we assume that almost surely $\sum_{h=1}^H r_h \le 1$.

Running Example. *As a running example, we consider* factored MDPs *[Kearns and Koller, 1999]. Let $d \in \mathbb{N}$ and let $\mathcal{O}$ be a small finite set. Define the context space $\mathcal{X} \triangleq [H] \times \mathcal{O}^d$, with the natural partition by time. For a state $x \in \mathcal{X}$ we use $x[i]$ for $i \in [d]$ to denote the value of x on the i^{th} state variable (ignoring the time step h), and similar notation for a subset of state variables. For each state variable $i \in [d]$, the parents of i, $\text{pa}_i \subseteq [d]$ are the subset of state variables that directly influence i. In factored MDPs, the transition dynamics P factorize according to the parent relationships:*

$$\forall h, x \in \mathcal{X}_h, x' \in \mathcal{X}_{h+1}, a \in \mathcal{A}, \;\; P(x' \mid x, a) = \prod_{i=1}^{d} P^{(i)}[x'[i] \mid x[\text{pa}_i], a, h] \tag{6.1}$$

for conditional distributions $\{P^{(i)}\}_{i=1}^d$ of the appropriate dimensions. Note that we always condition on the time point h to allow for non-stationary transitions. This transition operator has $L \triangleq \sum_{i=1}^d HK \cdot |\mathcal{O}|^{1+|\text{pa}_i|}$ parameters, which can be much smaller than $dHK|\mathcal{O}|^{1+d}$ for an unfactorized process on $|\mathcal{O}|^d$ states.[3] *When $|\text{pa}_i|$ is small for all i, we can expect algorithms with low sample complexity. Indeed Kearns and Koller [1999] show that factored MDPs can be PAC learned with poly$(H, K, L, \epsilon, \log(1/\delta))$ samples in the average and discounted reward settings. For more recent development in this line of research, we refer the readers to Diuk et al. [2009], Guo and Brunskill [2017], Nguyen et al. [2013], Osband and Van Roy [2014b] and the references therein.*

6.2.2 Model Class

Since we are interested in general CDPs with large state spaces (i.e., non-tabular setting), we equip model-based algorithms with a model class $\mathcal{M}$, where all models in $\mathcal{M}$ share the same $\mathcal{X}$ and $\mathcal{A}$ but can differ in reward function R and transition operator P. The environment reward and dynamics are called the *true model* and denoted $M^\star \triangleq (R^\star, P^\star)$. For a model $M \in \mathcal{M}$, π_M, V_M, Q_M, and v_M are the optimal policy, value function, action-value function, and value *in the model M*, respectively. These objects are purely functions of M and do not depend on the environment. For the true model $M^\star$, these quantities are denoted $\pi^\star, V^\star, Q^\star, v^\star$, suppressing subscripts. For $M \triangleq (R, P)$, we denote $(r, x') \sim M_{x,a}$ as sampling a reward and next context from M: $r \sim R(x, a), x' \sim P_{x,a}$. We use $x_h \sim \pi$ to denote a state sampled by executing π in the true environment $M^\star$ for $h-1$ steps.

We use $\texttt{OP}$ (for Optimal Planning) to represent the operator that maps a model M to its optimal Q function, that is $\texttt{OP}(M) \triangleq Q_M$. We denote $\texttt{OP}(\mathcal{M}) \triangleq \{Q : \exists M \in \mathcal{M} \; s.t. \; \texttt{OP}(M) = Q\}$ as the set of optimal Q functions derived from the class $\mathcal{M}$, Throughout the paper, when we compare model-based and model-free methods, we use $\mathcal{M}$ as input for the former and $\texttt{OP}(\mathcal{M})$ for the latter.

We assume the model class has *finite (but exponentially large) cardinality and is realizable.*

Assumption 2 (Realizability of $\mathcal{M}$)**.** *We assume the model class $\mathcal{M}$ contains the true model $M^\star$.*

[3]Actually the full unfactored process has $HK|\mathcal{O}|^{2d}$ parameters. Here we are assuming that the state variables are conditionally independent given the previous state and action.

The finiteness assumption is made only to simplify presentation and can be relaxed using standard techniques; see Theorem 22 for a result with infinite model classes. While realizability can also be relaxed (as in Jiang et al. [2017]), it is impossible to avoid it altogether (that is, to compete with $\{\pi_M : M \in \mathcal{M}\}$ for arbitrary $\mathcal{M}$) due to exponential lower bounds [Krishnamurthy et al., 2016].

Running Example. *For factored MDPs, it is standard to assume the factorization, formally* pa_i *for all* $i \in [d]$*, and the reward function are known [Kearns and Koller, 1999]. Thus the natural model class* $\mathcal{M}$ *is just the set of all dynamics of the form* (6.1)*, which obey the factorization, with shared reward function. While this class is infinite, our techniques apply as shown in the proof of Theorem 22.*

6.3 Why Model-based RL?

This section contains our first main result, that model-based methods can be *exponentially* more sample-efficient than model-free ones. To our knowledge, this is the first result of this form.

To show such separation, we must prove a lower bound against all model-free methods, and, to do so, we first formally define this class of algorithms. Strehl et al. [2006] define model-free algorithms to be those with $o(|\mathcal{X}|^2|\mathcal{A}|)$ space, but this definition is specialized to the tabular setting and provides little insight for algorithms employing function approximation. In contrast, our definition is information-theoretic: Intuitively, a model-free algorithm does not operate on the context x directly, but rather through the evaluations of an action-value function class $\mathcal{Q}$. Formally:

Definition 3 (Model-free algorithm)**.** *Given a (finite) function class* $\mathcal{Q} : (\mathcal{X} \times \mathcal{A}) \to \mathbb{R}$*, define the* Q-profile $\Phi_{\mathcal{Q}} : \mathcal{X} \to \mathbb{R}^{|\mathcal{Q}|\times|\mathcal{A}|}$ *by* $\Phi_{\mathcal{Q}}(x) := [Q(x,a)]_{Q\in\mathcal{Q},a\in\mathcal{A}}$*. An algorithm is* model-free *using* $\mathcal{Q}$ *if it accesses* x *exclusively through* $\Phi_{\mathcal{Q}}(x)$ *for all* $x \in \mathcal{X}$ *during its entire execution.*

This definition agrees with an intuitive notion of model-free algorithms, and captures value-based algorithms such as Olive which works for general CDPs, optimistic Q-learning [Jin et al., 2018] and Delayed Q-learning [Strehl et al., 2006] for tabular MDPs, and even *policy-based* algorithms like policy gradient methods [Williams, 1992]. In Appendix 9.5.4, we show that for tabular environments with a fully expressive $\mathcal{Q}$ class, the underlying context/state can be recovered from the Q-profile, and hence Definition 3 introduces no restriction whatsoever in this setting. We also discuss implementation of popular algorithms relying on function approximation using Q-profiles in the appendix. However, beyond tabular settings, the Q-profile can obfuscate the true context from the agent and may even introduce partial observability. This can lead to a significant loss of information, which can have a dramatic effect on the sample complexity of RL. Such information loss is formalized in the following theorem.

Theorem 19. *Fix* $\delta, \epsilon \in (0, 1]$*. There exists a family* $\mathcal{M}$ *of CDPs with horizon* H*, all with the same reward function, and satisfying* $|\mathcal{M}| \leq 2^H$*,such that*

1. *For any CDP in the family, with probability at least* $1 - \delta$*, a model based algorithm using* $\mathcal{M}$ *as the model class (Algorithm 12, Appendix 9.5.5) outputs* $\hat{\pi}$ *satisfying* $v^{\hat{\pi}} \geq v^\star - \epsilon$ *using at most poly*$(H, 1/\epsilon, \log(1/\delta))$ *trajectories.*

2. *With $\mathcal{Q} = \texttt{OP}(\mathcal{M})$ as the Q-function class, any model-free algorithm using $o(2^H)$ trajectories outputs a policy $\hat{\pi}$ with $v^{\hat{\pi}} < v^\star - 1/2$ with probability at least $1/3$ on some CDP in the family.*

See Appendix 9.5.3 for the proof. Informally, the result shows that there are CDPs where model-based methods can be exponentially more sample-efficient than any model-free method, when given access to a $\mathcal{Q}$ satisfying $Q^\star \in \mathcal{Q}$. As concrete instances of such methods, the lower bound applies to several recent value-based algorithms for CDPs [Dann et al., 2018, Jiang et al., 2017, Krishnamurthy et al., 2016] as well as any future algorithms developed assuming just realizable optimal value functions. On the other hand, it leaves room for sample efficient model-free techniques that require stronger representation conditions on $\mathcal{Q}$.[4] To our knowledge, this is the first information-theoretic separation result for any broad class of model-based/model-free algorithms. Indeed, even the definition of model-free methods is new here.[5]

Given this result, it might seem that model-based RL should be always preferred to model-free RL. However, it is worth also comparing the assumptions required to enable the two paradigms in general. Since $M^\star \in \mathcal{M}$ for each CDP in the family, we also have $Q^\star \in \texttt{OP}(\mathcal{M})$. This latter *value-function realizability* assumption is standard in model-free RL with function approximation [Antos et al., 2008, Krishnamurthy et al., 2016], but our model-based analog in Assumption 2 can be substantially stronger. As such, model-based methods operating with realizability typically require more powerful function approximation than model-free methods. Thus, while Theorem 19 formalizes an argument in favor of model-based methods, realizability considerations provide an important caveat.

Running Example. *The construction in the proof of Theorem 19 is a simple factored MDP with $d = H$, $|\mathcal{O}| = 4$, $|pa_i| = 1$ for all i, and deterministic dynamics. As we see, our algorithm has polynomial sample complexity in all factored MDPs (and a broad class of other environments).*

The construction implies that model-free methods cannot succeed in factored MDPs. To our knowledge, no information theoretic lower bounds for factored MDPs exist, but the result does agree with known computational and representational barriers, namely (a) that planning is NP-hard [Mundhenk et al., 2000], (b) that $Q^\star$ and $\pi^\star$ may not factorize [Guestrin et al., 2003], and (c) that $\pi^\star$ cannot be represented by a polynomially sized circuit [Allender et al., 2003]. Our result provides a new form of hardness, namely statistical complexity, for model-free RL in factored MDPs.

6.4 Witnessed Model Misfit

In this section we introduce *witnessed model misfit*, a measure of model error, which we later use to eliminate candidate models in our algorithm.

To verify the validity of a candidate model, a natural idea is to compare the samples from the environment with synthetic samples generated from a model M. To formalize this comparison approach, we use Integral Probability Metrics (IPM) [Müller, 1997]: for two probability distributions $P_1, P_2 \in \Delta(\mathcal{Z})$ over $z \in \mathcal{Z}$ and a function class $\mathcal{F} : \mathcal{Z} \to \mathbb{R}$ that is symmetric (i.e. if $f \in \mathcal{F}$ then $-f \in \mathcal{F}$ also holds), the IPM with respect $\mathcal{F}$ is: $\sup_{f \in \mathcal{F}} \mathbb{E}_{z \sim P_1}[f(z)] - \mathbb{E}_{z \sim P_2}[f(z)]$. In words, we search for a test function from $\mathcal{F}$ whose expectation under P_1 is maximally different from the expectation under P_2. We use IPMs to define witnessed model misfit.

[4] However, a few natural assumptions we have inspected do not seem to break the lower bound.

[5] Sutton and Barto [2018, Section 11.6] have a closely related definition (where the learner can only observe state features), but the definition is specialized to linear function approximation and is subsumed by ours.

Definition 4 (Witnessed Model Misfit). *For a class $\mathcal{F} : \mathcal{X}\times\mathcal{A}\times\mathcal{R}\times\mathcal{X} \to \mathbb{R}$, models $M, M' \in \mathcal{M}$ and a time step $h \in [H]$, the Witnessed Model Misfit of M' witnessed by M at level h is:*

$$\mathcal{W}(M, M', h; \mathcal{F}) \triangleq \sup_{f\in\mathcal{F}} \mathop{\mathbb{E}}_{\substack{x_h\sim\pi_M \\ a_h\sim\pi_{M'}}} \left[\mathop{\mathbb{E}}_{(r,x')\sim M'_h}[f(x_h, a_h, r, x')] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[f(x_h, a_h, r, x')] \right], \tag{6.2}$$

where for a model $M = (R, P)$, $(r, x') \sim M_h$ is shorthand for $r \sim R_{x_h,a_h}, x' \sim P_{x_h,a_h}$.

$\mathcal{W}(M, M', h; \mathcal{F})$ is the IPM between two distributions over $\mathcal{X}\times\mathcal{A}\times\mathcal{R}\times\mathcal{X}$ with the same marginal over $\mathcal{X}\times\mathcal{A}$ but two different conditionals over (r, x'), according to M' and the true model $M^\star$, respectively. The marginal over $\mathcal{X} \times \mathcal{A}$ is the distribution over context-action pairs when π_M, the optimal policy of another candidate model M, is executed in the true environment. We call this witnessed model misfit since M' might successfully masquerade as $M^\star$ unless we find the right context distribution to witness its discrepancy. Below we illustrate the definition with some examples.

Example (Total Variation). *When $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$, the witnessed model misfit becomes*

$$\mathcal{W}(M, M', h; \mathcal{F}) = \mathbb{E}\left[\left\| R'_{x_h,a_h} \circ P'_{x_h,a_h} - R^\star_{x_h,a_h} \circ P^\star_{x_h,a_h} \right\|_{TV} | \, x_h \sim \pi_M, a_h \sim \pi_{M'} \right], \tag{6.3}$$

where $R_{x,a} \circ P_{x,a}$ is the distribution over $\mathcal{R} \times \mathcal{X}$ with $r \sim R_{x,a}, x' \sim P_{x,a}$ independently. This is just the total variation distance[6] *between $R' \circ P'$ and $R^\star \circ P^\star$, averaged over context-action pairs $x \sim \pi_M, a \sim \pi_{M'}(\cdot|x)$ sampled from the true environment.*

Example (Exponential Family). *Suppose the models $M \triangleq (R, P)$ are from a conditional exponential family: conditioned on $(x, a) \in \mathcal{X} \times \mathcal{A}$, we have $R_{x,a} \circ P_{x,a} \triangleq \exp\left(\langle\theta_{x,a}, \mathrm{T}(r, x')\rangle\right)/Z(\theta_{x,a})$ for parameters $\theta_{x,a} \in \Theta \triangleq \{\theta : \|\theta\| \le 1\} \subset \mathbb{R}^m$ with partition function $Z(\theta_{x,a})$ and sufficient statistics $\mathrm{T} : \mathcal{R} \times \mathcal{X} \to \mathbb{R}^m$. With $\mathcal{G} = \{\mathcal{X} \times \mathcal{A} \to \Theta\}$, we design $\mathcal{F} = \{(x, a, r, x') \mapsto \langle g(x, a), \mathrm{T}(r, x')\rangle : g \in \mathcal{G}\}$. In this setting, witnessed model misfit is*

$$\mathcal{W}(M, M', h; \mathcal{F}) = \mathop{\mathbb{E}}_{x_h\sim\pi_M, a_h\sim\pi_{M'}} \left[\left\| \mathop{\mathbb{E}}_{(r,x')\sim M'_h}[\mathrm{T}(r, x')] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[\mathrm{T}(r, x')] \right\|_\star \right],$$

with $\|x\|_\star \triangleq \sup\{\langle x, \theta\rangle | \, \|\theta\| \le 1\}$. Here, we measure distance, in the dual norm, between the expected sufficient statistics of (r, x') sampled from M' and the true model $M^\star$. See Appendix 9.5.10.

Example (MMD). *When $\mathcal{F}$ is a unit ball in an RKHS, we obtain MMD [Gretton et al., 2012].*

Witnessed model misfit is also closely related to the *average Bellman error*, introduced by Jiang et al. [2017]. Given Q functions, Q and Q', the average Bellman error at time step h is:

$$\mathcal{E}_B(Q, Q', h) \triangleq \mathbb{E}\left[Q'(x_h, a_h) - r_h - Q'(x_{h+1}, a_{h+1}) \,\middle|\, x_h \sim \pi_Q, a_{h:h+1} \sim \pi_{Q'} \right] \tag{6.4}$$

where π_Q is the greedy policy associated with Q, i.e., $\pi_Q(a|x) \triangleq \mathbf{1}\{a = \mathrm{argmax}_{a'} Q(x, a')\}$, and the random trajectories (w.r.t. which we take expectation) are generated in the true environment $M^\star$.

[6] We use $\|P_1 - P_2\|_{\mathrm{TV}} = \sum_{x\in\mathcal{X}} |P_1(x) - P_2(x)|$, differing from the standard definition of $\|\cdot\|_{\mathrm{TV}}$ by a factor of 2.

When the Q functions are derived from a model class, meaning that $\mathcal{Q} = \texttt{OP}(\mathcal{M})$, we can extend the definition to any pair of models $M, M' \in \mathcal{M}$, using Q_M and $Q_{M'}$. In such cases, the average Bellman error for M, M' is just the model misfit witnessed by the function $f_{M'}(x, a, r, x') = r + V_{M'}(x')$. We conclude this section with an assumption about the class $\mathcal{F}$.

Assumption 3 (Bellman domination using $\mathcal{F}$)**.** *$\mathcal{F}$ is symmetric, finite in size,*[7] $\forall f \in \mathcal{F}$: $\|f\|_\infty \leq 2$, *and the witnessed model misfit* (6.2) *satisfies* $\forall M, M' \in \mathcal{M}$: $\mathcal{W}(M, M', h; \mathcal{F}) \geq \mathcal{E}_B(Q_M, Q_{M'}, h)$.

As discussed above, one easy way to satisfy this assumption is to ensure that the special functions $r + V_M(x')$ are contained in $\mathcal{F}$ for all $M \in \mathcal{M}$, but this is not the only way as we see in Section 6.6.[8] The Bellman domination condition in Assumption 3 plays an important role in the algorithm we present next, as it allows us to detect the suboptimality of a model in terms of the value attained by its optimal policy in the actual MDP.

6.5 A Model-based Algorithm

In this section, we present our main algorithm and sample complexity results. We start by describing the algorithm. Then, working towards a statistical analysis, we introduce the *witness rank*, a new structural complexity measure. We end this section with the main sample complexity bounds.

6.5.1 Algorithm

Since we do not have access to $M^\star$, we must estimate the witnessed model misfit from samples. Since $\mathcal{F}$ will always be clear from the context, we drop it from the arguments to the model misfit for succinctness. Given a dataset $\mathcal{D} \triangleq \{(x_h^{(n)}, a_h^{(n)}, r_h^{(n)}, x_{h+1}^{(n)})\}_{n=1}^N$ with

$$x_h^{(n)} \sim \pi_M,\ a_h^{(n)} \sim U(\mathcal{A}),\ (r_h^{(n)}, x_{h+1}^{(n)}) \sim M_h^\star,$$

denote the importance weight $\rho^{(n)} \triangleq K\pi_{M'}(a_h^{(n)}|x_h^{(n)})$. We simply use the empirical model misfit:

$$\widehat{\mathcal{W}}(M, M', h) \triangleq \max_{f \in \mathcal{F}} \sum_{n=1}^N \frac{\rho^{(n)}}{N} \mathop{\mathbb{E}}_{(r,x') \sim M'_h} \left[f(x_h^{(n)}, a_h^{(n)}, r, x') - f(x_h^{(n)}, a_h^{(n)}, r_h^{(n)}, x_{h+1}^{(n)}) \right]. \quad (6.5)$$

Here the importance weight $\rho^{(n)}$ accounts for distribution mismatch, since we are sampling from $U(\mathcal{A})$ instead of $\pi_{M'}$. Via standard uniform convergence arguments (in Appendix 9.5.1) we show that $\widehat{\mathcal{W}}(M, M', h)$ provides a high-quality estimate of $\mathcal{W}(M, M', h)$ under Assumption 3.

We also require an estimator for the average Bellman error $\mathcal{E}_B(M, M, h)$. Given a data set $\{(x_n^{(h)}, a_h^{(n)}, r_h^{(i)}, x_{h+1}^{(n)})\}_{n=1}^N$ where $x_h^{(n)} \sim \pi_M, a_h^{(n)} \sim \pi_M$, and $(r_h^{(n)}, x_{h+1}^{(n)}) \sim M_h^\star$, we

[7] As before, our results apply whenever $\mathcal{F}$ has bounded statistical complexity, for example any Glivenko-Cantelli class. We make these complexity assumptions just to enable a simple algorithm. We also describe a more complicated algorithm with no dependence on the complexity of $\mathcal{F}$ in the appendix.

[8] We allow the ℓ_∞ bound of 2 to accommodate these functions whose range can be 2 under our assumptions.

Algorithm 8 Inputs: $(\mathcal{M}, \mathcal{F}, n, n_e, \epsilon, \delta, \phi)$

```
 1: $\mathcal{M}_0 = \mathcal{M}$
 2: for $t = 1, 2, ...$ do
 3:     Choose model optimistically: $M^t = \mathrm{argmax}_{M \in \mathcal{M}_{t-1}} v_M$, set $\pi^t = \pi_{M^t}$
 4:     Execute $\pi^t$ to collect $n_e$ trajectories $\{(x_h^i, a_h^i, r_h^i)_{h=1}^H\}_{i=1}^{n_e}$ and set $\hat{v}^{\pi^t} = \frac{1}{n_e}\sum_{i=1}^{n_e}\left(\sum_{h=1}^H r_h^i\right)$
 5:     if $|\hat{v}^{\pi^t} - v_{M^t}| \le \epsilon/2$ then Terminate and output $\pi^t$ end if
 6:     Find $h_t$ such that $\widehat{\mathcal{E}}_B(M^t, M^t, h_t) \ge \frac{\epsilon}{4H}$ (See (6.6))
 7:     Collect trajectories $\{(x_h^{(i)}, a_h^{(i)}, r_h^{(i)})_{h=1}^H\}_{i=1}^n$ where $a_h^{(i)} \sim \pi^t$ for $h \ne h_t$ and $a_{h_t}^{(i)} \sim U(\mathcal{A})$
 8:     for $M' \in \mathcal{M}_{t-1}$ do Compute $\widehat{\mathcal{W}}(M^t, M', h_t)$ (See (6.5)) end for
 9:     Set $\mathcal{M}_t = \{M \in \mathcal{M}_{t-1} : \widehat{\mathcal{W}}(M^t, M, h_t) \le \phi\}$
10: end for
```

form an unbiased estimate of $\mathcal{E}_B(M, M, h)$ as

$$\widehat{\mathcal{E}}_B(M, M, h) \triangleq \frac{1}{N}\sum_{n=1}^{N}\left[Q_M(x_h^{(n)}, a_h^{(n)}) - \left[r_h^{(n)} + V_M(x_{h+1}^{(n)})\right]\right]. \tag{6.6}$$

Here V_M, Q_M are obtained through a query to `OP` using model M.

The pseudocode is displayed in Algorithm 8. The algorithm is round-based, maintaining a version space of models and eliminating a model from the version space when the discrepancy between the model and the ground truth $M^\star$ is witnessed. The witness distributions are selected using a form of *optimism*: at each round, we select, from all surviving models, the one with the highest predicted value, and use the associated policy for data collection. If the policy achieves a high value in the environment, we simply return it. Otherwise we estimate the witnessed model misfit on the context distributions induced by the policy, and we shrink the version space by eliminating all incorrect models. Then we proceed to the next iteration.

Intuitively, using a simulation lemma analogous to Lemma 1 of Jiang et al. [2017], if M^t is the optimistic model at round t and we do not terminate, then there must exist a time step h_t (line 6) where the average Bellman error is large. Using Assumption 3, this also implies that the witness model misfit for M^t witnessed by M^t itself must be large. Thus, if t is a non-terminal round, we ensure that M^t and potentially many other models are eliminated.

The algorithm is similar to Olive, which uses average Bellman error instead of witnessed model misfit to shrink the version space. However, by appealing to Assumption 3, witness model misfit provides a more aggressive elimination criterion, since a large average Bellman error on a distribution immediately implies a large witnessed model misfit on the same distribution, but the converse does not necessarily hold. Since the algorithm uses an aggressive elimination rule, it often requires fewer iterations than Olive, as discussed below.

Computational considerations. In this work, we focus on the sample complexity of model-based RL, and Algorithm 8, as stated, admits no obvious efficient implementation. The main bottleneck, for efficiency, is the optimistic computation of the next model in line 3 where we perform a constrained optimization, restricted to the class of all models not eliminated so far. The objective in this problem encapsulates a planning oracle to map models from our class to their values, and the constraints involve enforcing small values of witness model misfit on the prior context distributions. While the witness model misfit is linear in the transition dynamics, finding an optimistic value function induces bilinear, non-convex constraints even in a tabular setting. This resembles known computational difficulties with Olive, but we note that the recent hardness result of Dann et al. [2018] for Olive does not apply to Algorithm 8, leaving the possibility of an efficient implementation open.

6.5.2 A structural complexity measure

So far, we have imposed realizability and expressivity assumptions (Assumption 2 and Assumption 3) on $\mathcal{M}$ and $\mathcal{F}$. Unfortunately, these alone do not enable tractable reinforcement learning with polynomial sample complexity, as verified by the following simple lower bound.

Proposition 2. *Fix $H, K \in \mathbb{N}^+$ with $K \geq 2$ and $\epsilon \in (0, \sqrt{1/8})$. There exists a family of MDPs, classes $\mathcal{M}, \mathcal{F}$ satisfying Assumption 2 and Assumption 3 for all MDPs in the family with $|\mathcal{M}| = |\mathcal{F}| = K^{H-1}$, and a constant $c > 0$, such that the following holds: For any algorithm that takes $\mathcal{M}$, $\mathcal{F}$ as inputs and uses $T \leq cK^{H-1}/\epsilon^2$ episodes, the algorithm outputs a policy $\hat{\pi}$ with $v^{\hat{\pi}} < v^\star - \epsilon$ with probability at least $1/3$ for some MDP in the family.*

The proof, provided in Appendix 9.5.3, adapts a construction from Krishnamurthy et al. [2016] for showing that value-based realizability is insufficient for model-free algorithms. The result suggests that we must introduce further structure to obtain polynomial sample complexity guarantees. We do so with a new structural complexity measure, the *witness rank*.

For any matrix $B \in \mathbb{R}^{n\times n}$, define $\text{rank}(B, \beta)$ to be the smallest integer k such that $B = UV^\top$ with $U, V \in \mathbb{R}^{n\times k}$ and for every pair of rows u_i, v_j, we have $\|u_i\|_2 \cdot \|v_j\|_2 \leq \beta$. This generalizes the standard definition of matrix rank, with a condition on the row norms of the factorization.

Definition 5 (Witness Rank). *Given a model class $\mathcal{M}$, test functions $\mathcal{F}$, and $\kappa \in (0, 1]$, for $h \in [H]$, define the set of matrices $\mathcal{N}_{\kappa,h}$ such that any matrix $A \in \mathcal{N}_{\kappa,h}$ satisfies:*

$$A \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{M}|}, \quad \kappa\mathcal{E}_B(M, M', h) \leq A(M, M') \leq \mathcal{W}(M, M', h), \forall M, M' \in \mathcal{M},$$

We define the witness rank as

$$\text{W}(\kappa, \beta, \mathcal{M}, \mathcal{F}, h) \triangleq \min_{A\in\mathcal{N}_{\kappa,h}} \textit{rank}(A, \beta).$$

We typically suppress the dependence on β because it appears only logarithmically in our sample complexity bounds. Any β that is polynomial in other parameters (K, H, and the rank itself) suffices.

To build intuition for the definition, first consider the extreme where $A(M, M') = \mathcal{W}(M, M', h)$. The rank of this matrix corresponds to the number of context distributions required to verify non-zero witnessed model misfit for all incorrect models. This follows from the fact that there are at most $\text{rank}(\mathcal{W})$ linearly independent *rows* (context distributions), so any non-zero column (an incorrect model) must have a non-zero in at least one of these rows. Algorithmically, if we can find the policies π_M corresponding to these rows, we can eliminate *all* incorrect models to find $M^\star$ and hence $\pi^\star$.

At the other extreme, we have $A(M, M') = \kappa\mathcal{E}_B(M, M', h)$, the *Bellman error matrix* . The rank of this matrix, called *Bellman rank*, provides an upper bound on the witness rank by construction, and is known to be small for many natural RL settings, including tabular and low-rank MDPs, reactive POMDPs, and reactive PSRs (see Section 2 of Jiang et al. [2017] for details). The minimum over all sandwiched A matrices in the definition of the witness rank allows a smooth interpolation between these extremes in general. We further note that the choice of the class $\mathcal{F}$ defining the IPM also affects the witness model misfit and hence the witness rank. Adapting this class to the problem structure yields another useful knob to control the witness rank, as we show for the running example of factored MDPs in Section 6.6.

6.5.3 Sample complexity results

We now present a sample complexity guarantee for Algorithm 8 using the witness rank. Denote $\mathrm{W}_\kappa \triangleq \max_{h\in[H]} \mathrm{W}(\kappa, \beta, \mathcal{M}, \mathcal{F}, h)$. The main guarantee is the following theorem.

Theorem 20. *Under Assumption 2 and Assumption 3, for any $\epsilon, \delta, \kappa \in (0, 1]$, set $\phi = \frac{\kappa\epsilon}{48H\sqrt{\mathrm{W}_\kappa}}$, and denote $T = H\mathrm{W}_\kappa \log(\beta/2\phi)/\log(5/3)$. Run Algorithm 8 with inputs $(\mathcal{M}, \mathcal{F}, n_e, n, \epsilon, \delta, \phi)$, where $n_e = \Theta\big(H^2 \log(HT/\delta)/\epsilon^2\big)$ and $n = \Theta\big(H^2 K\mathrm{W}_\kappa \log(T|\mathcal{M}||\mathcal{F}|/\delta)/(\kappa\epsilon)^2\big)$. Then with probability at least $1 - \delta$, Algorithm 8 outputs a policy π such that $v^\pi \geq v^\star - \epsilon$. The number of trajectories collected is at most $\tilde{O}\left(\frac{H^3\mathrm{W}_\kappa^2 K}{\kappa^2\epsilon^2} \log\left(\frac{T|\mathcal{F}||\mathcal{M}|}{\delta}\right)\right)$.*

The proof is included in Appendix 9.5.1. Since, as we have discussed, many popular RL models admit low Bellman rank and hence low witness rank, Theorem 20 verifies that Algorithm 8 has polynomial sample complexity in all of these settings. A noteworthy case that does not have small Bellman rank but does have small witness rank is the factored MDP, which we discuss further in Section 6.6.

Comparison with Olive. The minimum sample complexity is achieved at $\inf_\kappa \mathrm{W}_\kappa/\kappa$, which is never larger than the Bellman rank. In fact when $\kappa = 1$, the sample complexity bounds match in all terms except (a) we replace Bellman rank with witness rank, and (b) we have a dependence on model and test-function complexity $\log(|\mathcal{M}||\mathcal{F}|)$ instead of Q-function complexity $\log|\mathtt{OP}(\mathcal{M})|$. The witness rank is never larger than the Bellman rank and it can be substantially smaller, which is favorable for Algorithm 8. However, we always have $\log|\mathcal{M}| \geq \log|\mathtt{OP}(\mathcal{M})|$ and since we require realizability, the model class can be much larger than the induced Q-function class. Thus the two results are in general incomparable, but for problems where modeling the environment is not much harder than modeling the optimal Q-function (in other words $\log(|\mathcal{M}||\mathcal{F}|) \approx \log|\mathtt{OP}(\mathcal{M})|$), Algorithm 8 can be substantially more sample-efficient than Olive.

Adapting to unknown witness rank. In Theorem 20, the algorithm needs to know the value of κ and W_κ, as they are used to determine ϕ and n. In Appendix 9.5.9, we show that a standard doubling trick can adapt to unknown κ and W_κ. The sample complexity for this adaptation is given by
$\tilde{O}(H^3\mathrm{W}_{\kappa^\star}^2 K/((\kappa^\star\epsilon)^2)\log(|\mathcal{M}||\mathcal{F}|/\delta))$, where $\kappa^\star \triangleq \arg\min_{\kappa\in(0,1]} \mathrm{W}_\kappa/\kappa$ minimizes the bound in Theorem 20. A similar technique was used to adapt Olive to handle unknown Bellman rank.

Extension to infinite $\mathcal{M}$. Theorem 20 as stated assumes that $\mathcal{M}$ and $\mathcal{F}$ are finite classes. It is desirable to allow rich classes $\mathcal{M}$ to have a better chance of satisfying realizability of $M^\star$ in Assumption 2. Indeed, it is possible to use standard covering arguments to handle the case of infinite $\mathcal{M}$. We demonstrate this in Theorem 22, where we study factored MDPs with $\mathcal{M}$ as the continuous space of all transition models obeying the factored structure. More generally, the rates obtained here hold up to an added model class complexity term, as long as the class $\mathcal{M}$ is parametric. As is standard, nonparametric model classes result in a loss of rates.

Extension to infinite $\mathcal{F}$. The result extends naturally to function classes $\mathcal{F}$ with bounded statistical complexity (e.g., VC Dimension, Rademacher Complexity) in a similar manner as discussed above. However, it is desirable to move beyond classes with bounded statistical complexity, especially for the test function class $\mathcal{F}$. For example, we would like to handle the total variation distance, which is the IPM induced by $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$. This class does not admit uniform convergence, so our analysis for Theorem 20 does not apply. To handle such rich classes, we borrow ideas from the Scheffé estimator and tournament of Devroye and Lugosi [2012],[9] and extend the method to handle conditional distributions and IPMs induced by an arbitrary class. The analysis here covers the total-variation based witnessed model misfit defined in (6.3) as a special case.

Theorem 21. *Under Assumption 2 and Assumption 3, but with no restriction on size of $\mathcal{F}$,[10] there exists an algorithm such that: For any $\epsilon, \delta \in (0, 1]$, with probability at least $1 - \delta$ the algorithm outputs a policy π such that $v^\pi \ge v^\star - \epsilon$ with at most $\tilde{O}\left(\frac{H^3 \mathrm{W}_\kappa^2 K}{\kappa^2 \epsilon^2} \log\left(\frac{T|\mathcal{M}|}{\delta}\right)\right)$ trajectories collected, where $T = H\mathrm{W}_\kappa \log(\beta/2\phi)/\log(5/3)$.*

The algorithm modifies Algorithm 8 to incorporate the Scheffé estimator instead of the direct empirical estimate for the witnessed model misfit (6.5). We defer the details of the algorithm and analysis to Appendix 9.5.2. The main improvement over Theorem 20 is that the sample complexity here has no dependence on $\mathcal{F}$, so we may use test function classes with unbounded statistical complexity.

6.6 Case Study on MDPs with Factored Transitions

In this section, we study the running example of factored MDPs in detail. Recall the definition of factored transition dynamics in (6.1). Following the formulation of Kearns and Koller [1999], we assume the reward distribution $R^\star$ is known and shared by all models in $\mathcal{M}$, so that models only disagree on the transition dynamics. For this setting, we have the following guarantee.

Theorem 22. *For MDPs with factored transitions and for any $\epsilon, \delta \in (0, 1]$, with probability at least $1 - \delta$ a modification of Algorithm 8 (Algorithm 12 in Appendix 9.5.5) outputs a policy π with $v^\pi \ge v^\star - \epsilon$ using at most $\tilde{O}(d^2 L^3 H K^2 \log(1/\delta)/\epsilon^2)$ trajectories.*

This result should be contrasted with the $\Omega(2^H)$ lower bound from Theorem 19 that actually applies precisely to this setting, where the lower bound construction has description length L polynomial in H (see Appendix 9.5.3 for details). Combining the two results

[9]The classical Scheffé tournament targets the following problem: given a set of distributions $\{P_i\}_{i=1}^K$ over $\mathcal{X}$, and a set of i.i.d samples $\{x_i\}_{i=1}^N$ from $P^\star \in \Delta(\mathcal{X})$, approximate the minimizer $\mathrm{argmin}_{i\in[K]} \|P_i - P^\star\|_{\mathrm{TV}}$.

[10]In fact, Assumption 3 holds automatically if we choose $\mathcal{F} = \{f : \|f\|_\infty \le 2\}$.

we have demonstrated exponential separation between model-based and model-free algorithms for MDPs with factored transitions.

Corollary 2. *For MDPs with factored transitions, the sample complexity of model-based methods is polynomial in all parameters, while all model-free methods must incur $\Omega(2^H)$ sample complexity.*

Comparing with Theorem 20, the main improvement is that we are working with an infinite model class of all possible factored transition operators. The linear scaling with H, which seems to be an improvement, is purely cosmetic as we have $L = \Omega(H)$ here. Theorem 22 involves a slight modification to Algorithm 8, in that it uses a slightly different notion of witnessed model misfit,

$$\mathcal{W}_F(M, M', h) = \max_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{\substack{x_h \sim \pi_M \\ a_h \sim U(\mathcal{A})}} \left[\mathop{\mathbb{E}}_{(r,x') \sim M'_h} [f(x_h, a_h, r, x')] - \mathop{\mathbb{E}}_{(r,x') \sim M^\star_h} [f(x_h, a_h, r, x')] \right]. \tag{6.7}$$

together with an $\mathcal{F}$ specially designed for factored MDPs (subscript of $\mathcal{W}_F$ indicates adaptation to factored MDPs). The main difference with (6.3) is that a_h is sampled from $U(\mathcal{A})$ rather than $\pi_{M'}$. This modification is crucial to obtain a low witness rank, since $\pi_{M'}$ is in general not guaranteed to be factored (recall the representation hardness discussed at the end of Section 6.3). Thanks to uniformly random actions and our choice of $\mathcal{F}$, $\mathcal{W}_F$ essentially computes the sum of the TV-distances across all factors, and the corresponding matrix naturally factorizes and yields low witness rank. On the other hand, the choice of $\pi_{M'}$ for the general case allows a direct comparison with Bellman rank and leads to better guarantees in general, so we do not use the definition (6.7) more generally. We defer the details of the algorithm and its analysis to Appendix 9.5.5.

6.7 Related Work

For tabular MDPs, a number of sample-efficient RL approaches exist, mostly model-based [Azar et al., 2017, Dann and Brunskill, 2015, Jaksch et al., 2010, Kearns and Singh, 2002, Szita and Szepesvári, 2010], but some are model-free [Jin et al., 2018, Strehl et al., 2006]. In contrast, our work focuses on more realistic rich-observation settings.[11] For our running example of factored MDPs, prior sample-efficient algorithms are model-based [Kearns and Koller, 1999, Osband and Van Roy, 2014b], and, as we show here, no sample-efficient model-free algorithms can exist. With rich observations, many prior works either focus on structured control settings like LQRs [Abbasi-Yadkori and Szepesvári, 2011, Dean et al., 2018] or Lipschitz-continuous MDPs [Kakade et al., 2003, Lakshmanan et al., 2015, Ortner and Ryabko, 2012, Pazis and Parr, 2013]. In LQRs, Tu and Recht [2018] show a gap between model-based and a particular model-free algorithm, but not an algorithm agnostic lower bound, as we show here for factored MDPs. We expect that our algorithm or natural variants are sample-efficient in many of these specific settings.

In more abstract settings, most sample-efficient algorithms are model-free [Dann et al., 2018, Jiang et al., 2017, Krishnamurthy et al., 2016, Wen and Van Roy, 2013]. Our work can be seen as a model-based analog to Jiang et al. [2017], which among the above references, studies the most general class of environments.

[11] In fact, our information-theoretic definition of model-free methods (Definition 3) is uninteresting in the tabular setting.

On the model-based side, Lattimore et al. [2013] and Osband and Van Roy [2014a] obtain sample complexity guarantees; the former makes no assumptions but the guarantee scales linearly with the model class size, and the latter makes continuity assumptions, so both results have more limited scope than ours. Ok et al. [2018] propose a complexity measure for structured RL problems, but their results are for asymptotic regret in tabular or Lipschitz MDPs.

On the empirical side, models are often used to speed up learning [see e.g., Aboaf et al., 1989, Deisenroth et al., 2011, for classical references in robotics]. Such results provide empirical evidence that models can be statistically valuable, which complement our theoretical results.

Finally, two recent papers share some technical similarities to our work. Farahmand et al. [2017] also use IPMs to detect model error but their analysis is restricted to test functions that form a ball in an RKHS, and they do not address exploration issues. Xu et al. [2018] devise a model-based algorithm with function approximation, but their algorithm performs local policy improvement and cannot find a globally optimal policy in a sample-efficient manner.

6.8 Discussion

In this chapter, we study model-based reinforcement learning in general contextual decision processes. We derive an algorithm for general CDPs and prove that it has sample complexity upper-bounded by a new structural notion called the witness rank, which is small in many settings of interest. Comparing model-based and model-free methods, we show that model-based methods can be exponentially more sample efficient in some settings, but they also require stronger function-approximation capabilities, which can result in worse sample complexity in other cases. Comparing the guarantees here with those derived by Jiang et al. [2017] precisely quantifies these tradeoffs, which we hope guides future design of RL algorithms.

Chapter 7

Towards Practically Efficient Model-Based RL: Dual Policy Iteration

A novel class of Approximate Policy Iteration (API) algorithms have recently demonstrated impressive practical performance (*e.g.*, ExIt Anthony et al. [2017], AlphaGo-Zero Silver et al. [2017]). This new family of algorithms maintains, and alternately optimizes, two policies: a fast, reactive policy (*e.g.*, a deep neural network) deployed at test time, and a slow, non-reactive policy (e.g., Tree Search), that can plan multiple steps ahead. The reactive policy is updated under supervision from the non-reactive policy, while the non-reactive policy is improved via guidance from the reactive policy. In this work we study this class of Dual Policy Iteration (DPI) strategy in an *alternating optimization framework* and provide a convergence analysis that extends existing API theory. We also develop a special instance of this framework which reduces the update of non-reactive policies to model-based optimal control using learned local models, and provides a theoretically sound way of unifying model-free and model-based RL approaches with unknown dynamics. We demonstrate the efficacy of our approach on various continuous control Markov Decision Processes.

The contributions in this chapter were first introduced in the joint work with Byron Boots and Drew Bagnell [Sun et al., 2018c]

7.1 Introduction

Approximate Policy Iteration (API) Bagnell et al. [2004], Bertsekas and Tsitsiklis [1995], Kakade and Langford [2002], Lazaric et al. [2010], Scherrer [2014], including conservative API (CPI) Kakade and Langford [2002], API driven by learned critics Rummery and Niranjan [1994], or gradient-based API with stochastic policies Bagnell and Schneider [2003], Baxter and Bartlett [2001], Kakade [2002], Schulman et al. [2015a], have played a central role in Reinforcement Learning (RL) for decades and motivated many modern practical RL algorithms. Several existing API methods Bagnell et al. [2004], Kakade and Langford [2002] can provide both local optimality guarantees and global guarantees under strong assumptions regarding the way samples are generated (e.g., access to a reset distribution that is similar to the optimal policy's state distribution). However, most modern practical API algorithms rely on myopic random exploration (e.g., REINFORCE Williams [1992] type policy gradient or ϵ-greedy). Sample inefficiency due to random exploration can cause even sophisticated RL methods to perform worse than simple black-box optimization with random search in parameter space Mania et al. [2018].

Recently, a new class of API algorithms, which we call *Dual Policy Iteration* (DPI), has begun to emerge. These algorithms follow a richer strategy for improving the policy, with two policies under consideration at any time during training: a reactive policy, usually learned by some form of function approximation, used for generating samples and deployed at test time, and an intermediate policy that can only be constructed or accessed during training, used as an expert policy to guide the improvement of the reactive policy. For example, ExIt Anthony et al. [2017] maintains and updates a UCT-based policy Kocsis and Szepesvári [2006] as an intermediate expert. ExIt then updates the reactive policy by directly imitating the tree-based policy which we expect would be *better* than the reactive policy as it involves a multi-step lookahead search. AlphaGo-Zero Silver et al. [2017] employs a similar strategy to achieve super-human performance at the ancient game of Go. The key difference that distinguishes ExIt and AlphaGo-Zero from previous APIs is that they *leverage models to perform systematic forward search*: the policy resulting from forward search acts as an expert and directly informs the improvement direction for the reactive policy. Hence the reactive policy improves by imitation instead of trial-and-error reinforcement learning. This strategy often provides better sample efficiency in practice compared to algorithms that simply rely on locally random search (e.g., AlphaGo-Zero abandons REINFORCE from AlphaGo Silver et al. [2016]).

In this work we provide a general framework for synthesizing and analyzing DPI by considering a particular *alternating optimization strategy* with different optimization approaches each forming a new family of approximate policy iteration methods. We additionally consider the extension to the RL setting with *unknown dynamics*. For example, we construct a simple instance of our framework, where the intermediate expert is computed from *Model-Based Optimal Control* (MBOC) locally around the reactive policy, and the reactive policy in turn is updated incrementally under the guidance of MBOC. The resulting algorithm iteratively learns a local dynamics model, applies MBOC to compute a locally optimal policy, and then updates the reactive policy by imitation and achieve larger policy improvement per iteration than classic APIs. The instantiation shares similar spirit from some previous works from robotics and control literature, including works from Atkeson [1994], Atkeson and Morimoto [2003] and Guided Policy Search (GPS) Levine and Abbeel [2014] (and its variants (e.g., Levine et al. [2016], Montgomery et al. [2017], Montgomery and Levine [2016])), i.e., using local MBOC to speed up learning global policies.

To evaluate our approach, we demonstrate our algorithm on discrete MDPs and continuous control tasks. We show that by integrating local model-based search with learned local dynamics into policy improvement via an imitation learning-style update, our algorithm is substantially more sample-efficient than classic API algorithms such as CPI Kakade and Langford [2002], as well as more recent actor-critic baselines Schulman et al. [2015b], albeit at the cost of slower computation per iteration due to the model-based search. We also apply the framework to a *robust policy optimization* setting Atkeson [2012], Bagnell and Schneider [2001] where the goal is to learn a *single* policy that can generalize across environments. In summary, the major practical difference between DPI and many modern practical RL approaches is that instead of relying on random exploration, the DPI framework integrates local model learning, local model-based search for advanced exploration, and an imitation learning-style policy improvement, to improve the policy in a more systematic way.

We also provide a general convergence analysis to support our empirical findings. Although our analysis is similar to CPI's, it has a key difference: as long as MBOC succeeds, we can provide a larger policy improvement than CPI at each iteration. Our analysis is general enough to provide theoretical intuition for previous successful practical DPI algorithms such as Expert Iteration (ExIt) Anthony et al. [2017]. We also analyze how predictive error from a learned local model can mildly affect policy improvement and show that locally accurate dynamics—a model that accurately predicts next states *under the current policy's state-action distribution*, is enough for improving the current policy. We believe our analysis of local model predictive error versus local policy improvement can shed light on further development of model-based RL approaches with learned local models. In summary, DPI operates in the middle of two extremes: (1) API type methods that update policies locally (e.g., first-order methods like policy gradient and CPI), (2) global model-based optimization where one attempts to learn a global model and perform model-based search. First-order methods have small policy improvement per iteration and learning a global model displays greater *model bias* and requires a dataset that covers the entire state space. DPI instead learns a local model and allows us to integrate models to leverage the power of model-based optimization to locally improve the reactive policy.

7.2 Preliminaries

A discounted infinite-horizon Markov Decision Process (MDP) is defined as $(\mathcal{S}, \mathcal{A}, P, c, \rho_0, \gamma)$. Here, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, and P is the transition dynamics: $P(s'|s,a)$ is the probability of transitioning to state s' from state s by taking action a. We use $P_{s,a}$ in short for $P(\cdot|s,a)$. We denote $c(s,a)$ as the cost of taking action a while in state s. Finally, ρ_0 is the initial distribution of states, and $\gamma \in (0,1)$ is the discount factor. Throughout this paper, we assume that we *know* the form of the cost function $c(s,a)$, but the transition dynamics P are *unknown*.We define a stochastic policy π such that for any state $s \in \mathcal{S}$, $\pi(\cdot|s)$ outputs a distribution over action space. The distribution of states at time step t, induced by running the policy π until and including t, is defined $\forall s_t$: $d_\pi^t(s_t) = \sum_{\{s_i,a_i\}_{i\leq t-1}} \rho_0(s_0)\prod_{i=0}^{t-1}\pi(a_i|s_i)P(s_{i+1}|s_i,a_i)$, where by definition $d_\pi^0(s) = \rho_0(s)$ for any π. The state visitation distribution can be computed $d_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_\pi^t(s)$. Denote $(d_\pi\pi)$ as the joint state-action distribution such that $d_\pi\pi(s,a) = d_\pi(s)\pi(a|s)$. We define the value function $V^\pi(s)$, state-action value function $Q^\pi(s,a)$, and the objective function $J(\pi)$ as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)|s_0 = s\right], Q^\pi(s,a) = c(s,a) + \gamma\mathbb{E}_{s'\sim P_{s,a}}\left[V^\pi(s')\right], J(\pi) = \mathbb{E}_{s\sim\rho_0}[V^\pi(s)].$$

With V^π and Q^π, the advantage function $A^\pi(s,a)$ is defined as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. As we work in the cost setting, in the rest of the paper we refer to A^π as the *disadvantage* function. The goal is to learn *a single stationary* policy π^* that minimizes $J(\pi)$: $\pi^* = \arg\min_{\pi\in\Pi} J(\pi)$.

For two distributions P_1 and P_2, $D_{TV}(P_1, P_2)$ denotes *total variation distance*, which is related to the L_1 norm as $D_{TV}(P_1, P_2) = \|P_1 - P_2\|_1/2$ (if we have a finite probability space) and $D_{KL}(P_1, P_2) = \int_x P_1(x)\log(P_1(x)/P_2(x))\mathrm{d}x$ denotes the KL divergence.

We introduce *Performance Difference lemma* (PDL) Kakade and Langford [2002], which will be used extensively in this work:

Lemma 7. *For any two policies π and π', we have: $J(\pi) - J(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d_\pi \pi}\left[A^{\pi'}(s,a)\right]$.*

7.3 Dual Policy Iteration

We propose an alternating optimization framework inspired by the PDL (Lemma 7). Consider the min-max optimization framework: $\min_{\pi\in\Pi}\max_{\eta\in\Pi}\mathbb{E}_{s\sim d_\pi}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A^\eta(s,a)\right]\right]$. It is not hard to see that the unique Nash equilibrium for the above equation is $(\pi,\eta) = (\pi^*,\pi^*)$. The above min-max proposes a general strategy, which we call Dual Policy Iteration (DPI): alternatively fix one policy and update the second policy. Mapping to previous practical DPI algorithms Anthony et al. [2017], Silver et al. [2017], π stands for the fast reactive policy and η corresponds to the tree search policy. For notation purposes, we use π_n and η_n to represent the two policies in the nth iteration. Below we introduce one instance of DPI for settings with unknown models (hence no tree search), first describe how to compute η_n from a given reactive policy π_n (Sec. 7.3.1), and then describe how to update π_n to π_{n+1} via imitating η_n (Sec. 7.3.2).

7.3.1 Updating η with MBOC using Learned Local Models

Given π_n, the objective function for η becomes: $\max_\eta \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_n(\cdot|s)}\left[A^\eta(s,a)\right]\right]$. From PDL we can see that updating η is equivalent to finding the optimal policy π^*: $\arg\max_\eta\left(J(\pi_n) - J(\eta)\right) \equiv \arg\min_\eta J(\eta)$, regardless of what π_n is. As directly minimizing $J(\eta)$ is as hard as the original problem, we update η locally by constraining it to a trust region around π_n:

$$\arg\min_\eta J(\eta), \quad s.t., \mathbb{E}_{s\sim d_{\pi_n}} D_{TV}[(\eta(\cdot|s), \pi_n(\cdot|s))] \le \alpha. \tag{7.1}$$

To solve the constraint optimization problem in Eq 7.1, we propose to learn $P_{s,a}$ and use it with any off-the-shelf model-based optimal control algorithm. Moreover, thanks to the trust region, we can simply learn a *local* dynamics model, *under the state-action distribution $d_{\pi_n}\pi_n$*. We denote the optimal solution to the above constrained optimization (Eq. 7.1) under the *real* model $P_{s,a}$ as η_n^*. Note that, due to the definition of the optimality, η_n^* must perform better than π_n: $J(\pi_n) - J(\eta_n^*) \ge \Delta_n(\alpha)$, where $\Delta_n(\alpha) \ge 0$ is the performance gain from η_n^* over π_n. When the trust region expands, i.e., α increases, then $\Delta_n(\alpha)$ approaches the performance difference between the optimal policy π^* and π_n.

To perform MBOC, we learn a locally accurate model—a model $\hat{P}$ that is close to P *under the state-action distribution induced by π_n*: we seek a model $\hat{P}$, such that the quantity $\mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n} D_{TV}(\hat{P}_{s,a}, P_{s,a})$ is small. Optimizing D_{TV} directly is hard, but note that, by

Pinsker's inequality, we have $D_{KL}(P_{s,a}, \hat{P}_{s,a}) \geq D_{TV}(\hat{P}_{s,a}, P_{s,a})^2$, which indicates that we can optimize a surrogate loss defined by a KL-divergence:

$$\arg\min_{\hat{P}\in\mathbf{P}} \mathbb{E}_{s\sim d_{\pi_n}, a\sim\pi_n(s)} D_{KL}(P_{s,a}, \hat{P}_{s,a}) = \arg\min_{\hat{P}\in\mathbf{P}} \mathbb{E}_{s\sim d_{\pi_n}, a\sim\pi_n(s), s'\sim P_{s,a}}[-\log \hat{P}_{s,a}(s')], \tag{7.2}$$

where we denote $\mathbf{P}$ as the model class. Hence we reduce the local model fitting problem into a classic maximum likelihood estimation (MLE) problem, where the training data $\{s, a, s'\}$ can be easily collected by executing π_n on the real system (i.e., $P_{s,a}$). As we will show later, to ensure policy improvement, we just need a learned model to perform well under $d_{\pi_n}\pi_n$ (i.e., no training and testing distribution mismatch as one will have for global model learning). For later analysis purposes, we denote $\hat{P}$ as the MLE in Eq. 7.2 and assume $\hat{P}$ is δ-optimal under $d_{\pi_n}\pi_n$:

$$\mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n} D_{TV}(\hat{P}_{s,a}, P_{s,a}) \leq \delta, \tag{7.3}$$

where $\delta \in \mathbb{R}^+$ is controlled by the complexity of model class $\mathbf{P}$ and by the amount of training data we sample using π_n, which can be analyzed by standard supervised learning theory. After achieving a locally accurate model $\hat{P}$, we solve Eq. 7.1 using any existing stochastic MBOC solvers. Assume a MBOC solver returns an optimal policy η_n under the estimated model $\hat{P}$ subject to trust-region:

$$\eta_n = \arg\min_{\pi} J(\pi), s.t., \ s_{t+1} \sim \hat{P}_{s_t,a_t}, \ \mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi, \pi_n) \leq \alpha. \tag{7.4}$$

At this point, a natural question is: If η_n is solved by an MBOC solver under $\hat{P}$, by how much can η_n outperform π_n when *executed under the real dynamics* P? Recall that the performance gap between the real optimal solution η_n^* (optimal under P) and π_n is denoted as $\Delta_n(\alpha)$. The following theorem quantifies the performance gap between η_n and π_n using the learned local model's predictive error δ:

Theorem 23. *Assume $\hat{P}_{s,a}$ satisfies Eq. 7.3, and η_n is the output of a MBOC solver for the optimization problem defined in Eq. 7.4, then we have:*

$$J(\eta_n) \leq J(\pi_n) - \Delta_n(\alpha) + O\left(\frac{\gamma\delta}{1-\gamma} + \frac{\gamma\alpha}{(1-\gamma)^2}\right).$$

The proof of the above theorem can be found in Appendix 9.6.2. Theorem 23 indicates that when the model is *locally accurate*, i.e., δ is small (e.g., $\mathbf{P}$ is rich and we have enough data from $d_{\pi_n}\pi_n$), α is small, and there exists a local optimal solution that is significantly better than the current policy π_n (i.e., $\Delta_n(\alpha) \in \mathbb{R}^+$ is large), then the OC solver with the learned model $\hat{P}$ finds a nearly local-optimal solution η_n that outperforms π_n. With a better η_n, now we are ready to improve π_n via imitating η_n.

7.3.2 Updating π via Imitating η

Given η_n, we compute π_{n+1} by performing the following constrained optimization procedure:

$$\arg\min_{\pi} \mathbb{E}_{s\sim d_{\pi_n}} \left[\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A^{\eta_n}(s,a)\right]\right], s.t., \mathbb{E}_{s\sim d_{\pi_n}}\left[D_{TV}(\pi(\cdot|s), \pi_n(\cdot|s))\right] \leq \beta \tag{7.5}$$

Note that the key difference between Eq. 7.5 and classic API policy improvement procedure is that we use η_n's disadvantage function A^{η_n}, i.e., we are performing imitation learning by treating η_n as an expert in this iteration Ross and Bagnell [2014], Sun et al. [2017c]. We can solve Eq. 7.5 by converting it to supervised learning problem such as cost-sensitive classification Kakade and Langford [2002] by sampling states and actions from π_n and estimating A^{η_n} via rolling out η_n, subject to an L1 constraint.

Note that a CPI-like update approximately solves the above constrained problem as well:

$$\pi_{n+1} = (1-\beta)\pi_n + \beta\pi_n^*, \text{ where } \pi_n^* = \arg\min_{\pi} \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\eta_n}(s,a)]\right]. \tag{7.6}$$

Note that π_{n+1} satisfies the constraint as $D_{TV}(\pi_{n+1}(\cdot|s), \pi_n(\cdot|s)) \leq \beta, \forall s$. Intuitively, the update in Eq. 7.6 can be understood as first solving the objective function to obtain π_n^* without considering the constraint, and then moving π_n towards π_n^* until the boundary of the constraint is reached.

7.3.3 DPI: Combining Updates on π and η

In summary, assume MBOC is used for Eq. 7.1, DPI operates in an iterative way: with π_n:

1. Fit MLE $\hat{P}$ on states and actions from $d_{\pi_n}\pi_n$ (Eq. 7.2).
2. $\eta_n \leftarrow$ MBOC($\hat{P}$), subject to trust region $\mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi, \pi_n) \leq \alpha$ (Eq. 7.4)
3. Update to π_{n+1} by imitating η_n, subject to trust region $\mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi, \pi_n) \leq \beta$ (Eq. 7.5).

The above framework shows how π and η are tightened together to guide each other's improvements: the first step corresponds classic MLE under π_n's state-action distribution: $d_{\pi_n}\pi_n$; the second step corresponds to model-based policy search around π_n ($\hat{P}$ is only locally accurate); the third step corresponds to updating π by imitating η (i.e., imitation). Note that in practice MBOC solver (e.g., a second order optimization method, as we will show in our practical algorithm below) could be computationally expensive and slow (e.g. tree search in ExIt and AlphaGo-Zero), but once $\hat{P}$ is provided, MBOC does not require additional samples from the real system.

Connections to Previous works We can see that the above framework generalizes several previous work from API and IL. **(a)** If we set $\alpha = 0$ in the limit, we reveal CPI (assuming we optimize with Eq. 7.6), i.e., no attempt to search for a better policy using model-based optimization. **(b)** Mapping to ExIt, our η_n plays the role of the tree-based policy, and our π_n plays the role of the apprentice policy, and MBOC plays the role of forward search. **(c)** when an optimal expert policy π^* is available during and only during training, we can set every η_n to be π^*, and DPI then reveals a previous IL algorithm—AggreVaTeD Sun et al. [2017c].

7.4 Analysis of Policy Improvement

We provide a general convergence analysis for DPI. The trust region constraints in Eq. 7.1 and Eq. 7.5 tightly combines MBOC and policy improvement together, and is the key to ensure monotonic improvement and achieve larger policy improvement per iteration than existing APIs.

Define $\mathbb{A}_n(\pi_{n+1})$ as the disadvantage of π^{n+1} over η_n under d_{π_n}: $\mathbb{A}_n(\pi_{n+1}) = \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}(\cdot|s)}\left[A^{\eta_n}(s,a)\right]\right]$. Note that $\mathbb{A}_n(\pi_{n+1})$ is at least non-positive (if π and η are from the same function class, or π's policy class is rich enough to include η), as if we set π_{n+1} to η_n. In that case, we simply have $\mathbb{A}_n(\pi_{n+1}) = 0$, which means we can hope that the IL procedure (Eq. 7.5) finds a policy π_{n+1} that achieves $\mathbb{A}_n(\pi_{n+1}) < 0$ (i.e., local improvement over η_n). The question we want to answer is: by *how much* is the performance of π_{n+1} improved over π_n by solving the two trust-region optimization procedures detailed in Eq. 7.1 and Eq. 7.5. Following Theorem 4.1 from Kakade and Langford [2002], we define $\varepsilon = \max_s |\mathbb{E}_{a\sim\pi_{n+1}(\cdot|s)}[A^{\eta_n}(s,a)]|$, which measures the maximum possible one-step improvement one can achieve from η_n. The following theorem states the performance improvement:

Theorem 24. *Solve Eq. 7.1 to get η_n and Eq. 7.5 to get π_{n+1}. The improvement of π_{n+1} over π_n is:*

$$J(\pi_{n+1}) - J(\pi_n) \leq \frac{\beta\varepsilon}{(1-\gamma)^2} - \frac{|\mathbb{A}_n(\pi_{n+1})|}{1-\gamma} - \Delta_n(\alpha). \tag{7.7}$$

The proof of Theorem 24 is provided in Appendix 9.6.3. When β is small, we are guaranteed to find a policy π_{n+1} where the total cost decreases by $\Delta_n(\alpha) + |\mathbb{A}_n(\pi_{n+1})|/(1-\gamma)$ compared to π_n. Note that classic CPI's per iteration improvement Kakade and Langford [2002], Schulman et al. [2015a] only contains a term that has the similar meaning and magnitude of the second term in the RHS of Eq. 7.7. Hence DPI can improve the performance of CPI by introducing an extra term $\Delta_n(\alpha)$, and the improvement could be substantial when there exists a locally optimal policy η_n that is much better than the current reactive policy π_n. Such $\Delta(\alpha)$ comes from the explicit introduction of a model-based search into the training loop, which does not exist in classic APIs. From a practical point view, modern MBOCs are usually second-order methods, while APIs are usually first-order (e.g., REINFORCE and CPI). Hence it is reasonable to expect $\Delta(\alpha)$ itself will be larger than API's policy improvement per iteration. Connecting back to ExIt and AlphaGo-Zero under model-based setting, $\Delta(\alpha)$ stands for the improvement of the tree-based policy over the current deep net reactive policy. In ExIt and AlphaGo Zero, the tree-based policy η_n performs fixed depth forward search followed by rolling out π_n (i.e., bottom up by $V^{\pi_n}(s)$), which ensures the expert η_n outperforms π_n.

When $|\Delta_n(\alpha)|$ and $|\mathbb{A}_n(\pi_{n+1})|$ are small, i.e., $|\Delta_n(\alpha)| \leq \xi$ and $|\mathbb{A}_n(\pi_{n+1})| \leq \xi$, then we can guarantee that η_n and π_n are good policies, *under the stronger assumption that the initial distribution ρ_0 happens to be a good distribution (e.g., close to d_{π^*}), and the realizable assumption*: $\min_{\pi\in\Pi}\mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\eta_n}(s,a)]\right] = \mathbb{E}_{s\sim d_{\pi_n}}\left[\min_{a\sim\mathcal{A}}\left[A^{\eta_n}(s,a)\right]\right]$, holds. We show in Appendix 9.6.4 that under the realizable assumption:

$$J(\eta_n) - J(\pi^*) \leq \left(\max_s\left(\frac{d_{\pi^*}(s)}{\rho_0(s)}\right)\right)\left(\frac{\xi}{\beta(1-\gamma)^2} + \frac{\xi}{\beta(1-\gamma)}\right).$$

The term $(\max_s(d_{\pi^*}(s)/\rho_0(s)))$ measures the distribution mismatch between the initial distribution ρ_0 and the optimal policy π^*, and appears in some previous API algorithms–CPI Kakade and Langford [2002] and PSDP Bagnell et al. [2004]. A ρ_0 that is closer to d_{π^*} (e.g., let experts reset the agent's initial position if possible) ensures better global performance guarantee. CPI considers a setting where a good reset distribution ν (different from ρ_0) is available, DPI can leverage such reset distribution by replacing ρ_0 by ν at training.

In summary, we can expect larger per-iteration policy improvement from DPI compared to CPI (and TRPO which has similar per iteration policy improvement as CPI), thanks to the introduction of local model-based search. The final performance bound of the learned policy is in par with CPI and PSDP.

7.5 An Instance of DPI

In this section, we dive into the details of each update step of DPI and suggest one practical instance of DPI, which can be used in continuous control settings. We denote T as the maximum possible horizon.[1] We denote the state space $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and action space $\mathcal{A} \subseteq \mathbb{R}^{d_a}$. We work on parameterized policies: we parameterize policy π as $\pi(\cdot|s;\theta)$ for any $s \in \mathcal{S}$ (e.g., a neural network with parameter θ), and parameterize η by a sequence of time-varying linear-Gaussian policies $\eta = \{\eta_t\}_{1\leq t\leq T}$, where $\eta_t(\cdot|s) = \mathcal{N}(K_t s + k_t, P_t)$ with control gain $K_t \in \mathbb{R}^{d_a \times d_s}$, bias term $k_t \in \mathbb{R}^{d_a}$ and Covariance $P_t \in \mathbb{R}^{d_a \times d_a}$.We will use $\Theta = \{K_t, k_t, P_t\}_{0\leq t\leq H}$ to represent the collection of the parameters of all the linear-Gaussian policies across the entire horizon. One approximation we make here is to replace the policy divergence measure $D_{TV}(\pi_n, \pi)$ (note total variation distance is symmetric) with the KL-divergence $D_{KL}(\pi_n, \pi)$, which allows us to leverage Natural Gradient Bagnell and Schneider [2003], Kakade [2002].[2] To summarize, π_n and η_n are short for π_{θ_n} and $\eta_{\Theta_n} = \{\mathcal{N}(K_t s + k_t, P_t)\}_t$, respectively. Below we first describe how to compute η_{Θ_n} given π_n (Sec. 7.5.1), and then describe how to update π via imitating η_{Θ_n} using Natural Gradient (Sec. 7.5.2).

7.5.1 Updating η_Θ with MBOC using Learned Time Varying Linear Models

Algorithm 9 AGGREVATED-GPS

```
1: Input: Parameters $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}^+$.
2: Initialized $\pi_{\theta_0}$
3: for n = 0 to ... do
4:     Execute $\pi_{\theta_n}$ to generate a set of trajectories
5:     Fit local linear dynamics $\hat{P}$ (Eq. 7.8) using $\{s_t, a_t, s_{t+1}\}$ collected from step 1
6:     Solve the minmax in Eq. 7.9 subject to $\hat{P}$ to obtain $\eta_{\Theta_n}$ and form disadvantage $A^{\eta_{\Theta_n}}$
7:     Compute $\theta_{n+1}$ by NGD (Eq. 7.12)
8: end for
```

We explain here how to find η_n given π_n using MBOC. In our implementation, we use Linear Quadratic Gaussian (LQG) optimal control Kwakernaak and Sivan [1972] as the black-box optimal control solver. We learn a sequence of time varying linear Gaussian transition models to represent $\hat{P}$: $\forall t \in [1, T]$,

$$s_{t+1} \sim \mathcal{N}(A_t s_t + B_t a_t + c_t, \Sigma_t), \tag{7.8}$$

where A_t, B_t, c_t, Σ_t can be learned using classic linear regression techniques on a dataset $\{s_t, a_t, s_{t+1}\}$ collected from executing π_n on the real system. Although the dynamics $P(s, a)$ may be complicated over the entire space, linear dynamics could locally approximate the dynamics well (after all, our theorem only requires $\hat{P}$ to have low predictive error under $d_{\pi_n}\pi_n$).

[1] Note T is the maximum possible horizon which could be long. Hence, we still want to output a single policy, especially when the policy is parameterized by complicated non-linear function approximators like deep nets.

[2] Small D_{KL} leads to small D_{TV}, as by Pinsker's inequality, $D_{KL}(q, p)$ (and $D_{KL}(p, q)) \geq D_{TV}(p, q)^2$.

Next, to find a locally optimal policy under linear-Gaussian transitions (i.e., Eq. 7.4), we add the KL constraint to the objective with Lagrange multiplier μ and form an equivalent min-max problem:

$$\min_{\eta} \max_{\mu \geq 0} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} c(s_t, a_t)\right] + \mu\Big(\sum_{t=1}^{T} \gamma^{t-1} \mathbb{E}_{s \sim d_{\eta}^{t}}[D_{KL}(\eta, \pi_n)] - \alpha\Big), \tag{7.9}$$

where μ is the Lagrange multiplier, which can be solved by alternatively updating η and μ Levine and Abbeel [2014]. For a fixed μ, using the derivation from Levine and Abbeel [2014], ignoring terms that do not depend on η, Eq. 7.9 can be written:

$$\arg\min_{\eta} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1}(c(s_t, a_t)/\mu - \log \pi_n(a_t|s_t))\right] - \sum_{t=1}^{T} \gamma^{t-1} \mathbb{E}_{s \sim d_{\eta}^{t}}[\mathcal{H}(\eta(\cdot|s))], \tag{7.10}$$

where $\mathcal{H}(\pi(\cdot|s)) = \sum_a \pi(a|s) \ln(\pi(a|s))$ is the negative entropy. Hence the above formulation can be understood as using a *new cost function*: $c'(s_t, a_t) = c(s_t, a_t)/\mu - \log(\pi_n(a_t|s_t))$, and an entropy regularization on π. It is well known in the optimal control literature that when c' is quadratic and dynamics are linear, the optimal sequence of linear Gaussian policies for the objective in Eq. 7.10 can be found exactly by a Dynamic Programming (DP) based approach, the *Linear Quadratic Regulator* (LQR) Kwakernaak and Sivan [1972]. Given a dataset $\{(s_t, a_t), c'(s_t, a_t)\}$ collected while executing π_n, we can fit a quadratic approximation of $c'(s, a)$ Levine and Abbeel [2014], Ziebart [2010]. With a quadratic approximation of c' and linear dynamics, we solve Eq. 7.10 for η exactly by LQR Ziebart [2010]. Once we get η, we go back to Eq. 7.9 and update the Lagrange multiplier μ, for example, by projected gradient ascent Zinkevich [2003]. Upon convergence, LQR gives us a sequence of time-dependent linear Gaussian policies together with a sequence of analytic quadratic cost-to-go functions $Q_t(s, a)$, and quadratic disadvantage functions $A_t^{\eta_{\Theta_n}}(s, a)$, for all $t \in [T]$.

7.5.2 Updating π_θ via imitating η_Θ using Natural Gradient

Performing a second order Taylor expansion of the KL constraint $\mathbb{E}_{s \sim d_{\pi_n}}[D_{KL}(\pi_n(\cdot|a), \pi(\cdot|s; \theta)))]$ around θ_n Bagnell and Schneider [2003], Kakade [2002], we get the following constrained optimization problem:

$$\min_{\theta} \mathbb{E}_{s \sim d_{\pi_{\theta_n}}}[\mathbb{E}_{a \sim \pi(\cdot|s;\theta)}[A^{\eta_{\Theta_n}}(s, a)]], s.t., (\theta - \theta_n)^T F_{\theta_n}(\theta - \theta_n) \leq \beta, \tag{7.11}$$

where F_{θ_n} is the Hessian of the KL constraint $\mathbb{E}_{s \sim d_{\pi_{\theta_n}}} D_{KL}(\pi_{\theta_n}, \pi_\theta)$ (i.e., Fisher information matrix), measured at θ_n. Denote the objective (i.e., the first term in Eq. 7.11) as $L_n(\theta)$, and denote ∇_{θ_n} as $\nabla_\theta L_n(\theta)|_{\theta=\theta_n}$, we can optimize θ by performing natural gradient descent (NGD):

$$\theta_{n+1} = \theta_n - \mu F_{\theta_n}^{-1} \nabla_{\theta_n}, \text{where } \mu = \sqrt{\beta / (\nabla_{\theta_n}^T F_{\theta_n}^{-1} \nabla_{\theta_n})}. \tag{7.12}$$

The specific μ above ensures the KL constraint is satisfied. More details about the imitation update on π can be found in Appendix 9.6.6.

Summary If we consider η as an expert, NGD is similar to natural gradient AggreVaTeD—a differential IL approach Sun et al. [2017c]. We summarizes the procedures presented in Sec. 7.3.1&7.5.2 in Alg. 9, which we name as AggreVaTeD-GPS, stands for the fact that we are using MBOC to Guide Policy Search Levine and Abbeel [2014], Montgomery and Levine [2016] via AggreVaTeD-type update. Every iteration, we run π_{θ_n} on P to gather samples. We estimate time dependent local linear dynamics $\hat{P}$ and then leverage an OC solver (e.g, LQR) to solve the Lagrangian in Eq. 7.9 to compute η_{Θ_n} and $A^{\eta_{\Theta_n}}$. We then perform NGD to update to π_{n+1}.

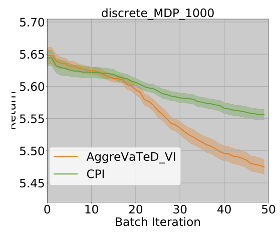
7.5.3 Additional Related Works

The most closely related work with respect to Alg. 9 is Guided Policy Search (GPS) for unknown dynamics Levine and Abbeel [2014] and its variants (e.g.,Levine et al. [2016], Montgomery et al. [2017], Montgomery and Levine [2016]). GPS (including its variants) demonstrates model-based optimal control approaches can be used to speed up training policies parameterized by rich non-linear function approximators (e.g., deep networks) in large-scale applications. While Alg. 9 in high level follows GPS's iterative procedure of alternating reactive policy improvement and MBOC, the main difference between Alg. 9 and GPSs are the update procedure of the reactive policy. Classic GPS, including the mirror descent version, phrases the update procedure of the reactive policy as a *behavior cloning procedure*, i.e., given an expert policy η, we perform $\min_{\pi} D_{KL}(d_{\mu}\mu||d_{\pi}\pi)$ [3]. Note that our approach to updating π is fundamentally on-policy, *i.e.,* we generate samples from π. Moreover, we update π by performing policy iteration against η, i.e., π approximately acts greedily with respect to A^{η}, which resulting a key difference: if we limit the power of MBOC, *i.e.*, set the trust region size in MBOC step to zero in both DPI and GPS, then our approach reduces to CPI and thus improves π to local optimality. GPS and its variants, by contrast, have no ability to improve the reactive policy in that setting.

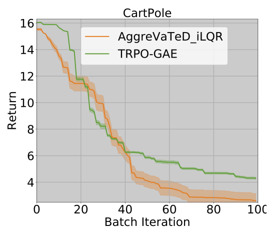
7.6 Experiments

We tested our approach on several MDPs: (1) a set of random discrete MDPs (Garnet problems Scherrer [2014]) (2) Cartpole balancing Sutton and Barto [1998b], (3) Helicopter Aerobatics (Hover and Funnel) Abbeel et al. [2005], (4) Swimmer, Hopper and Half-Cheetah from the MuJoCo physics simulator Todorov et al. [2012]. The goals of these experiments are: **(a)** to experimentally verify that using A^{η} from the intermediate expert η computed by model-based search to perform policy improvement is more sample-efficient than using A^{π}. **(b)** to show that our approach can be applied to robust policy search and can outperform existing approaches Atkeson [2012].

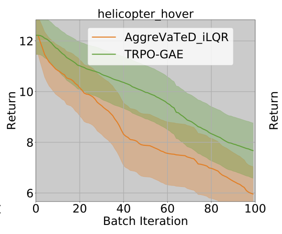
[3]See Line 3 in Alg.2 in Montgomery and Levine [2016], where in principle a behavior cloning on π uses samples from expert η (i.e., off-policy samples). We note, however, in actual implementation some variants of GPS tend to swap the order of π and η inside the KL, often resulting a on-policy sampling strategy (e.g.,Montgomery et al. [2017]). We also note a Mirror Descent interpretation and analysis to explain GPS's convergence Montgomery and Levine [2016] implies the correct way to perform a projection is to minimize the reverse KL, *i.e.*, $\arg\min_{\pi\in\Pi} D_{KL}(d_{\pi}\pi||d_{\eta}\eta)$. This in turn matches the DPI intuition: one should attempt to find a policy π that is similar to η under the state distribution of π itself.



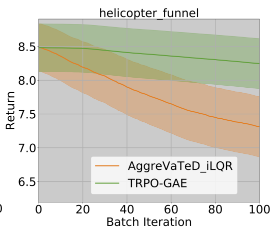
(a) Discrete MDP



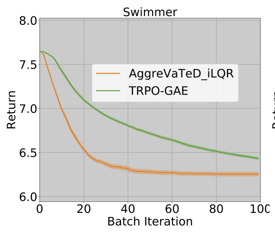
(b) Cart-Pole



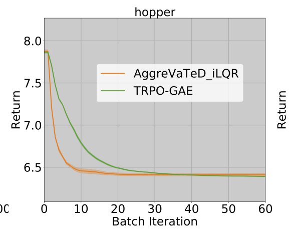
(c) Helicopter Hover



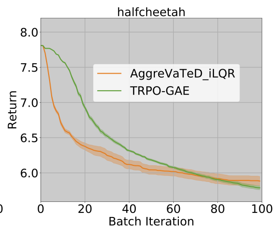
(d) Helicopter Funnel



(e) Swimmer



(f) Hopper



(g) Half-Cheetah

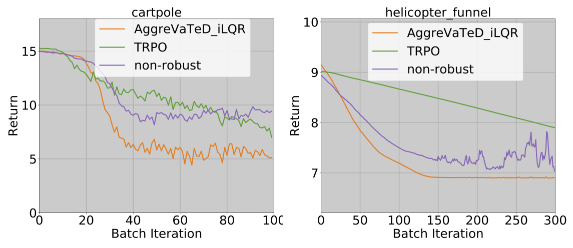
Figure 7.1: Performance (mean and standard error of cumulative cost in $\log_2$-scale on y-axis) versus number of episodes (n on x-axis).

7.6.1 Comparison to CPI on Discrete MDPs

Following Scherrer [2014], we randomly create ten discrete MDPs with 1000 states and 5 actions. Different from the techniques we introduced in Sec. 7.5.2 for continuous settings, here we use the conservative update shown in Eq. 7.6 to update the reactive policy, where each π_n^* is a linear classifier and is trained using regression-based cost-sensitive classification on samples from d_{π_n} Kakade and Langford [2002]. The feature for each state $\phi(s)$ is a binary encoding of the state ($\phi(s) \in \mathbb{R}^{\log_2(|\mathcal{S}|)}$). We maintain the estimated transition $\hat{P}$ in a tabular representation. The policy η is also in a tabular representation (hence expert η and reactive policy π have different feature representation) and is computed using exact VI under $\hat{P}$ and $c'(s,a)$ (hence we name our approach here as AggreVaTeD-VI). The setup and the conservative update implementation is detailed in Appendix 9.6.6. Fig. 7.1a reports the statistical performance of our approach and CPI over the 10 random discrete MDPs. Note that our approach is more sample-efficient than CPI, although we observed it is slower than CPI per iteration as we ran VI using learned model. We tune β and neither CPI nor our approach uses line search on β. The major difference between AggreVaTeD-VI and CPI here is that we used A^{η} instead of A^{π} to update π.

7.6.2 Comparison to Actor-Critic in Continuous Settings

We compare against TRPO-GAE Schulman et al. [2015b] on a set of continuous control tasks. The setup is detailed in Appendix 9.6.6. TRPO-GAE is a actor-critic-like approach where both actor and critic are updated using trust region optimization. We use a two-layer neural network to represent policy π which is updated by natural gradient descent. We use LQR as the underlying MBOC solver and we name our approach as AggreVaTeD-iLQR. Fig. 7.1 (b-g) shows the comparison between our method and TRPO-GAE over a set of continuous control tasks (confidence interval is computed from 20 random trials). As we can see, our



(a) Cart-Pole (b) Helicopter Funnel

Figure 7.2: Performance (mean in log-scale on y-axis) versus episodes (n on x-axis) in robust control.

method is significantly more sample-efficient than TRPO-GAE albeit slower per iteration as we perform MBOC. The major difference between our approach and TRPO-GAE is that we use A^{η} while TRPO-GAE uses A^{π} for the policy update. Note that both A^{η} and A^{π} are computed using the rollouts from π. The difference is that our approach uses rollouts to learn local dynamics and analytically estimates A^{η} using MBOC, while TRPO-GAE learns A^{π} using rollouts. Overall, our approach converges faster than TRPO-GAE (*i.e.*, uses less samples), which again indicates the benefit of using A^{η} in policy improvement.

7.6.3 Application on Robust Policy Optimization

One application for our approach is robust policy optimization Zhou et al. [1996], where we have multiple training environments that are all potentially different from, but similar to, the testing environments. The goal is to train a *single* reactive policy using the training environments and deploy the policy on a test environment *without any further training*. Previous work suggests a policy that optimizes all the training models simultaneously is stable and robust during testing Atkeson [2012], Bagnell and Schneider [2001], as the training environments together act as "regularization" to avoid overfitting and provide generalization.

More formally, let us assume that we have M training environments. At iteration n with π_{θ_n}, we execute π_{θ_n} on the i'th environment, generate samples, fit local models, and call MBOC associated with the i'th environment to compute $\eta_{\Theta_n^i}$, for all $i \in [M]$. With $A^{\eta_{\Theta_n^i}}$, for all $i \in [M]$, we consider all training environments equally and formalize the objective $L_n(\theta)$ as $L_n(\theta) = \sum_{i=1}^{M} \mathbb{E}_{s\sim d_{\pi_{\theta_n}}}[\mathbb{E}_{a\sim\pi(\cdot|s;\theta)}[A^{\eta_{\Theta_n^i}}]]$. We update θ_n to θ_{n+1} by NGD on $L_n(\theta)$. Intuitively, we update π_θ by imitating $\eta_{\Theta_n^i}$ simultaneously for all $i \in [M]$.

We consider two simulation tasks, cartpole balancing and helicopter funnel. For each task, we create ten environments by varying the physical parameters (e.g., mass of helicopter, mass and length of pole). We use 7 of the environments for training and the remaining three for testing. We compare our algorithm against TRPO, which could be regarded as a model-free, natural gradient version of the first-order algorithm proposed in Atkeson [2012]. We also ran our algorithm on a single randomly picked training environment, but still tested on test environments, which is denoted as *non-robust* in Fig. 7.2. Fig. 7.2 summarizes the comparison between our approach and baselines. Similar to the trend we saw in the previous section, our approach is more sample-efficient in the robust policy optimization setup as well. It is interesting to see the "non-robust" approach fails to further converge, which illustrates the overfitting phenomenon: the learned policy overfits to one particular training environment.

7.6.4 Comparison to Guided Policy Search: A Simple Thought Experiment & An Experimental Validation

Guided policy search (GPS) uses imitation learning methods that are not cost-aware to update the parameterized policy. Namely, at any iteration, after one obtains η_Θ, GPS often updates the parameterized policy via minimizing some kind of divergence between η_Θ and π_θ:

$$\min_\theta D_{KL}(d_{\eta_\Theta}\eta_\Theta || d_{\pi_\theta}\pi_\theta), \tag{7.13}$$

which corresponds to Behavior Cloning η_Θ, or sometimes optimizing some "divergence":

$$\min_\theta \mathbb{E}_{s\sim d_{\pi_\theta}}[D_{KL}(\eta_\Theta(\cdot|s)||\pi_\theta(\cdot|s))], \tag{7.14}$$

which is neither $D_{KL}(d_{\eta_\Theta}\eta_\Theta||d_{\pi_\theta}\pi_\theta)$, nor $D_{KL}(d_{\pi_\theta}\pi_\theta||d_{\eta_\Theta}\eta_\Theta)$

To illustrate the main difference between GPS and DPI, let us play the following simple thought game. Let us simply set $\alpha = 0$—namely, we eliminate the power of MBOC. Since the trust region of the MBOC is zero, then the MBOC will simply return the parameterized policy itself: $\eta_n = \pi_n$. Now, let us assume that we have access to infinite many samples and computation such that we can solve the divergence minimization above (Eq. 7.13 or Eq. 7.14), i.e., driving the divergence to zero exactly. Since the divergence is zero, we will have $\pi_{n+1} = \eta_n$, simply because η_n is the optimal solution. Now chaining the results, we have $\pi_{n+1} = \eta_n = \pi_n = \dots \pi_0$, where again the first equality comes from the fact that we can perfectly solving the above divergence minimization formulations, and the second equality comes from the fact that we set the trust region in MBOC step to zero, and the last equality follows from recursion. Hence, GPS with $\alpha = 0$ basically will never improve the parameterized policy ever.

Under the same condition, what will DPI do. Again, since we set $\alpha = 0$, we must have $\eta_n = \pi_n$. However, DPI performs on-policy, cost-aware imitation learning, i.e., DPI performs policy iteration against η_n. As $\eta_n = \pi_n$ here, DPI will perform classic Policy Iteration:

$$\min_\pi \mathbb{E}_{s\sim d_{\pi_n}} \mathbb{E}_{a\sim\pi} A^{\pi_n}(s,a), s.t., \mathbb{E}_{s\sim d_{\pi_n}}[D(\pi||\pi_n)] \leq \beta.$$

For parameterized policy π_θ, with D being the KL divergence, the above update step reveals Natural Policy Gradient (up to second order approximation). Hence, even we set $\alpha = 0$—namely eliminate the power of MBOC, DPI still has the ability to gradually improve the parameterized policy and finally converge to a stationary point. DPI has the ability to get a policy improvement from both MBOC and the parameterized policy update procedure—hence the name Dual Policy Iteration.

The thought experiment above clearly demonstrates the advantage of DPI. To support the above simple thought experiment, we perform the following experimental validation in simulation. We set $\alpha = 10^{-7}$. For GPS we update π_θ via minimizing the forward KL objective (Eq. 7.13). We also increase the number of rolling out trajectories in each batch in order to reduce the variance caused by finite samples. Fig. 7.3 confirms our above simple thought experiment. As we can see, when $\alpha = 10^{-7}$, GPS barely improves the performance of its policies, while DPI can still steadily improve its policy performance.

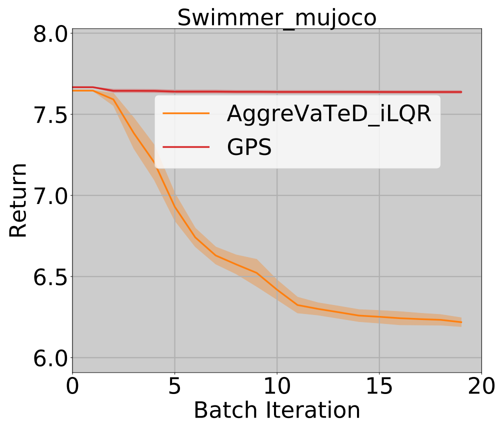


Figure 7.3: Comparison of DPI (AggreVaTeD-iLQR) and GPS, with $\alpha = 10^{-7}$. The mean and the standard error is averaged over 5 random runs.

7.7 Conclusion

In this chapter, we present and analyze Dual Policy Iteration—a framework that alternatively computes a non-reactive policy via more advanced and systematic search, and updates a reactive policy via imitating the non-reactive one. Recent algorithms that have been successful in practice, like AlphaGo-Zero and ExIt, are subsumed by the DPI framework. We then provide a simple instance of DPI for RL with unknown dynamics, where the instance integrates local model fit, local model-based search, and reactive policy improvement via imitating the teacher–the nearly local-optimal policy resulting from model-based search. We theoretically show that integrating model-based search and imitation into policy improvement could result in larger policy improvement at each step. We also experimentally demonstrate the improved sample efficiency compared to strong baselines.

Our work also opens some new problems. In theory, the performance improvement during one call of optimal control with the local accurate model depends on a term that scales quadratically with respect to the horizon $1/(1-\gamma)$. We believe the dependency on horizon can be brought down by leveraging system identification methods focusing on multi-step prediction Sun et al. [2016c], Venkatraman et al. [2015]. On the practical side, our specific implementation has some limitations due to the choice of LQG as the underlying OC algorithm. LQG-based methods usually require the dynamics and cost functions to be somewhat smooth so that they can be locally approximated by polynomials. We also found that LQG planning horizons must be relatively short, as the approximation error from polynomials will likely compound over the horizon. We plan to explore the possibility of learning a non-linear dynamics and using more advanced non-linear optimal control techniques such as Model Predictive Control (MPC) for more sophisticated control tasks.

Part IV

Future Work and Conclusion

Chapter 8

Conclusion and Future Work

Following the structure of this thesis, we conclude the contributions and discuss open problems and interesting future research directions in the order of Imitation Learning, Model-Free Reinforcement Learning, and Model-Based Reinforcement Learning.

8.1 Imitation Learning

In this thesis, we study two settings of imitation learning: (1) Interactive Imitation Learning with access to reward signals, and (2) imitation learning from observations alone. While the interactive setting assumes a much stronger expert than the one assumed in the second setting, we also achieve much stronger performance guarantees in the first setting than the ones in the second setting.

There are many works need to be done in the Imitation Learning from Observations Alone setting. The proposed algorithm in this thesis, FAIL, in general, can only guarantee near-optimality: it suffers from the *inherent Bellman Error*, which potentially could be large, and is hard to measure as well in practice. Hence, for FAIL to succeed, one has to hope that the Inherent Bellman Error is small. Inherent Bellman Error also appears in existing works related to Value Iteration with Function Approximation. FAIL essentially could be understood as Value Iteration with a set of discriminators approximating value functions. Can we design a new algorithm for this setting which does not suffer from Inherent Bellman Error?

What is the role of model learning in the setting of Imitation Learning with Observations Alone? Previous work on Agnostic System Identification suggests that if we have a set of expert demonstrations consisting of states and expert's actions, then one just need to learn a model that can predicts the next state accurately under the even mixture of the expert's state-action distribution, and the state-action distribution of the optimal policy resulting from the model. However, in Imitation Learning from Observations Alone setting, we do not have recorded expert's actions. Hence we cannot even learn a model that can guarantee to predict the next state accurately under the state-action distribution induced from the expert.

8.2 Model-Free Reinforcement Learning

In this thesis, we studied model-free RL from two perspectives: policy evaluation and policy optimization. For policy evaluation, we proposed a reduction that reduces online policy evaluation to no-regret and stable online learning. Such reduction creates a new family of online policy evaluation algorithms that enjoy provable predictive error guarantees. For policy optimization, we studied two most popular simple exploration strategies: exploration in parameter space (e.g., zeroth-order policy optimization), and exploration in action space (e.g., policy gradient methods such as REINFORCE). Theoretically, we found that the sample complexity of exploration in parameter space usually scales polynomially with respect to the dimension of the parameter space, while the sample complexity of exploration in action space is independent of the dimension of the parameter space, but instead polynomially dependent on both the horizon and the dimension of the action space. Our empirical study also supports our theoretical findings.

One interesting combination of imitation learning and model-free policy iteration is worth considering as a future work. Consider the setting where one has access to reward signals during training and has access to a set of expert's demonstrations consisting of only the observations of the states. The expert's demonstrations form an empirical estimation of the state distribution resulting from the expert policy. We can use imitation learning approaches (e.g., FAIL) to first learn a policy whose state distribution is similar to the expert's state distribution. This learned policy can be understood as an exploration policy. With this exploration policy, now we can run approximate value function based reinforcement learning methods such as CPI [Kakade and Langford, 2002] (if we are working on the infinite horizon discounted setting) and PSDP [Bagnell et al., 2004] (if we are working on the finite horizon setting).

8.3 Model-Based Reinforcement Learning

In this thesis, we first studied model-based reinforcement learning in Contextual Decision Processes, where, to the best of our knowledge, we identify the first exponential separation in terms of sample complexity between model-based RL and model-free RL. We proposed *witnessed model rank*, which is a general complexity measure of the underlying MDPs. We show that problems with low model rank is PAC learnable by designing an efficient model-based RL algorithm whose sample complexity scales quadratically with respect to the witnessed model rank.

We then proposed a general framework, Dual Policy Iteration (DPI), that integrates model learning, optimal control, and imitation learning together. We demonstrated the efficacy of the two instances of DPI on a set of simulated continuous control tasks, and a set of randomly generated discrete MDPs.

One important future direction is to study if there exists a computationally efficient and statistically efficient model-based RL algorithms for MDPs with low witness model rank. In other words, are MDPs with low witness model rank efficiently PAC solvable? To study the computational efficiency, we must relax our requirement and first study oracle-efficient computation. Namely, assume we have access to classic oracles such as optimal planner and supervised learning oracle (e.g., regression, classification), can we design a model-based algorithm that only needs a reasonable number of oracle calls?

Another interesting direction is to study multi-task reinforcement learning where all tasks share the same transition dynamics but have different reward functions which themselves are i.i.d sampled from some fixed, but unknown distribution. During training time, we have access to a set of training tasks. The goal is to quickly solve a new task during test time, where we assume the test task is also sampled from the same fixed distribution. The idea here is that if during training time, one can learn a model such that for each training task, the optimal policy of the learned model for this task can optimize the task well, then by generalization, we can hope that for any new task sampled from the same distribution, the learned model can procedure a reasonably good policy with respect to the test task, without any further training.

Part V

Appendix

Chapter 9

Appendix

9.1 Missing Proofs and Details in Chapter 2

9.1.1 A relation between AggreVaTeD with Natural Gradient and AggreVaTe with Weighted Majority

First, we show that Weighted Majority can be leveraged for imitation learning in discrete MDPs. Then we extend Weighted Majority to continuous MDPs, where we show that, with three steps of first-order approximations, WM leads to a natural gradient update procedure.

Weighted Majority in Discrete MDPs

For notation simplicity, for each state $s \in \mathcal{S}$, we represent the policy $\pi(\cdot|s)$ as a discrete probability vector $\pi^s \in \Delta(A)$. We also represent d_t^{π} as a S-dimension probability vector from S-d simplex, consisting of $d_t^{\pi}(s), \forall s \in \mathcal{S}$. For each s, we use $Q_t^*(s)$ to denote the A-dimension vector consisting of the state-action cost-to-go $Q_t^*(s,a)$ for all $a \in \mathcal{A}$. With this notation, the loss function $\ell_n(\pi)$ from Eq. 2.6 can now be written as: $\ell_n(\pi) = \frac{1}{H}\sum_{t=1}^{H}\sum_{s\in\mathcal{S}} d_t^{\pi_n}(s)(\pi^s \cdot Q_t^*(s))$, where $a \cdot b$ represents the inner product between vectors a and b. Weighted Majority updates π as follows: $\forall s \in \mathcal{S}$,

$$\pi_{n+1} = \arg\min_{\pi^s\in\Delta(A),\forall s\in\mathcal{S}} \frac{1}{H}\sum_{t=1}^{H}\sum_{s\in\mathcal{S}} d_t^{\pi_n}(s)\big(\pi^s \cdot Q_t^*(s)\big) + \sum_{s\in\mathcal{S}} \frac{\bar{d}^{\pi_n}(s)}{\eta_{n,s}} KL(\pi_s \| \pi_n^s), \tag{9.1}$$

where $KL(q\|p)$ is the KL-divergence between two probability distributions q and p. This leads to the following closed-form update:

$$\pi_{n+1}^s[i] = \frac{\pi_n^s[i]\exp\big(-\eta_{n,s}\tilde{Q}_s^e[i]\big)}{\sum_{j=1}^{|\mathcal{A}|}\pi_n^s[j]\exp\big(-\eta_{n,s}\tilde{Q}_s^e[j]\big)}, i \in [|\mathcal{A}|], \tag{9.2}$$

where $\tilde{Q}_s^e = \sum_{t=1}^{H} d_t^{\pi_n}(s) Q_t^*(s)/(H\bar{d}^{\pi_n}(s))$. We refer readers to Shalev-Shwartz et al. [2012] or Appendix 9.1.3 for the derivations of the above closed-form updates.

From Discrete to Continuous

We now consider how to update the parametrized policy π_θ for continuous MDPs. Replacing summations by integrals, Eq. 9.1 can be written as:

$$\theta = \arg\min_{\theta} \frac{1}{H}\sum_{t=1}^{H} \mathbb{E}_{s\sim d_t^{\pi_{\theta_n}}} \mathbb{E}_{a\sim\pi(\cdot|s;\theta)}[Q_t^*(s,a)] + \mathbb{E}_{s\sim\bar{d}^{\pi_{\theta_n}}} KL(\pi_\theta||\pi_{\theta_n})/\eta_n. \tag{9.3}$$

In order to solve for θ from Eq. 9.3, we apply several first-order approximations. We first approximate $\ell_n(\theta)$ (the first part of the RHS of the above equation) by its first-order Taylor expansion: $\ell_n(\theta) \approx \ell_n(\theta_n) + \nabla_{\theta_n}\ell_n(\theta_n)\cdot(\theta - \theta_n)$. When θ and θ_n are close, this is a valid local approximation.

Second, we replace $KL(\pi_\theta||\pi_{\theta_n})$ by $KL(\pi_{\theta_n}||\pi_\theta)$, which is a local approximation since $KL(q||p)$ and $KL(p||q)$ are equal up to the second order Kakade and Langford [2002], Schulman et al. [2015a].

Third, we approximate $KL(\pi_{\theta_n}||\pi_\theta)$ by a second-order Taylor expansion around θ_n, such that we can approximate the penalization using the Fisher information matrix:

$$\mathop{\mathbb{E}}_{s\sim \bar{d}^{\pi_{\theta_n}}} KL(\pi_{\theta_n}||\pi_\theta) \approx (1/2)(\theta-\theta_n)^T I(\theta_n)(\theta-\theta_n),$$

where the Fisher information matrix $I(\theta_n) = \mathbb{E}_{s,a\sim \bar{d}^{\pi_{\theta_n}}\pi_{\theta_n}(a|s)}\big(\nabla_{\theta_n}\log(\pi_{\theta_n}(a|s))\big)\big(\nabla_{\theta_n}\log(\pi_{\theta_n}(a|s)\big)^T$.

Inserting these three approximations into Eq. 9.3, and solving for θ, we reach the following update rule $\theta_{n+1} = \theta_n - \eta_n I(\theta_n)^{-1}\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$, which is similar to the natural gradient update rule developed in Kakade [2002] for the RL setting. Bagnell and Schneider [2003] provided an equivalent representation for Fisher information matrix:

$$I(\theta_n) = \frac{1}{H^2}\mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_{\theta_n}}} \nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))^T, \tag{9.4}$$

where $\nabla_\theta \log(\rho_{\pi_\tau}(\tau))$ is the gradient of the log likelihood of the trajectory τ which can be computed as $\sum_{t=1}^{H}\nabla_\theta\log(\pi_\theta(a_t|s_t))$. In the remainder of the paper, we use this Fisher information matrix representation, which yields much faster computation of the descent direction δ_θ, as we will explain in the next section.

9.1.2 Derivation of Eq. 2.9

Starting from Eq. 2.6 with parametrized policy π_θ, we have:

$$\begin{aligned}
\ell_n(\theta) &= \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}}}\Big[\mathop{\mathbb{E}}_{a_t\sim\pi(\cdot|s_t;\theta)}[Q_t^*(s_t,a_t)]\Big]\\
&= \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}}}\Big[\int_a \pi(a|s_t;\theta)Q_t^*(s_t,a)da\Big]\\
&= \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}}}\Big[\int_a \pi(a|s_t;\theta_n)\frac{\pi(a|s_t;\theta)}{\pi(a|s_t;\theta_n)}Q_t^*(s_t,a)da\Big]\\
&= \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}}}\Big[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s_t;\theta_n)}\frac{\pi(a|s_t;\theta)}{\pi(a|s_t;\theta_n)}Q_t^*(s_t,a)\Big]\\
&= \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_{\theta_n}},a_t\sim\pi(a|s_t;\theta_n)}\Big[\frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)}Q_t^*(s_t,a_t)\Big].
\end{aligned}\tag{9.5}$$

9.1.3 Derivation of Weighted Majority Update in Discrete MDP

We show the detailed derivation of Eq. 9.2 for AggreVaTeD with WM in discrete MDP. Recall that with KL-divergence as the penalization, one update the policy in each episode as:

$$\{\pi_{n+1}^s\}_{s\in\mathcal{S}} = \arg\min_{\{\pi^s\in\Delta(A),\forall s\}}\frac{1}{H}\sum_{t=1}^{H}\sum_{s\in\mathcal{S}} d_t^{\pi_n}(s)\big(\pi^s\cdot Q_t^*(s)\big) + \sum_{s\sim\mathcal{S}}\frac{\bar{d}^{\pi_n}(s)}{\eta_{n,s}}KL(\pi_s\|\pi_n^s)$$

Note that in the above equation, for a particular state s, optimizing π^s is in fact independent of $\pi^{s'},\forall s'\neq s$. Hence the optimal sequence $\{\pi^s\}_{s\in\mathcal{S}}$ can be achieved by optimizing π^s

independently for each $s \in \mathcal{S}$. For π^s, we have the following update rule:

$$\begin{aligned}
\pi^s_{n+1} =& \arg\min_{\pi^s \in \Delta(A)} \frac{1}{H}\sum_{t=1}^{H} d_t^{\pi_n}(s)(\pi^s \cdot Q_t^*(s)) + \frac{\bar{d}^{\pi_n}(s)}{\eta_{n,s}} KL(\pi_s \| \pi_n^s) \\
&= \arg\min_{\pi^s \in \Delta(A)} \pi^s \cdot (\sum_{t=1}^{H} d_t^{\pi_n}(s) Q_t^*(s)/H) + \frac{\bar{d}^{\pi_n}(s)}{\eta_{n,s}} KL(\pi_s \| \pi_n^s) \\
&= \arg\min_{\pi^s \in \Delta(A)} \pi^s \cdot (\sum_{t=1}^{H} d_t^{\pi_n}(s) Q_t^*(s)/(H\bar{d}^{\pi_n}(s))) + \frac{1}{\eta_{n,s}} KL(\pi_s \| \pi_n^s) \\
&= \arg\min_{\pi^s \in \Delta(A)} \pi^s \cdot \tilde{Q}^e(s) + \frac{1}{\eta_{n,s}} \sum_{j=1}^{A} \pi^s[j](\log(\pi^s[j]) - \log(\pi_n^s[j])) \qquad (9.6)
\end{aligned}$$

Take the derivative with respect to $\pi^s[j]$, and set it to zero, we get:

$$\tilde{Q}^e(s)[j] + \frac{1}{\eta_{n,s}}(\log(\pi^s[j]/\pi_n^s[j]) + 1) = 0, \qquad (9.7)$$

this gives us:

$$\pi^s[j] = \pi_n^s[j]\exp(-\eta_{n,s}\tilde{Q}^e(s)[j] - 1). \qquad (9.8)$$

Since $\pi^s \in \Delta(A)$, after normalization, we get:

$$\pi^s[j] = \frac{\pi_n^s[j]\exp(-\eta_{n,s}\tilde{Q}^e(s)[j])}{\sum_{i=1}^{A}\pi_n^s[i]\exp(-\eta_{n,s}\tilde{Q}^e(s)[i])} \qquad (9.9)$$

9.1.4 Lemmas

Before proving the theorems, we first present the *Performance Difference Lemma* Kakade and Langford [2002], Ross and Bagnell [2014] which will be used later:

Lemma 8. *For any two policies π_1 and π_2, we have:*

$$\mu(\pi_1) - \mu(\pi_2) = H\sum_{t=1}^{H} \mathbb{E}_{s_t \sim d_t^{\pi_1}}\left[\mathbb{E}_{a_t \sim \pi_1(\cdot|s_t)}[Q_t^{\pi_2}(s_t, a_t) - V_t^{\pi_2}(s_t)]\right]. \qquad (9.10)$$

We refer readers to Ross and Bagnell [2014] for the detailed proof of the above lemma.

The second known result we will use is the analysis of Weighted Majority Algorithm. Let us define the linear loss function as $\ell_n(w) = w \cdot y_n$, for any $y_n \in \mathbb{R}^d$, and $w \in \Delta(d)$ from a probability simplex. Running Weighted Majority Algorithm on the sequence of losses $\{w \cdot y_n\}$ to compute a sequence of decisions $\{w_n\}$, we have:

Lemma 9. *The sequence of decisions $\{w_n\}$ computed by running Weighted Majority with step size μ on the loss functions $\{w \cdot y_n\}$ has the following regret bound:*

$$\sum_{n=1}^{N} w_n \cdot y_n - \min_{w^* \in \Delta(d)} \sum_{n=1}^{N} w^* \cdot y_n \leq \frac{\ln(d)}{\mu} + \frac{\mu}{2}\sum_{n=1}^{N}\sum_{i=1}^{d} w_n[i] y_n[i]^2. \qquad (9.11)$$

We refer readers to Shalev-Shwartz et al. [2012] for detailed proof.

9.1.5 Proof of Theorem 5

Proof. We construct a reduction from stochastic Multi-Arm Bandits (MAB) to the MDP $\tilde{\mathcal{M}}$. A stochastic MAB is defined by S arms denoted as $I^1, ..., I^S$. Each arm I^t's cost c_i at any time step t is sampled from a fixed but unknown distribution. A bandit algorithm picks an arm I_t at iteration t and then receives an unbiased sample of the picked arm's cost c_{I_t}. For any bandit algorithm that picks arms $I_1, I_2, ..., I_N$ in N rounds, the expected regret is defined as:

$$\mathbb{E}[R_N] = \mathbb{E}[\sum_{n=1}^{N} c_{I_n}] - \min_{i\in[S]} \sum_{n=1}^{N} \bar{c}_i, \tag{9.12}$$

where the expectation is taken with respect to the randomness of the cost sampling process and possibly the randomness of the bandit algorithm. It has been shown that there exists a set of distributions from which the arms' costs sampled from, the expected regret $\mathbb{E}[R_N]$ is at least $\Omega(\sqrt{SN})$ Bubeck et al. [2012].

Consider a MAB with 2^K arms. To construct a MDP from a MAP, we construct a $K+1$-depth binary-tree structure MDP with $2^{K+1}-1$ nodes. We set each node in the binary tree as a state in the MDP. The number of actions of the MDP is two, which corresponds to go left or right at a node in the binary tree. We associate each leaf nodes with arms in the original MAB: the cost of the i'th leaf node is sampled from the cost distribution for the i'th arm, while the non-leaf nodes have cost always equal to zero. The initial distribution ρ_0 concentrates on the root of the binary tree. Note that there are total 2^K trajectories from the root to leafs, and we denote them as $\tau_1, ...\tau_{2^K}$. We consider finite horizon ($H = K+1$) episodic RL algorithms that outputs $\pi_1, \pi_2, ..., \pi_N$ at N episodes, where π_n is any deterministic policy that maps a node to actions *left* or *right*. Any RL algorithm must have the following regret lower bound:

$$\mathbb{E}[\sum_{n=1}^{N} \mu(\pi_n)] - \min_{\pi^*} \sum_{n=1}^{N} \mu(\pi^*) \geq \Omega(\sqrt{SN}), \tag{9.13}$$

where the expectation is taken with respect to the possible randomness of the RL algorithms. Note that any deterministic policy π identifies a trajectory in the binary tree when rolling out from the root. The optimal policy π^* simply corresponds to the trajectory that leads to the leaf with the mininum expected cost. Note that each trajectory is associated with an arm from the original MAB, and the expected total cost of a trajectory corresponds to the expected cost of the associated arm. Hence if there exists an RL algorithm that achieves regret $O(\sqrt{SN})$, then we can solve the original MAB problem by simply running the RL algorithm on the constructed MDP. Since the lower bound for MAB is $\Omega(\sqrt{SN})$, this concludes that Eq. 9.13 holds. □

9.1.6 Proof of Theorem 6

Proof. For notation simplicity we denote a_l as the go-left action while a_r is the go-right action. Without loss of generality, we assume that the leftmost trajectory has the lowest total cost (e.g., s_3 in Fig. 2.1 has the lowest average cost). We consider the deterministic policy class Π that contains all policy $\pi : \mathcal{S} \to \{a_l, a_r\}$. Since there are S states and 2 actions, the total number of policies in the policy class is 2^S. To prove the upper bound $R_N \leq O(\log(S))$, we claim that for any $e \leq K$, at the end of episode e, AggreVaTe with FTL identifies the e'th state on the best trajectory, i,e, the leftmost trajectory $s_0, s_1, s_3, ..., s_{(2^{K-1}-1)}$. We can prove the claim by induction.

At episode $e = 1$, based on the initial policy, AggreVaTe picks a trajectory τ_1 to explore. AggreVaTe with FTL collects the states s at τ_1 and their associated cost-to-go vectors $[Q^*(s, a_l), Q^*(s, a_r)]$. Let us denote D_1 as the dataset that contains the state,cost-to-go pairs: $D_1 = \{(s, [Q^*(s, a_l), Q^*(s, a_l)])\}$, for $s \in \tau_1$. Since s_0 is visited, the state-cost pair $(s_0, [Q^*(s_0, a_l), Q^*(s_0, a_r)])$ must be in D_1. To update policy from π_1 to π_2, AggreVaTe with FTL runs cost-sensitive classification D_1 as:

$$\pi_2 = \arg\min_{\pi} \sum_{k=1}^{|D_1|} Q^*(s_k, \pi(s_k)), \tag{9.14}$$

where s_k stands for the k'th data point collected at dataset D_1. Due to the construction of policy class Π, we see that π_2 must picks action a_l at state s_0 since $Q(s_0, a_l) < Q(s_0, a_r)$. Hence at the end of the episode $e = 1$, π_2 identifies s_1 (i.e., running π_2 from root s_0 leads to s_1), which is on the optimal trajectory.

Now assume that at the end of episode $n - 1$, the newly updated policy π_n identifies the state $s_{(2^{n-1}-1)}$: namely at the beginning of episode n, if we roll-in π_n, the algorithm will keep traverse along the leftmost trajectory till at least state $s_{(2^{n-1}-1)}$. At episode n, let D_n as the dataset contains all data points from D_{n-1} and the new collected state, cost-to-go pairs from τ_n: $D_n = D_{n-1} \cup \{(s, [Q^*(s, a_l), Q^*(s, a_r)])\}$, for all $s \in \tau_n$. Now if we compute policy π_{n+1} using cost-sensitive classification (Eq. 9.14) over D_n, we must learn a policy π_{n+1} that identifies action a_l at state $s_{(2^j-1)}$, since $Q^e(s_{(2^j-1)}, a_l) < Q^*(s_{(2^j-1)}, a_r)$, and $s_{(2^j-1)}$ is included in D_n, for $j = 1, ..., n - 1$. Hence at the end of episode n, we identify a policy π_{n+1} such that if we roll in policy π_{n+1} from s_0, we will traverse along the left most trajectory till we reach $s_{(2^n-1)}$.

Hence by the induction hypothesis, at the end of episode $K - 1$, π_K will reach state $s_{(2^{K-1}-1)}$, the end of the best trajectory.

Since AggreVaTe with FTL with policy class Π identifies the best trajectory with at most $K - 1$ episodes, the cumulative regret is then at most $O(K)$, which is $O(\log(S))$ (assuming the average cost at each leaf is a bounded constant), as S is the number of nodes in the binary-tree structure MDP $\tilde{\mathcal{M}}$. $\square$

9.1.7 Proof of Theorem 7

Since in Theorem 7 we assume that we only have access to the noisy, but unbiased estimate of Q^*, the problem becomes more difficult since unlike in the proof of Theorem 6, we cannot simply eliminate states completely since the cost-to-go of the states queried from expert is noisy and completely eliminate nodes will potentially result elimination of low cost trajectories. Hence here we consider a different policy representation. We define 2^K deterministic base policies $\pi^1, ..., \pi^{2^K}$, such that rolling out policy π^i at state s_0 will traverse along the trajectory ending at the i'th leaf. We define the policy class Π as the convex hull of the base policies $\Pi = \{\pi : \sum_{i=1}^{2^K} w_i \pi^i, \sum_{i}^{2^K} w_i = 1, w_i \geq 0, \forall i\}$. Namely each $\pi \in \Pi$ is a stochastic policy: when rolling out, with probability w_i, π execute the i'th base policy π^i from s_0. Below we prove that AggreVaTeD with Weighted Majority achieves the regret bound $O(\sqrt{\ln(S)N})$.

Proof. We consider finite horizon, episodic imitation learning setting where at each episode n, the algorithm can roll in the current policy π_n once and only once and traverses through trajectory τ_n . Let us define $\tilde{\ell}_n(w) = \frac{1}{K+1}\sum_{s\in\tau_n}\sum_{j=1}^{2^K} w_j\tilde{Q}^e(s,\pi^j(s))$, where τ_n is the trajectory traversed by rolling out policy π_n starting at s_0, and $\tilde{Q}^e$ is a noisy but unbiased estimate of Q^*. We simply consider the setting where $\tilde{Q}^e$ is bounded $|\tilde{Q}^e| \leq l_{\max}$ (note that we can easily extend our analysis to a more general case where $\tilde{Q}^e$ is from a sub-Gaussian distribution). Note that $\tilde{\ell}_n(w)$ is simply a linear loss with respect to w:

$$\tilde{\ell}_n(w) = w\cdot q_n, \tag{9.15}$$

where $q_n[j] = \sum_{s\in\tau_n}\tilde{Q}^e(s,\pi^j(s))/(K+1)$. AggreVaTeD with WM updates w using Exponential gradient descent. Using the result from lemma 9, we get:

$$\sum_{n=1}^{N}(\tilde{\ell}_n(w_n) - \tilde{\ell}_n(w^*)) = \sum_{n=1}^{N}(w_n\cdot q_n - w^*\cdot q_n) \leq \frac{\ln(2^K)}{\mu} + \frac{\mu}{2}\sum_{n=1}^{N}\sum_{j=1}^{2^K} w_n[j]q_n[j]^2 \leq \frac{\ln(2^K)}{\mu} + \frac{\mu}{2}\sum_{n=1}^{N} l_{\max}^2$$
$$= \frac{\ln(2^K)}{\mu} + \frac{\mu N l_{\max}^2}{2} \leq l_{\max}\sqrt{\ln(S)N}. \tag{9.16}$$

Note that $S = 2^{K+1}-1$. The above inequality holds for any $w^*\in\Delta(2^K)$, including the w^e that corresponds to the expert (i.e., $w^e[1]=1, w^e[i]=0, i\neq 1$ as we assumed without loss of generality the left most trajectory is the optimal trajectory).

Now let us define $\ell_n(w)$ as follows:

$$\ell_n(w) = \frac{1}{K+1}\sum_{t=1}^{K+1}\sum_{s\sim\mathcal{S}} d_t^{\pi_n}(s)\sum_{j=1}^{2^K} w_j Q^*(s,\pi^j(s)). \tag{9.17}$$

Note $\ell_n(w)$ can be understood as first rolling out π_n *infinitely many times* and then querying for the exact cost-to-go Q^* on all the visited states. Clearly $\tilde{\ell}_n(w)$ is an unbiased estimate of $\ell_n(w)$: $\mathbb{E}[\tilde{\ell}_n(w)] - \ell_n(w) = 0$, where the expectation is over the randomness of the roll-in and sampling procedure of $\tilde{Q}^e$ at iteration n, conditioned on all events among the previous $n-1$ iterations. Also note that $|\tilde{\ell}_n(w) - \ell_n(w)| \leq 2l_{\max}$, since $\ell_n(w)\leq l_{\max}$. Hence $\{\tilde{\ell}_n(w_n) - \ell_n(w_n)\}$ is a bounded martingale difference sequence. Hence by Azuma-Heoffding inequality, we get with probability at least $1-\delta/2$:

$$\sum_{n=1}^{N}\ell_n(w_n) - \tilde{\ell}_n(w_n) \leq 2l_{\max}\sqrt{\log(2/\delta)N}, \tag{9.18}$$

and with probability at least $1-\delta/2$:

$$\sum_{n=1}^{N}\tilde{\ell}_n(w^e) - \ell_n(w^e) \leq 2l_{\max}\sqrt{\log(2/\delta)N}. \tag{9.19}$$

Combine the above inequality using union bound, we get with probability at least $1-\delta$:

$$\sum_{n=1}^{N}(\ell_n(w_n) - \ell_n(w^e)) \leq \sum_{n=1}^{N}(\tilde{\ell}_n(w_n) - \tilde{\ell}_n(w^e)) + 4l_{\max}\sqrt{\log(2/\delta)N}. \tag{9.20}$$

Now let us apply the Performance Difference Lemma (Lemma 8), we get with probability at least $1-\delta$:

$$\sum_{n=1}^{N}\mu(\pi_n)-\sum_{n=1}^{N}\mu(\pi^*)=\sum_{n=1}^{N}(K+1)\big(\ell_n(w_n)-\ell_n(w^e)\big)\leq (K+1)(l_{\max}\sqrt{\ln(S)N}+4l_{\max}\sqrt{\log(2/\delta)N}), \tag{9.21}$$

rearrange terms we get:

$$\sum_{n=1}^{N}\mu(\pi_n)-\sum_{n=1}^{N}\mu(\pi^*)\leq \log(S)l_{\max}(\sqrt{\ln(S)N}+\sqrt{\log(2/\delta)N})\leq O(\ln(S)\sqrt{\ln(S)N}), \tag{9.22}$$

with probability at least $1-\delta$. $\square$

9.1.8 Proof of Theorem 8

The proof of theorem 8 is similar to the one for theorem 7. Hence we simply consider the infinitely many roll-ins and exact query of Q^* case. The finite number roll-in and noisy query of Q^* case can be handled by using the martingale difference sequence argument as shown in the proof of theorem 7.

Proof. Recall that in general setting, the policy π consists of probability vectors $\pi^{s,t}\in\Delta(A)$, for all $s\in\mathcal{S}$ and $t\in[H]$: $\pi=\{\pi^{s,t}\}_{\forall s\in\mathcal{S},t\in[H]}$. Also recall that the loss functions WM is optimizing are $\{\ell_n(\pi)\}$ where:

$$\ell_n(\pi)=\frac{1}{H}\sum_{t=1}^{H}\sum_{s\in\mathcal{S}}d_t^{\pi_n}(s)(\pi^{s,t}\cdot Q_t^*(s))=\sum_{t=1}^{H}\sum_{s\in\mathcal{S}}\pi^{s,t}\cdot q_n^{s,t} \tag{9.23}$$

where as we defined before $Q_t^*(s)$ stands for the cost-to-go vector $Q_t^*(s)[j]=Q_t^*(s,a_j)$, for the j'th action in $\mathcal{A}$, and $q_n^{s,t}=\frac{d_t^{\pi_n}(s)}{H}Q_t^*(s)$.

Now if we run Weighted Majority on ℓ_n to optimize $\pi^{s,t}$ for each pair of state and time step independently, we can get the following regret upper bound by using Lemma 9:

$$\sum_{n=1}^{N}\ell_n(\pi)-\min_{\pi}\sum_{n=1}^{N}\ell_n(\pi_n)\leq\sum_{t=1}^{H}\sum_{s\in\mathcal{S}}\big(\frac{\ln(A)}{\mu}+\frac{\mu}{2}\sum_{n=1}^{N}\sum_{j=1}^{A}\pi^{s,t}[j]q_n^{s,t}[j]^2\big). \tag{9.24}$$

Note that we can upper bound $(q_n^{s,t}[j])^2$ as:

$$(q_n^{s,t}[j])^2\leq\frac{d_t^{\pi_n}(s)^2}{H^2}(Q^*_{\max})^2\leq\frac{d_t^{\pi_n}(s)}{H^2}(Q^2_{\max})^2 \tag{9.25}$$

Substitute it back, we get:

$$\begin{aligned}\sum_{n=1}^{N}(\ell_n(\pi_n)-\ell_n(\pi^*))&\leq\sum_{t=1}^{H}\sum_{s\in\mathcal{S}}\big(\frac{\ln(A)}{\mu}+\frac{\mu}{2}\sum_{n=1}^{N}\sum_{j=1}^{A}\pi^{s,t}[j]d_t^{\pi_n}(s)\frac{(Q^*_{\max})^2}{H^2}\big)\\&=\sum_{t=1}^{H}\big(\frac{S\ln(A)}{\mu}+\frac{\mu(Q^*_{\max})^2}{2H^2}\sum_{n=1}^{N}\sum_{s\in\mathcal{S}}d_t^{\pi_n}(s)\sum_{j=1}^{A}\pi^{s,t}[j]\big)=\sum_{t=1}^{H}\big(\frac{S\ln(A)}{\mu}+\frac{\mu(Q^*_{\max})^2}{2H^2}N\big)\end{aligned}$$

$$\leq \frac{Q^*_{\max}}{H}\sqrt{2S\ln(A)N}, \tag{9.26}$$

if we set $\mu = \sqrt{(Q^*_{\max})^2 NS\ln(A)/(2H^2)}$.

Now let us apply the performance difference lemma (Lemma 8), we get:

$$R_N = \sum_{n=1}^{N}\mu(\pi_n) - \sum_{n=1}^{N}\mu(\pi^*) = H\sum_{n=1}^{N}(\ell_n(w_n) - \ell_n(w^e)) \leq HQ^e_{\max}\sqrt{S\ln(A)N}. \tag{9.27}$$

□

9.1.9 Proof of Theorem 9

Let us use $\tilde{Q}^e(s)$ to represent the noisy but unbiased estimate of $Q^*(s)$.

Proof. For notation simplicity, we denote $\mathcal{S} = \{s_1, s_2, ..., s_S\}$. We consider a finite MDP with time horizon $H = 1$. The initial distribution $\rho_0 = \{1/S, ..., 1/S\}$ puts $1/S$ weight on each state. We consider the algorithm setting where at every episode n, a state $s^n \in \mathcal{S}$ is sampled from ρ_0 and the algorithm uses its current policy $\pi_n^{s_n} \in \Delta(A)$ to pick an action $a \in \mathcal{A}$ for s^n and then receives a noisy but unbiased estimate $\tilde{Q}^e(s^n)$ of $Q^*(s^n) \in \mathbb{R}^{|\mathcal{A}|}$. The algorithm then updates its policy from $\pi_n^{s^n}$ to $\pi_{n+1}^{s^n}$ for s^n while keep the other polices for other s unchanged (since the algorithm did not receive any feedback regarding $Q^*(s)$ for $s \neq s^n$ and the sample distribution ρ_0 is fixed and uniform). For expected regret $\mathbb{E}[R_N]$ we have the following fact:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s^n\sim\rho_0,\forall n}\Big[\mathop{\mathbb{E}}_{\tilde{Q}^e(s_n)\sim P_{s_n},\forall n}\big[\sum_{n=1}^{N}(\pi_n^{s^n}\cdot\tilde{Q}^e(s^n) - \pi^*_{s^n}\cdot\tilde{Q}^e(s^n))\big]\Big]\\
&= \mathop{\mathbb{E}}_{s^n\sim\rho_0,\forall n}\Big[\sum_{n=1}^{N}\mathop{\mathbb{E}}_{\tilde{Q}^e_i(s_i)\sim P_{s_i},i\leq n-1}\big[(\pi_n^{s^n}\cdot Q^*(s^n) - \pi^e_{s^n}\cdot Q^*(s^n))\big]\Big]\\
&= \sum_{n=1}^{N}\mathop{\mathbb{E}}_{s^i\sim\rho_0,i\leq n-1}\Big[\mathop{\mathbb{E}}_{\tilde{Q}^e_i(s_i)\sim P_{s_i},i\leq n-1}\big[\mathop{\mathbb{E}}_{s\sim\rho_0}(\pi_n^s\cdot Q^*(s) - \pi^*_s\cdot Q^*(s))\big]\Big]\\
&= \mathbb{E}\big[\sum_{n=1}^{N}\mathop{\mathbb{E}}_{s\sim\rho_0}\pi_n^s\cdot Q^*(s) - \mathop{\mathbb{E}}_{s\sim\rho_0}\pi^*_s\cdot Q^*(s)\big]\\
&= \mathbb{E}\sum_{n=1}^{N}[\mu(\pi_n) - \mu(\pi^*)],
\end{aligned} \tag{9.28}$$

where the expectation in the final equation is taken with respect to random variables $\pi_i, i \in [N]$ since each π_i is depend on $\tilde{Q}^e_j$, for $j < i$ and s^j, for $j < i$.

We first consider $\mathbb{E}_{\tilde{Q}^e(s^n)\sim P_{s^n},\forall n}\big[\sum_{n=1}^{N}(\pi_n^{s^n}\cdot\tilde{Q}^e(s^n) - \pi^*_{s^n}\cdot\tilde{Q}^e(s^n))\big]$ conditioned on a given sequence of $s^1, ..., s^N$. Let us define that among N episodes, the set of the index of the episodes that state s_i is sampled as $\mathcal{N}_i$ and its cardinality as N_i, and we then have $\sum_{i=1}^{S} N_i = N$ and $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$,for $i \neq j$.

$$\mathop{\mathbb{E}}_{\tilde{Q}^e(s^n)\sim P_{s^n},\forall n}\big[\sum_{n=1}^{N}(\pi_n^{s^n}\cdot\tilde{Q}^e(s^n) - \pi^*_{s^n}\cdot\tilde{Q}^e(s^n))\big]$$

$$= \sum_{i=1}^{S} \sum_{j\in\mathcal{N}_i} \mathop{\mathbb{E}}_{\tilde{Q}^e_j(s_i)\sim P_{s_i}} (\pi_j^{s_i} \cdot \tilde{Q}^e_j(s_i) - \pi^e_{s_i}\tilde{Q}^e_j(s_i)) \tag{9.29}$$

Note that for each state s_i, at the rounds from $\mathcal{N}_i$, we can think of the algorithm running any possible online linear regression algorithm to compute the sequence of policies $\pi_j^{s_i}, \forall j \in \mathcal{N}_i$ for state s_i. Note that from classic online linear regression analysis, we can show that for state s_i there exists a distribution P_{s_i} such that for any online algorithm:

$$\mathop{\mathbb{E}}_{\tilde{Q}^e_j(s_i)\sim P_{s_i}, \forall j\in\mathcal{N}_i} \big[\sum_{j\in\mathcal{N}_i} (\pi_j^{s_i} \cdot \tilde{Q}^e_j(s_i) - \pi^e_{s_i} \cdot \tilde{Q}^e_j(s_i))\big] \geq c\sqrt{\ln(A)N_i}, \tag{9.30}$$

for some non-zero positive constant c. Substitute the above inequality into Eq. 9.29, we have:

$$\mathop{\mathbb{E}}_{\tilde{Q}^e(s_n)\sim P_{s_n}, \forall n} \big[\sum_{n=1}^{N}(\pi_n^{s^n} \cdot \tilde{Q}^e(s^n) - \pi^*_{s^n} \cdot \tilde{Q}^e(s^n))\big] \geq \sum_{i=1}^{S} c\sqrt{\ln(A)N_i} = c\sqrt{\ln(A)}\sum_{i=1}^{S}\sqrt{N_i}. \tag{9.31}$$

Now let us put the expectation $\mathbb{E}_{s^i\sim\rho_0,\forall i}$ back, we have:

$$\mathop{\mathbb{E}}_{s^n\sim\rho_0,\forall n}\Big[\mathop{\mathbb{E}}_{\tilde{Q}^e(s_n)\sim P_{s_n}}\big[\sum_{n=1}^{N}(\pi_n^{s^n} \cdot \tilde{Q}^e(s^n) - \pi^*_{s^n} \cdot \tilde{Q}^e(s^n))|s^1, ..., s^n\big]\Big] \geq c\sqrt{\ln(A)}\sum_{i=1}^{N}\mathbb{E}[\sqrt{N_i}]. \tag{9.32}$$

Note that each N_i is sampled from a Binomial distribution $\mathcal{B}(N, 1/S)$. To lower bound $\mathbb{E}_{n\sim\mathcal{B}(N,1/S)}\sqrt{n}$, we use Hoeffding's Inequality here. Note that $N_i = \sum_{n=1}^{N} a_n$, where $a_n = 1$ if s_i is picked at iteration n and zero otherwise. Hence a_i is from a Bernoulli distribution with parameter $1/S$. Using Hoeffding bound, for N_i/N, we get:

$$P(|N_i/N - 1/S| <= \epsilon) \geq 1 - \exp(-2N\epsilon^2). \tag{9.33}$$

Let $\epsilon = 1/(2S)$, and substitute it back to the above inequality, we get:

$$P(0.5(N/S) \leq N_i \leq 1.5(N/S)) = P(\sqrt{0.5(N/S)} \leq \sqrt{N_i} \leq \sqrt{1.5(N/S)}) \geq 1 - \exp(-2N/S^2). \tag{9.34}$$

Hence, we can lower bound $\mathbb{E}[\sqrt{N_i}]$ as follows:

$$\mathbb{E}[\sqrt{N_i}] \geq \sqrt{0.5N/S}(1 - \exp(-2N/S^2)). \tag{9.35}$$

Take N to infinity, we get:

$$\lim_{N\to\infty}\mathbb{E}[\sqrt{N_i}] \geq \sqrt{0.5N/S}. \tag{9.36}$$

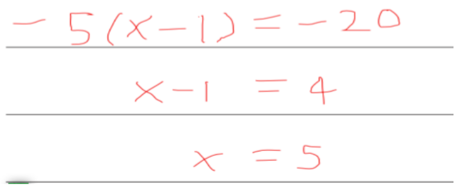
Substitute this result back to Eq. 9.32 and use the fact from Eq. 9.28, we get:

$$\lim_{N\to\infty}\mathbb{E}[R_N] = \lim_{N\to\infty}\mathop{\mathbb{E}}_{s^n\sim\rho_0,\forall n}\Big[\mathop{\mathbb{E}}_{\tilde{Q}^e(s_n)\sim P_{s_n},\forall n}\big[\sum_{n=1}^{N}(\pi_n^{s^n} \cdot \tilde{Q}^e(s^n) - \pi^*_{s^n} \cdot \tilde{Q}^e(s^n))\big]\Big] \geq c\sqrt{\ln(A)}\sum_{i=1}^{S}\mathbb{E}[\sqrt{N_i}]$$
$$\geq c\sqrt{\ln(A)}S\sqrt{0.5N/S} = \Omega(\sqrt{S\ln(A)N}).$$

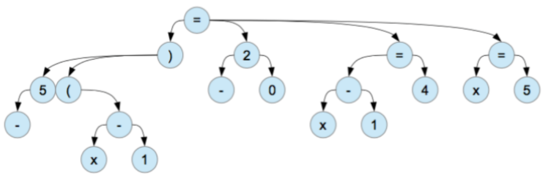
Hence we prove the theorem. □

9.1.10 Details of Dependency Parsing for Handwritten Algebra

In Fig. 9.1, we show an example of set of handwritten algebra equations and its dependency tree from a arc-hybird sequence $slssslssrrllslsslssrrslssrlssrrslssrr$. The preprocess step cropped individual symbols one by one from left to right and from the top equation to the bottom one, centered them, scaled symbols to 40 by 40 images, and finally formed them as a sequence of images.



(a) Handwritten algebra equations



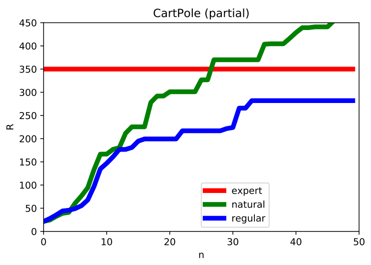
(b) Dependency tree

Figure 9.1: An example of a set of handwritten algebra equations (a) and its corresponding dependency tree (b).

Since in the most common dependency parsing setting, there is no immediate reward at every parsing step, the reward-to-go $Q^*(s, a)$ is computed by using UAS as follows: start from s and apply action a, then use expert π^* to roll out til the end of the parsing process; $Q^*(s, a)$ is the UAS score of the final configuration. Hence AggreVaTeD can be considered as directly maximizing the UAS score, while previous approaches such as DAgger or SMILe Ross et al. [2011] tries to mimic expert's actions and hence are not directly optimizing the final objective.

9.1.11 Additional Experiments on Partial Observable Setting

We test AggreVaTeD with Gated Recurrent Unit (GRU) based policies on a partially observable CartPole environment. Again the expert has access to the full state while the observation excludes the velocity information of the cart.



(a)

Figure 9.2: AggreVaTeD with GRU on the partial observable CartPole setting.

Fig. 9.2 shows that even under partial observable setting, AggreVaTeD with RNN-based policies can also outperform sub-optimal experts.

9.2 Missing Proofs in Chapter 3

9.2.1 Proof of Theorem 10

Before proving the theorem, we introduce some notations and useful lemmas.

Given a pair of π and f, denote random variable $v_i = K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - f(\tilde{x}_{h+1}^i)$, recall the definition of the utility function $u(\pi, f)$:

$$u(\pi, f) = \frac{1}{N}\sum_{i=1}^{N} K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - \frac{1}{N}\sum_{n=1}^{N} f(\tilde{x}_{h+1}^i) = \frac{1}{N}\sum_{n=1}^{N} v_i.$$

Denote $v = \mathbb{E}_{x\sim\nu_h, a\sim\pi(\cdot|x), x'\sim P^\star_{x,a}}[f(x')] - \mathbb{E}_{x\sim\mu^\star_{h+1}}[f(x)]$. It is easy to verify that $\mathbb{E}_i v_i = v$. We also have $|v_i - v| \leq 4K$. We can further bound the variance of $v_i - v$ as:

$$\begin{aligned}
\text{Var}_i(v_i - v) &= \mathbb{E}_i(v_i - v)^2 = \mathbb{E}_i v_i^2 - v^2 \leq \mathbb{E}_i v_i^2 \\
&= \mathbb{E}_i(K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - f(\tilde{x}_{h+1}^i))^2 \\
&\leq \mathbb{E}_i K^2\pi(a_h^i|x_h^i)f(x_{h+1}^i) - \mathbb{E}_i K\pi(a_h^i|x_h^i)f(x_{h+1}^i)f(\tilde{x}_{h+1}^i) + \mathbb{E}_i f(\tilde{x}_{h+1}^i)^2 \\
&\leq K + 1 + 1 \leq 2K,
\end{aligned}$$

where we used the fact that $|f(x)| \leq 1, \forall x$, $\pi(a|x) \leq 1, \forall x, a$, and the fact that a_h^i is sampled from a uniform distribution over $\mathcal{A}$. With that, we can apply Bernstein's inequality to $\{v_i\}$ together with a union bound over Π and $\mathcal{F}$, we will have the following lemma:

Lemma 10. *Given dataset $\mathcal{D} = \{x_h^i, a_h^i, x_{h+1}^i\}$ with $x_h^i \sim \nu_h, a_h^i \sim U(\mathcal{A}), x_{h+1}^i \sim P^\star_{x_h^i, a_h^i}$, and $\mathcal{D}^e = \{\tilde{x}_{h+1}^i\}$ with $\tilde{x}_{h+1}^i \sim \mu^\star_{h+1}$, for any pair $\pi \in \Pi, f \in \mathcal{F}$, with probability at least $1-\delta$,*

$$\begin{aligned}
&\left|\left(\frac{1}{N}\sum_{i=1}^{N} K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N} f(\tilde{x}_{h+1}^i)\right) - \left(\mathbb{E}_{(x,a,x')\sim\nu_h\pi P^\star}[f(x')] - \mathbb{E}_{x\sim\mu^\star_{h+1}}[f(x)]\right)\right| \\
&\leq 4\sqrt{\frac{2K\log(2|\Pi||\mathcal{F}|/\delta)}{N}} + \frac{8K\log(2|\Pi||\mathcal{F}|/\delta)}{N}. \qquad (9.37)
\end{aligned}$$

Let us define two loss functions for π and f:

$$\begin{aligned}
\ell_t(\pi) &= (1/N)\sum_{i=1}^{N} K\pi(a_h^i|x_h^i)f^t(x_{h+1}^i) \\
c_t(f) &= (1/N)\sum_{i=1}^{N} K\pi^t(a_h^i|x_h^i)f(x_{h+1}^i) - (1/N)\sum_{n=1}^{N} f(\tilde{x}_{h+1}^i).
\end{aligned}$$

For any $f, g : \mathcal{X}\times\mathcal{A}\times\mathcal{X} \to \mathbb{R}$, define $\langle f, g\rangle = \mathbb{E}_{(x,a)\sim\mathcal{D}_{x,a}} f(x,a)g(x,a)$, where we overload the notation and denote $\mathcal{D}$ as the empirical distribution over the dataset $\mathcal{D}$ (i.e., put probability $1/|\mathcal{D}|$ over each data point in $\mathcal{D}$), and $\mathcal{D}_{x,a}$ as the marginal distribution over x, a. With this notation, we can see that $\ell_t(\pi)$ can be written as a linear functional with respect to π:

$$\ell_t(\pi) = \langle \pi, Kf^t\rangle, \qquad (9.38)$$

where Kf_t is defined such that $Kf^t(x,a) = K\sum_{i=1}^N \mathbf{1}[x = x_h^i, a = a_h^i]f^t(x_{h+1}^i)$. Under this definition of inner product, we have:

$$\max_\pi \|\pi\| \leq 1, \quad \max \|Kf^t\| \leq K.$$

It is easy to verify that Algorithm 2 is running Best Response on loss $\{c_t(f)\}_t$ and running FTRL on loss $\{\ell_t(\pi)\}_t$. Using the no-regret guarantee from FTRL, for $\{\pi^t\}$, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\ell_t(\pi^t) - \min_{\pi\in\Pi}\frac{1}{T}\sum_{t=1}^{T}\ell_t(\pi) \leq \frac{K}{\sqrt{T}}. \tag{9.39}$$

Denote $\hat{\pi}^\star$ and $\hat{f}^\star$ as the minimizer and maximizer of Eqn **??**, i.e.,

$$(\hat{\pi}^\star, \hat{f}^\star) = \arg\min_{\pi\in\Pi}\arg\max_{f\in\mathcal{F}}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f(\tilde{x}_{h+1}^i)\right). \tag{9.40}$$

The following lemma quantifies the performance of $\bar{\pi} = \sum_{t=1}^{T}\pi^t/T$ and $\bar{f} = \sum_{t=1}^{T}f^t/T$:

Lemma 11. *Denote* $\bar{\pi} = \sum_{t=1}^{T}\pi^t/T$ *and* $\bar{f}^\star = \max_{f\in\mathcal{F}}\left(\mathbb{E}_{(x,a,x')\sim\nu_n\bar{\pi}P^\star}[f(x')] - \mathbb{E}_{x\sim\mu^\star_{h+1}}[f(x)]\right)$.
We have:

$$\frac{1}{N}\sum_{i=1}^{N}K\bar{\pi}(a_h^i|x_h^i)\bar{f}^\star(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{T}\bar{f}^\star(\tilde{x}_{h+1}^i) \leq \frac{1}{N}\sum_{i=1}^{N}K\hat{\pi}^\star(a_h^i|x_h^i)\hat{f}^\star(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}\hat{f}^\star(\tilde{x}_{h+1}^i) + \frac{K}{\sqrt{T}},$$

where $\hat{\pi}^\star, \hat{f}^\star$ *is defined in* (9.40).

Proof. Using the definition of ℓ_t and the no-regret property on $\{\pi_t\}$, we have:

$$\begin{aligned}&\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi^t(a_h^i|x_h^i)f^t(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f^t(\tilde{x}_{h+1}^i)\right)\\ &\leq \min_{\pi\in\Pi}\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi(a_h^i|x_h^i)f^t(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f^t(\tilde{x}_{h+1}^i)\right) + \frac{K}{\sqrt{T}}.\end{aligned}$$

Since $f^t = \arg\max_{f\in\mathcal{F}} c_t(f)$, we have:

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N}K\bar{\pi}(a_h^i|x_h^i)\bar{f}^\star(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}\bar{f}^\star(\tilde{x}_{h+1}^i) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi^t(a_h^i|x_h^i)\bar{f}^\star(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}\bar{f}^\star(\tilde{x}_{h+1}^i)\right)\\ &\leq \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi^t(a_h^i|x_h^i)f^t(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f^t(\tilde{x}_{h+1}^i)\right)\end{aligned}$$

We also have:

$$\begin{aligned}&\min_{\pi\in\Pi}\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\pi(a_h^i|x_h^i)f^t(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f^t(\tilde{x}_{h+1}^i)\right)\\ &\leq \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}K\hat{\pi}^\star(a_h^i|x_h^i)f^t(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f^t(\tilde{x}_{h+1}^i)\right)\\ &\leq \max_{f\in\{f^1,\dots,f^T\}}\left(\frac{1}{N}\sum_{i=1}^{N}K\hat{\pi}^\star(a_h^i|x_h^i)f(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^{N}f(\tilde{x}_{h+1}^i)\right)\end{aligned}$$

$$\leq \left(\frac{1}{N} \sum_{i=1}^{N} K \hat{\pi}^\star(a_h^i | x_h^i) \hat{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} \hat{f}^\star(\tilde{x}_{h+1}^i) \right),$$

where the first inequality uses the definition of $\min_{\pi \in \Pi}$, the second inequality uses the fact that the maximum is larger than the average, and the last inequality uses the fact that $\hat{f}^\star$ is the maximizer with respect to $\hat{\pi}^\star$.

Combining the above results, we have:

$$\frac{1}{N} \sum_{i=1}^{N} K \bar{\pi}(a_h^i | x_h^i) \bar{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} \bar{f}^\star(\tilde{x}_{h+1}^i) \leq \frac{1}{N} \sum_{i=1}^{N} K \hat{\pi}^\star(a_h^i | x_h^i) \hat{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} \hat{f}^\star(\tilde{x}_{h+1}^i) + \frac{K}{\sqrt{T}}.$$

□

Denote $\pi^\star$ and $f^\star$ as

$$\pi^\star, f^\star = \arg\min_{\pi \in \Pi} \arg\max_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim v, a \sim \pi, x' \sim P^\star_{x,a}} [f(x')] - \mathbb{E}_{x \sim \mu_h^{\pi^\star}} [f(x)] \right) \tag{9.41}$$

Now we are ready to prove Theorem 10

Proof of Theorem 10. Denote $C_N = 4\sqrt{\frac{2K \log(2|\Pi||\mathcal{F}|/\delta)}{N}} + \frac{8K \log(2|\Pi||\mathcal{F}|/\delta)}{N}$. First, using the concentration result from Lemma 10, we have:

$$\left| \left(\frac{1}{N} \sum_{i=1}^{N} K \bar{\pi}(a_h^i | x_h^i) \bar{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{T} \bar{f}^\star(\tilde{x}_{h+1}^i) \right) - \left(\mathbb{E}_{(x,a,x') \sim \nu_h \bar{\pi} P^\star} [\bar{f}^\star(x')] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [\bar{f}^\star(x)] \right) \right|$$
$$\leq \frac{1}{T} \sum_{t=1}^{T} \left| \left(\frac{1}{N} \sum_{i=1}^{N} K \pi^t(a_h^i | x_h^i) \bar{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{T} \bar{f}^\star(\tilde{x}_{h+1}^i) \right) - \left(\mathbb{E}_{(x,a,x') \sim \nu_h \pi^t P^\star} [\bar{f}^\star(x')] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [\bar{f}^\star(x)] \right) \right|$$
$$\leq \frac{1}{T} \sum_{t=1}^{T} (C_N) = C_N.$$

On the other hand, for $\pi^\star, f^\star$, we have:

$$\left| \left(\frac{1}{N} \sum_{i=1}^{N} K \pi^\star(a_h^i | x_h^i) f^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} f^\star(\tilde{x}_{h+1}^i) \right) - \left(\mathbb{E}_{(x,a,x') \sim \nu_h \pi^\star P^\star} [f^\star(x')] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [f^\star(x)] \right) \right| \tag{9.42}$$
$$\leq C_N. \tag{9.43}$$

Define $\hat{f}' = \max_{f \in \mathcal{F}} (\frac{1}{N} \sum_{i=1}^{N} K \pi^\star(a_h^i | x_h^i) f(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} f(\tilde{x}_{h+1}^i))$. Combine the above inequalities together, we have:

$$\max_{f \in \mathcal{F}} \mathbb{E}_{(x,a,x') \sim \nu_h \bar{\pi} P^\star} [f(x')] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [f(x)] = \mathbb{E}_{(x,a,x') \sim \nu_h \bar{\pi} P^\star} [\bar{f}^\star(x')] - \mathbb{E}_{x \sim \mu^\star_{h+1}} [\bar{f}^\star(x)]$$
$$\leq \frac{1}{N} \sum_{i=1}^{N} K \bar{\pi}(a_h^i | x_h^i) \bar{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{T} \bar{f}^\star(\tilde{x}_{h+1}^i) + C_N$$
$$\leq \frac{1}{N} \sum_{i=1}^{N} K \hat{\pi}^\star(a_h^i | x_h^i) \hat{f}^\star(x_{h+1}^i) - \frac{1}{N} \sum_{i=1}^{N} \hat{f}^\star(\tilde{x}_{h+1}^i) + \frac{K}{\sqrt{T}} + C_N$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} K\pi^\star(a_h^i|x_h^i)\hat{f}'(x_{h+1}^i) - \frac{1}{N}\sum_{n=1}^{N} \hat{f}'(\tilde{x}_{h+1}^i) + \frac{K}{\sqrt{T}} + C_N$$
$$\leq \mathbb{E}_{(x,a,x')\sim\nu_h\pi^\star P^\star}[\hat{f}'(x')] - \mathbb{E}_{x\sim\mu_{h+1}^\star}[\hat{f}'(x)] + 2C_N + \frac{K}{\sqrt{T}}$$
$$\leq \mathbb{E}_{(x,a,x')\sim\nu_h\pi^\star P^\star}[f^\star(x')] - \mathbb{E}_{x\sim\mu_{h+1}^\star}[f^\star(x)] + 2C_N + \frac{K}{\sqrt{T}},$$

where the first equality uses the definition of $\bar{f}^\star$, the second inequality uses Lemma 11, the third inequality uses the fact that $\hat{\pi}^\star$ and $\hat{f}^\star$ are the min-max solution of (9.40), and the fifth inequality uses the fact that $f^\star$ is the maximizer of (**??**) given $\pi^\star$. Hence, we prove the theorem.

□

9.2.2 Proof of Theorem 11

Lemma 12. *There exists a distribution $D \in \mathcal{X}$, such that for any two datasets $S_1 = \{x_1, \ldots, x_K\}$ and $S_2 = \{x'_1, \ldots, x'_K\}$ where x_i and x'_i are drawn i.i.d from D, as long as $K = O(\log(|\mathcal{X}|))$, then:*

$$\lim_{|\mathcal{X}|\to\infty} \Pr\left([S_1 \cap S_2 = \emptyset]\right) = 1.$$

Proof. We simply set D to be a uniform distribution of $\mathcal{X}$. Denote $|\mathcal{X}| = N$, and $K = O(\log(N))$. The probability of S_1 and S_2 does not have any overlap samples can be easily computed as:

$$\mathrm{P}(S_1 \cap S_2 = \emptyset) \geq \mathrm{P}(S_1 \cap S_2 = \emptyset \text{ and } S_1 \text{ does not have repeated samples}).$$

Note that the probability that S_1 does not have repeated samples can be computed as:

$$\mathrm{P}(S_1 \text{ does not have repeated samples}) = (1 - 1/N)(1 - 2/N)(1 - (K-1)/N).$$

When $N \to \infty$ and $K = O(\log N)$, we have:

$$\lim_{N\to\infty} \mathrm{P}(S_1 \text{ does not have repeated samples}) = 1.$$

Now, conditioned on the event that S_1 does not contain repeated samples, we have:

$$\mathrm{P}(S_1 \cap S_2 = \emptyset) = (1 - K/N)^K = (1 - K/N)^{(N/K)*(K^2/N)}$$

Take $N \to \infty$, we know that $\lim_{x\to\infty}(1 - 1/x)^x = 1/e$ and $\lim_{N\to\infty} K^2/N = 0$, hence we have:

$$\lim_{N\to\infty}(1 - K/N)^K = \lim_{N\to\infty}(1 - K/N)^{(N/K)(K^2/N)} = \lim_{N\to\infty}(1/e)^{K^2/N} = 1.$$

Hence we prove the lemma by coming two results above. □

We construct the MDP using the above lemma. The MDP has $H = 2$, two actions $\{a, a^\star\}$, and the initial distribution ρ assigns probability one to a unique state $\hat{x} \in \mathcal{X}$. The expert policy $\pi^\star$ is designed to be $\pi^\star(a^\star|\hat{x}) = 1$, i.e., the expert's action at time step $h = 1$ is $a^\star$. We split the state space $\mathcal{X}$ into half and half, denoted as $\mathcal{X}_1$ and $\mathcal{X}_2$, such that $\mathcal{X}_1 \cup$

$\mathcal{X}_2 = \emptyset$ and $|\mathcal{X}_1| = |\mathcal{X}_2| = N/2$. We design the MDP's dynamics such that $P(\cdot|\hat{x}, a)$ assigns probability $2/N$ to each state in $\mathcal{X}_1$ and assigns probability 0 to any other state in $\mathcal{X}_2$. We design $P(\cdot|\hat{x}, a^\star)$ such that it assigns probability $2/N$ to each state in $\mathcal{X}_2$ and zero to each state in $\mathcal{X}_1$.

Denote $\mathcal{D}^\star = \{\tilde{x}_2^{(i)}\}_{i=1}^K$ as the states generated from the expert by executing $a^\star$ at $\hat{x}$. For any two policies π and π', such that $\pi(a^\star|\hat{x}) = 1$ and $\pi'(a|\hat{x}) = 1$, denote $\mathcal{D} = \{x_2^i\}_{i=1}^K$ as the dataset sampled from executing π at $\hat{x}$ K many times, and $\mathcal{D}' = \{x_2^{\prime(i)}\}_{i=1}^K$ as the dataset sampled from executing π' at $\hat{x}$ K many times. From Lemma 12, we know that

$$\lim_{N\to\infty} \mathrm{P}(\mathcal{D} \cap \mathcal{D}^\star = \emptyset) = 1, \quad \mathrm{P}(\mathcal{D} \cap \mathcal{D}' = \emptyset) = 1.$$

Hence asymptotically either $\mathcal{D}$ nor $\mathcal{D}'$ will overlap with $\mathcal{D}^\star$, unless $K = \mathrm{poly}(N)$.

9.2.3 Proof of Theorem 12

We first present some extra notations and useful lemmas below.

Lemma 13 (Performance Difference Lemma [Kakade and Langford, 2002]). *Consider a policy $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_H\}$ and $\boldsymbol{\pi}^\star = \{\pi_1^\star, \ldots, \pi_H^\star\}$. We have:*

$$J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) = \sum_{h=1}^H \mathbb{E}_{x\sim\mu_h^\pi}\left[\mathbb{E}_{a\sim\pi_h(\cdot|x)} Q_h^\star(x,a) - V_h^\star(x)\right].$$

Note that under our setting, i.e., the cost function does not depend on actions, the above equation can be simplified to:

$$\begin{aligned} J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) &= \sum_{h=1}^H \mathbb{E}_{x\sim\mu_h^\pi}\left[\mathbb{E}_{a\sim\pi_h(\cdot|x)} Q_h^\star(x,a) - V_h^\star(x)\right] && (9.44)\\ &= \sum_{h=1}^H \mathbb{E}_{x\sim\mu_h^\pi}\left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right], && (9.45) \end{aligned}$$

where we use Bellman equations, i.e., $Q_h^\star(x,a) = c(x) + \mathbb{E}_{x'\sim P_{x,a}} V_{h+1}^\star(x')$ and $V_h^\star(x) = c(x) + \mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}} V_{h+1}^\star(x')$.

Note that for any h, we have:

$$\begin{aligned} &\left|\mathbb{E}_{x\sim\mu_h^\pi}\left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right]\right| \\ &\le \left|\mathbb{E}_{x\sim\mu_h^\pi}\left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right] - \mathbb{E}_{x\sim\mu_h^\star}\mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right| \\ &\quad + \left|\mathbb{E}_{x\sim\mu_h^\star}\mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{x\sim\mu_h^\pi}\mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right| \\ &\le d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) + \left|\mathbb{E}_{x\sim\mu_h^\star}\mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{x\sim\mu_h^\pi}\mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right| \\ &\le d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) + \Delta_h + 2\epsilon_{\mathrm{be}}, \qquad (9.46) \end{aligned}$$

where the first inequality comes from the triangle inequality, the second inequality comes from the fact that $V_{h+1}^\star \in \mathcal{F}_{h+1}$, and in the third inequality, we denote $\Delta_h = \max_{f\in\mathcal{F}}\left|\mathbb{E}_{x\sim\mu_h^\star}[f(x)] - \mathbb{E}_{x\sim\mu_h^\pi}[f(x)]\right|$, and ϵ_{be} is introduced because $\Gamma_h V_{h+1}^\star$ might not in $\mathcal{F}_h$.

Now we are ready to prove the main theorem.

Proof of Theorem 12. We consider the h'th iteration. Let us denote $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_{h-1}\}$ and $\mu_h^{\boldsymbol{\pi}}$ as the observation distribution at time step h of following policies $\boldsymbol{\pi}$ starting from the initial distribution ρ. Denote $\mu_{h+1}^\star$ as the observation distribution of the expert policy $\boldsymbol{\pi}^\star$ at time step $h+1$ starting from the initial distribution ρ. Note that the dataset $\{\tilde{x}_{h+1}^{(i)}\}_{i=1}^n$ is generated from distribution $\mu_{h+1}^\star$. The data at $\mathcal{D}$ is generated i.i.d by first drawing sample $x_h^{(i)}$ from $\mu_h^{\boldsymbol{\pi}}$ (i.e., executing $\pi_1, \dots \pi_{h-1}$), and then sample action $a_h^{(i)} \sim U(\mathcal{A})$, and then sample $x_{h+1}^{(i)}$ from the real system $P_{x_h^{(i)}, a_h^{(i)}}$.

Mapping the above setup to the setup in Theorem 10, i.e., set

$$\nu_h = \mu_h^{\boldsymbol{\pi}}, \;\; T = \Theta(4K^2/\epsilon^2), \;\; N = \Theta(K \log(|\Pi||\mathcal{F}|/\delta)/\epsilon^2),$$

Algorithm 2 will output a policy π_h such that with probability at least $1-\delta$:

$$d_{\mathcal{F}_{h+1}}(\pi_h | \mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) \leq \min_{\pi \in \Pi_h} d_{\mathcal{F}_{h+1}}(\pi | \mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) + \epsilon.$$

Recall the definition of inherent Bellman Error ϵ_{be} with respect to $\mathcal{F}_h$ and $\boldsymbol{\pi}^\star$:

$$\epsilon_{\mathrm{be},h} = \max_{g \in \mathcal{F}_{h+1}} \min_{f \in \mathcal{F}_h} \|f - \Gamma_h g\|_{(\mu_h^{\boldsymbol{\pi}} + \mu_h^\star)/2}.$$

Denote $\hat{f}$ as:

$$\hat{f} = \arg\max_{f \in \mathcal{F}_{h+1}} \left(\mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[f(x')] \right] - \mathbb{E}_{x \sim \mu_h^\star} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[f(x')] \right] \right),$$

and $\hat{g}$ as:

$$\hat{g} = \arg\min_{g \in \mathcal{F}_h} \|g - \Gamma_h \hat{f}\|_{(\mu_h^{\boldsymbol{\pi}} + \mu_h^\star)/2}$$

Now we upper bound $\min_{\pi \in \Pi_h} d_{\mathcal{F}_{h+1}}(\pi | \mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star)$ as follows.

$$\begin{aligned}
&\min_{\pi \in \Pi_h} d_{\mathcal{F}_{h+1}}(\pi | \mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) \leq d_{\mathcal{F}_{h+1}}(\pi_h^\star | \mu_h^{\boldsymbol{\pi}}, \mu_{h+1}^\star) \\
&= \max_{f \in \mathcal{F}_{h+1}} \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}, a \sim \pi_h^\star, x' \sim P_{x,a}} f(x') - \mathbb{E}_{x \sim \mu_h^\star, a \sim \pi_h^\star, x' \sim P_{x,a}} f(x') \right| \\
&= \max_{f \in \mathcal{F}_{h+1}} \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[f(x')] \right] - \mathbb{E}_{x \sim \mu_h^\star} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[f(x')] \right] \right| \\
&= \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[\hat{f}(x')] \right] - \mathbb{E}_{x \sim \mu_h^\star} \left[\mathbb{E}_{a \sim \pi_h^\star(\cdot|x), x' \sim P_{x,a}}[\hat{f}(x')] \right] \right| \\
&\leq \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}}[\hat{g}(x)] - \mathbb{E}_{x \sim \mu_h^\star}[\hat{g}(x)] \right| + \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}}[\hat{g}(x) - \mathbb{E}_{a \sim \pi_h^\star, x' \sim P_{x,a}} \hat{f}(x')] \right| + \left| \mathbb{E}_{x \sim \mu_h^\star}[\hat{g}(x) - \mathbb{E}_{a \sim \pi_h^\star, x' \sim P_{x,a}} \hat{f}(x')] \right| \\
&\leq \max_{f \in \mathcal{F}_h} \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}}[f(x)] - \mathbb{E}_{x \sim \mu_h^\star}[f(x)] \right| + 2\mathbb{E}_{x \sim (\mu_h^{\boldsymbol{\pi}} + \mu_h^\star)/2} \left[|\hat{g}(x) - \mathop{\mathbb{E}}_{a \sim \pi_h^\star, x' \sim P_{x,a}} \hat{f}(x')| \right] \\
&\leq \max_{f \in \mathcal{F}_h} \left| \mathbb{E}_{x \sim \mu_h^{\boldsymbol{\pi}}}[f(x)] - \mathbb{E}_{x \sim \mu_h^\star}[f(x)] \right| + 2\epsilon_{\mathrm{be}} \\
&= \Delta_h + 2\epsilon_{\mathrm{be}},
\end{aligned}$$

where the first inequality comes from the realizable assumption that $\pi_h^\star \in \Pi$, the second inequality comes from an application of triangle inequality, and the third inequality comes from the definition of ϵ_{be} and the fact that $\hat{g} \in \mathcal{F}_h$.

After learn π_h, $\boldsymbol{\pi}$ is updated to $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_h\}$. For Δ_{h+1}, we have:

$$\begin{aligned}
\Delta_{h+1} &= \max_{f \in \mathcal{F}_{h+1}} \left| \mathbb{E}_{x \sim \mu^{\boldsymbol{\pi}}_{h+1}}[f(x)] - \mathbb{E}_{x \sim \mu^\star_{h+1}}[f(x)] \right| \\
&= \max_{f \in \mathcal{F}_{h+1}} \left| \mathbb{E}_{x \sim \mu^{\boldsymbol{\pi}}_h, a \sim \pi_h, x' \sim P_{x,a}}[f(x')] - \mathbb{E}_{x \sim \mu^\star_h, a \sim \pi^\star_h, x' \sim P_{x,a}}[f(x')] \right| \\
&= d_{\mathcal{F}_{h+1}}(\pi_h | \mu^{\boldsymbol{\pi}}_h, \mu^\star_{h+1}) \leq \min_{\pi \in \Pi_h} d_{\mathcal{F}_{h+1}}(\pi | \mu^{\boldsymbol{\pi}}_h, \mu^\star_{h+1}) + O(\epsilon) \leq \Delta_h + 2\epsilon_{\text{be}} + O(\epsilon).
\end{aligned}$$

Define $\Delta_0 = \max_f |\mathbb{E}_{x \sim \rho}[f(x)] - \mathbb{E}_{x \sim \rho}[f(x)]| = 0$, we have for any h,

$$\Delta_h \leq 2h\epsilon_{\text{be}} + O(h\epsilon).$$

Now we link Δ_h to the performance of the policy $J(\boldsymbol{\pi})$. From (9.46), we know that:

$$\begin{aligned}
&\left| \mathbb{E}_{x \sim \mu^{\boldsymbol{\pi}}_h} \left[\mathbb{E}_{a \sim \pi_h, x' \sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{a \sim \pi^\star_h, x' \sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] \right| \\
&\leq d_{\mathcal{F}_{h+1}}(\pi_h | \mu^{\boldsymbol{\pi}}_h, \mu^\star_{h+1}) + \Delta_h + 2\epsilon_{\text{be}} \\
&\leq \Delta_h + O(\epsilon) + 2\epsilon_{\text{be}} + \Delta_h + 2\epsilon_{\text{be}} = 2\Delta_h + 4\epsilon_{\text{be}} + O(\epsilon) \\
&\leq 4h\epsilon_{\text{be}} + O(2h\epsilon) + 4\epsilon_{\text{be}} + O(\epsilon) = O(h\epsilon_{\text{be}}) + O(h\epsilon).
\end{aligned}$$

Using Performance Difference Lemma (Lemma 13), we know that:

$$\begin{aligned}
J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) &\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x \sim \mu^{\boldsymbol{\pi}}_h} \left[\mathbb{E}_{a \sim \pi_h, x' \sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{a \sim \pi^\star_h, x' \sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] \right| \\
&\leq \sum_{h=1}^{H} 4h\epsilon_{\text{be}} + 4\epsilon_{\text{be}} + O(2h\epsilon) + O(\epsilon) \\
&\leq 4H^2\epsilon_{\text{be}} + 2H\epsilon_{\text{be}} + O(2H^2\epsilon) + O(H\epsilon) = O(H^2\epsilon_{\text{be}}) + O(H^2\epsilon)
\end{aligned}$$

□

9.2.4 Proof of Proposition 1

We first show the construction of the MDP below. The MDP has horizon H, $2^H - 1$ many states, and two actions $\{l, r\}$ standing for *go left* and *go right* respectively. All states are organized in a perfect balanced binary tree, with 2^{H-1} many leafs at level $h = H$, and the first level $h = 1$ contains only a root. The transition is deterministic such that at any internal state, taking action l leads to the state's left child, and taking action r leads to the state's right child. Each internal node has cost zero, and all leafs will have nonzero cost which we will specify later. Note that in such MDP, any sequence of actions $\{a_1, \dots, a_{H-1}\}$ with $a_i \in \{l, r\}$ deterministically leads to one and only one leaf, and the total cost of the sequence of actions is only revealed once the leaf is reached.

The first part of the proposition is proved by reducing the problem to Best-arm identification in multi-armed bandit (MAB) setting. We use the following lower bound of best-arm identification in MAB from Krishnamurthy et al. [2016]:

Lemma 14 (Lower bound for best arm identification in stochastic bandits from Krishnamurthy et al. [2016])**.** *For any $K \geq 2$ and $\epsilon \in (0, \sqrt{1/8}]$, and any best-arm identification algorithm, there exists a multi-armed bandit problem for which the best arm $i^\star$ is ϵ better than all others, but for which the estimate $\hat{i}$ of the best arm must have* $\mathrm{P}(\hat{i} \neq i^\star) \geq 1/3$ *unless the number of samples collected T is at least $K/(72\epsilon^2)$.*

Given any MAB problem with K arms, without loss of generality, let us assume $K = 2^H - 1$ for some $H \in \mathbb{N}^+$. Any such MAB problem can be reduced to the above constructed binary tree MDP with horizon H, and $2^H - 1$ leafs. Each arm in the original MAB will be encoded by a unique sequence of actions $\{a_h\}_{h=1}^{H-1}$ with $a_h \in \{l, r\}$, and its corresponding leaf. We assign each leaf the cost distribution of the corresponding arm. The optimal policy in the MDP, i.e., the sequence of actions leading to the leaf that has the smallest expected cost, is one-to-one corresponding to the best arm, i.e., the arm that has the smallest expected cost in the MAB. Hence, without any further information about the MDP, any RL algorithm that aims to find the near-optimal policy must suffer the lower bound presented in Lemma 14, as otherwise one can solve the original MAB by first converting the MAB to the MDP and then running an RL algorithm. Hence, we prove the first part of the proposition.

For the second part, let us denote the sequence of the observations from the expert policy as $\{\tilde{x}_h\}_{h=1}^{H}$, i.e., the sequence of states corresponding to the optimal sequence of actions where the last state $\tilde{x}_H$ has the smallest expected cost. We design an IL algorithm as follows.

We initialize a sequence of actions $\boldsymbol{a} = \emptyset$. At every level h, staring at $h = 1$, we try any sequence of actions with prefix $\boldsymbol{a} \circ l$ ($\boldsymbol{a} \circ a$ means we append action a to end of the sequence), record the observed observation x_{h+1}^l; we then reset and try any sequence of actions with prefix $\boldsymbol{a} \circ r$, and record the observed observation x_{h+1}^l. If $x_{h+1}^l = \tilde{x}_{h+1}$, then we append l to $\boldsymbol{a}$, i.e., $\boldsymbol{a} = \boldsymbol{a} \circ l$, otherwise, we append r, i.e., $\boldsymbol{a} = \boldsymbol{a} \circ r$. We continue the above procedure until $h = H - 1$, and we output the final action sequence $\boldsymbol{a}$.

Due to the deterministic transition, by induction, it is easy to verify that the outputted sequence of actions $\boldsymbol{a}$ is exactly equal to the optimal sequence of actions executed by the expert policy. Note that in each level h, we only drew two trajectories from the MDP. Hence the total number trajectories before finding the optimal sequence of actions is at most $2(H - 1)$. Hence we prove the proposition.

9.2.5 Reduction to LP

Let us denote a set $\{y_i\}_{i=1}^{2N}$ such that $\{y_1, \ldots, y_N\} = \{x_1, \ldots, x_N\}$, and $\{y_{N+1} \ldots, y_{2N}\} = \{x'_1, \ldots, x'_N\}$. Denote $d_{i,j} = \mathcal{D}(y_i, y_j)$ for $i \neq j$, and $c_i = 1/N$ for $i \in [N]$ and $c_i = -1/N$ for $i \in [N+1, 2N]$. We formulate the following LP with $2N$ variables and $O(N^2)$ many constraints:

$$\max_{\alpha_1,\ldots,\alpha_{2N}} \sum_{i=1}^{2N} c_i \alpha_i, \quad s.t., \forall i \neq j, -L d_{i,j} \leq \alpha_i - \alpha_j \leq L d_{i,j}, \quad \forall i, -1 \leq \alpha_i \leq 1. \tag{9.47}$$

Denote the solution of the above LP as $\alpha_i^\star$. We will have the following claim:

Claim 25 (LP Oracle)**.** *Given $\mathcal{F}$ in* (3.5)*, $\{x_i\}_{i=1}^N$, and $\{x'_i\}_{i=1}^N$, denote $\{\alpha_i^\star\}_{i=1}^{2N}$ as the solution of the LP from* (9.47)*, we have:* $\sup_{f\in\mathcal{F}} \left(\sum_{i=1}^N f(x_i)/N - \sum_{i=1}^N f(x'_i)/N \right) = \sum_{i=1}^{2N} c_i \alpha_i^\star$.

Proof of Claim 25. Given the solutions $\{\alpha_i^\star\}_{i=1}^{2N}$, we first are going to construct a function $\hat{f} : \mathcal{X} \to \mathbb{R}$, such that for any y_i, we have $\hat{f}(y_i) = \alpha_i^\star$, and $\hat{f} \in \mathcal{F}$. Denote $L^\star = \max_{i\neq j} |\alpha_i^\star - \alpha_j^\star|/d_{i,j}$. Note that $L^\star \leq L$. The function $\hat{f}$ is constructed as:

$$\hat{f}(x) = \max\left(-1, \min\left(1, \min_{i\in[2N]} L^\star \mathcal{D}(y_i, x) + \alpha_i^\star\right)\right)$$

First of all, we show that for any y_i, we have $\hat{f}(y_i) = \alpha_i^\star$. For any $j \neq i$, we have:

$$L^\star \mathcal{D}(y_j, y_i) + \alpha_j^\star \geq |\alpha_j^\star - \alpha_i^\star| + \alpha_j^\star \geq \alpha_i^\star,$$

where the first inequality uses the definition of $L^\star$. Also we know that $-1 \leq \alpha_i^\star \leq 1$. Hence we have that for y_i, $\max(-1, \min(1, \min_{j\in[2N]} L^\star \mathcal{D}(y_j, y_i) + \alpha_j^\star)) = \alpha_i^\star$.

Now we need to prove that $\hat{f}$ is L-Lipschitz continuous. Note that we just need to prove that $\min_i L^\star \mathcal{D}(y_i, x) + \alpha_i^\star$ is L-Lipschitz continuous, since for any L-Lipschitz continuous function $f(x)$, we have $\max(1, f(x))$ and $\min(-1, f(x))$ to be L-Lipschitz continuous as well.

Consider any two points x and x' such that $x \neq x'$. Denote $\hat{i}$ as $\arg\min_i L^\star \mathcal{D}(y_i, x) + \alpha_i^\star$ and $\hat{i}' = \arg\min_i L^\star \mathcal{D}(y_i, x') + \alpha_i^\star$. We have:

$$\begin{aligned}
&\hat{f}(x) - \hat{f}(x') = L^\star \mathcal{D}(y_{\hat{i}}, x) + \alpha_{\hat{i}}^\star - (L^\star \mathcal{D}(y_{\hat{i}'}, x') + \alpha_{\hat{i}'}^\star) \\
&\leq L^\star \mathcal{D}(y_{\hat{i}'}, x) + \alpha_{\hat{i}'}^\star - (L^\star \mathcal{D}(y_{\hat{i}'}, x') + \alpha_{\hat{i}'}^\star) \\
&\leq L^\star \mathcal{D}(x, x'),
\end{aligned}$$

where for the first inequality we used the definition of $\hat{i}$, and the second inequality uses the triangle inequality. Similarly, one can show that

$$\hat{f}(x) - \hat{f}(x') \geq -L^\star \mathcal{D}(x, x').$$

Combine the above two inequalities and the fact that $L^\star \leq L$, we conclude that $\hat{f}$ is L-Lipschitiz continuous.

Now we have constructed $\hat{f}$ such that $\hat{f}(y_i) = \alpha_i^\star$ for all $i \in [2N]$ and $\hat{f} \in \mathcal{F}$. Now suppose that there exists a function $f' \in \mathcal{F}$, such that $|\sum_{i=1}^N f'(x_i)/N - \sum_{i=1}^N f'(x_i')/N| > |\sum_{i=1}^N \hat{f}(x_i)/N - \sum_{i=1}^N \hat{f}'(x_i')/N|$, then we must have for some $i \in [2N]$, $f'(y_i) \neq \alpha_i^\star$. However, since $f' \in \mathcal{F}$, we must have that $\{f'(y_i)\}_{i=1}^{2N}$ satisfies all constrains in the LP in (9.47). Hence the assumption that $\sum_{i=1}^{2N} c_i f'(y_i) > \sum_{i=1}^{2N} c_i \alpha_i^\star$ contradicts to the fact that $\{\alpha_i\}_{i=1}^{2N}$ is the maximum solution of the LP formulation in (9.47). Hence we prove the claim.

□

9.2.6 Proof of Corollary 1

Since in this setting, $\mathcal{F}_h$ for all $h \in [H]$ contains infinitely many functions, we need to discretize $\mathcal{F}_h$ before we can apply the proof techniques from the proof of Theorem 12. We use covering number.

Denote $\mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})$ as the ϵ-cover of the metric space $(\mathcal{X}, \mathcal{D})$. Namely, for any $x \in \mathcal{X}$, there exists a $x' \in \mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})$ such that $D(x', x) \leq \epsilon$. Consider any function class $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}, \|f\|_L \leq L, \|f\|_\infty \leq 1\}$ with $L \in \mathbb{R}^+$. Below we construct the ϵ-cover over $\mathcal{F}$.

For any $f \in \mathcal{F}$, denote $\bar{f} \in \mathbb{R}^{|\mathcal{N}(\mathcal{X},\epsilon,\mathcal{D})|}$ with the i-th element $\bar{f}_i$ being the function value $f(x_i)$ measured at the i-th element x_i from $\mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})$. Hence $\bar{\mathcal{F}} \triangleq \{\bar{f} : f \in \mathcal{F}\} \in \mathbb{R}^{|\mathcal{N}(\mathcal{X},\epsilon,\mathcal{D})|}$, and $\|\bar{f}\|_\infty \leq C$ for any $\bar{f} \in \bar{\mathcal{F}}$. Denote $\bar{\mathcal{N}}(\bar{\mathcal{F}}, \alpha, \|\cdot\|_\infty)$ as the α-cover of $\bar{\mathcal{F}}$. Let us denote the set $\mathcal{N} \triangleq \{f \in \mathcal{F} : \bar{f} \in \bar{\mathcal{N}}(\tilde{\mathcal{F}}, \alpha, \|\cdot\|_\infty)\}$.

Claim 26. *With the above set up, for $\mathcal{F}$'s $(\alpha + 2L\epsilon)$-cover, i.e., $\mathcal{N}(\mathcal{F}, \alpha + 2L\epsilon, \|f\|_\infty)$, we have*

$$|\mathcal{N}(\mathcal{F}, \alpha + 2L\epsilon, \|\cdot\|_\infty)| \leq \left|\bar{\mathcal{N}}(\bar{\mathcal{F}}, \alpha, \|\cdot\|_\infty)\right| \leq \left(\frac{1}{\alpha}\right)^{|\mathcal{N}(\mathcal{X},\epsilon,\mathcal{D})|}.$$

Proof. By definition, we know that for any $\bar{f} \in \bar{\mathcal{F}}$, we have that there exists a $\bar{f}^\star \in \bar{\mathcal{F}}$ such that $\|\bar{f} - \bar{f}^\star\|_\infty \leq \alpha$. Now consider $\|f - f^\star\|_\infty$. Denote $x^\star = \arg\max_x |f(x) - f^\star(x)|$ and $x^{\star\prime}$ is its closest point in $\mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})$. We have:

$$\begin{aligned}
&\sup_x |f(x) - f^\star(x)| = |f(x^\star) - f^\star(x^\star)| \leq \left|f(x^\star) - f(x^{\star\prime})\right| + \left|f(x^{\star\prime}) - f^\star(x^\star)\right| \\
&\leq \left|f(x^\star) - f(x^{\star\prime})\right| + \left|f(x^{\star\prime}) - f^\star(x^{\star\prime})\right| + \left|f^\star(x^{\star\prime}) - f^\star(x^\star)\right| \\
&\leq L\epsilon + \alpha + L\epsilon = 2L\epsilon + \alpha,
\end{aligned}$$

where the last inequality comes from the fact that $f, f^\star$ are L-Lipschitz continuous, $\mathcal{D}(x^\star, x^{\star\prime}) \leq \epsilon$, $\|\bar{f} - \bar{f}^\star\|_\infty \leq \alpha$ and $x^{\star\prime} \in \mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})$. Hence, we just identify a subset of $\mathcal{F}$ such that it forms a $\alpha + 2L\epsilon$ cover for $\mathcal{F}$ under norm $\|\cdot\|_\infty$.

Note that $\bar{\mathcal{N}}$ is a α-cover with $\|\cdot\|_\infty$ for $\bar{\mathcal{F}}$ which is a subset of $|\mathcal{N}(\mathcal{X}, \epsilon, \mathcal{D})|$-dimension space. By standard discretization along each dimension, we prove the claim. □

From the above claim, setting up α and ϵ properly, we have:

$$|\mathcal{N}(\mathcal{F}, \epsilon/K, \|\cdot\|_\infty)| \leq \left(\frac{K}{\epsilon}\right)^{|\mathcal{N}(\mathcal{X}, \epsilon/(3KL), \mathcal{D})|}.$$

Extending the analysis of Theorem 12 simply results extending the concentration result in Lemma 10. Specifically via Bernstein's inequality and a union bound over $\Pi \times \mathcal{N}(\mathcal{F}, \epsilon/K, \|\cdot\|_\infty)$, we have that for any $\pi \in \Pi$, $\tilde{f} \in \mathcal{N}(\mathcal{F}, \epsilon/K, \|\cdot\|_\infty)$, with probability at least $1 - \delta$,

$$\begin{aligned}
&\left|\left(\frac{1}{N}\sum_{i=1}^N K\pi(a_h^i|x_h^i)\tilde{f}(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^N \tilde{f}(\tilde{x}_{h+1}^i)\right) - \left(\mathbb{E}_{(x,a,x')\sim\nu_h\pi P^\star}[\tilde{f}(x')] - \mathbb{E}_{x\sim\mu_{h+1}^\star}[\tilde{f}(x)]\right)\right| \\
&\leq 4\sqrt{\frac{2K\,|\mathcal{N}(\mathcal{X}, \epsilon/(3KL), \mathcal{D})|\log(2|\Pi|K/\epsilon(\delta))}{N}} + \frac{8K\,|\mathcal{N}(\mathcal{X}, \epsilon/(3KL), \mathcal{D})|\log(2|\Pi|K/(\epsilon\delta))}{N}.
\end{aligned}$$

Now using the fact that $\mathcal{N}(\mathcal{F}, \epsilon/K, \|\cdot\|_\infty)$ is an ϵ-cover under norm $\|\cdot\|_\infty$, we have that for any $\pi \in \Pi$, $f \in \mathcal{F}$, with probability at least $1 - \delta$,

$$\begin{aligned}
&\left|\left(\frac{1}{N}\sum_{i=1}^N K\pi(a_h^i|x_h^i)f(x_{h+1}^i) - \frac{1}{N}\sum_{i=1}^N f(\tilde{x}_{h+1}^i)\right) - \left(\mathbb{E}_{(x,a,x')\sim\nu_h\pi P^\star}[f(x')] - \mathbb{E}_{x\sim\mu_{h+1}^\star}[f(x)]\right)\right| \\
&\leq 4\sqrt{\frac{2K\,|\mathcal{N}(\mathcal{X}, \epsilon/(3KL), \mathcal{D})|\log(2|\Pi|3C/\epsilon(\delta))}{N}} + \frac{8K\,|\mathcal{N}(\mathcal{X}, \epsilon/(3KL), \mathcal{D})|\log(2|\Pi|3C/(\epsilon\delta))}{N} + 2\epsilon.
\end{aligned}$$

The rest of the proof is the same as the proof of Theorem 12.

9.2.7 Fail in Interactive Setting

Recall that with $\{\pi_1, \ldots, \pi_{h-1}\}$ being fixed, we denote ν_h as resulting observation distribution resulting at time step h. The interactiveness comes from the ability we can query expert to generate next observation conditioned on states sampled from ν_h—the states that would be visited by learner at time step h. Let us define $d(\pi|\nu_h, \pi_h^\star)$ as:

$$d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \pi_h^\star) \triangleq \max_{f\in\mathcal{F}_{h+1}} \left(\mathbb{E}_{x\sim\nu_h}\mathbb{E}_{a\sim\pi, x'\sim P_{x,a}}[f(x')] - \mathbb{E}_{x\sim\nu_h}\mathbb{E}_{a\sim\pi^\star, x'\sim P_{x,a}}[f(x')]\right).$$

Algorithm 10 iFail(Π, $\mathcal{F}$, ϵ, n, T)

```
 1: Set $\boldsymbol{\pi} = \emptyset$
 2: for $h = 1$ to $H - 1$ do
 3:     $\mathcal{D} = \emptyset, \tilde{\mathcal{D}} = \emptyset$
 4:     for $i = 1$ to $n$ do
 5:         Reset $x_1^{(i)} \sim \rho$ and from $x_i^{(1)}$ execute $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{h-1}\}$ to generate $x_h^{(i)}$
 6:         Execute $a \sim U(\mathcal{A})$ to generate $x_{h+1}^{(i)}$ and add it to $\mathcal{D}$
 7:         Reset again $\tilde{x}_1^{(i)} \sim \rho$ and from $\tilde{x}_1^{(i)}$ execute $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{h-1}\}$ to generate $\tilde{x}_h^{(i)}$
 8:         Ask expert to execute its policy at $\tilde{x}_h^{(i)}$ for one step, observe $\tilde{x}_{h+1}^{(i)}$, and add it to $\tilde{\mathcal{D}}$
 9:     end for
10:     Set $\pi_h$ to be the return of Algorithm 2 with inputs $\left(\tilde{\mathcal{D}}, \mathcal{D}, \Pi, \mathcal{F}, T\right)$
11:     Append $\pi_h$ to $\boldsymbol{\pi}$
12: end for
```

Note that different from $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \mu_{h+1}^\star)$, in $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \pi_h^\star)$, the marginal distributions on x are the same for both π and $\pi_h^\star$ and we directly access $\pi_h^\star$ to generate epxert observations at $h+1$ rather than thorough the expert observation distribution $\mu_{h+1}^\star$. In other words, we use IPM to compare the observation distribution at time step $h+1$ after applying π and the observation distribution at time step $h+1$ after applying $\pi^\star$, *conditioned on the distribution ν_h generated by the previously learned policies* $\{\pi_1, \ldots, \pi_{h-1}\}$. In Algorithm 10, at every time step h, to find a policy π_h that approximately minimizes $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \pi_h^\star)$, we replace expectations in $d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \pi_h^\star)$ by proper samples (line 6 and line 8), and then call Algorithm 2 (line 10).

Proof. Recall the definition of $d(\pi|\nu_h, \pi_h^\star)$,

$$d_{\mathcal{F}_{h+1}}(\pi|\nu_h, \pi_h^\star) = \max_{f \in \mathcal{F}_{h+1}} \left(\mathbb{E}_{x\sim\nu_h}\mathbb{E}_{a\sim\pi, x'\sim P_{x,a}}[f(x')] - \mathbb{E}_{x\sim\nu_h}\mathbb{E}_{a\sim\pi^\star, x'\sim P_{x,a}}[f(x')]\right),$$

with ν_h being the distribution over $\mathcal{X}_h$ resulting from executing policies $\{\pi_1, \ldots, \pi_{h-1}\}$.

We will use Lemma 10, Lemma 11, and Lemma 13 below.

The Performance Difference Lemma (Lemma 13) tells us that:

$$\begin{aligned}
J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) &= \sum_{h=1}^{H} \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}}\left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right] \\
&\le \sum_{h=1}^{H} \left|\mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}}\left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right] - \mathbb{E}_{a\sim\pi_h^\star, x'\sim P_{x,a}}\left[V_{h+1}^\star(x')\right]\right]\right| \\
&\le \sum_{h=1}^{H} d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \pi_h^\star), \qquad (9.48)
\end{aligned}$$

where the last inequality comes from the realizable assumption that $V_h^\star \in \mathcal{F}_h$.

At every time step h, mapping to Theorem 10 with $\nu_h = \mu_h^{\boldsymbol{\pi}}$, $T = 4K^2/\epsilon^2$, $n = K\log(|\Pi||\mathcal{F}|/\delta)/\epsilon^2$, we have that with probability at least $1-\delta$:

$$d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \pi_h^\star) \le \min_{\pi\in\Pi_h} d_{\mathcal{F}_{h+1}}(\pi|\mu_h^{\boldsymbol{\pi}}\pi_h^\star), +\epsilon.$$

Note that $\min_{\pi\in\Pi_h} d_{\mathcal{F}_{h+1}}(\pi|\mu_h^{\boldsymbol{\pi}}, \pi_h^\star) \le d_{\mathcal{F}_{h+1}}(\pi_h^\star|\mu_h^{\boldsymbol{\pi}}, \pi_h^\star) = 0$, since $\pi_h^\star \in \Pi_h$ by the realizable assumption. Hence, we have that:

$$d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \pi_h^\star) \le \epsilon.$$

Hence, using (9.48), and a union bound over all time steps $h \in [H]$, we have that with probability at least $1-\delta$,

$$J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \le H\epsilon,$$

with $T = 4K^2/\epsilon^2$, and $N = K\log(H|\Pi||\mathcal{F}|/\delta)/\epsilon^2$. Since in every round h, we need to draw N many trajectories, hence, the total number of trajectories we need is at most $HK\log(H|\Pi||\mathcal{F}|/\delta)/\epsilon^2$. □

9.2.8 Relaxation of Assumption 2

Our theoretical results presented so far rely on the realizable assumption (Assumption 2). While equipped with recent advances in powerful non-linear function approximators (e.g., deep neural networks), readability can be ensured, in this section, we relax the realizable assumption and show that our algorithms' performance only degenerates mildly. We relax Assumption 2 as follows:

Assumption 4 (Approximate Realizability)**.** *We assume Π and $\mathcal{F}$ is approximate realizable in a sense that for any $h \in [H]$, we have $\min_{\pi\in\Pi_h}\max_{x,a}\|\pi(a|x) - \pi_h^\star(a|x)\| \le \epsilon_\Pi$ and $\min_{f\in\mathcal{F}_h}\|f - V_h^\star\|_\infty \le \epsilon_{\mathcal{F}}$.*

The above assumption does not require $\mathcal{F}$ and Π to contain the exact $V_h^\star$ and $\pi_h^\star$, but assumes $\mathcal{F}$ and Π are rich enough to contain functions that can approximate $V_h^\star$ and $\pi_h^\star$ uniformly well (i.e., $\epsilon_{\mathcal{F}}$ and ϵ_Π are small). Without any further modification of Algorithm 3 and Algorithm 10 for non-interactive and interactive setting, we have the following corollary.

Corollary 3. *Under Assumption 4, for $\epsilon \in (0,1)$ and $\delta \in (0,1)$, with $T = \Theta(K/\epsilon^2)$, $n = \Theta(K\log(|\Pi||\mathcal{F}|H/\delta)/\epsilon^2)$, with probability at least $1-\delta$, (1) for non-interactive setting,* FAIL *(Algorithm 3) outputs a policy $\boldsymbol{\pi}$ with $J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \le O\left(H^2(\epsilon_{\mathrm{be}} + \epsilon) + H(\epsilon_{\mathcal{F}} + \epsilon_\Pi)\right)$, and (2) for interactive setting,* IFAIL *(Algorithm 10) outputs a policy $\boldsymbol{\pi}$ with $J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \le O\left(H\epsilon + H\epsilon_{\mathcal{F}} + H\epsilon_\Pi\right)$, by using at most $\tilde{O}((HK/\epsilon^2)\log(|\Pi||\mathcal{F}|/\delta))$ many trajectories under both settings.*

The proof is deferred to Appendix 9.2.9.

9.2.9 Proof of Corollary 3

Proof of Corollary 3. For any h, denote g_h as

$$g_h = \arg\min_{g\in\mathcal{F}} \|g - V_h^\star\|_\infty$$

Below we prove the first bullet in Corollary 3, i.e., the results for non-interactive setting.

Non-Interactive Setting Using PDL (Lemma 13), we have

$$\begin{aligned}
&J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) \\
&\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] \right| \\
&\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] - \mathbb{E}_{x\sim\mu_h^\star} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right| \\
&\quad + \left| \mathbb{E}_{x\sim\mu_h^\star} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right| \\
&\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] \right] - \mathbb{E}_{x\sim\mu_h^\star} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] \right| \\
&\quad + \left| \mathbb{E}_{x\sim\mu_h^\star} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] - \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] \right| + 4\epsilon_{\mathcal{F}} \\
&\leq \sum_{h=1}^{H} \left(d_{\mathcal{F}_{h+1}}(\pi_h | \mu_h^{\boldsymbol{\pi}}, \mu_h^\star) + \Delta_h + 4\epsilon_{\mathcal{F}} \right).
\end{aligned}$$

Now repeat the same recursive analysis for $d_{\mathcal{F}_{h+1}}(\pi_h|\mu_h^{\boldsymbol{\pi}}, \mu_h^\star)$ as we did in proof of Theorem 12 in Appendix 9.2.3, we can prove the first bullet in the corollary.

Now we prove the second bullet in Corollary 3, i.e., the results for interactive setting.

Interactive Setting Again, using Performance Difference Lemma (Lemma 13), we have

$$\begin{aligned}
J(\boldsymbol{\pi}) - J(\boldsymbol{\pi}^\star) &= \sum_{h=1}^{H} \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] \\
&\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] - \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[V^\star_{h+1}(x') \right] \right] \right| \\
&\leq \sum_{h=1}^{H} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] - \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[g_{h+1}(x') \right] \right] \right| \\
&+ \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} [V^\star_{h+1}(x') - g_{h+1}(x')] \right| + \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} [V^\star_{h+1}(x') - g_{h+1}(x')] \right| \\
&\leq \sum_{h=1}^{H} \max_{f\in\mathcal{F}_{h+1}} \left| \mathbb{E}_{x\sim\mu_h^{\boldsymbol{\pi}}} \left[\mathbb{E}_{a\sim\pi_h, x'\sim P_{x,a}} \left[f(x') \right] - \mathbb{E}_{a\sim\pi^\star_h, x'\sim P_{x,a}} \left[f(x') \right] \right] \right| + 2\epsilon_{\mathcal{F}} \\
&= \sum_{h=1}^{H} \left(d_{\mathcal{F}_{h+1}}(\pi_h | \mu_h^{\boldsymbol{\pi}}, \pi_h^\star) + 2\epsilon_{\mathcal{F}} \right)
\end{aligned}$$

Now repeat the same steps from the proof of Theorem 13 after (9.48) in proof of Theorem 13, we can prove the second bullet in the corollary.

□

9.3 Missing Proofs in Chapter 4

9.3.1 Proof of Theorem. 14

Proof. To prove Theorem. 14, we provide an example where the sequence of predictors $\{f_t\}$ is no-regret on loss $\{l_t(f_t)\}$ but Eq. 4.3 does not hold.

Let us assume there exist a $f^* \in \mathcal{F}$ such that $f^*(x_t) = v_t = \sum_{s=t}^{T} \gamma^{s-t} r_t$. Namely we assume that the best predictor in $\mathcal{F}$ can predict long-term reward exactly. Note that this f^* minimizes the PE and BE simultaneously as $f^* = \arg\min_{f\in\mathcal{F}} \sum (f(\mathbf{x}_t) - v_t)^2$ and $f^* = \arg\min_{f\in\mathcal{F}} \sum l_t(f)$.

Let us assume that $f_t(\mathbf{x}_t) = v_t + a$ and $f_t(\mathbf{x}_{t+1}) = v_{t+1} + \frac{1}{\gamma}a$, $\forall t$, for $a \in \mathbb{R}^+$. Then we have:

$$b_t = f_t(\mathbf{x}_t) - r_t - \gamma f_t(\mathbf{x_{t+1}}) = v_t + a - r_t - \gamma v_{t+1} - \gamma \frac{1}{\gamma} a = 0. \tag{9.49}$$

Hence, for regret, we have:

$$\text{Regret} = \sum l_t(f_t) - \min_{f\in\mathcal{F}} \sum l_t(f) = \sum l_t(f_t) - l_t(f^*) = \sum b_t^2 - b_t^{*2} = 0, \tag{9.50}$$

which means that this sequence of predictor $\{f_t\}$ is no-regret with respect to the loss functions $\{l_t(f)\}$.

However, on the other hand, when we check the predictor error e_t, we have $e_t = f_t(\mathbf{x}_t) - v_t = a$, which makes the sum of prediction error $\sum e_t^2$ increase linearly: $\sum e_t^2 = (T)a^2$ and $(1/T)\sum e_t^2 = a$. Since we have $e_t^* = 0$ for all t and thus $\sum e_t^{*2} = 0$, there is no such constant $C \in \mathbb{R}^+$ that could make the Eq. 4.3 hold. □ □

This example presents a sequence of predictors that does not satisfy the online stability condition. In fact, it is this example that motivates us to study stability condition of online algorihtms.

9.3.2 Proof of Lemma. 2

Proof. Note that $b_t^* = f^*(\mathbf{x}_t) - v_t + v_t - r_t - \gamma f^*(\mathbf{x}_{t+1}) = e_t^* - \gamma e_{t+1}^*$. Squaring both sides and summing over from $t = 0$ to $T-1$, we have:

$$\begin{aligned}
\sum b_t^{*2} &= \sum (e_t^* - \gamma e_{t+1}^*)^2 \\
&= \sum e_t^{*2} + \sum \gamma^2 e_{t+1}^{*2} - 2\gamma \sum e_t^* e_{t+1}^* \\
&\leq \sum e_t^{*2} + \sum \gamma^2 e_{t+1}^{*2} + \sum \gamma e_t^{*2} + \sum \gamma e_{t+1}^{*2} \\
&= (1+\gamma)^2 \sum e_t^{*2} + (\gamma^2 + \gamma)(e_T^{*2} - e_0^{*2}).
\end{aligned}$$

Again, the first inequality is obtained by applying Young's inequality to $-2e_t^* e_{t+1}^*$ to get $-2e_t^* e_{t+1}^* \leq e_t^{*2} + e_{t+1}^{*2}$. □ □

9.3.3 Proof of Lemma. 3

Proof. To show $l_t(f)$ is convex with respect to f, we only need the assumption that $\mathcal{F}$ belongs to a vector space. Since the function space $\mathcal{F}$ belongs to a vector space, for any two function $f \in \mathcal{F}$ and $g \in \mathcal{F}$, and a scalar $a \in R$ and $\mathbf{x}$, we have:

$$(f+g)(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \tag{9.51}$$

$$(af)(\mathbf{x}) = af(\mathbf{x}). \tag{9.52}$$

To prove the convexity of the loss functional $l_t(f)$, we show that for any $\alpha \in [0,1]$, we have $l_t(\alpha f + (1-\alpha)g) \leq \alpha l_t(f) + (1-\alpha)l_t(g)$. For $l_t(\alpha f + (1-\alpha)g)$, we have:

$$l_t(\alpha f + (1-\alpha)g) = ((\alpha f + (1-\alpha)g)(\mathbf{x}_t) - r_t - \gamma(\alpha f + (1-\alpha)g)(\mathbf{x}_{t+1}))^2 \tag{9.53}$$
$$= (\alpha(f(\mathbf{x}_t) - \gamma f(\mathbf{x}_{t+1}) - r_t) + (1-\alpha)(g(\mathbf{x}_t) - \gamma g(\mathbf{x}_{t+1}) - r_t))^2 \tag{9.54}$$
$$= \alpha^2(f(\mathbf{x}_t - \gamma f(\mathbf{x}_{t+1}) - r_t))^2 + (1-\alpha)^2(g(\mathbf{x}_t) - \gamma g(\mathbf{x}_{t+1}) - r_t)^2 \tag{9.55}$$
$$+ 2\alpha(1-\alpha)(f(\mathbf{x}_t) - \gamma f(\mathbf{x}_{t+1}) - r_t)(g(\mathbf{x}_t) - \gamma g(\mathbf{x}_{t+1}) - r_t). \tag{9.56}$$

For $\alpha l_t(f) + (1-\alpha)l_t(g)$, we have:

$$\alpha l_t(f) + (1-\alpha)l_t(g) = \alpha(f(\mathbf{x}_t) - \gamma f(\mathbf{x}_{t+1}) - r_t)^2 + (1-\alpha)(g(\mathbf{x}_t) - \gamma g(\mathbf{x}_{t+1}) - r_t)^2. \tag{9.57}$$

Define $b(f) = (f(\mathbf{x}_t) - \gamma f(\mathbf{x}_{t+1}) - r_t)$ and $b(g) = (g(\mathbf{x}_t) - \gamma g(\mathbf{x}_{t+1}) - r_t)$. Subtract Eq. 9.57 from Eq. 9.56, we have:

$$l_t(\alpha f + (1-\alpha)g) - (\alpha l_t(f) + (1-\alpha)l_t(g)) \tag{9.58}$$
$$= (\alpha^2 - \alpha)b(f)^2 + ((1-\alpha)^2 - (1-\alpha))b(g)^2 + 2(\alpha(1-\alpha))b(f)g(f) \tag{9.59}$$
$$= (\alpha^2 - \alpha)(b(f)^2 + b(g)^2 - 2b(f)g(f)) = (\alpha^2 - \alpha)(b(f) - g(f))^2 \leq 0. \tag{9.60}$$

Now we prove $l_t(f)$ is Lipschitz continuous. First, consider the case when $\mathcal{F}$ is in RKHS. $l_t(f)$ is differentiable and its gradient is:

$$\nabla l_t(f) = (f(\mathbf{x}_t) - r_t - \gamma f(\mathbf{x}_{t+1}))(K(\mathbf{x}_t, \cdot) - \gamma K(\mathbf{x}_{t+1}, \cdot)). \tag{9.61}$$

Note that the norm of $\nabla l_t(f)$ is:

$$\|\nabla l_t(f)\| = (f(\mathbf{x}_t) - r_t - \gamma f(\mathbf{x}_{t+1}))^2(1 + \gamma^2 - 2\gamma K(\mathbf{x}_t, \mathbf{x}_{t+1})). \tag{9.62}$$

Under the assumption that $|f(\mathbf{x})| \leq P$, $|r| \leq R$, $|K(\mathbf{x}_t, \mathbf{x}_{t+1})| \leq K$, it is easy to see that $\|\nabla l_t(f)\|$ is upper bounded by some postive constant. The fact that a function is differentialbe and has bounded gradient implies the function is Lipschitz continuous.

For the case when $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, we have $l_t(\mathbf{w})$ is differentiable and the gradient is:

$$\nabla l_t(\mathbf{w}) = (\mathbf{w}^T\mathbf{x}_t - r_t - \gamma \mathbf{w}^T\mathbf{x}_t)(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}). \tag{9.63}$$

Under the assumptions that $\|\mathbf{w}\|_2 \leq W$, $\|\mathbf{x}\|_2 \leq X$, $|r| \leq R$, it is easy to see that $\|\nabla l_t(\mathbf{w})\|_2$ is bounded. Hence, $l_t(\mathbf{w})$ is Lipschitz continuous with respect to L_2 norm.

To see that $l_t(\mathbf{w})$ is also Lipschitz continuous with respect to L_1 norm, note that $\|\nabla l_t(\mathbf{w})\|_\infty$ must be upper bounded, since $|f(\mathbf{x})| \leq P$, $|r| \leq R$, and $|\mathbf{x}^i| \leq X$, where $\mathbf{x}^i$ stands for the i'th entry of the vector $\mathbf{x}$. □ □

9.3.4 Proof of Lemma. 4

Proof. Without loss of generality, we assume the regularization $R(f)$ is 1-strongly convex with respect to norm $\|\cdot\|$. Due to strong convexity, we have:

$$\sum_{i=0}^{t} l_i(f_t) + \frac{1}{\mu}R(f_t) \geq \sum_{i=0}^{t} l_t(f_{t+1}) + \frac{1}{\mu}R(f_{t+1})$$

$$+\frac{1}{2\mu}\|f_t - f_{t+1}\|. \tag{9.64}$$

The inequality follows from the fact that $\sum l_t + \frac{1}{\mu}R$ is a strongly convex function and f_{t+1} is a minimizer of $\sum_{i=0}^{t} l_t + \frac{1}{\mu}R$. Similarly, We also have:

$$\sum_{i=0}^{t-1} l_i(f_{t+1}) + \frac{1}{\mu}R(f_{t+1}) \geq \sum_{i=0}^{t-1} l_i(f_t) + \frac{1}{\mu}R(f_t) + \frac{1}{2\mu}\|f_t - f_{t+1}\|, \tag{9.65}$$

because f_t is a minimizer of $\sum_{i=0}^{t-1} l_i + \frac{1}{\mu_{t-1}}R$. Adding (9.64) and (9.65) together side by side and cancelling out repeated terms from both sides, we get:

$$\begin{aligned}(1/\mu)\|f_t - f_{t+1}\|^2 &\leq l_t(f_t) - l_t(f_{t+1}) \\ &\leq |l_t(f_t) - l_t(f_{t+1})| \leq L\|f_t - f_{t+1}\|\end{aligned} \tag{9.66}$$

Setting $z = \|f_t - f_{t+1}\|$, and solve the above quadratic inequality with respect to z, we get $\|f_t - f_{t+1}\| \leq L\mu$. Sum from $t = 0$ to T, set $\mu = 1/\sqrt{T}$ and take the limit $T \to \infty$, we get to Eq. 4.17 □ □

9.3.5 Proof of Lemma. 5

Proof. $l_t(\mathbf{w})$ is a quadratic function with respect $\mathbf{w}$. Hence, taking the Tayplor expension of $l_t(\mathbf{w})$ at $\mathbf{w}'$, we have:

$$l_t(\mathbf{w}) = l_t(\mathbf{w}') + \nabla l_t(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}') + \frac{1}{2}(\mathbf{w} - \mathbf{w}')^T \nabla\nabla l_t(\mathbf{w}')(\mathbf{w} - \mathbf{w}'). \tag{9.67}$$

Note that the Hessian $\nabla\nabla l_t(\mathbf{w}') = 2(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})^T$, which can be writted as:

$$\begin{aligned}\nabla\nabla l_t(\mathbf{w}') &= 2(\mathbf{w}'^T\mathbf{x}_t - r_t - \gamma\mathbf{w}'^T\mathbf{x}_{t+1})^2 \frac{(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})^T}{(\mathbf{w}'^T\mathbf{x}_t - r_t - \gamma\mathbf{w}'^T\mathbf{x}_{t+1})^2} \\ &= \frac{1}{2(\mathbf{w}'^T\mathbf{x}_t - r_t - \gamma\mathbf{w}'^T\mathbf{x}_{t+1})^2}\nabla l_t(\mathbf{w}')\nabla l_t(\mathbf{w}')^T \geq \frac{1}{2M}\nabla l_t(\mathbf{w}')\nabla l_t(\mathbf{w}')^T,\end{aligned} \tag{9.68}$$

where $M = \sup_{\mathbf{w},\mathbf{x}_t,\mathbf{x}_{t+1},r_t}(\mathbf{w}^T\mathbf{x}_t - r_t - \gamma\mathbf{w}^T\mathbf{x}_{t+1})^2$. The derivation in Eq. 9.68 implicitly assumes that $(\mathbf{w}'^T\mathbf{x}_t - r_t - \gamma\mathbf{w}'^T\mathbf{x}_{t+1}) \neq 0$. But when $(\mathbf{w}'^T\mathbf{x}_t - r_t - \gamma\mathbf{w}'^T\mathbf{x}_{t+1}) = 0$, the final result from Eq. 9.68 still holds ($\nabla l_t(\mathbf{w}')\nabla l_t(\mathbf{w}')^T = 0$).

Note that M is a positive constant since we assume that $\|\mathbf{w}\|$, $\|\mathbf{x}\|$ and $|r_t|$ are all bounded. Hence, we have:

$$l_t(\mathbf{w}) \geq l_t(\mathbf{w}') + \nabla l_t(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}') + \frac{1}{4M}(\mathbf{w} - \mathbf{w}')^T\nabla l_t(\mathbf{w}')\nabla l_t(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}'). \tag{9.69}$$

Setting $\lambda \leq 1/2M$ we prove the lemma. □ □

9.3.6 Proof of Lemma. 6

Proof. We next show that online newton method satisfies the online stability condition. For convenience, define $\mathbf{y}_{t+1} = \mathbf{w}_t - \frac{1}{\lambda}A_t^{-1}\nabla l_t(\mathbf{w}_t)$, we have:

$$\sum \|\mathbf{w}_t - \mathbf{w}_{t+1}\|_{A_t}^2 \leq \sum \|\mathbf{w}_t - \mathbf{y}_{t+1}\|_{A_t}^2 = \sum \|\frac{1}{\lambda}A_t^{-1}\nabla l_t(\mathbf{w}_t)\|_{A_t}^2$$
$$= \sum \frac{1}{\lambda^2}\nabla l_t(\mathbf{w}_t)^T A_t^{-1} A_t A_t^{-1}\nabla l_t(\mathbf{w}_t) = \frac{1}{\lambda^2}\sum \nabla l_t(\mathbf{w}_t)^T A_t^{-1}\nabla l_t(\mathbf{w}_t).$$

Following the proof in Hazan et al. [2006], it can be shown that:

$$\sum \nabla l_t(\mathbf{w}_t)^T A_t^{-1}\nabla l_t(\mathbf{w}_t) \leq n\log(\frac{TG^2}{\epsilon}+1),$$

where $G \in \mathbb{R}^+$ and $G \geq \|\nabla l_t\|_2$. We simply set $\epsilon = G^2$. Hence, the online stability condition is satisfied as:

$$\frac{1}{T}\sum(\mathbf{w}_t^T\mathbf{x}_{t+1} - \mathbf{w}_{t+1}^T\mathbf{x}_{t+1})^2 \leq \frac{X^2}{T}\sum\|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2$$
$$\leq \frac{1}{T}\frac{X^2}{\epsilon}\sum\|\mathbf{w}_t - \mathbf{w}_{t+1}\|_{A_t}^2 \leq \frac{1}{T}\frac{X^2}{G^2\lambda^2}n\log(T+1) = 0, \quad T\to\infty.$$

The first inequality comes from Cauchy-Schwarz inequality and the assumption that $\|\mathbf{x}\|_2 \leq X$. The second inequality follows from the fact that the smallest eigenvalues of A_t's are bigger than or equal to ϵ. □ □

9.3.7 Disccusion and Results for Online Algorithms on TD-loss Functions $\{\tilde{l}_t(f)\}$

First of all, we show similar to our analysis for Bellman Residual minimization, simply being no-regret on the TD-loss functions $\{\tilde{l}_t(f)\}$ under general function approximation (not necessarily linear) is not sufficient to small predictive errors (Eq. 4.3 doesn't hold):

Theorem 27. *There exists a sequence of $\{f_t\}$ that is no-regret with respect to the TD-loss functions $\{\tilde{l}_t(f)\}$, but no $C \in \mathbb{R}^+$ exists that makes Eq. 4.3 hold.*

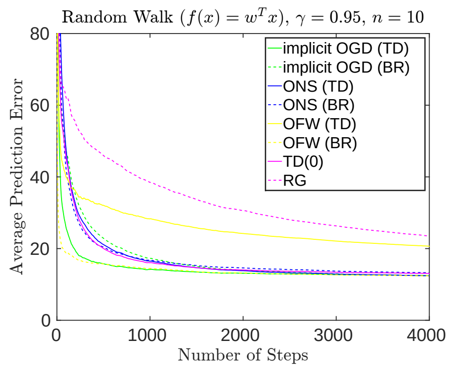
Proof. Again, we assume that $f_t(\mathbf{x}_t) = v_t + a$ and $f_t(\mathbf{x}_{t+1}) = v_{t+1} + \frac{1}{\gamma}a$, where $a \in \mathbb{R}^+$ and $v_t = \sum_{s=t}^{T}\gamma^{s-t}r_t$ is the long-term reward. Under this setting, the TD-loss $\tilde{l}_t(f_t)$ becomes:

$$\tilde{l}_t(f_t) = (f_t(\mathbf{x}_t) - r_t - \gamma f_t(\mathbf{x}_{t+1}))^2 = 0. \quad (9.70)$$

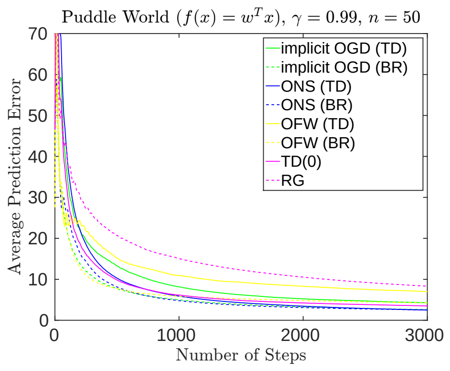
Hence, this sequence of predictors $\{f_t\}$ is no-regret:

$$\sum\tilde{l}_t(f_t) - \sum\tilde{l}_t(f^*) \leq \sum\tilde{l}_t(f_t) = 0, \quad \forall f^* \in \mathcal{F}. \quad (9.71)$$

However this sequence of predictors performly badly in terms of prediction error $e_t^2 = (f_t(\mathbf{x}_t) - v_t)^2 = a^2$. Under the assumption that the function space $\mathcal{F}$ (hypothesis class) is broad enough to have f^* that perfectly predicts long-term reward ($f^*(\mathbf{x}_t) = v_t, \forall t$), we always have $\frac{1}{T}\sum e_t^2 = a^2 > \frac{1}{T}e_t^{*2} = 0$. Hence, it is impossible to find a postive constant C such that Eq. 4.3 will hold. □ □



(a) Linear



(b) RKHS

Figure 9.3: Convergence of prediction error. We applied a set of online algorithms (OGD, implicit OGD, ONS, OFW) on BE loss functions $\{l_t(\mathbf{w})\}$ (dot line) and TD-loss functions $\{\tilde{l}_t(\mathbf{w})\}$ (solid line) for Random walk (left) and Puddle World (right).

Note that in the above proof, the constructed predictors are not stable in a sense that $f_t(\mathbf{x}_{t+1})$ and $f_{t+1}(\mathbf{x}_{t+1})$ varies a lot and hence it does not satisfies the online stability condition.

We conjecture that together with a similar stability analysis as we did for Bellman Residual minimization, we could achieve similar predictive guarantees as in Theorem. 15. We leave it as a open question here and we currently are working on it.

9.3.8 Empirical Results

We applied several stable no-regret online learning algorithms including ONS, OFW, implicit OGD to TD-loss functions $\tilde{l}_t(f)$ with linear function approximation ($f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$).

Fig. 9.3 shows the results of applying the set of algorithms (OGD, implicit OGD, ONS, and OFW) to BE and TD-loss for Random Walk and Puddle World. We compare their performance to $TD(0)$ and $RG(0)$. Although we currently do not have sound predictive guarantees, these empirical results suggest that applying stable no-regret online algorithms to TD-loss functions $\{\tilde{l}_t(\mathbf{w})\}$ in practice may give competitive performance compared to Bellman Residual minimization algorithms and the $TD(0)$.

9.4 Missing Proofs in Chapter 5

9.4.1 Proof of Theorem 16

Proof of Theorem 16. To prove Eq. 5.3 for Alg. 4, we use the proof techniques from Flaxman et al. [2005]. The proof is more simpler than the one in Flaxman et al. [2005] as we do not have to deal with shrinking and reshaping the predictor set Θ.

Denote $u \sim \mathbb{B}_b$ as uniformly sampling u from a b-dim unit ball, $u \sim \mathbb{S}_b$ as uniformly sampling u from the b-dim unit sphere, and $\delta \in (0,1)$. Consider the loss function $\hat{c}_i(w_i) = \mathbb{E}_{v\sim\mathbb{B}_b}[c_i(\theta_i + \delta v)]$, which is a smoothed version of $c_i(w_i)$. It is shown in Flaxman et al. [2005] that the gradient of $\hat{c}_i$ with respect to θ is:

$$\begin{aligned}
&\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i}\\
&= \frac{b}{\delta}\mathbb{E}_{u\sim\mathbb{S}_b}[c_i(\theta_i + \delta u)u]\\
&= \frac{b}{\delta}\mathbb{E}_{u\sim\mathbb{S}_b}[((\theta_i + \delta u)^T s_i - a_i)^2 u].
\end{aligned}$$

Hence, the descent direction we take in Alg. 4 is actually an unbiased estimate of $\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i}$. So Alg. 4 can be considered as running OGD with an unbiased estimate of gradient on the sequence of loss $\hat{c}_i(\theta_i)$. It is not hard to show that for an unbiased estimate of $\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i} = \frac{b}{\delta}((\theta_i + \delta u)^T s_i - a_i)^2 u$, the norm is bounded as $b(C^2 + C_s^2)/\delta$. Now we can directly applying Lemma 3.1 from Flaxman et al. [2005], to get:

$$\mathbb{E}\left[\sum_{i=1}^{T}\hat{c}_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^{T}\hat{c}_i(\theta^\star) \leq \frac{C_\theta b(C^2 + C_s^2)}{\delta}\sqrt{T}. \tag{9.72}$$

We can bound the difference between $\hat{c}_i(\theta)$ and $c_i(\theta)$ using the Lipschitiz continuous property of c_i:

$$\begin{aligned}
|\hat{c}_i(\theta) - c_i(\theta)| &= |\mathbb{E}_{v\sim\mathbb{B}_b}[c_i(\theta + \delta v) - c_i(\theta)]|\\
&\leq \mathbb{E}_{v\sim\mathbb{B}_b}[|c_i(\theta + \delta v) - c_i(\theta)|] \leq L\delta.
\end{aligned} \tag{9.73}$$

Substitute the above inequality back to Eq. 9.72, rearrange terms, we get:

$$\mathbb{E}\left[\sum_{i=1}^{T}c_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^{T}c_i(w^\star) \leq \frac{C_\theta b(C^2 + C_s^2)}{\delta}\sqrt{T} + 2LT\delta. \tag{9.74}$$

By setting $\delta = T^{-0.25}\sqrt{\frac{C_\theta b(C^2+C_s^2)}{2L}}$, we get:

$$\mathbb{E}\left[\sum_{i=1}^{T}c_i(\theta_i)\right] - \min_{w^\star\in\Theta}\sum_{i=1}^{T}c_i(w^\star) \leq \sqrt{C_\theta b(C^2 + C_s^2)L}T^{3/4}.$$

To prove Eq. 5.4 for Alg. 7, we follow the similar strategy in the proof of Alg. 4.

Denote $\epsilon \sim [-1,1]$ as uniformly sampling ϵ from the interval $[-1,1]$, $e \sim \{-1,1\}$ as uniformly sampling e from the set containing -1 and 1. Consider the loss function $\tilde{c}_i(\theta) = \mathbb{E}_{\epsilon\sim[-1,1]}[(\theta^T s_i + \delta\epsilon - a_i)^2]$. One can show that the gradient of $\tilde{c}_i(\theta)$ with respect to θ is:

$$\nabla_\theta \tilde{c}_i(\theta) = \frac{1}{\delta}\mathbb{E}_{e\sim\{-1,1\}}[e(\theta^\top s_i + \delta e - a_i)^2 s_i]. \tag{9.75}$$

As we can see that the descent direction we take in Alg. 7 is actually an unbiased estimate of $\nabla_\theta \tilde{c}_i(\theta)|_{\theta=\theta_i}$. Hence Alg. 7 can be considered as running OGD with unbiased estimates of gradients on the sequence of loss functions $\tilde{c}_i(\theta)$. For an unbiased estimate of the gradient, $\frac{1}{\delta}e(\theta_i^\top s_i + \delta e - a_i)^2 s_i$, its norm is bounded as $(C^2+1)C_s/\delta$. Note that different from Alg. 4, here the maximum norm of the unbiased gradient *is independent of feature dimension b*. Now we apply Lemma 3.1 from Flaxman et al. [2005] on $\tilde{c}_i$, to get:

$$\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^*\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \leq \frac{C_\theta(C^2+1)C_s}{\delta}\sqrt{T}. \tag{9.76}$$

Again we can bound the difference between $\tilde{c}_i(\theta)$ and $c_i(\theta)$ for any θ using the fact that $(\hat{a}_i - a_i)^2$ is Lipschitz continuous with respect to prediction $\hat{a}_i$ with Lipschitz constant C:

$$\begin{aligned}|\tilde{c}_i(\theta) - c_i(\theta)| &= |\mathbb{E}_{\epsilon\sim[-1,1]}[(\theta^\top s_i + \delta\epsilon - a_i)^2 - (\theta^\top s_i - a_i)^2]| \\ &\leq \mathbb{E}_{\epsilon\sim[-1,-1]}[C\delta|\epsilon|] \leq C\delta.\end{aligned} \tag{9.77}$$

Substitute the above inequality back to Eq. 9.76, rearrange terms:

$$\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^*\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \leq \frac{C_\theta(C^2+1)C_s}{\delta}\sqrt{T} + 2C\delta T.$$

Set $\delta = T^{-0.25}\sqrt{\frac{C_\theta(C^2+1)C_s}{2C}}$, we get:

$$\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^*\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \leq \sqrt{C_\theta(C^2+1)C_s C}T^{3/4}.$$

□

9.4.2 Proof of Theorem 17

We first present some useful lemmas below.

Consider the smoothed objective given by $\hat{J}(\theta) = \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta+\delta v)]$ where $\mathbb{B}_d$ is the unit ball in d dimensions and δ is a positive constant. Using the assumptions stated in Section 5.4.1, we obtain the following useful lemma:

Lemma 15. *If the objective $J(\theta)$ satisfies the assumptions in Section 5.4.1 and the smoothed objective $\hat{J}(\theta)$ is as given above, then we have that*

1. *$\hat{J}(\theta)$ is also G-Lipschitz and L-smooth*
2. *For all $\theta \in \mathbb{R}^d$, $\|\nabla_\theta J(\theta) - \nabla_\theta \hat{J}(\theta)\| \leq L\delta$*

Proof of Lemma 15. Consider for any $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\begin{aligned}|\hat{J}(\theta_1) - \hat{J}(\theta_2)| &= |\mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_1+\delta v) - J(\theta_2+\delta v)]| \\ &\leq \mathbb{E}_{v\sim\mathbb{B}_d}[|J(\theta_1+\delta v) - J(\theta_2+\delta v)|] \\ &\leq \mathbb{E}_{v\sim\mathbb{B}_d}[G\|\theta_1-\theta_2\|] \\ &= G\|\theta_1-\theta_2\|\end{aligned}$$

The above inequalities are due to the fact that expectation of absolute value is greater than absolute value of expectation, and the G-lipschitz assumption on $J(\theta)$. Thus, the smoothened loss function $\hat{J}(\theta)$ is also G-lipschitz. Similarly consider,

$$\begin{aligned}
&\|\nabla_\theta \hat{J}(\theta_1) - \nabla_\theta \hat{J}(\theta_2)\| \\
&\qquad = \|\nabla_\theta \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_1 + \delta v)] - \nabla_\theta \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_2 + \delta v)]\| \\
&\qquad = \|\mathbb{E}_{v\sim\mathbb{B}_d}[\nabla_\theta J(\theta_1 + \delta v) - \nabla_\theta J(\theta_2 + \delta v)]\| \\
&\qquad \leq \mathbb{E}_{v\sim\mathbb{B}_d}[\|\nabla_\theta J(\theta_1 + \delta v) - \nabla_\theta J(\theta_2 + \delta v)\|] \\
&\qquad \leq \mathbb{E}_{v\sim\mathbb{B}_d}[L\|\theta_1 - \theta_2\|] \\
&\qquad = L\|\theta_1 - \theta_2\|
\end{aligned}$$

The above inequalities are due to the fact that expectation of norm is greater than norm of expectation, and the L-smoothness assumption on $J(\theta_1)$. We interchange the expectation and derivative using the assumptions on $J(\theta_1)$ and the dominated convergence theorem. Thus, the smoothened loss function $\hat{J}(\theta_1)$ is also L-smooth.

We know,

$$\begin{aligned}
\nabla_\theta \hat{J}(\theta) &= \nabla_\theta \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta + \delta v)] \\
&= \mathbb{E}_{v\sim\mathbb{B}_d}[\nabla_\theta J(\theta + \delta v)]
\end{aligned}$$

Note that the expectation and derivative can be interchanged using the dominated convergence theorem. Hence, we have

$$\begin{aligned}
\|\nabla_\theta \hat{J}(\theta) - \nabla_\theta J(\theta)\| &= \|\mathbb{E}_{u\sim\mathbb{B}_d}[\nabla_\theta J(\theta + \delta v)] - \nabla_\theta J(\theta)\| \\
&\leq \mathbb{E}_{u\sim\mathbb{B}_d}\|\nabla_\theta J(\theta + \delta v) - \nabla_\theta J(\theta)\| \\
&\leq \mathbb{E}_{u\sim\mathbb{B}_d}[L||\delta v||] \\
&\leq L\delta
\end{aligned}$$

□

The above lemma will be very useful later when we try to relate the convergence rate for the smoothed objective and the true objective. It is shown in [Agarwal et al., 2010, Flaxman et al., 2005] that the gradient estimate g_i is an unbiased estimator of the gradient $\nabla_\theta \hat{J}(\theta_i)$. Hence, Algorithm 6 is performing SGD on the smoothed objective $\hat{J}(\theta)$. Using this insight, we can use the convergence rate of SGD for nonconvex functions to stationary points from [Ghadimi and Lan, 2013] which is given as follows

Lemma 16 ([Ghadimi and Lan, 2013]). *Consider running SGD on the objective $\hat{J}(\theta)$ that is L-smooth and G-Lipschitz for T steps. Fix initial solution θ_0 and denote $\Delta_0 = \hat{J}(\theta_0) - \hat{J}(\theta^*)$ where θ^* is the point at which $\hat{J}(\theta)$ attains global minimum. Also, assume that the gradient estimate g_i is unbiased and has a bounded variance, i.e. for all i, $\mathbb{E}_i[\|g_i - \nabla_\theta \hat{J}(\theta_i)\|_2^2] \leq V \in \mathbb{R}^+$ where $\mathbb{E}_i$ denotes expectation with randomness only at iteration i conditioned on history upto iteration $i-1$. Then we have,*

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V + G^2)}}{\sqrt{T}} \tag{9.78}$$

For completeness, we include a proof of the above lemma below.

Proof of Lemma 16. Denote $\xi_i = g_i - \nabla_\theta \hat{J}(\theta_i)$. Note that $\mathbb{E}_i[\xi_i] = 0$ since the stochastic gradient g_i is unbiased. From $\theta_{i+1} = \theta_i - \alpha g_i$, we have:

$$\begin{aligned}
\hat{J}(\theta_{i+1}) &= \hat{J}(\theta_i - \alpha g_i) \\
&\leq \hat{J}(\theta_i) - \nabla_\theta \hat{J}(\theta_i)^\top (\alpha g_i) + \frac{L\alpha^2}{2}\|g_i\|_2^2 \\
&= \hat{J}(\theta_i) - \alpha \nabla_\theta \hat{J}(\theta_i)^\top g_i + \frac{L\alpha^2}{2}\|\xi_i + \nabla_\theta \hat{J}(\theta_i)\|_2^2 \\
&= \hat{J}(\theta_i) - \alpha \nabla_\theta \hat{J}(\theta_i)^\top g_i + \frac{L\alpha^2}{2}(\|\xi_i\|_2^2 + 2\xi_i^\top \nabla_\theta \hat{J}(\theta_i) + \|\nabla_\theta \hat{J}(\theta_i)\|_2^2)
\end{aligned}$$

The first inequality above is obtained since the loss function $\hat{J}(\theta)$ is L-smooth. Adding $\mathbb{E}_i$ on both sides and using the fact that $\mathbb{E}_i[\xi_i] = 0$, we have:

$$\begin{aligned}
\mathbb{E}_i[\hat{J}(\theta_{i+1})] &= \hat{J}(\theta_i) - \alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}\left(\mathbb{E}_i[\|\xi_i\|_2^2] + \|\nabla_\theta \hat{J}(\theta_i)\|_2^2\right) \\
&\leq \hat{J}(\theta_i) - \alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}\left(\mathbb{E}_i[\|\xi_i\|_2^2] + G^2\right)
\end{aligned}$$

where the inequality is due to the lipschitz assumption. Rearranging terms, we get:

$$\begin{aligned}
\alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &= \hat{J}(\theta_i) - \mathbb{E}_i[\hat{J}(\theta_{i+1})] + \frac{L\alpha^2}{2}(\mathbb{E}_i[\|\xi_i\|_2^2] + G^2) \\
&\leq \hat{J}(\theta_i) - \mathbb{E}_i[\hat{J}(\theta_{i+1})] + \frac{L\alpha^2}{2}(V + G^2)
\end{aligned}$$

Sum over from time step 1 to T, we get:

$$\alpha \sum_{t=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta_T)] + \frac{LT\alpha^2}{2}(V + G^2)$$

Divide α on both sides, we get:

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &\leq \frac{1}{\alpha}\mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta_T)] + LT\alpha(V + G^2) \\
&\leq \frac{1}{\alpha}\mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta^*)] + LT\alpha(V + G^2) \\
&= \frac{1}{\alpha}\Delta_0 + LT\alpha(V + G^2) \\
&\leq \sqrt{\frac{\Delta_0 LT(V + G^2)}{2}} + \sqrt{2\Delta_0 LT(V + G^2)} \\
&\leq 2\sqrt{2\Delta_0 LT(V + G^2)}
\end{aligned}$$

with $\alpha = \sqrt{\frac{2\Delta_0}{LT(V+G^2)}}$. Hence, we have:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V + G^2)}}{\sqrt{T}}$$

$\square$

The above lemma is useful as it gives us the following result:

$$\min_{1\leq i\leq T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2$$
$$\leq \frac{2\sqrt{2\Delta_0 L(V+G^2)}}{\sqrt{T}} \tag{9.79}$$

since the minimum is always less than the average. We have then that using SGD to minimize a nonconvex objective finds a θ_i that is 'almost' a stationary point in bounded number of steps provided the stochastic gradient estimate has bounded variance.

We now show that the gradient estimate g_i used in Algorithm 6 indeed has a bounded variance. Observe that the estimate g_i in the algorithm is a two-point estimate, which should have substantially less variance than one-point estimates [Agarwal et al., 2010]. However, the two evaluations, resulting in J_i^+ and J_i^-, have different independent noise. This is due to the fact that in policy search, stochasticity arises from the environment and cannot be controlled and we cannot obtain the significant variance reduction that is typical of two-point estimators. The following lemma quantifies the bound on the variance of gradient estimate g_i:

Lemma 17. *Consider a smoothed objective $\hat{J}(\theta) = \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta+\delta v)]$ where $\mathbb{B}_d$ is the unit ball in d dimensions, $\delta > 0$ is a scalar and the true objective $J(\theta)$ is G-lipschitz. Given gradient estimate $g_i = \frac{d(J_i^+ - J_i^-)}{2\delta}u$ where u is sampled uniformly from a unit sphere $\mathbb{S}_d$ in d dimensions, $J_i^+ = J(\theta_i + \delta u) + \eta_i^+$ and $J_i^- = J(\theta - \delta u) + \eta_i^-$ for zero mean random i.i.d noises η_i^+, η_i^-, we have*

$$\mathbb{E}_i[\|g_i - \nabla_\theta \hat{J}(\theta_i)\|_2^2] \leq 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2} \tag{9.80}$$

where σ^2 is the variance of the random noise η.

Proof of Lemma 17. From Shamir [2017], we know that g_i is an unbiased estimate of the gradient of $\hat{J}(\theta_i)$, i.e. $\mathbb{E}_{u_i\sim\mathbb{S}_d}[g_i] = \nabla\hat{J}(\theta_i)$. Thus, we have

$$\begin{aligned}
&\mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i - \nabla\hat{J}(\theta_i)\|^2 \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}[\|g_i\|^2 + \|\nabla\hat{J}(\theta)_i\|^2 - 2g_i^T\nabla\hat{J}(\theta_i)] \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 + \|\nabla\hat{J}(\theta_i)\|^2 - 2\|\nabla\hat{J}(\theta_i)\|^2 \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 - \|\nabla\hat{J}(\theta_i)\|^2 \\
&\quad\leq \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 \\
&\quad= \frac{d^2}{4\delta^2}\mathbb{E}_{u_i\sim\mathbb{S}_d}\|(J(\theta_i+\delta u_i) - J(\theta_i - \delta u_i) + (\eta_i^+ - \eta_i^-))u_i\|^2 \\
&\quad\leq \frac{d^2}{2\delta^2}[\mathbb{E}_{u_i\sim\mathbb{S}_d}\|(J(\theta_i+\delta u_i) - J(\theta_i-\delta u_i)u_i\|_2^2 + \mathbb{E}_{u_i\sim\mathbb{S}_d}\|(\eta_i^+ - \eta_i^-))u_i\|^2] \\
&\quad\leq \frac{d^2}{2\delta^2}[\mathbb{E}_{u_i\sim\mathbb{S}_d}4G^2\delta^2\|u_i\|^2 + 4\mathbb{E}_{u_i\sim\mathbb{S}_d}\|\eta_i^+\|_2^2\|u_i\|_2^2] \\
&\quad= 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2}
\end{aligned}$$

where the second inequality is true as $\|a+b\|_2^2 \leq 2(\|a\|_2^2 + \|b\|_2^2)$ and the last inequality is due to the Lipschitz assumption on $J(\theta)$. □

We are ready to prove Theorem 17.

Proof of Theorem 17. Fix initial solution θ_0 and denote $\Delta_0 = \hat{J}(\theta_0) - \hat{J}(\theta^*)$ where $\hat{J}(\theta)$ is the smoothed objective and θ^* is the point at which $\hat{J}(\theta)$ attains global minimum. Since the gradient estimate g_i used in Algorithm 6 is an unbiased estimate of the gradient $\nabla_\theta \hat{J}(\theta_i)$, we know that Algorithm 6 performs SGD on the smoothed objective. Moreover, from Lemma 17, we know that the variance of the gradient estimate g_i is bounded. Hence, we can use Lemma 16 on the smoothed objective $\hat{J}(\theta)$ to get

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V+G^2)}}{\sqrt{T}} \tag{9.81}$$

where $V \leq 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2}$ (from Lemma 17). We can relate $\nabla_\theta \hat{J}(\theta)$ and $\nabla_\theta J(\theta)$ - the quantity that we ultimately care about, as follows:

$$\begin{aligned}
&\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \\
&= \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i) - \nabla_\theta \hat{J}(\theta_i) + \nabla_\theta \hat{J}(\theta_i)\|_2^2 \\
&\leq \frac{2}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i) - \nabla_\theta \hat{J}(\theta_i)\|_2^2 + \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2
\end{aligned}$$

We can use Lemma 15 to bound the first term and Equation 9.81 to bound the second term. Thus, we have

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \frac{2}{T}[TL^2\delta^2 + 2\sqrt{2\Delta_0 L(V+G^2)T}]$$

Substituting the bound for V from Lemma 17, using the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \in \mathbb{R}^+$, optimizing over δ, and using $\Delta_0 \leq \mathcal{Q}$ we get

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(\mathcal{Q}^{\frac{1}{2}} d T^{\frac{-1}{2}} + \mathcal{Q}^{\frac{1}{3}} d^{\frac{2}{3}} T^{\frac{-1}{3}}\sigma)$$

□

9.4.3 Proof of Theorem

The bound on the bias of the gradient estimate is given by the following lemma:

Lemma 18. *If the assumptions in Section 5.4.2 are satisfied, then for the gradient estimate g_i used in Algorithm 7 and the gradient of the objective $J(\theta)$ given in equation 5.6, we have*

$$\|\mathbb{E}[g_i] - \nabla_\theta J(\theta_i)\| \leq KUH\delta \tag{9.82}$$

Proof of Lemma 18. To prove that the bias is bounded, let's consider for any i

$$\|\mathbb{E}[g_i] - \nabla_\theta J(\theta_i)\|_2 = \| \sum_{t=0}^{H-1} \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}} [\nabla_\theta \pi(\theta_i, s_t) \nabla_a (\mathbb{E}_{v \sim \mathbb{B}_p} Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v) - Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t)))]\|_2$$

$$\leq \sum_{t=0}^{H-1} \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}, v \sim \mathbb{B}_p} \|\nabla_\theta \pi(\theta_i, s_t)\|_2 \|[\nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v) - \nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t))]\|_2$$

$$\leq \sum_{t=0}^{H-1} KU\delta \mathbb{E}_{v \sim \mathbb{B}_p} \|v\|_2$$

$$\leq KUH\delta$$

The first inequality above is obtained by using the fact that $\|\mathbb{E}[X]\|_2 \leq \mathbb{E}\|X\|_2$, and the second inequality using the K-lipschitz assumption on $\pi(\theta, s)$ and U-smooth assumption on $Q^t_{\pi_\theta}(s, a)$ in a. Also, observe that we interchanged the derivative and expectation above by using the assumptions on $Q^t_{\pi_\theta}$ as stated in Section 5.4.2. □

We will now show that the gradient estimate g_i used in Algorithm 7 has a bounded variance. Note that the gradient estimate constructed in Algorithm 7 is a one-point estimate, unlike policy search in parameter space where we had a two-point estimate. Thus, the variance would be higher and the bound on the variance of such a one-point estimate is given below

Lemma 19. *Given a gradient estimate g_i as shown in Algorithm 7, the variance of the estimate can be bounded as*

$$\mathbb{E}\|g_i - \mathbb{E}[g_i]\|_2^2 \leq \frac{2H^2p^2K^2}{\delta^2}((\mathcal{Q} + W\delta)^2 + \sigma^2) \tag{9.83}$$

where σ^2 is the variance of the random noise $\tilde{\eta}$.

Proof of Lemma 19. To bound the variance of the gradient estimate g_i in Algorithm 7, lets consider

$$\begin{aligned}
\mathbb{E}_i\|g_i - \mathbb{E}[g_i]\|_2^2 &= \mathbb{E}_i\|g_i\|_2^2 - \|\mathbb{E}_i[g_i]\|_2^2 \leq \mathbb{E}_i\|g_i\|_2^2 \\
&= \frac{H^2p^2}{\delta^2}\mathbb{E}_i\|\nabla_\theta \pi(\theta_i, s_t)(Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u) + \tilde{\eta}_i)u\|_2^2 \\
&\leq \frac{K^2p^2H^2}{\delta^2}\mathbb{E}_i\|Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u)u + \tilde{\eta}_i u\|_2^2
\end{aligned}$$

where $\mathbb{E}_i$ denotes expectation with respect to the randomness at iteration i and the inequality is obtained using K-lipschitz assumption on $\pi(\theta, s)$. Note that we can express $Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u) \leq Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t)) + W\delta\|u\|_2 \leq \mathcal{Q} + W\delta$ where we used the W-lipschitz assumption on $Q^t_{\pi_\theta}(s, a)$ in a and that it is bounded everywhere by constant $\mathcal{Q}$. Thus, we have

$$\begin{aligned}
&\mathbb{E}_i\|g_i - \mathbb{E}[g_i]\|_2^2 \\
&\leq \frac{K^2p^2H^2}{\delta^2}\mathbb{E}_i\|(\mathcal{Q} + W\delta)u + \tilde{\eta}_i u\|_2^2 \\
&\leq \frac{2K^2p^2H^2}{\delta^2}(\mathbb{E}_i\|(\mathcal{Q} + W\delta)u\|_2^2 + \mathbb{E}_i\|\tilde{\eta}_i u\|_2^2
\end{aligned}$$

$$\leq \frac{2K^2p^2H^2}{\delta^2}((\mathcal{Q}+W\delta)^2+\sigma^2)$$

□

We are now ready to prove theorem 18

Proof of Theorem 18. Fix initial solution θ_0 and denote $\Delta_0 = J(\theta_0) - J(\theta^*)$ where θ^* is the point at which $J(\theta)$ attains global minimum. Denote $\xi_i = g_i - \mathbb{E}_i[g_i]$ and $\beta_i = \mathbb{E}_i[g_i] - \nabla_\theta J(\theta_i)$. From Lemma 18, we know $\|\beta_i\| \leq KUH\delta$ and from lemma 19, we know $\mathbb{E}\|\xi_i\|_2^2 = V \leq \frac{2K^2p^2H^2}{\delta^2}((\mathcal{Q}+W\delta)^2+\sigma^2)$ and $\mathbb{E}_i[\xi_i]=0$ from definition. From $\theta_{i+1} = \theta_i - \alpha g_i$ we have:

$$\begin{aligned}
J(\theta_{i+1}) &= J(\theta_i - \alpha g_i)\\
&\leq J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i + \frac{L\alpha^2}{2}\|g_i\|_2^2\\
&= J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i + \frac{L\alpha^2}{2}\|\xi_i + \mathbb{E}_i[g_i]\|_2^2\\
&= J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i + \frac{L\alpha^2}{2}(\|\mathbb{E}_i[g_i]\|_2^2 + \|\xi_i\|_2^2 + 2\mathbb{E}_i[g_i]^T\xi_i)
\end{aligned}$$

Taking expectation on both sides with respect to randomness at iteration i, we have

$$\begin{aligned}
&\mathbb{E}_i[J(\theta_{i+1})] = J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T\mathbb{E}_i[g_i] + \frac{L\alpha^2}{2}(\|\mathbb{E}_i[g_i]\|_2^2 + \mathbb{E}_i\|\xi_i\|_2^2 + 2\mathbb{E}_i[g_i]^T\mathbb{E}_i[\xi_i])\\
&\leq J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T(\beta_i + \nabla_\theta J(\theta_i)) + \frac{L\alpha^2}{2}(\|\beta_i + \nabla_\theta J(\theta_i)\|_2^2 + V)\\
&= J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(\|\nabla_\theta J(\theta_i)\|_2^2 + V + \|\beta_i\|_2^2) + (L\alpha^2 - \alpha)\nabla_\theta J(\theta_i)^T\beta_i\\
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2 - \alpha)\nabla_\theta J(\theta_i)^T\beta_i\\
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2 + \alpha)\|\nabla_\theta J(\theta_i)\|\|\beta_i\|\\
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2 + \alpha)GKUH\delta
\end{aligned}$$

Rearranging terms and summing over timestep 1 to T, we get

$$\begin{aligned}
&\alpha\sum_{i=1}^T\|\nabla_\theta J(\theta_i)\|_2^2 \leq J(\theta_0) - \mathbb{E}_T[J(\theta_T)] + \frac{LT\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2+\alpha)GKUHT\delta\\
&\leq \Delta_0 + \frac{LT\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2+\alpha)GKUHT\delta\\
&\sum_{i=1}^T\|\nabla_\theta J(\theta_i)\|_2^2 \leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha+1)GKUHT\delta\\
&\leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G^2 + K^2H^2U^2\delta^2 + 2GKUH\delta) + GKUHT\delta + \frac{LT\alpha}{2}V\\
&\leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G + KHU\delta)^2 + GKUHT\delta + \frac{LT\alpha K^2p^2H^2}{\delta^2}((\mathcal{Q}+W\delta)^2+\sigma^2)
\end{aligned}$$

$$\leq \frac{\Delta_0}{\alpha} + LT\alpha(G^2 + K^2H^2U^2\delta^2) + GKUHT\delta + 2\frac{LT\alpha K^2p^2H^2}{\delta^2}(\mathcal{Q}^2 + W^2\delta^2 + \sigma^2)$$

Using $\Delta_0 \leq \mathcal{Q}$ and optimizing over α and δ, we get $\alpha = \mathcal{O}(\mathcal{Q}^{\frac{3}{4}}T^{-\frac{3}{4}}H^{-1}p^{-\frac{1}{2}}(\mathcal{Q}^2 + \sigma^2)^{-\frac{1}{4}})$ and $\delta = \mathcal{O}(T^{-\frac{1}{4}}p^{\frac{1}{2}}(\mathcal{Q}^2 + \sigma^2)^{\frac{1}{4}})$. This gives us

$$\frac{1}{T}\sum_{i=1}^{T}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(T^{-\frac{1}{4}}Hp^{\frac{1}{2}}(\mathcal{Q}^3 + \sigma^2\mathcal{Q})^{\frac{1}{4}}) \tag{9.84}$$

□

9.4.4 Implementation Details

Tuning Hyperparameters for ARS

We tune the hyperparameters for ARS [Mania et al., 2018] in both MNIST and linear regression experiments, by choosing a candidate set of values for each hyperparameter: stepsize, number of directions sampled, number of top directions chosen and the perturbation length along each direction. The candidate hyperparameter values are shown in Table 9.1.

Hyperparameter	**Candidate Values**
Stepsize	$0.001, 0.005, 0.01, 0.02, 0.03$
# Directions	$10, 50, 100, 200, 500$
# Top Directions	$5, 10, 50, 100, 200$
Perturbation	$0.001, 0.005, 0.01, 0.02, 0.03$

Table 9.1: Candidate hyperparameters used for tuning in ARS experiments

We use the hyperparameters shown in Table 9.2 chosen through this tuning for each of the experiments in this work. The hyperparameters are chosen by averaging the test squared loss across three random seeds (different from the 10 random seeds used in actual experiments) and chosing the setting that has the least mean test squared loss after 100000 samples.

Experiment	**Stepsize**	**# Dir.**	**# Top Dir.**	**Perturbation**
MNIST	0.02	50	20	0.03
LR $d = 10$	0.03	10	10	0.03
LR $d = 100$	0.03	10	10	0.02
LR $d = 1000$	0.03	200	200	0.03

Table 9.2: Hyperparameters chosen for ARS in each experiment. LR is short-hand for Linear Regression.

MNIST Experiments

The CNN architecture used is as shown in Figure 9.4[1]. The total number of parameters in this model is $d = 21840$. For supervised learning, we use a cross-entropy loss on the softmax

[1]This figure is generated by adapting the code from `https://github.com/gwding/draw_convnet`

Experiment	Learning Rate	Batch size
MNIST	0.001	512
LR $d = 10$	0.08	512
LR $d = 100$	0.03	512
LR $d = 1000$	0.01	512

Table 9.3: Learning rate and batch size used for REINFORCE experiments. We use an ADAM [Kingma and Ba, 2014] optimizer for these experiments.

Experiment	Learning Rate	Batch size
LR $d = 10$	2.0	512
LR $d = 100$	2.0	512

Table 9.4: Learning rate and batch size used for Natural REINFORCE experiments. Note that we decay the learning rate after each batch by $\sqrt{T}$ where T is the number of batches seen.

output with respect to the true label. To train this model, we use a batch size of 64 and a stochastic gradient descent (SGD) optimizer with learning rate of 0.01 and a momentum factor of 0.5. We evaluate the test accuracy of the model over all the 10000 images in the MNIST test dataset.

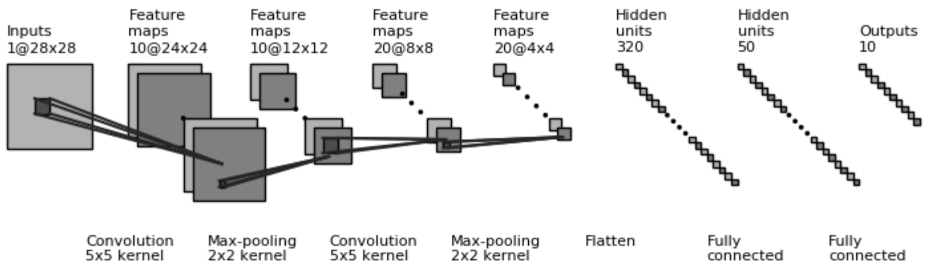


Figure 9.4: CNN architecture used for the MNIST experiments

For REINFORCE, we use the same architecture as before. We train the model by sampling from the categorical distribution parameterized by the softmax output of the model and then computing a ± 1 reward based on whether the model predicted the correct label. The loss function is the REINFORCE loss function given by,

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} r_i \log(\mathbb{P}(\hat{y}_i|x_i, \theta)) \tag{9.85}$$

where θ is the parameters of the model, r_i is the reward obtained for example i, $\hat{y}_i$ is the predicted label for example i and x_i is the input feature vector for example i. The reward r_i is given by $r_i = 2 * \mathbb{I}[\hat{y}_i = y_i] - 1$, where $\mathbb{I}$ is the $0-1$ indicator function and y_i is the true label for example i.

For ARS, we use the same architecture and reward function as before. The hyperparameters used are shown in Table 9.2 and we closely follow the algorithm outlined in [Mania et al., 2018].

Linear Regression Experiments

We generate training and test data for the linear regression experiments as follows: we sampled a random $d + 1$ dimensional vector w where d is the input dimensionality. We also sampled a random $d \times d$ covariance matrix C. The training and test dataset consists of $d + 1$ vectors x whose first element is always 1 (for the bias term) and the rest of the d terms are sampled from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix C. The target vectors y are computed as $y = w^T x + \epsilon$ where ϵ is sampled from a univariate normal distribution with mean 0 and standard deviation 0.001.

We implemented both SGD and Newton Descent on the mean squared loss, for the supervised learning experiments. For SGD, we used a learning rate of 0.1 for $d = 10, 100$ and a learning rate of 0.01 for $d = 1000$, and a batch size of 64. For Newton Descent, we also used a batch size of 64. To frame it as a one-step MDP, we define a reward function r which is equal to the negative of mean squared loss. Both REINFORCE and ARS use this reward function. To compute the REINFORCE loss, we take the prediction of the model $\hat{w}^T x$, add a mean 0 standard deviation $\beta = 0.5$ Gaussian noise to it, and compute the reward (negative mean squared loss) for the noise added prediction. The REINFORCE loss function is then given by

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} r_i \frac{-(y_i - \hat{w}^T x_i)^2}{2\beta^2} \tag{9.86}$$

where $r_i = -(y_i - \hat{y}_i)^2$, $\hat{y}_i$ is the noise added prediction and $\hat{w}^T x_i$ is the prediction by the model. We use an Adam optimizer with learning rate and batch size as shown in Table 9.3. For the natural REINFORCE experiments, we estimate the fisher information matrix and compute the descent direction by solving the linear system of equations $Fx = g$ where F is the fisher information matrix and g is the REINFORCE gradient. We use SGD with a $O(1/\sqrt{T})$ learning rate, where T is the number of batches seen, and batch size as shown in Table 9.4.

For ARS, we closely follow the algorithm outlined in [Mania et al., 2018].

9.4.5 Multi-step Control Experiments

Tuning Hyperparameters for ARS

We tune the hyperparameters for ARS [Mania et al., 2018] in both mujoco and LQR experiments, similar to the one-step control experiments. The candidate hyperparameter values are shown in Tables 9.5 and 9.6. We have observed that using all the directions in ARS is always preferable under the low horizon settings that we explore. Hence, we do not conduct a hyperparameter search over the number of top directions and instead keep it the same as the number of directions.

We use the hyperparameters shown in Tables 9.7 and 9.8 chosen through tuning for each of the multi-step experiments. The hyperparameters are chosen by averaging the total reward obtained across three random seeds (different from the 10 random seeds used in experiments presented in Figures 7.1e, 7.1g, 5.3c) and chosing the setting that has the highest total reward after 10000 episodes of training..

Hyperparameter	**Swimmer-v2**	**HalfCheetah-v2**
Stepsize	$0.03, 0.05, 0.08, 0.1, 0.15$	$0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	$5, 10, 20$	$5, 10, 20$
Perturbation	$0.05, 0.1, 0.15, 0.2$	$0.01, 0.03, 0.05, 0.08$

Table 9.5: Candidate hyperparameters used for tuning in ARS experiments

Hyperparameter	**LQR**
Stepsize	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	10
Perturbation	$0.01, 0.05, 0.1$

Table 9.6: Candidate hyperparameters used for tuning in ARS experiments

Tuning Hyperparameters for ExAct

We tune the hyperparameters for ExAct (Algorithm 7) in both mujoco and LQR experiments, similar to ARS. The candidate hyperparameter values are shown in Tables 9.9 and 9.10. Similar to ARS, we do not conduct a hyperparameter search over the number of top directions and instead keep it the same as the number of directions.

Hyperparameter	**Swimmer-v2**	**HalfCheetah-v2**
Stepsize	$0.005, 0.008, 0.01, 0.015, 0.02, 0.025, 0.03$	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.002, 0.003$
# Directions	$5, 10, 20$	$5, 10, 20$
Perturbation	$0.15, 0.2, 0.3, 0.5$	$0.15, 0.2, 0.3, 0.5$

Table 9.9: Candidate hyperparameters used for tuning in ExAct experiments

We use the hyperparameters shown in Tables 9.11 and 9.12 chosen through tuning for each of the multi-step experiments, similar to ARS.

Mujoco Experiments

For all the mujoco experiments, both ARS and ExAct use a linear policy with the same number of parameters as the dimensionality of the state space. The hyperparameters for both algorithms are chosen as described above. Each algorithm is run on both environments (Swimmer-v2 and HalfCheetah-v2) for 10000 episodes of training across 10 random seeds (different from the ones used for tuning). This is repeated for each horizon value $H \in \{1, 2, \cdots, 15\}$. In each experiment, we record the mean evaluation return obtained after training and plot the results in Figures 7.1e, 7.1g. For more details on the environments used, we refer the reader to [Brockman et al., 2016a].

LQR Experiments

In the LQR experiments, we constructed a linear dynamical system $x_{t+1} = Ax_t + Bu_t + \xi_t$ where $x_t \in \mathbb{R}^{100}$, $A \in \mathbb{R}^{100\times 100}$, $B \in \mathbb{R}^{100}$, $u_t \in \mathbb{R}$ and the noise $\xi_t \sim \mathcal{N}(0_{100}, cI_{100\times 100})$ with a small constant $c \in \mathbb{R}^+$. We explicitly make sure that the maximum eigenvalue of A is less than 1 to avoid instability. We fix a quadratic cost function $c(x, u) = x^TQx + uRu$, where

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.15	5	0.2
$H = 2$	0.08	5	0.2
$H = 3$	0.15	5	0.2
$H = 4$	0.08	5	0.2
$H = 5$	0.05	5	0.2
$H = 6$	0.08	5	0.2
$H = 7$	0.08	5	0.2
$H = 8$	0.08	5	0.2
$H = 9$	0.1	5	0.2
$H = 10$	0.08	5	0.2
$H = 11$	0.08	5	0.2
$H = 12$	0.1	5	0.2
$H = 13$	0.08	5	0.2
$H = 14$	0.08	5	0.2
$H = 15$	0.08	10	0.2

Table 9.7: Hyperparameters chosen for multi-step experiments for ARS in Swimmer-v2

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.001	20	0.08
$H = 2$	0.008	5	0.08
$H = 3$	0.008	10	0.08
$H = 4$	0.003	5	0.05
$H = 5$	0.003	5	0.05
$H = 6$	0.003	10	0.05
$H = 7$	0.008	20	0.05
$H = 8$	0.008	5	0.05
$H = 9$	0.01	20	0.03
$H = 10$	0.005	10	0.03
$H = 11$	0.008	20	0.03
$H = 12$	0.005	5	0.05
$H = 13$	0.008	20	0.03
$H = 14$	0.01	10	0.03
$H = 15$	0.008	20	0.03

Table 9.8: Hyperparameters chosen for multi-step experiments for ARS in HalfCheetah-v2

$Q = 10^{-3} I_{100\times 100}$ and $R = 1$. The hyperparameters chosen for both algorithms are chosen as described above.

For each algorithm, we run it for noise covariance values $c \in \{10^{-4}, 5\times 10^{-4}, 10^{-3}, 5\times 10^{-3}, 10^{-2}, 5\times 10^{-2}, 10^{-1}, 5\times 10^{-1}\}$ until we reach a stationary point where $\|\nabla_\theta J(\theta)\|_2^2 \leq 0.05$. The number of interactions with the environment allowed is capped at 10^6 steps for each run. This is repeated across 10 random seeds (different from the ones used for tuning). The number of interactions needed to reach the stationary point as the noise covariance is increased is recorded and shown in Figure 5.3c.

Hyperparameter	**LQR**
Stepsize	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	10
Perturbation	$0.01, 0.05, 0.1$

Table 9.10: Candidate hyperparameters used for tuning in ExAct experiments

Horizon	**Stepsize**	**# Directions**	**Perturbation**
$H = 1$	0.02	5	0.2
$H = 2$	0.02	5	0.2
$H = 3$	0.015	10	0.2
$H = 4$	0.015	10	0.2
$H = 5$	0.01	10	0.2
$H = 6$	0.015	10	0.2
$H = 7$	0.01	20	0.2
$H = 8$	0.015	20	0.2
$H = 9$	0.02	20	0.2
$H = 10$	0.008	5	0.2
$H = 11$	0.02	5	0.15
$H = 12$	0.02	20	0.2
$H = 13$	0.015	5	0.15
$H = 14$	0.02	10	0.15
$H = 15$	0.01	5	0.1

Table 9.11: Hyperparameters chosen for multi-step experiments for ExAct in Swimmer-v2

9.5 Missing Proofs in Chapter 6

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.0001	20	0.2
$H = 2$	0.001	5	0.2
$H = 3$	0.001	5	0.2
$H = 4$	0.001	5	0.2
$H = 5$	0.001	10	0.2
$H = 6$	0.001	5	0.2
$H = 7$	0.001	10	0.2
$H = 8$	0.001	5	0.2
$H = 9$	0.001	5	0.2
$H = 10$	0.001	5	0.2
$H = 11$	0.0008	5	0.15
$H = 12$	0.001	5	0.2
$H = 13$	0.001	10	0.2
$H = 14$	0.001	5	0.2
$H = 15$	0.0008	10	0.2

Table 9.12: Hyperparameters chosen for multi-step experiments for ExAct in HalfCheetah-v2

9.5.1 Proof of Theorem 20

We first present several lemmas that are useful for proving Theorem 20.

Fact 1. *For any two models M, M', the corresponding average Bellman error can be written as*

$$\begin{aligned}&\mathcal{E}_B(M, M', h) \triangleq \mathcal{E}_B(Q_M, Q_{M'}, h)\\ =&\mathbb{E}_{x_h \sim \pi_M, a_h \sim \pi_{M'}} \left[\mathbb{E}_{(r,x') \sim M'_h} \left[r + V_{M'}(x')\right] - \mathbb{E}_{(r,x') \sim M^\star_h} \left[r + V_{M'}(x')\right]\right]. \end{aligned} \tag{9.87}$$

Lemma 20 (Lemma 11 of Jiang et al. [2017])**.** *Consider a closed and bounded set $V \subset \mathbb{R}^d$ and a vector $p \in \mathbb{R}^d$. Let B be any origin-centered enclosing ellipsoid of V. Suppose there exists $v \in V$ such that $p^\top v \geq \kappa$ and define B_+ as the minimum volume enclosing ellipsoid of $\{v \in B : |p^\top v| \leq \frac{\kappa}{3\sqrt{d}}\}$. With vol$(\cdot)$ denoting the (Lebesgue) volume, we have:*

$$\frac{vol(B_+)}{vol(B)} \leq \frac{3}{5}.$$

Recall that V_M, π_M denote the optimal value function and policy derived from model M, and that v_M denotes π_M's value in M. For any policy π, v^π denotes the policy π's value in the true environment.

Lemma 21 (Simulation Lemma)**.** *Fix a model M. Under Assumption 3, we have*

$$v_M - v^{\pi_M} = \sum_{h=1}^{H} \mathcal{E}_B(M, M, h), \qquad \textit{and} \qquad v_M - v^{\pi_M} \leq \sum_{h=1}^{H} \mathcal{W}(M, M, h).$$

Proof. Start at time step $h = 1$,

$$\mathbb{E}_{x_1 \sim P_0}[V_M(x_1) - V^{\pi_M}(x_1)]$$

$$= \mathbb{E}_{x_1\sim P_0, a_1\sim\pi_M}\left[\mathbb{E}_{(r,x_2)\sim M_{x_1,a_1}}\left[r + V_M(x_2)\right] - \mathbb{E}_{(r,x_2)\sim M^\star_{x_1,a_1}}\left[r + V^{\pi_M}(x_2)\right]\right]$$
$$= \mathbb{E}_{x_1\sim P_0, a_1\sim\pi_M}\left[\mathbb{E}_{(r,x_2)\sim M_{x_1,a_1}}\left[r + V_M(x_2)\right] - \mathbb{E}_{(r,x_2)\sim M^\star_{x_1,a_1}}\left[r + V_M(x_2)\right]\right]$$
$$+ \mathbb{E}_{x_1\sim P_0, a_1\sim\pi_M}\left[\mathbb{E}_{(r,x_2)\sim M^\star_{x_1,a_1}}\left[V_M(x_2)\right] - \mathbb{E}_{(r,x_2)\sim M^\star_{x_1,a_1}}\left[V^{\pi_M}(x_2)\right]\right],$$

where the first equality is based on applying Bellman's equation to V_M in M and V^{π_M} in $M^\star$. Now, by Fact 1, the first term above is exactly $\mathcal{E}_B(M, M, 1)$. The second term can be expressed as,

$$\mathbb{E}\left[V_M(x_2) - V^{\pi_M}(x_2) | x_2 \sim \pi_M\right],$$

which we can further expand by applying the same argument recursively to obtain the identity involving the average Bellman errors. For the bound involving the witness model misfit, since $V_M \in \mathcal{F}$, we simply observe that $\mathcal{E}_B(M, M, h) \le \mathcal{W}(M, M, h)$. □

Next, we present several concentration results.

Lemma 22. *Fix a policy π, and fix $\epsilon, \delta \in (0,1)$. Sample $n_e = \frac{\log(2/\delta)}{(2\epsilon)^2}$ trajectories $\{(x_h^{(i)}, a_h^{(i)}, r_h^{(i)})_{h=1}^H\}_{i=1}^{n_e}$ by executing π and set $\hat{v}^\pi = \frac{1}{n_e}\sum_{i=1}^{n_e}\sum_{h=1}^H r_h^{(i)}$. With probability at least $1-\delta$, we have $|\hat{v}^\pi - v^\pi| \le \epsilon$.*

The proof is a direct application of Hoeffding's inequality on the random variables $\sum_{h=1}^H r_h^{(i)}$.

Recall the definitions of $\widehat{\mathcal{W}}$ and $\widehat{\mathcal{E}}_B$ from (6.5) and (6.6), and the shorthand notation $(r, x') \sim M_h$, which stands for $r \sim R_{x_h,a_h}$ and $x' \sim P_{x_h,a_h}$ (with $(R,P) = M$) whenever the identities of x_h and a_h are clear from context.

Lemma 23 (Deviation Bound for $\widehat{\mathcal{E}}_M$)**.** *Fix h and model $M \in \mathcal{M}$. Sample a dataset $\mathcal{D} = \left\{(x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)})\right\}_{i=1}^N$ with $x_h^{(i)} \sim \pi_M, a_h^{(i)} \sim U(\mathcal{A}), (r_h^{(i)}, x_{h+1}^{(i)}) \sim M_h^\star$ of size N. Then with probability at least $1-\delta$, we have for all $M' \in \mathcal{M}$:*

$$\left|\widehat{\mathcal{W}}(M, M', h) - \mathcal{W}(M, M', h)\right| \le \sqrt{\frac{2K\log(2|\mathcal{M}||\mathcal{F}|/\delta)}{N}} + \frac{2K\log(2|\mathcal{M}||\mathcal{F}|/\delta)}{3N}.$$

Proof. Fix $M' \in \mathcal{M}$ and $f \in \mathcal{F}$, define the random variable $z_i(M', f)$ as:

$$z_i(M', f) \triangleq K\pi_{M'}(a_h^{(i)}|x_h^{(i)})\left(\mathop{\mathbb{E}}_{(r,x')\sim M'_h} f(x_h^{(i)}, a_h^{(i)}, r, x') - f(x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)})\right).$$

The expectation of $z_i(M', f)$ is

$$\mathbb{E}[z_i(M', f)] = \underbrace{\mathop{\mathbb{E}}_{x_h\sim\pi_M, a_h\sim\pi_{M'}}[\mathop{\mathbb{E}}_{(r,x')\sim M'_h}[f(x_h, a_h, r, x')] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[f(x_h, a_h, r, x')]]}_{\triangleq d(M', M^\star, f)},$$

and it is easy to verify that $\mathrm{Var}(z_i(M', f)) \le 4K$. Hence, we can apply Bernstein's inequality, so that with probability at least $1-\delta$, we have

$$\left|\frac{1}{N}\sum_{i=1}^N z_i(M', f) - d(M', M^\star, f)\right| \le \sqrt{\frac{2K\log(2/\delta)}{N}} + \frac{2K\log(2/\delta)}{3N}.$$

Via a union bound over $\mathcal{M}$ and $\mathcal{F}$, we have that for all pairs $M' \in \mathcal{M}, f \in \mathcal{F}$, with probability at least $1-\delta$:

$$\left| \frac{1}{N} \sum_{i=1}^{N} z_i(M', f) - d(M', M^\star, f) \right| \leq \sqrt{\frac{2K \log(2|\mathcal{M}||\mathcal{F}|/\delta)}{N}} + \frac{2K \log(2|\mathcal{M}||\mathcal{F}|/\delta)}{3N}. \quad (9.88)$$

For fixed M', we have shown uniform convergence over $\mathcal{F}$, and this implies that the empirical and the population maxima must be similarly close, which yields the result. $\square$

Lemma 24 (Deviation Bound on $\widehat{\mathcal{E}}_B$)**.** *Fix model $M \in \mathcal{M}$. Sample a dataset $\mathcal{D} = \left\{ (x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)}) \right\}_{i=1}^{N}$ with $x_h^{(i)} \sim \pi_M, a_h^{(i)} \sim \pi_M, (r_h^{(i)}, x_{h+1}^{(i)}) \sim M_h^\star$ of size N. Then with probability at least $1-\delta$, for any $h \in [H]$, with probability at least $1-\delta$, we have:*

$$\left| \mathcal{E}_B(M, M, h) - \widehat{\mathcal{E}}_B(M, M, h) \right| \leq \sqrt{\frac{\log(2H/\delta)}{2N}}.$$

The result involves a standard application of Hoeffding's inequality with a union bound over $h \in [H]$, which can also be found in Jiang et al. [2017].

Lemma 25 (Terminate or Explore)**.** *Suppose that for any round t, $\hat{v}^{\pi^t}$ satisfies $\left| v^{\pi^t} - \hat{v}^{\pi^t} \right| \leq \epsilon/8$ and $M^\star$ is never eliminated. Then in any round t, one of the following two statements must hold:*

1. *The algorithm does not terminate and there exists a $h \in [H]$ such that $\mathcal{E}_B(M^t, M^t, h) \geq \frac{3\epsilon}{8H}$;*
2. *The algorithm terminates and outputs a policy π^t which satisfies $v^{\pi^t} \geq v^\star - \epsilon$.*

Proof. Let us first consider the situation where the algorithm does not terminate, i.e., $|\hat{v}^{\pi^t} - v_{M^t}| \geq \epsilon/2$. Via Lemma 21, we must have

$$\sum_{h=1}^{H} \mathcal{E}_B(M^t, M^t, h) \geq \left| v^{\pi^t} - v_{M^t} \right| = \left| v^{\pi^t} - \hat{v}^{\pi^t} + \hat{v}^{\pi^t} - v_{M^t} \right| \geq \left| \hat{v}^{\pi^t} - v_{M^t} \right| - \left| v^{\pi^t} - \hat{v}^{\pi^t} \right| \geq 3\epsilon/8.$$

By the pigeonhole principle, there must exist $h \in [H]$, such that

$$\mathcal{E}_B(M^t, M^t, h) \geq \frac{3\epsilon}{8H},$$

so we obtain the first claim. For the second claim, if the algorithm terminates at round t, we must have $|\hat{v}^{\pi^t} - v_{M^t}| \leq \epsilon/2$. Based on the assumption that $M^\star$ is never eliminated, and M^t is the optimistic model, we may deduce

$$v^{\pi^t} \geq \hat{v}^{\pi^t} - \frac{\epsilon}{8} \geq v_{M^t} - \frac{5\epsilon}{8} \geq v^\star - \frac{5\epsilon}{8} \geq v^\star - \epsilon. \quad (9.89)$$

$\square$

Recall the definition of the *witness rank* (Definition 5):

$$\mathrm{W}(\kappa, \beta, \mathcal{M}, \mathcal{F}, h) = \inf\{\mathrm{rank}(A) : \kappa \mathcal{E}_B(M, M', h) \leq A(M, M') \leq \mathcal{W}(M, M', h), \forall M, M' \in \mathcal{M}\}.$$

Let us denote $A^\star_{\kappa,h}$ as the matrix that achieves the witness rank $\mathrm{W}(\kappa, \beta, \mathcal{M}, \mathcal{F}, h)$ at time step h. Denote the factorization by $A^\star_{\kappa,h}(M, M') = \langle \zeta_h(M), \chi_h(M') \rangle$ with $\zeta_h, \chi_h \in \mathbb{R}^{\mathrm{W}(\kappa,\beta,\mathcal{M},\mathcal{F},h)}$. Finally, recall that $\beta \geq \max_{M,M',h} \|\zeta_h(M)\|_2 \|\chi_h(M')\|_2$.

Lemma 26. *Fix round t and assume that* $\left|\widehat{\mathcal{E}}_B(M^t, M^t, h) - \mathcal{E}_B(M^t, M^t, h)\right| \leq \frac{\epsilon}{8H}$ *for all* $h \in [H]$ *and* $\left|v^{\pi^t} - \hat{v}^{\pi^t}\right| \leq \epsilon/8$ *hold. If Algorithm 8 does not terminate, then we must have* $A^\star_{\kappa,h_t}(M^t, M^t) \geq \frac{\kappa\epsilon}{8H}$.

Proof. We first verify the existence of h_t in the selection rule line 6 in Algorithm 8. From Lemma 25, we know that there exists $h \in [H]$ such that $\mathcal{E}_B(M^t, M^t, h) \geq \frac{3\epsilon}{8H}$, and for this h, we have

$$\widehat{\mathcal{E}}_B(M^t, M^t, h) \geq \frac{3\epsilon}{8H} - \frac{\epsilon}{8H} = \frac{\epsilon}{4H}. \tag{9.90}$$

While this h may not be the one selected in line 6, it verifies that h_t exists, and further we do know that for h_t

$$\mathcal{E}_B(M^t, M^t, h_t) \geq \frac{2\epsilon}{8H} - \frac{\epsilon}{8H} = \frac{\epsilon}{8H},$$

Now the constraints defining $A^\star_{\kappa,h_t}$ give $A^\star_{\kappa,h_t}(M^t, M^t) \geq \kappa\mathcal{E}_B(M^t, M^t, h_t)$, which proves the lemma. □

Recall the model elimination criteria at round t: $\mathcal{M}_t = \{M \in \mathcal{M}_{t-1} : \widehat{\mathcal{W}}(M^t, M, h_t) \leq \phi\}$.

Lemma 27. *Suppose that* $\left|\widehat{\mathcal{W}}(M^t, M, h_t) - \mathcal{W}(M^t, M, h_t)\right| \leq \phi$ *holds for all* t, h_t*, and* $M \in \mathcal{M}$*. Then*

1. $M^\star \in \mathcal{M}_t$*, for all t.*
2. *Denote* $\widetilde{\mathcal{M}}_t = \{M \in \widetilde{\mathcal{M}}_{t-1} : A^\star_{\kappa,h_t}(M^t, M) \leq 2\phi\}$ *with* $\widetilde{\mathcal{M}}_0 = \mathcal{M}$*. We have* $\mathcal{M}_t \subseteq \widetilde{\mathcal{M}}_t$ *for all t.*

Observe $\widetilde{\mathcal{M}}_t$ is defined via the matrix $A^\star_{\kappa,h}$.

Proof. Recall that we have $\mathcal{W}(M^t, M^\star, h_t) = 0$. Assuming $M^\star \in \mathcal{M}_{t-1}$ and via the assumption in the statement, for every t, we have

$$\widehat{\mathcal{W}}(M^t, M^\star, h_t) \leq \mathcal{W}(M^t, M^\star, h_t) + \phi = \phi,$$

so $M^\star$ will not be eliminated at round t.

For the second result, we know that $\widetilde{\mathcal{M}}_0 = \mathcal{M}$. Assume inductively that, we have $\mathcal{M}_{t-1} \subset \widetilde{\mathcal{M}}_{t-1}$, and let us prove that $\mathcal{M}_t \subset \widetilde{\mathcal{M}}_t$. Towards a contradiction, let us assume that there exists $M \in \mathcal{M}_t$ such that $M \notin \widetilde{\mathcal{M}}_t$. Since $M \in \mathcal{M}_t \subset \mathcal{M}_{t-1} \subset \widetilde{\mathcal{M}}_{t-1}$, the update rule for $\widetilde{\mathcal{M}}_t$ implies that

$$A^\star_{\kappa,h_t}(M^t, M) > 2\phi.$$

But, using the deviation bound and the definition of $A_{\kappa^\star,h}$, we get

$$\widehat{\mathcal{W}}(M^t, M, h_t) \geq \mathcal{W}(M^t, M, h_t) - \phi \geq A^\star_{\kappa,h_t}(M^t, M) - \phi > \phi,$$

which contradicts the fact that $M \in \mathcal{M}_t$. Thus, by induction we obtain the result. □

With our choice of $\phi = \frac{\kappa\epsilon}{48H\sqrt{\mathrm{W}_\kappa}}$, we may now quantify the number of rounds of Algorithm 8 using $\widetilde{\mathcal{M}}_t$.

Lemma 28 (Iteration complexity)**.** *Suppose that*

$$\left|\widehat{\mathcal{W}}(M^t, M, h_t) - \mathcal{W}(M^t, M, h_t)\right| \leq \phi, \qquad \left|\widehat{\mathcal{E}}_B(M^t, M^t, h) - \mathcal{E}_B(M^t, M^t, h)\right| \leq \frac{\epsilon}{8H},$$

hold for all t, h_t, $h \in [H]$, and $M \in \mathcal{M}$, then the number of rounds of Algorithm 8 is at most $H\mathrm{W}_\kappa \log(\frac{\beta}{2\phi})/\log(5/3)$.

Proof. From Lemma 26, if the algorithm does not terminate at round t, we find M^t and h_t such that

$$A^\star_{\kappa,h_t}(M^t, M^t) = \left\langle \zeta_{h_t}(M^t), \chi_{h_t}(M^t)\right\rangle \geq \frac{\kappa\epsilon}{8H} = 6\sqrt{\mathrm{W}_\kappa}\phi,$$

which uses the value of $\phi = \frac{\kappa\epsilon}{48H\sqrt{\mathrm{W}_\kappa}}$.

Recall the recursive definition of $\widetilde{\mathcal{M}}_t = \{M \in \widetilde{\mathcal{M}}_{t-1} : A^\star_{\kappa,h_t}(M^t, M) \leq 2\phi\}$ from Lemma 27. For the analysis, we maintain and update H origin-centered ellipsoids where the h^{th} ellipsoid contains the set $\{\chi_h(M) : M \in \widetilde{\mathcal{M}}_t\}$. Denote O^h_t as the origin-centered minimum volume enclosing ellipsoid (MVEE) of $\{\chi_h(M) : M \in \widetilde{\mathcal{M}}_t\}$. At round t, for $\zeta_{h_t}(M^t)$, we just proved that there exists a vector $\chi_{h_t}(M^t) \in O^{h_t}_{t-1}$ such that $\langle\zeta_{h_t}(M^t), \chi_{h_t}(M^t)\rangle \geq 6\sqrt{\mathrm{W}_\kappa}\phi$. Denote $O^{h_t}_{t-1,+}$ as the origin-centered MVEE of $\{v \in O^{h_t}_{t-1} : \langle\zeta_{h_t}(M^t), v\rangle \leq 2\phi\}$. Based on Lemma 20, and the fact that $O^{h_t}_t \subset O^{h_t}_{t-1,+}$, by the definition of $\widetilde{\mathcal{M}}_t$, we have:

$$\frac{\mathrm{vol}(O^{h_t}_t)}{\mathrm{vol}(O^{h_t}_{t-1})} \leq \frac{\mathrm{vol}(O^{h_t}_{t-1,+})}{\mathrm{vol}(O^{h_t}_{t-1})} \leq 3/5,$$

which shows that if the algorithm does not terminate, then we shrink the volume of $O^{h_t}_t$ by a constant factor.

Denote $\Phi \triangleq \sup_{M\in\mathcal{M},h} \|\zeta_h(M)\|_2$ and $\Psi \triangleq \sup_{M\in\mathcal{M},h} \|\chi_h(M)\|_2$. For O^h_0, we have $\mathrm{vol}(O^h_0) \leq c_{\mathrm{W}_\kappa}\Psi^{\mathrm{W}_\kappa}$ where c_{W_κ} is the volume of the unit Euclidean ball in W_κ-dimensions. For any t, we have

$$O^h_t \supseteq \{q \in \mathbb{R}^{\mathrm{W}_\kappa} : \max_{p:\|p\|_2\leq\Phi} \langle q, p\rangle \leq 2\phi\} = \{q \in \mathbb{R}^{\mathrm{W}_\kappa} : \|q\|_2 \leq 2\phi/\Phi\}$$

Hence, we must have that at termination, $\mathrm{vol}(O^h_T) \geq c_{\mathrm{W}_\kappa}(2\phi/\Phi)^{\mathrm{W}_\kappa}$. Using the volume of O^h_0 and the lower bound of the volume of O^h_T and the fact that every round we shrink the volume of $O^{h_t}_t$ by a constant factor, we must have that for any $h \in [H]$, the number of rounds for which $h_t = h$ is at most:

$$\mathrm{W}_\kappa \log(\frac{\Phi\Psi}{2\phi})/\log(5/3). \tag{9.91}$$

Using the definition $\beta \geq \Phi\Psi$, this gives an iteration complexity of $H\mathrm{W}_\kappa \log\left(\frac{\beta}{2\phi}\right)/\log(5/3)$. □

We are now ready to prove Theorem 20. Note that we are using $A^\star_\kappa$, rather than relying on $\mathcal{E}_B$ or $\mathcal{W}$.

Proof of Theorem 20. Below we condition on three events: (1) $\left|\widehat{\mathcal{W}}(M^t, M, h_t) - \mathcal{W}(M^t, M, h_t)\right| \leq \phi$ for all t and $M \in \mathcal{M}$, (2) $\left|\widehat{\mathcal{E}}_B(M^t, M^t, h) - \mathcal{E}_B(M^t, M^t, h)\right| \leq \frac{\epsilon}{8H}$ for all t and $h \in [H]$, and (3) $\left|v^{\pi^t} - \hat{v}^{\pi^t}\right| \leq \epsilon/8$ for all t.

Under the first and second condition, from the lemma above, we know that the algorithm must terminate in at most $T = \mathrm{W}_\kappa H \log(\beta/(2\phi))/\log(5/3)$ rounds. Once the algorithm terminates, based on Lemma 25, we know that we must have found a policy that is ϵ-optimal.

Now, we show that with our choices for n, n_e, and ϕ, the above conditions hold with probability at least $1-\delta$. Based on value of $n_e = 32\frac{H^2\log(6HT/\delta)}{\epsilon^2}$, and Lemma 22, we can verify that the third condition $|v^{\pi^t} - \hat{v}^{\pi^t}| \leq \epsilon/8$ for all $t \in [T]$ with probability $1-\delta/3$, and the condition $\left|\hat{\mathcal{E}}_B(M^t, M^t, h) - \mathcal{E}_B(M^t, M^t, h)\right| \leq \epsilon/(8H)$ holds for all $t \in [T]$ and $h \in [H]$ with probability at least $1-\delta/3$. Based on the value of $n = 18432H^2K\mathrm{W}_\kappa \log(12T|\mathcal{M}||\mathcal{F}|/\delta)/(\kappa\epsilon)^2$, the value of ϕ, and the deviation bound from Lemma 23, we can verify that the condition $\left|\widehat{\mathcal{W}}(M^t, M, h_t) - \mathcal{W}(M^t, M, h_t)\right| \leq \phi$ holds for all $t \in [T]$, $M \in \mathcal{M}$ with probability at least $1-\delta/3$. Together these ensure the algorithm terminates in T iterations. The number trajectories is at most $(n_e + n)\cdot T$, and the result follows by substitute the value of n_e, n, and T. $\square$

9.5.2 Proof of Theorem 21

Algorithm 11 Extension to $\mathcal{F}$ with Unbounded Complexity. Arguments: $(\mathcal{M}, \mathcal{F}, \epsilon, \delta, \epsilon)$

```
1: Compute $\widetilde{\mathcal{F}}$ from $\mathcal{F}$ and $\mathcal{M}$ via (9.92)
2: Set $\phi = \kappa\epsilon/(48H\sqrt{\mathrm{W}_\kappa})$ and $T = H\mathrm{W}_\kappa\log(\beta/2\phi)/\log(5/3)$
3: Set $n_e = \Theta(H^2\log(6HT/\delta)/\epsilon^2)$ and $n = \Theta(H^2K\mathrm{W}_\kappa\log(12T|\mathcal{M}||\widetilde{\mathcal{F}}|/\delta)/(\kappa^2\epsilon^2))$
4: Run Algorithm 8 with inputs $(\mathcal{M}, \widetilde{\mathcal{F}}, n_e, n, \epsilon, \delta, \phi)$ and return the found policy.
```

We are interested in generalizing Theorem 20 to accommodate a broader class of test functions $\mathcal{F}$, for example $\{f : \|f\|_\infty \leq 1\}$ that induces the total-variation distance. This class is not a Glivenko-Cantelli class, so it does not enable uniform convergence, and we cannot simply use empirical mean estimator as in (6.5).

The key is to define a much smaller function class $\widetilde{\mathcal{F}} \subset \mathcal{F}$ that does enjoy uniform convergence, and at the same time is expressive enough such that the witnessed model misfit w.r.t. $\widetilde{\mathcal{F}}$ is the same as that w.r.t. $\mathcal{F}$. To define $\widetilde{\mathcal{F}}$, we need one new definition. For a model M and a policy π, we use $x_h \sim (\pi, M)$ to denote that x_h is sampled by executing π *in the model* M, instead of the true environment, for h steps. With this notation, define $f_{\pi,M_1,M_2,h}$ as:

$$\operatorname*{argmax}_{f\in\mathcal{F}} \mathbb{E}\left[\mathop{\mathbb{E}}_{(r,x_{h+1})\sim M_2}[f(x_h,a_h,r,x_{h+1})] - \mathop{\mathbb{E}}_{(r,x_{h+1})\sim M_1}[f(x_h,a_h,r,x_{h+1})] \mid x_h \sim (\pi, M_1), a_h \sim \pi_{M_2}\right].$$

Note that the maximum over $\mathcal{F}$ is always attained due to the boundedness assumption on $f \in \mathcal{F}$, and hence this definition is without loss of generality. Now we define

$$\widetilde{\mathcal{F}} \triangleq \left\{\pm f_{\pi_{M_3},M_1,M_2,h} : M_1, M_2, M_3 \in \mathcal{M}, h \in [H]\right\}. \tag{9.92}$$

This construction is based on the Scheffé estimator, which was originally developed for density estimation in total variation [Devroye and Lugosi, 2012]. As we have done here, the idea is to define a smaller function class containing just the potential maximizers. Importantly, this smaller function class is computed *independently* of the data, so there is no risk of overfitting. The main innovation here is that we extend the Scheffé estimator to conditional distributions, and also to handle arbitrary classes $\mathcal{F}$.

Lemma 29. *For any true model $M^\star \in \mathcal{M}$, policy π_M, $h \in [H]$, and target model M', we have*

$$\mathcal{W}(M, M', h; \mathcal{F}) = \mathcal{W}(M, M', h; \widetilde{\mathcal{F}}).$$

Moreover $|\widetilde{\mathcal{F}}| \le 2|\mathcal{M}|^3 H$.

Proof. The bound on $|\widetilde{\mathcal{F}}|$ is immediate. For the other claim, by the realizability assumption for $\mathcal{M}$, $\widetilde{\mathcal{F}}$ contains the functions $f_{\pi_M, M^\star, M', h}$ for each (M, M', h) pair. These are precisely the test functions that maximize the witness model misfit for $\mathcal{F}$, and so the IPM induced by $\widetilde{\mathcal{F}}$ achieves exactly the same values. $\square$

Replacing $\widehat{\mathcal{W}}(M, M', h)$ in (6.5), which uses $\mathcal{F}$, to instead use $\widetilde{\mathcal{F}}$, we obtain Algorithm 11 and Theorem 21 as a corollary to Theorem 20. The key is that we have eliminated the dependence on $|\mathcal{F}|$ in the bound.

9.5.3 Lower Bounds and the Separation Result

Proof of Proposition 2

To prove Proposition 2, we need the following lower bound for best-arm identification in stochastic multi-armed bandits.

Lemma 30 (Theorem 2 from Krishnamurthy et al. [2016])**.** *For $K \ge 2$, $\epsilon < \sqrt{1/8}$, and any best-arm identification algorithm, there exists a multi-armed bandit problem for which the best arm $i^\star$ is ϵ better than all others, but for which the estimate $\hat{i}$ of the best arm must have $\mathbb{P}[\hat{i} \neq i^\star] \ge 1/3$ unless the number of samples collected is at least $K/(72\epsilon^2)$.*

Proof of Proposition 2. Below we explicitly give the construction of $\mathcal{M}$. Every MDP in this family shares the same reward function, and actually also shares the same transition structure for all levels $h \in [H-1]$. The models only differ in their transition at the last time step.

Fix H and $K \ge 2$. Each MDP $M^{\mathbf{a}^\star} \in \mathcal{M}$ corresponds to an action sequence $\mathbf{a}^\star = \{a_1^\star, a_2^\star, \ldots, a_{H-1}^\star\}$ where $a_i^\star \in [K]$. Thus there are K^{H-1} models. The reward function, which is shared by all models, is

$$R(x) \triangleq \mathbf{1}\{x = x^\star\} \tag{9.93}$$

where $x^\star$ is a special state that only appears at level H. Let x' denote another special state at level H.

For any model $M^{\mathbf{a}^\star}$, at any level $h < H-1$, the state x_h is simply the history of actions $x_h \triangleq \{a_1, a_2, \ldots a_{h-1}\}$ applied so far, and taking $a \in \mathcal{A}$ at state x_h deterministically transitions to $x_h \circ a \triangleq \{a_1, a_2, \ldots, a_{h-1}, a\}$. The transition at level $h = H-1$ is defined as follows:

$$P^{\mathbf{a}^\star}(x_H | x_{H-1}, a_{H-1}) \triangleq \begin{cases} 0.5 + \epsilon \mathbf{1}\{x_{H-1} \circ a_{H-1} = \mathbf{a}^\star\}, & x_H = x^\star \\ 0.5 - \epsilon \mathbf{1}\{x_{H-1} \circ a_{H-1} = \mathbf{a}^\star\}, & x_H = x'. \end{cases} \tag{9.94}$$

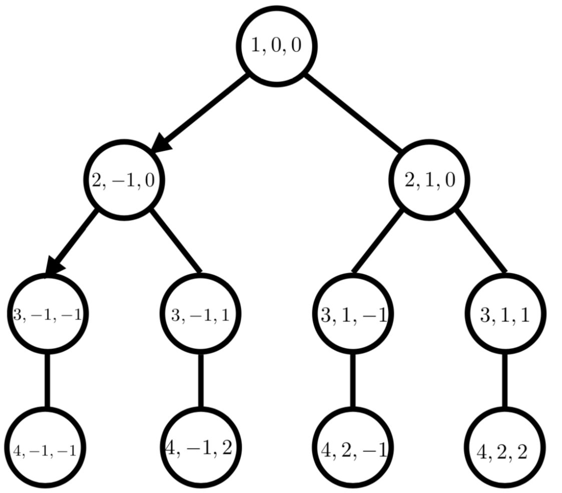


Figure 9.5: An example of the factored MDP construction in the proof of Theorem 19, with $d = 2$ and $H = 4$. All models are deterministic, and each model is uniquely indexed by a sequence of actions **p**. (Here $\mathbf{p} = \{-1, -1\}$, as indicated by the black arrows.) The first coordinate in each state encodes the level h. Each state at level $h \leq H - 1$ encodes the sequence of actions leading to it using bits from the second to the last (padded with 0's). The last transition is designed such that the agent always lands in a state that contains "2" unless it follows path **p**.

Thus, in each model $M^{\mathbf{a}^\star}$, each action sequence $\{a_1, a_2, \ldots, a_{H-1}\}$ can be regarded as an arm in MAB problem with K^{H-1} arms, where all the arms yield Ber(0.5) reward except for the optimal arm $\mathbf{a}^\star$ which yields Ber$(0.5+\epsilon)$ reward. In fact, this construction is information-theoretically equivalent to the construction used in the standard MAB lower bound, which appears in the proof of Lemma 30. That lower bound directly applies and since we have K^{H-1} arms here, the result follows. $\square$

Proof of Theorem 19

Theorem 19 has two claims: (1) There exists a family of MDPs in which Algorithm 12 achieves polynomial sample complexity, and (2) *Any* model-free algorithm will incur exponential sample complexity in this family. As we have discussed, the actual result is stronger in that the model class consists of factored MDPs under a particular structure, and our algorithm can handle any class of factored MDPs with an arbitrary (but known) structure.

The rest of this subsection is organized as follows: Appendix 9.5.3 describes the family of MDPs we construct. Since the MDPs obey a factored structure, we can learn this family using our Algorithm 12 and its guarantees in Theorem 22 immediately applies, which proves the second claim. Then, the first claim is proved in Appendix 9.5.3, where we leverage the definition of model-free algorithm (Definition 3) to induce information-theoretic hardness.

Model Class Construction and Sample Efficiency of Algorithm 12

Model Class Construction. We prove the claim by constructing a family of factored MDPs (recall (6.1)) that share the same reward function R but differ in their transition operators. The set of such transition operators is denoted as $\mathcal{P}$, and we use $P \in \mathcal{P}$ to refer to an MDP instance.

Fix $d > 2$ and set $H \triangleq d+2$. The state variables take values in $\mathcal{O} = \{-1, 0, 1, 2\}$. The state space is $\mathcal{X} = [H] \times \mathcal{O}^d$ with the natural partition across time steps and the action space is $\mathcal{A} = \{-1, +1\}$. The initial state is fixed as $x = 1 \circ [0]^d$, where $[a]^d$ stands for a d-dimensional vector where every coordinate is a and $\circ$ denotes concatenation. Our model class contains 2^d models, each of which is uniquely indexed by an action sequence (or a *path*) of length d, $\mathbf{p} = \{p_1, \dots, p_d\}$ with $p_i \in \{-1, 1\}$. Fixing $\mathbf{p}$, we describe the transition dynamics for $P^{\mathbf{p}}$ below. All models share the same reward function, which will be described afterwards.

In $P^{\mathbf{p}}$, the parent of the i^{th} factor is itself so that each factor evolves independently. Furthermore, all transitions are deterministic, so we abuse notation and let $P_h^{\mathbf{p},i}(\cdot,\cdot)$ denote the deterministic value of the i^{th} factor at time step $h+1$, as a function of its value at step h and action a. That is, if at time step h we are in state $(h, x_1, \dots, x_d)$, upon taking action a we will transition deterministically to $(h+1, P_h^{\mathbf{p},1}(x_1, a), \dots, P_h^{\mathbf{p},d}(x_d, a))$.

Levels 1 to $H-1$ form a complete binary tree; see Figure 9.5 for an illustration. For any layer $h \le H-2$,

$$P_h^{\mathbf{p},i}(v, a) = v, \quad \forall v \in \mathcal{O}, a \in \mathcal{A},\ i \neq h;$$
$$P_h^{\mathbf{p},i}(v, a) = a, \quad \forall v \in \mathcal{O}, a \in \mathcal{A},\ i = h.$$

In words, any internal state at level $h \le H-1$ simply encodes the sequence of actions that leads to it. These transitions *do not* depend on the planted path $\mathbf{p}$ and are identical across all models. Note that it is not possible to have $x_i = 2$ for any $i \in [d]$, $h \le H-1$.

Now we define the transition from level $H-1$ to H, where each state only has 1 action, say $+1$:

$$P_{H-1}^{\mathbf{p},i}(p_i, +1) = p_i, \quad \forall i \in [d], \qquad \text{and} \qquad P_{H-1}^{\mathbf{p},i}(\bar{p}_i, +1) = 2, \quad i \in [d].$$

Here $\bar{p}_i$ is the negation of p_i. In words, the state at level H simply copies the state at level $H-1$, except that the i^{th} factor will take value 2 if it disagrees with p_i (see Figure 9.5). Thus, the agent arrives at a state without the symbol "2" at level H only if it follows the action sequence $\mathbf{p}$.

The reward function is shared across all models. Non-zero rewards are only available at level H, where each state only has 1 action. The reward is 1 if x does not contain the symbol "2" and the reward is 0 otherwise. Formally

$$R((h, x_1, \dots, x_d)) \triangleq \mathbf{1}\{h = H\} \prod_{i=1}^{d} \mathbf{1}\{x_i \neq 2\}. \tag{9.95}$$

Sample Efficiency of Algorithm 12 For this family of factored MDPs, we have $K = 2$ and $d = H-2$. The remaining parameter of interest is L, on which we provide a coarse upper bound: $L \le dH|\mathcal{A}||\mathcal{O}|^2 = O(H^2)$ since $|\mathrm{pa}_i| = 1$ for all i and $|\mathcal{O}| = 4$. Given that our Algorithm 12 works for factored MDPs of any structure, the guarantees in Theorem 22 immediately applies and we obtain a sample complexity that is polynomial in H and $\log(1/\delta)$. This proves the first claim of Theorem 19.

Sample Inefficiency of Model-free Algorithms

We prove the second claim by showing that any model-free algorithm—that is, any algorithm that always accesses state x exclusively through $[Q(x,\cdot)]_{Q\in\mathcal{Q}}$—will incur exponential sample complexity when given $\mathcal{Q} = \texttt{OP}(\mathcal{P})$ as input. To show this, we construct another class of non-factored models, such that (1) learning in this new class is intractable, and (2) the two families are indistinguishable to any model-free algorithm. The new model class is obtained by transforming each $P^{\mathbf{p}} \in \mathcal{P}$ into $\tilde{P}^{\mathbf{p}}$. $\tilde{P}^{\mathbf{p}}$ has the same state space and transitions as $P^{\mathbf{p}}$, except for the transition from level $H-1$ to H. This last transition is:

$$\tilde{P}^{\mathbf{p}}_h((H-1,x_1,\ldots,x_d)) = \begin{cases}(H,x_1,\ldots,x_d) & \text{if } x_i = \mathbf{p}_i \ \forall i \in [d] \\ H \circ [2]^d & \text{otherwise.}\end{cases}$$

The reward function is the same as in the original model class, given in (9.95). This construction is equivalent to a multi-armed bandit problem with one optimal arm among 2^{H-2} arms, so the sample complexity of *any algorithm* (not necessarily restricted to model-free ones) is $\Omega(2^H)$.[2] In fact this model class is almost identical to the one used in the proof of Proposition 2.

To prove that the two model families are indistinguishable for model-free algorithms (Definition 3), we show that the Q-profiles in $P^{\mathbf{p}}$ are identical to those in $\tilde{P}^{\mathbf{p}}$. This implies that the behavior of a model-free algorithm is identical in $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$, so that the sample complexity must be identical, and hence $\Omega(2^H)$.

Let $\mathcal{M} = \{P^{\mathbf{p}}\}_{\mathbf{p}\in\{-1,1\}^d}$ and $\widetilde{\mathcal{M}} = \{\tilde{P}^{\mathbf{p}}\}_{\mathbf{p}\in\{-1,1\}^d}$. Let $\mathcal{Q} \triangleq \texttt{OP}(\mathcal{M})$ and $\widetilde{\mathcal{Q}} \triangleq \texttt{OP}(\widetilde{\mathcal{M}})$. Since all MDPs of interest have fully deterministic dynamics, and non-zero rewards only occur at the last step, it suffices to show that for any deterministic sequence of actions, $\mathbf{a}$, (1) the final reward has the same distribution for $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$, and (2) the Q-profiles $[Q(x_h,\cdot)]_{Q\in\mathcal{Q}}$ and $[Q(x_h,\cdot)]_{Q\in\tilde{\mathcal{Q}}}$ are equivalent at all states generated by taking $\mathbf{a}$ in $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$, respectively. The reward equivalence is obvious, so it remains to study the Q-profiles.

In $P^{\mathbf{p}}$ and at level H, since the reward function is shared, the Q-profile is $[1]^{|\mathcal{Q}|}$ for the state without "2" and $[0]^{|\mathcal{Q}|}$ otherwise. Thus, upon taking $\mathbf{a} = \mathbf{p}$ we see the Q-profile $[1]^{|\mathcal{Q}|}$ and otherwise we see $[0]^{|\mathcal{Q}|}$. Similarly, in $\tilde{P}^{\mathbf{p}}$ the Q-profile is $[0]^{|\tilde{\mathcal{Q}}|}$ if the state is $H \circ [2]^d$ and it is $[1]^{|\tilde{\mathcal{Q}}|}$ otherwise. The equivalence here is obvious as $|\mathcal{Q}| = |\widetilde{\mathcal{Q}}| = 2^d$.

For level $H-1$, no matter the true model path $\mathbf{p}$, the $Q^{\mathbf{p}'}$ associated with path $\mathbf{p}'$ has value $Q^{\mathbf{p}'}(\mathbf{a},+1) = \mathbf{1}\{\mathbf{a} = \mathbf{p}'\}$ at state $\mathbf{a}$. Hence the Q-profile at $\mathbf{a}$ can be represented as $[\mathbf{1}\{\mathbf{a} = \mathbf{p}'\}]_{\mathbf{p}'\in\{-1,1\}^d}$, for both $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$. Note that the Q-profile does not depend on the true model $\mathbf{p}$ because all models agree on the dynamics before the last step. Similarly, for $h < H-1$ where each state has two actions $\{-1,1\}$, we have:

$$Q^{\mathbf{p}'}(\mathbf{a}_{1:h-1},-1) = \mathbf{1}\left\{\mathbf{a}_{1:h-1} \circ \text{-}1 = \mathbf{p}'_{1:h}\right\}, Q^{\mathbf{p}'}(\mathbf{a}_{1:h-1},1) = \mathbf{1}\left\{\mathbf{a}_{1:h-1} \circ 1 = \mathbf{p}'_{1:h}\right\}.$$

Hence, the Q-profile can be represented as:

$$[(\mathbf{1}\left\{\mathbf{a}_{1:h-1} \circ \text{-}1 = \mathbf{p}'_{1:h}\right\}, \mathbf{1}\left\{\mathbf{a}_{1:h-1} \circ 1 = \mathbf{p}'_{1:h}\right\})]_{\mathbf{p}'\in\{-1,1\}^d},$$

again with no difference between $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$. Thus, the model $P^{\mathbf{p}}$ and $\tilde{P}^{\mathbf{p}}$ induce exactly the same Q-profile for all paths, implying that any model-free algorithm (in the sense

[2]Note that the reward function is known and non-random, so we do not have any dependence on an accuracy parameter ϵ.

Algorithm 12 Variant of Algorithm 8 for factored MDPs. Arguments: $(\mathcal{M}, n, n_e, \epsilon, \delta, \phi)$

```
1: Run Algorithm 8 with $\mathcal{F}$ in (9.96), except in line 8, estimate $\widehat{\mathcal{W}}_F(M^t, M', h_t)$ via (9.97).
```

of Definition 3), must behave identically on both. Since the family $\widetilde{\mathcal{M}} = \{\tilde{P}^{\mathbf{p}}\}_{\mathbf{p}}$ admits an information-theoretic sample complexity lower bound of $\Omega(2^H)$, this same lower bound applies to $\mathcal{M} = \{P^{\mathbf{p}}\}_{\mathbf{p}}$ for model-free algorithms.

9.5.4 Q-profiles in tabular settings

Here we show that the Q-profile yields no information loss in tabular environments. Thus from the perspective of Definition 3, model-based and model-free algorithms are information-theoretically equivalent.

In tabular settings, the state space $\mathcal{X}$ and action space $\mathcal{A}$ are both finite and discrete. It is also standard to use a fully expressive Q-function class, that is $\mathcal{Q} = \{Q : \mathcal{X} \times \mathcal{A} \to [0,1]\}$, where the range here arises due to the bounded reward. For each state $x \in \mathcal{X}$ define the function Q^x such that for all $a \in \mathcal{A}$, $Q^x(x', a) = \mathbf{1}\{x = x'\}$. Observe that since $\mathcal{Q}$ is fully expressive, we are ensured that $Q^x \in \mathcal{Q}, \forall x \in \mathcal{X}$.

At any state $x' \in \mathcal{X}$, from the Q-profile $\Phi_{\mathcal{Q}}(x')$ we can always extract the values $[Q^x(x', a)]_{x\in\mathcal{X}}$ for some fixed action a. By construction of the Q^x functions, exactly one of these values will be one, while all others will be zero, and thus we can recover the state x' simply by examining a few values in $\Phi_{\mathcal{Q}}(x')$. In other words, the mapping $x \mapsto \Phi_{\mathcal{Q}}(x)$ is invertible in the tabular case, and so there is no information lost through the projection. Hence in tabular setting, one can run classic model-free algorithms such as Q-learning [Watkins and Dayan, 1992] under our definition.

Our definition can also be applied to parameterized Q-function class $\mathcal{Q} \triangleq \{Q(\cdot,\cdot|\theta) : \theta \in \Theta \subset \mathbb{R}^d\}$. To perform gradient-based update on the parameter θ, we can use Q-profile as follows. Given any state-action pair (x, a), we can approximate $\nabla_{\theta_i} Q(x, a|\theta), \forall i \in [d]$, to an arbitrary accuracy, using finite differencing:

$$\nabla_{\theta_i} Q(x, a|\theta) = \lim_{\delta \to 0} \frac{Q(x, a|\theta + \delta e_i) - Q(x, a|\theta - \delta e_i)}{2\delta},$$

where e_i is the vector with zero everywhere except one in the i-th entry, and $Q(x, a|\theta + \delta e_i)$ and $Q(x, a|\theta - \delta e_i)$ can be extracted from the Q-profile $\Phi_{\mathcal{Q}}(x)$.

The Q-profile can also be used to estimate policy gradient on policies induced from the parameterized Q functions. Denote $\Pi_{\mathcal{Q}}$ as the policy class induced from $\mathcal{Q}$, e.g., $\pi(a|x;\theta) \propto \exp(Q(x, a|\theta))$. Policy gradient method often involves computing the gradient of the log likelihood of the policy (e.g., REINFORCE [Williams, 1992]): $\nabla_{\theta_i} \log(\pi(x|a;\theta)), \forall i \in [d]$, which via chain rule, is determined by $\nabla_\theta Q(x, a|\theta)$. Hence, with the finite differencing technique we introduced above for computing $\nabla_\theta Q(x, a|\theta)$, we can use Q-profile to compute $\nabla_\theta \log \pi(x|a;\theta)$.

9.5.5 Proof of Theorem 22

Here we prove Theorem 22, which states that Algorithm 12 can handle factored MDPs, where $\mathcal{M}$ is the infinite class of all possible factored MDPs under the given structure (i.e., $\{\mathrm{pa}_i\}$ are known). Since the only difference between two models is their transitions, we use

$\mathcal{P} = \{P : (R^\star, P) \in \mathcal{M}\}$ to represent the model class, and use P and M interchangeably sometimes.

As an input to the algorithm, we supply an $\mathcal{F}$ tailored for factored MDPs that always guarantees Bellman domination (up to a multiplicative constant; see Lemma 35). In particular,

$$\mathcal{F} = \{g_1 + \dots g_d : g_i \in \mathcal{G}_i\}, \tag{9.96}$$

where each $\mathcal{G}_i = (\mathcal{O}^{|\mathrm{pa}_i|} \times \mathcal{A} \times [H] \times \mathcal{O} \to \{-1, 1\})$. Note that functions in $\mathcal{F}$ operate on (x, a, r, x') and here we are using a slightly incorrect but intuitive notation: $g_i \in \mathcal{G}_i$ takes (x, a, r, x') as input, and only looks at $(x[\mathrm{pa}_i], h, a, x'[i])$ to determine a binary output value, and $\mathcal{G}_i$ is the set of all functions of this form. The IPM induced by $\mathcal{F}$ is the sum of total variation for each factor, and

$$|\mathcal{F}| = \prod_{i=1}^{d} 2^{HK|\mathcal{O}|^{1+|\mathrm{pa}_i|}} = 2^L,$$

so its logarithmic size is polynomial in L and allows uniform convergence. One slightly unusual property of $\mathcal{F}$, compared to how it is used in other results in the main text, is that functions in $\mathcal{F}$ has ℓ_∞ norm bounded by d instead of a constant, and this magnitude will be manifested in the sample complexity through concentration bounds.

Besides the specific choice of $\mathcal{F}$, we also need an important change in how we estimate the model misfit $\mathcal{W}_F$ defined in (6.7). Since $\mathcal{W}_F$ is defined w.r.t. uniformly random actions, we change our estimate accordingly by simply dropping the importance weight in line 8: Given dataset $\{(x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)})\}_{i=1}^n$ generated in line 7 of Algorithm 8 using roll-in policy π_M, the new estimator is

$$\widehat{\mathcal{W}}_F(M, M', h) \triangleq \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathop{\mathbb{E}}_{(r,x') \sim M'_h} [f(x_h^{(i)}, a_h^{(i)}, r, x')] - f(x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)}) \right). \tag{9.97}$$

We now state a formal version of Theorem 22, which includes the specification of input parameters to Algorithm 12, and prove it in the remainder of this section.

Theorem 28 (Formal version of Theorem 22). *Let $M^\star$ be a factored MDP with known structure (6.1). For any $\epsilon, \delta \in (0, 1]$, set $\beta = O(L/K)$, $\kappa = 1/K$, $\mathrm{W}_{\kappa,F} = L_h/|\mathcal{O}|$,[3] $\phi = \frac{\kappa\epsilon}{48H\sqrt{\mathrm{W}_{\kappa,F}}}$, and $T = H\mathrm{W}_{\kappa,F}\log(\beta/2\phi)/\log(5/3)$. Run Algorithm 12 with inputs $(\mathcal{M}, n_e, n, \epsilon, \delta, \phi)$, where $\mathcal{M}$ is the infinite class of all possible factored MDPs with the given structure, and*

$$n_e = \Theta\left(\frac{H^2 \log(HT/\delta)}{\epsilon^2}\right), n = \Theta\left(\frac{d^2(L\log(\frac{dKL}{\epsilon}) + \log(3T/\delta))}{|\mathcal{O}|\epsilon^2} LHK^2\right),$$

then with probability at least $1 - \delta$, the algorithm outputs a policy π such that $v^\pi \geq v^\star - \epsilon$, using at most the following number of sample trajectories:

$$\tilde{O}\left(\frac{d^2 L^3 HK^2 \log(1/\delta)}{\epsilon^2}\right).$$

[3]Here we treat $\mathrm{W}_{\kappa,F}$ as an algorithm parameter, and its value is an upper bound on the actual witness rank (4).

9.5.6 Concentration Result

Cover construction

We prove uniform convergence by discretizing the CPTs in facotred MDPs and constructing a cover of $\mathcal{P}$. Let $\alpha \in (0,1)$ be the discretization resolution, whose precise value will be set later. For convenience we also assume that $2/\alpha$ is an odd integer. Recall that a factored MDP is fully specified by the CPTs:

$$\{P^{(i)}[o \,|\, x[\mathrm{pa}_i], a, h] : o \in \mathcal{O}, x[\mathrm{pa}_i] \in \mathcal{O}^{|\mathrm{pa}_i|}, a \in \mathcal{A}, h \in [H]\}_{i=1}^{d}.$$

Since each of these probabilities takes value in $[0,1]$, we start with an improper cover of $\mathcal{P}$ by discretizing this range and considering cover centers $\{\alpha/2, 3\alpha/2, 5\alpha/2, \ldots, 1-\alpha/2\}$ for each $o \in \mathcal{O}, x[\mathrm{pa}_i] \in \mathcal{O}^{|\mathrm{pa}_i|}, a \in \mathcal{A}, h \in [H], i \in [d]$. Note that any number in $[0,1]$ will be $(\alpha/2)$-close to one of these $(1/\alpha)$ values. Altogether the discretization yields

$$\prod_{i=1}^{d} (1/\alpha)^{HK|\mathcal{O}|^{1+|\mathrm{pa}_i|}} = (1/\alpha)^{L}$$

(possibly unnormalized) CPTs. For the purpose of cover construction, the distance between two CPTs P and P' is defined as

$$\max_{o \in \mathcal{O}, x[\mathrm{pa}_i] \in \mathcal{O}^{|\mathrm{pa}_i|}, a \in \mathcal{A}, h \in [H]} |P^{(i)}[o \,|\, x[\mathrm{pa}_i], a, h] - P'^{(i)}[o \,|\, x[\mathrm{pa}_i], a, h]|. \tag{9.98}$$

Under this distance, any MDP in $\mathcal{P}$ will be $(\alpha/2)$-close to one of the discretized CPTs, hence we say the discretization yields a $(\alpha/2)$-cover of $\mathcal{P}$ with size $(1/\alpha)^L$.

Note that the above cover is improper because many cover centers violate the normalization constraints. We convert this improper cover to a proper one by (1) discarding all cover centers whose $\alpha/2$ radius ball contains no valid models, and (2) replacing every remaining invalid cover center with a valid model in its $\alpha/2$ radius ball. This yields an α-cover with size $(1/\alpha)^L$ whose cover centers are all valid models. We denote the set of cover centers as $\mathcal{P}_c$.

Uniform Convergence of $\widehat{\mathcal{W}}_F$

Recall the definition of $\widehat{\mathcal{W}}_F$ from (9.97). Our main concentration result is the following lemma.

Lemma 31 (Concentration of $\widehat{\mathcal{W}}_F$ in factored MDPs)**.** *Fix h and model $P \in \mathcal{P}$. Sample a dataset $\mathcal{D} = \left\{(x_h^{(i)}, a_h^{(i)}, r_h^{(i)}, x_{h+1}^{(i)})\right\}_{i=1}^{n}$ with $x_h^{(i)} \sim \pi_M$, $a_h^{(i)} \sim U(\mathcal{A})$, $(r_h^{(i)}, x_{h+1}^{(i)}) \sim M_h^\star$ of size n. Fix any ϕ and $\delta > 0$. With probability at least $1-\delta$, we have for all $P' \in \mathcal{P}$: $\left|\widehat{\mathcal{W}}_F(M, M', h) - \mathcal{W}_F(M, M', h)\right| \le \phi$, as long as*

$$n \ge \frac{8d^2(L\log(\frac{8d|\mathcal{O}|}{\phi}) + \log(2/\delta))}{\phi^2}.$$

We first prove a helper lemma, which quantifies the error introduced by approximating $\mathcal{P}$ with $\mathcal{P}_c$.

Lemma 32. *For any $P' \in \mathcal{P}$, let P'_c be its closest model in $\mathcal{P}_c$. For any $f \in \mathcal{F}$ and any x, a,*

$$|\mathop{\mathbb{E}}_{(r,x')\sim M'_{(x,a)}}[f(x,a,r,x')] - \mathop{\mathbb{E}}_{(r,x')\sim (M'_c)_{(x,a)}}[f(x,a,r,x')]| \leq d|\mathcal{O}|\alpha, \tag{9.99}$$

where $(r, x') \sim M'_{(x,a)}$ is the shorthand for $r \sim R'(x,a), x' \sim P'_{(x,a)}$.

Proof. Recall the definition of $f \in \mathcal{F}$ tailored for factored MDPs: $f = g_1 + \cdots + g_d$, with each $\|g_i\|_\infty \leq 1$. By triangle inequality, we have:

$$\begin{aligned}
\text{LHS} &\leq \sum_{i=1}^{d} \left| \mathop{\mathbb{E}}_{r,x'\sim M'_{(x,a)}}[g_i(x,a,r,x')] - \mathop{\mathbb{E}}_{r,x'\sim (M'_c)_{(x,a)}}[g_i(x,a,r,x')] \right| \\
&\leq \sum_{i=1}^{d} \left\| (P')^{(i)}_{x,a} - (P'_c)^{(i)}_{x,a} \right\|_{\text{TV}}, && \text{(Hölder)} \\
&= \sum_{i=1}^{d} \sum_{o\in\mathcal{O}} \left| (P')^{(i)}[o|x[\text{pa}_i], a, h] - (P'_c)^{(i)}[o|x[\text{pa}_i], a, h] \right| \\
&\leq d|\mathcal{O}|\alpha. && \text{($\mathcal{P}_c$ yields α-cover under distance defined in (9.98))}
\end{aligned}$$

□

Now we are ready to prove the main concentration result for factored MDPs.

Proof of Lemma 31. To argue uniform convergence for $\mathcal{P}$, we first apply Hoeffding's inequality and union bound to $\mathcal{P}_c$. For any fixed f, $\widehat{\mathcal{W}}_F$ is the average of i.i.d. random variables with range $[-\|f\|_\infty, \|f\|_\infty]$. For the $\mathcal{F}$ that we use for factored MDPs, $\|f\|_\infty \leq d$, so with probability at least $1-\delta$, $\forall P'_c \in \mathcal{P}_c$,

$$\left|\widehat{\mathcal{W}}_F(M, M'_c, h) - \mathcal{W}_F(M, M'_c, h)\right| \leq 2d\sqrt{\frac{\log(2|\mathcal{P}_c||\mathcal{F}|/\delta)}{2n}}.$$

We then follow a standard argument to decompose the estimation error for any $P' \in \mathcal{P}$ into three terms:

$$\begin{aligned}
\left|\widehat{\mathcal{W}}_F(M, M', h) - \mathcal{W}_F(M, M', h)\right| &\leq \left|\widehat{\mathcal{W}}_F(M, M'_c, h) - \mathcal{W}_F(M, M'_c, h)\right| \\
+ \left|\widehat{\mathcal{W}}_F(M, M', h) - \widehat{\mathcal{W}}_F(M, M'_c, h)\right| &+ \left|\mathcal{W}_F(M, M', h) - \mathcal{W}_F(M, M'_c, h)\right|.
\end{aligned}$$

We have an upper bound on the first term , so it suffices to upper-bound the other two terms. For the second term,

$$\begin{aligned}
&\left|\widehat{\mathcal{W}}_F(M, M', h) - \widehat{\mathcal{W}}_F(M, M'_c, h)\right| \\
&\leq \frac{1}{n}\max_{f\in\mathcal{F}} \left| \sum_{i=1}^{n} \mathop{\mathbb{E}}_{(r,x')\sim M'_h}[f(x_h^{(i)}, a_h^{(i)}, r, x')] - \sum_{i=1}^{n} \mathop{\mathbb{E}}_{(r,x')\sim (M'_c)_h}[f(x_h^{(i)}, a_h^{(i)}, r, x')] \right|.
\end{aligned}$$

where we use the fact that for any functionals μ_1, μ_2, we have $|\max_f \mu_1(f) - \max_f \mu_2(f)| \leq \max_f |\mu_1(f) - \mu_2(f)|$. Now using Lemma 32, we can show that:

$$\left|\widehat{\mathcal{W}}_F(M, M', h) - \widehat{\mathcal{W}}_F(M, M'_c, h)\right| \leq \frac{1}{n}(nd\,|\mathcal{O}|\,\alpha) = d\,|\mathcal{O}|\,\alpha.$$

$|\mathcal{W}_F(M, M', h) - \mathcal{W}_F(M, M'_c, h)|$ has the same upper bound using exactly the same argument. So finally we conclude that for all $P' \in \mathcal{P}$,

$$\left|\widehat{\mathcal{W}}_F(M, M', h) - \mathcal{W}_F(M, M', h)\right| \leq 2d\sqrt{\frac{\log\left(2|\mathcal{P}_c||\mathcal{F}|/\delta\right)}{2n}} + 2d|\mathcal{O}|\alpha.$$

To guarantee that the deviation is no more than ϕ, we back up the necessary sample size n from the above expression. Let $\alpha = \frac{\phi}{4d|\mathcal{O}|}$, so $2d|\mathcal{O}|\alpha \leq \phi/2$. We then want

$$2d\sqrt{\frac{\log\left(2(8d|\mathcal{O}|/\phi)^L/\delta\right)}{2n}} \leq \phi/2.$$

It is easy to verify that the sample size given in the lemma statement satisfies this inequality. □

9.5.7 Low Witness Rank and Bellman Domination

In this subsection we establish several important properties of $\mathcal{W}_F$ which will be directly useful in proving Theorem 22. To start, we provide a form of $\mathcal{W}_F$ that is equivalent to the definition provided in (6.7). The proof is elementary and omitted.

Lemma 33 (Alternative definition of $\mathcal{W}_F$).

$$\mathcal{W}_F(M, M', h) = \mathbb{E}\left[\sum_{i=1}^{d}\left\|P'^{(i)}(\cdot|x_h[\text{pa}_i], a_h) - P^{\star(i)}(\cdot|x_h[\text{pa}_i], a_h)\right\|_{TV} |x_h \sim \pi_P, a_h \sim U(\mathcal{A})\right]$$

Using this lemma, we show two important properties of $\mathcal{W}_F$: (1) that the matrix $\mathcal{W}_F$ has rank at most $\sum_{i=1}^{d} K\,|\mathcal{O}|^{|\text{pa}_{(i)}|}$ (Proposition 3), which is less than L, the description length of the factored MDP, and (2) that we can upper-bound $\mathcal{E}_B$ using $\mathcal{W}_F$ (Lemma 35). For the remainder, it will be convenient to use the notation $L_h = \sum_{i=1}^{d} K|\mathcal{O}|^{1+|\text{pa}_{(i)}|}$ to be the number of parameters needed to specify the conditional probability table at a single level h.

Proposition 3. *There exists $\zeta_h : \mathcal{P} \to \mathbb{R}^{L_h/|\mathcal{O}|}$ and $\chi_h : \mathcal{P} \to \mathbb{R}^{L_h/|\mathcal{O}|}$, such that for any $P, P' \in \mathcal{P}$, and $h \in [H]$, (recall that $M = (R, P)$ and $M' = (R, P')$)*

$$\mathcal{W}_F(M, M', h) = \langle \zeta_h(M), \chi_h(M') \rangle\,,$$

and $\|\zeta_h(M)\|_2 \cdot \|\chi_h(M')\|_2 \leq O(L_h/K)$.

Proof. Given any policy π, let us denote $\eta_h^\pi(x) \in \Delta(\mathcal{X}_h)$ as the state distribution resulting from π at time step h. Then we can write $\eta_h^\pi(x) = \eta_h^\pi(x[u])\eta_h^\pi(x[-u]|x[u])$, where for a subset $u \subset [d]$, we write $x[u]$ to denote the corresponding assignment of those state variables in x, and $-u = [d] \setminus u$ is the set of remaining variables. We use η_h^π to denote the probability mass function and we use $\mathbb{P}_h^\pi$ to denote the distribution.

For any $P, P' \in \mathcal{P}$, we can factorize $\mathcal{W}_F(M, M', h)$ as follows:

$$\mathcal{W}_F(M, M', h) = \mathbb{E}\left[\sum_{i=1}^{d}\left\|P'^{,(i)}(\cdot|x_h[\text{pa}_i], a_h) - P^{\star,(i)}(\cdot|x_h[\text{pa}_i], a_h)\right\|_{\text{TV}} |x_h \sim \pi_M, a_h \sim U(\mathcal{A})\right]$$

$$
\begin{aligned}
&= \frac{1}{K}\sum_{i=1}^{d}\sum_{x_h,a} \eta_h^{\pi_M}(x_h)\left\|P^{\star,(i)}(\cdot|x_h[\mathrm{pa}_i],a) - P^{\prime,(i)}(\cdot|x_h[\mathrm{pa}_i],a)\right\|_{\mathrm{TV}} \\
&= \frac{1}{K}\sum_{i=1}^{d}\sum_{x_h,a} \eta_h^{\pi_M}(x_h[\mathrm{pa}_i])\eta_h^{\pi_M}(x_h[-\mathrm{pa}_i]|x_h[\mathrm{pa}_i])\left\|P^{\star,(i)}(\cdot|x_h[\mathrm{pa}_i],a) - P^{\prime,(i)}(\cdot|x_h[\mathrm{pa}_i],a)\right\|_{\mathrm{TV}} \\
&= \frac{1}{K}\sum_{i=1}^{d}\sum_{a}\sum_{u\in\mathcal{O}^{|\mathrm{pa}_i|}} \mathbb{P}_h^{\pi_M}[x_h[\mathrm{pa}_i]=u]\left\|P^{\star,(i)}(\cdot|u,a) - P^{\prime,(i)}(\cdot|u,a)\right\|_{\mathrm{TV}} \\
&= \langle \zeta_h(M), \chi_h(M')\rangle\,.
\end{aligned}
$$

Here $\zeta_h(M)$ is indexed by $(i,a,u)\in[d]\times\mathcal{A}\times\mathcal{O}^{|\mathrm{pa}_i|}$ with value

$$\zeta_h(i,a,u;M) \triangleq \mathbb{P}_h^{\pi_M}(x_h[\mathrm{pa}_i]=u)/K.$$

$\chi_h(M')$ is also indexed by i,a,u, with value

$$\chi_h(i,a,u;M') \triangleq \left\|P^{\star,(i)}(\cdot|u,a) - P^{\prime,(i)}(\cdot|u,a)\right\|_{\mathrm{TV}}.$$

Note that ζ_h's value only depends on M, while χ_h's values only depend on M'. Moreover the dimensions of ζ_h and χ_h are $\sum_{i=1}^{d} K|\mathcal{O}|^{|\mathrm{pa}_i|} = L_h/|\mathcal{O}|$, each entry of ζ_h is bounded by $1/K$, and each entry of χ_h is at most 2. Hence, we must have $\beta = \sup_{\zeta,\chi}\|\zeta\|_2\cdot\|\chi\|_2 \le O(L/K)$. Note that we omit $|\mathcal{O}|$ from the denominator as L_h has a higher exponent on $|\mathcal{O}|$ and the quantity is being treated as a constant in the big-oh notation. □

We now proceed to prove Bellman domination (up to a constant), which relies on the following lemma on the tensorization property of total variation:

Lemma 34. *Let $P_1,\ldots,P_n$ and $Q_1,\ldots,Q_n$ be distributions where $P_i\in\Delta(\mathcal{S}_i)$ for finite sets $\mathcal{S}_i$. Define the product measures $P^{(n)},Q^{(n)}$ as $P^{(n)}(s_1,\ldots,s_n)\triangleq\prod_{i=1}^{n}P_i(s_i)$. Then*

$$\left\|P^{(n)}-Q^{(n)}\right\|_{TV} \le \sum_{i=1}^{n}\|P_i-Q_i\|_{TV}.$$

Proof. Define $W_i\in\Delta(\mathcal{S}_1\times\cdots\times\mathcal{S}_n)$ with $W_i(s_{1:n}) = \prod_{j=1}^{i}P_j(s_j)\prod_{j=i+1}^{n}Q_j(s_j)$, with $i\in\{0,\ldots,n\}$. This gives $W_0=Q^{(n)}$, and $W_n=P^{(n)}$. Now, by telescoping, we have

$$\left\|P^{(n)}-Q^{(n)}\right\|_{\mathrm{TV}} = \|W_0-W_n\|_{\mathrm{TV}} \le \sum_{i=0}^{n-1}\|W_i-W_{i+1}\|_{\mathrm{TV}}.$$

For $\|W_i-W_{i+1}\|_{\mathrm{TV}}$, we have

$$\|W_i-W_{i+1}\|_{\mathrm{TV}} = \left\|\prod_{j=1}^{i}P_j\prod_{j=i+1}^{n}Q_j - \prod_{j=1}^{i+1}P_j\prod_{j=i+2}^{n}Q_j\right\|_{\mathrm{TV}} = \|Q_{i+1}-P_{i+1}\|_{\mathrm{TV}}.$$

□

With this helper lemma, the following lemma shows the Bellman domination.

Lemma 35. $\frac{1}{K}\mathcal{E}_B(Q_M, Q_{M'}, h) \le \mathcal{W}_F(M, M', h)$.

Proof.

$$\begin{aligned}
\mathcal{E}_B(Q_M, Q_{M'}, h) &= \mathbb{E}\left[Q'(x_h, a_h) - r_h - Q'(x_{h+1}, a_{h+1}) \,\middle|\, x_h \sim \pi_M, a_{h:h+1} \sim \pi_{M'}\right] \\
&= \mathbb{E}\left[\mathop{\mathbb{E}}_{x_{h+1}\sim P'_{x_h,a_h}}[V_{M'}(x_{h+1})] - \mathop{\mathbb{E}}_{x_{h+1}\sim P^\star_{x_h,a_h}}[V_{M'}(x_{h+1})] \mid x_h \sim \pi_M, a_h \sim \pi_{M'}\right] \\
&\le \mathbb{E}\left[\sum_{a_h} \pi_{M'}(a_h|x_h)\left|\mathop{\mathbb{E}}_{x_{h+1}\sim P'_{x_h,a_h}}[V_{M'}(x_{h+1})] - \mathop{\mathbb{E}}_{x_{h+1}\sim P^\star_{x_h,a_h}}[V_{M'}(x_{h+1})]\right| \mid x_h \sim \pi_M\right] \\
&\le \mathbb{E}\left[\sum_{a_h} \pi_{M'}(a_h|x_h)\left\|P'_{x_h,a_h} - P^\star_{x_h,a_h}\right\|_{\text{TV}} \mid x_h \sim \pi_M\right] \quad \text{(Hölder and boundedness of } V_{M'}\text{)} \\
&\le K\,\mathbb{E}\left[\mathbb{E}_{a_h\sim U(\mathcal{A})}\left[\left\|P'_{x_h,a_h} - P^\star_{x_h,a_h}\right\|_{\text{TV}}\right] \mid x_h \sim \pi_M\right] \le K\mathcal{W}_F(M, M', h).
\end{aligned}$$

The first step follows as we expand the definition of $Q'(x_h, a_h)$ by Bellman equation in M', and realize that the immediate reward cancels out with r_h in expectation as the reward function is known. The last step follows from Lemma 34 and Lemma 33. □

Combining Proposition 3 and Lemma 35 we arrive at the following corollary:

Corollary 4. *Recall the definition of witness rank in Definition 5. Set $\kappa = \frac{1}{K}$, we have $\mathrm{W}_{\kappa,F} \triangleq \mathrm{W}(\frac{1}{K}, \beta, \mathcal{M}, \mathcal{F}, h) \le L_h/|\mathcal{O}|$,[4] for $\beta = O(L/K)$.*

9.5.8 Proof of Theorem 22

The proof is largely the same as that of Theorem 20, and the only difference is that we use $\widehat{\mathcal{W}}_F$ as the estimator and handle infinite $\mathcal{M}$, whose uniform convergence property is provided in Section 9.5.6. Following the proof of Theorem 20, within the high probability events,the algorithm must terminate in $T = \mathrm{W}_{\kappa,F} H \log(\beta/(2\phi))/\log(5/3)$ iterations, where $\kappa = 1/K$ and $\mathrm{W}_{\kappa,F} \le L_h/|\mathcal{O}|$ (Corollary 4) . As in the previous proof we still set $\phi = \frac{\kappa\epsilon}{48H\sqrt{\mathrm{W}_{\kappa,F}}}$. Plugging this value into Lemma 31 and requiring that each of these estimation events succeeds with probability at least $1 - \delta/3T$, we have

$$n = \Theta\left(\frac{d^2(L\log(\frac{dKL}{\epsilon}) + \log(3T/\delta))}{|\mathcal{O}|\epsilon^2} LHK^2\right) = \tilde{O}\left(\frac{d^2L^2HK^2\log(1/\delta)}{\epsilon^2}\right).$$

Here the $|\mathcal{O}|^2$ on the denominator is dropped due to its negligible magnitude compared to L. The rest of the proof is unchanged: Since estimating $\mathcal{W}_F$ requires much more samples than other estimation events, the order of the overall sample complexity is determined by the above expression multiplied by T, which gives the desired sample complexity.

9.5.9 Extension to Unknown Witness Rank

Algorithm 8 and its analysis assumes that we know κ and W_κ (in fact any finite upper bound of W_κ), which could be a strong assumption in some cases. In this section, we show that we can apply a standard doubling trick to handle the situation where κ and W_κ are unknown.

[4] Note that we abuse the definition of W to use $\mathcal{W}_F$ instead of $\mathcal{W}$ in Definition 5 here.

Algorithm 13 Guessing $\mathrm{W}_{\kappa^\star}/\kappa^\star$, Arguments: $(\mathcal{M}, \mathcal{F}, \epsilon, \delta)$

```
1: for epoch $i = 1, 2, ...$ do
2:   Set $N_i = 2^{i-1}$ and $\delta_i = \delta/(i(i+1))$
3:   for $j = 1, 2, \ldots$ do
4:     Set $\kappa_{i,j} = (1/2)^{j-1}$, $\delta_{i,j} = \delta_i/(j(j+1))$, and $\mathrm{W}_{i,j} = N_i \kappa_{i,j}$
5:     if $\mathrm{W}_{i,j} < 1$ then
6:       Break
7:     end if
8:     Set $T_{i,j} = H\mathrm{W}_{i,j} \log(\beta/(2\phi))/\log(5/3)$ and $\phi_{i,j} = \epsilon\kappa_{i,j}/(48H\sqrt{\mathrm{W}_{i,j}})$
9:     Set $n_{e_{i,j}} = \Theta\left(\frac{H^2 \log(6HT_{i,j}\delta_{i,j})}{\epsilon^2}\right)$ and $n_{i,j} = \Theta\left(\frac{H^2 K\mathrm{W}_{i,j} \log(12T_{i,j}|\mathcal{M}||\mathcal{F}|\delta_{i,j})}{\kappa_{i,j}^2\epsilon^2}\right)$
10:    Run Algorithm 8 with $(\mathcal{M}, \mathcal{F}, n_{i,j}, n_{e_{i,j}}, \epsilon, \delta_{i,j}, \phi)$ for $T_{i,j}$ iterations
11:    If Algorithm 8 returns a policy, then break and return the policy
12:  end for
13: end for
```

Let us consider the quantity W_κ/κ. Let us denote $\kappa^\star = \arg\min_{\kappa\in(0,1]} \mathrm{W}_\kappa/\kappa$. Note that the sample complexity of Algorithm 8 is minimized at $\kappa^\star$. Algorithm 13 applies the doubling trick to guess $\mathrm{W}_{\kappa^\star}$ and $\kappa^\star$ jointly with Algorithm 8 as a subroutine. In the algorithm, N_i in the outer loop denotes a guess for W/κ as a whole, and in the inner loop we use $\kappa_{i,j}$ to guess κ, while setting $\mathrm{W}_{i,j} = N_i\kappa_{i,j}$, which we use to set the parameter ϕ and n. The following theorem characterizes the its sample complexity.

Theorem 29. *For any $\epsilon, \delta \in (0, 1)$, with $\mathcal{M}$ and $\mathcal{F}$ satisfying Assumption 2 and Assumption 3, with probability at least $1 - \delta$, Algorithm 13 terminates and outputs a policy π with $v^\pi \geq v^\star - \epsilon$, using at most*

$$\tilde{O}\left(\frac{H^3 K\mathrm{W}_{\kappa^\star}^2 \log(|\mathcal{M}||\mathcal{F}|/\delta)}{(\kappa^\star\epsilon)^2}\right) \textit{ trajectories.}$$

Proof. Consider the j^{th} iteration in the i^{th} epoch. Based on the value of $\phi_{i,j}$, $n_{e_{i,j}}$, $n_{i,j}$, using Lemma 23 and Lemma 22, with probability at least $1 - \delta_{i,j}$, for any $t \in [1, T_{i,j}]$ during the run of Algorithm 8, we have

$$|v^{\pi^t} - \hat{v}^{\pi^t}| \leq \epsilon/8, \tag{9.100}$$

$$|\hat{\mathcal{E}}_B(M^t, M^t, h) - \mathcal{E}_B(M^t, M^t, h)| \leq \epsilon/(8H), \forall h \in [H], \tag{9.101}$$

$$|\widehat{\mathcal{W}}(M^t, M', h_t) - \mathcal{W}(M^t, M', h_t)| \leq \phi_{i,j}, \forall M' \in \mathcal{M}. \tag{9.102}$$

The first condition above ensures that if Algorithm 8 terminates in the j^{th} iteration and the i^{th} epoch and outputs π, then π must be near-optimal, based on Lemma 25. The third inequality above together with the elimination criteria in Algorithm 8 ensures that $M^\star$ is never eliminated.

Denote $i_\star$ as the epoch where $2\mathrm{W}_{\kappa^\star}/\kappa^\star \leq N_{i_\star} \leq 4\mathrm{W}_{\kappa^\star}/\kappa^\star$, and $j_\star$ as the iteration inside the $i_\star^{\text{th}}$ epoch where $\kappa^\star/2 \leq \kappa_{i_\star,j_\star} \leq \kappa^\star$. Since $\mathrm{W}_{i_\star,j_\star} = N_{i_\star}\kappa_{i_\star,j_\star}$, we have:

$$\mathrm{W}_{\kappa^\star} \leq \mathrm{W}_{i_\star,j_\star} \leq 4\mathrm{W}_{\kappa^\star}. \tag{9.103}$$

Below we condition on the event that $M^\star$ is not eliminated during any epoch before $i_\star$, and any iteration before $j_\star$ in the $i_\star^{\text{th}}$ epoch. We analyze the $j_\star^{\text{th}}$ iteration in the $i_\star^{\text{th}}$ epoch below.

Since $N_{i_\star} = 2^{i_\star - 1}$ and $N_{i_\star} \leq 4\mathrm{W}_{\kappa^\star}/\kappa^\star$, we must have $i_\star \leq 1 + \log_2(4\mathrm{W}_{\kappa^\star}/\kappa^\star)$. Also note that we have with these settings that $\mathrm{W}_{i^\star,j^\star}/(\kappa_{i^\star,j^\star})^2 \geq \mathrm{W}_{\kappa^\star}/(\kappa^\star)^2$, so that the number of samples we use at round $(i^\star, j^\star)$is at least as large, and the parameter ϕ is no larger than what we would have if we knew $\kappa^\star$ and used it in Algorithm 8.

Based on the value of $\phi_{i_\star,j_\star}$, $n_{i_\star,j_\star}$, $n_{e_{i_\star,j_\star}}$ and $T_{i_\star,j_\star}$, we know with probability at least $1 - \delta_{i_\star,j_\star}$, for any $t \in [1, T_{i_\star,j_\star}]$ in the execution of Algorithm 8, inequalities (9.100), (9.101), and (9.102) hold. Conditioned on this event and since $\phi_{i^\star,j^\star}$ is small enough as observed above, similar to the proof of Lemma 28, we can show that Algorithm 8 must terminate in at most $H\mathrm{W}_{\kappa^\star}\log(\beta/2\phi)/\log(5/3)$ many rounds in this iteration.

From (9.103), we know that $\mathrm{W}_{i_\star,j_\star} \geq \mathrm{W}_{\kappa^\star}$, which implies that $T_{i_\star,j_\star} \geq H\mathrm{W}_{\kappa^\star}\log(\beta/2\phi)/\log(5/3)$. In other words, in the $j_\star^{\text{th}}$ iteration of the $i_\star^{\text{th}}$ epoch, we run Algorithm 8 long enough to guarantee that it terminates and outputs a policy. We have already ensured that if it terminates, it must output a policy π with $v^\pi \geq v^\star - \epsilon$ (this is true for any (i, j) pair).

Now we calculate the sample complexity. In the i^{th} epoch, since we terminate when $\mathrm{W}_{i,j} < 1$, the number of iterations is at most $\log_2 N_i < i$. Hence the number of trajectories collected in this epoch is at most

$$\begin{aligned}
&\sum_{j=1}^{i}(n_{e_{i,j}} + n_{i,j})T_{i,j} = \sum_{j=1}^{i} O\left(H^3 K \mathrm{W}_{i,j}^2 \log(T_{i,j}|\mathcal{M}||\mathcal{F}|/\delta_{i,j})/(\epsilon\kappa_{i,j})^2\right) \\
&= O\left(iH^3 K N_i^2 \log(T_{i,1}|\mathcal{M}||\mathcal{F}|/\delta_{i,i})/\epsilon^2\right),
\end{aligned}$$

where we used the fact that $N_i = \mathrm{W}_{i,j}/\kappa_{i,j}$, $T_{i,1} \geq T_{i,j}$, and $\delta_{i,i} \leq \delta_{i,j}$. Note that $\sum_{i=1}^{i_\star - 1} iN_i^2 = \sum_{i=1}^{i_\star - 1} i(2^{i-1})^2 \leq (i_\star - 1)(2^{i_\star - 1})^2/3 = O(i_\star N_{i_\star}^2)$. Hence the sample complexity in the $i_\star^{\text{th}}$ epoch dominates the total sample complexity, which is

$$\tilde{O}\left((1 + \log_2(4\mathrm{W}_{\kappa^\star}/\kappa^\star))H^3 K \mathrm{W}_{\kappa^\star}^2 \log(T_{i_\star,0}|\mathcal{M}||\mathcal{F}|/\delta_{i_\star,i_\star})/(\kappa^\star\epsilon)^2\right),$$

where we used the fact that $i_\star \leq 1 + \log_2(2\mathrm{W}_{\kappa^\star}/\kappa^\star)$, and $N_{i_\star} \leq 2\mathrm{W}_{\kappa^\star}/\kappa^\star$. Applying a union bound over (i, j), with $i \leq i_\star$, since we have $\sum_{i=1}^{i_\star}\sum_{j=1}^{i}\delta_{i,j} \leq \sum_{i=1}^{i_\star}\delta_{i_\star,1} = \sum_{i=1}^{i_\star}\delta/(i(i+1)) \leq \delta$, the failure probability is at most δ, which proves the theorem. □

9.5.10 Details on Exponential Family Model Class

For any model $M \in \mathcal{M}$, conditioned on $(x, a) \in \mathcal{X}\times\mathcal{A}$, we assume $M_{x,a} \triangleq \exp(\langle\theta_{x,a}, \mathrm{T}(r, x')\rangle)/Z(\theta_{x,a})$ with $\theta_{x,a} \in \Theta \subset \mathbb{R}^m$. Without loss of generality, we assume $\|\theta_{x,a}\| \leq 1$, and $\Theta = \{\theta : \|\theta\| \leq 1\}$. We design $\mathcal{G} = \{\mathcal{X} \times \mathcal{A} \to \Theta\}$, i.e., $\mathcal{G}$ contains all mappings from $(\mathcal{X} \times \mathcal{A})$ to Θ. We design $\mathcal{F} = \{(x, a, r, x') \mapsto \langle g(x, a), \mathrm{T}(r, x')\rangle : g \in \mathcal{G}\}$. Using Definition 4, we have:

$$\begin{aligned}
\mathcal{W}(M, M', h) &= \sup_{f\in\mathcal{F}} \mathop{\mathbb{E}}_{x_h\sim\pi_M, a_h\sim\pi_{M'}}\left[\mathop{\mathbb{E}}_{(r,x')\sim M'_h}[f(x_h, a_h, r, x')] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[f(x_h, a_h, r, x')]\right] \\
&= \mathop{\mathbb{E}}_{x_h\sim\pi_M, a_h\sim\pi_{M'}}\left[\sup_{\theta\in\Theta}\left(\mathop{\mathbb{E}}_{(r,x')\sim M'_h}[\langle\theta, \mathrm{T}(r, x')\rangle] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[\langle\theta, \mathrm{T}(r, x')\rangle]\right)\right] \\
&= \mathop{\mathbb{E}}_{x_h\sim\pi_M, a_h\sim\pi_{M'}}\left[\left\|\mathop{\mathbb{E}}_{(r,x')\sim M'_h}[\mathrm{T}(r, x')] - \mathop{\mathbb{E}}_{(r,x')\sim M^\star_h}[\mathrm{T}(r, x')]\right\|_\star\right],
\end{aligned}$$

where the second equality uses the fact that $\mathcal{G}$ contains all possible mappings from $\mathcal{X}\times\mathcal{A} \to \Theta$.

We assume that for any $\theta \in \Theta$, the hessian of the log partition function $\nabla^2 \log(Z(\theta))$ is positive definite with eigenvalues bounded between $[\gamma, \beta]$ with $0 \leq \gamma \leq \beta$. Below, we show that under the above assumptions, Bellman domination required for Assumption 3 still holds up to a constant.

Claim 30 (Bellman Domination for Exponential Families)**.** *In the exponential family setting, we have*

$$\frac{\gamma}{2\sqrt{2\beta}} \mathcal{E}_B(M, M', h) \leq \mathcal{W}(M, M', h).$$

Proof. We leverage Theorem 3.2 from Gao et al. [2018], which implies that

$$\frac{\gamma}{\sqrt{\beta}} \mathop{\mathbb{E}}_{x_h \sim \pi_M, a_h \sim \pi_{M'}} \sqrt{D_{KL}(M'_{x_h,a_h} || M^\star_{x_h,a_h})} \leq \mathcal{W}(M, M', h).$$

By Pinsker's inequality, we have:

$$\frac{\gamma}{\sqrt{2\beta}} \mathop{\mathbb{E}}_{x_h \sim \pi_M, a_h \sim \pi_{M'}} \left\| M'_{x_h,a_h} - M^\star_{x_h,a_h} \right\|_{\text{TV}} \leq \mathcal{W}(M, M', h),$$

where the LHS is the witness model misfit defined using total variation directly (6.3).

On the other hand, we know that $r + V_M(x)$ for any $M \in \mathcal{M}$ is upper bounded by 2 via our regularity assumption on the reward. Hence, the TV-based witness model misfit upper bounds Bellman error as follows:

$$2 \mathop{\mathbb{E}}_{x_h \sim \pi_M, a_h \sim \pi_{M'}} \left\| M'_{x_h,a_h} - M^\star_{x_h,a_h} \right\|_{\text{TV}} \geq \mathcal{E}_B(M, M', h),$$

which concludes the proof.

□

Note that the constant $\gamma/(2\sqrt{2\beta})$ can be absorbed into κ in the definition of witness rank (Definition 5).

9.6 Missing Proofs and Details in Chapter 7

9.6.1 Useful Lemmas

As we work in finite probability space, we will use the following fact regarding total variation distance and L1 distance for any two probability measures P and Q:

$$\|P - Q\|_1 = 2D_{TV}(P, Q). \tag{9.104}$$

Recall that $d_\pi = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_\pi^t$. The following lemma shows that if two policies are close with each other in terms of the trust region constraint we defined in the paper, then the state visitations of the two policies are not that far away.

Lemma 36. *Given any two policy π_1 and π_2 such that $\mathbb{E}_{s\sim d_{\pi_1}}\left[D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))\right] \leq \alpha$, then we have:*

$$\|d_{\pi_1} - d_{\pi_2}\|_1 \leq \frac{2\alpha}{1-\gamma}. \tag{9.105}$$

Proof. Fix a state s and time step t, let us first consider $d_{\pi_1}^t(s) - d_{\pi_2}^t(s)$.

$$\begin{aligned}
&d_{\pi_1}^t(s) - d_{\pi_2}^t(s)\\
&= \sum_{s_0,s_1,..,s_{t-1}}\sum_{a_0,a_1,..,a_{t-1}}\Big(\rho(s_0)\pi_1(a_0|s_0)P_{s_0,a_0}(s_2)...\pi_1(a_{t-1}|s_{t-1})P_{s_{t-1},a_{t-1}}(s)\\
&- \rho(s_0)\pi_2(a_0|s_0)P_{s_0,a_0}(s_1)...\pi_2(a_{t-1}|s_{t-1})P_{s_{t-1},a_{t-1}}(s)\Big)\\
&= \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)\sum_{s_1}P_{s_0,a_0}(s_1)...\sum_{a_{t-1}}\pi_1(a_{t-1}|s_{t-1})P_{s_{t-1},a_{t-1}}(s)\\
&\quad - \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_2(a_0|s_0)\sum_{s_1}P_{s_0,a_0}(s_1)...\sum_{a_{t-1}}\pi_2(a_{t-1}|s_{t-1})P_{s_{t-1},a_{t-1}}(s)\\
&= \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)P(s_t = s|s_0,a_0;\pi_1) - \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_2(a_0|s_0)P(s_t = s|s_0,a_0;\pi_2),
\end{aligned} \tag{9.106}$$

where $P(s_t = s|s_0, a_0; \pi)$ stands for the probability of reaching state s at time step t, starting at s_0 and a_0 and then following π. Continue, we have:

$$\begin{aligned}
&|d_{\pi_1}^t(s) - d_{\pi_2}^t(s)|\\
&= |\sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)P(s_t = s|s_0,a_0;\pi_1) - \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_2(a_0|s_0)P(s_t = s|s_0,a_0;\pi_2)|\\
&\leq |\sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)P(s_t = s|s_0,a_0;\pi_1) - \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)P(s_t = s|s_0,a_0;\pi_2)|\\
&\quad + |\sum_{s_0}\rho(s_0)\sum_{a_0}\pi_1(a_0|s_0)P(s_t = s|s_0,a_0;\pi_2) - \sum_{s_0}\rho(s_0)\sum_{a_0}\pi_2(a_0|s_0)P(s_t = s|s_0,a_0;\pi_2)|\\
&\leq |\sum_{s_1}d_{\pi_1}^1(s_1)\left(P(s_t = s|s_1;\pi_1) - P(s_t = s|s_1;\pi_2)\right)| + \mathbb{E}_{s_0\sim\rho}\sum_{a_0}|\pi_1(a_0|s_0) - \pi_2(a_0|s_0)|P(s_t = s|s_0,a_0;\pi_2)
\end{aligned} \tag{9.107}$$

Add $\sum_s$ on both sides of the above equality, we get the following inequality:

$$\sum_s |d_{\pi_1}^t(s) - d_{\pi_2}^t(s)|$$

$$\leq \mathbb{E}_{s_1 \sim d^1_{\pi_1}} \sum_s |P(s_t = s|s_1;\pi_1) - P(s_t = s|s_1;\pi_2)| + \mathbb{E}_{s_0\sim\rho}\|\pi_1(\cdot|s_0) - \pi_2(\cdot|s_0)\|_1 \quad (9.108)$$

We can apply similar operations on $P(s_t = s|s_1;\pi_1) - P(s_t = s|s_1;\pi_2)$ as follows:

$$\begin{aligned}
&\mathbb{E}_{s_1 \sim d^1_{\pi_1}} \sum_s |P(s_t = s|s_1;\pi_1) - P(s_t = s|s_1;\pi_2)| \\
&= \mathbb{E}_{s \sim d^1_{\pi_1}} \sum_s |\sum_{a_1} [\pi_1(a_1|s_1)P(s_t = s|s_1, a_1;\pi_1) - \pi_2(a_1|s_1)P(s_t = s|s_1, a_2;\pi_2)]\,| \\
&\leq \mathbb{E}_{s_2 \sim d^2_{\pi_1}} \sum_s |P(s_t = s|s_2;\pi_1) - P(s_t = s|s_2;\pi_2)| + \mathbb{E}_{s_1 \sim d^1_{\pi_1}}[\|\pi_1(\cdot|s_1) - \pi_2(\cdot|s_1)\|_1]
\end{aligned}$$

Again, if we continue expand $P(s_t = s|s_2;\pi_1) - P(s_t = s|s_2;\pi_2)$ till time step t, we get:

$$\sum_s |d^t_{\pi_1}(s) - d^t_{\pi_2}(s)| \leq \sum_{i=0}^{t-1} \mathbb{E}_{s_i \sim d^i_{\pi_1}}[\|\pi_1(\cdot|s_i) - \pi_2(\cdot|s_i)\|_1] \quad (9.109)$$

Hence, for $\|d_{\pi_1} - d_{\pi_2}\|_1$, we have:

$$\begin{aligned}
&\|d_{\pi_1} - d_{\pi_2}\|_1 \leq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \|d^t_{\pi_1} - d^t_{\pi_2}\|_1 \\
&\leq \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s\sim d^t_{\pi_1}}[\|\pi_1(\cdot|s) - \pi_2(\cdot|s)\|_1] \leq \sum_{t=0}^{\infty} 2\gamma^t \mathbb{E}_{s\sim d^t_{\pi_1}}[D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))] \leq \frac{2\alpha}{1-\gamma}.
\end{aligned} \quad (9.110)$$

□

Lemma 37. *For any two distribution P and Q over $\mathcal{X}$, and any bounded function $f : \mathcal{X} \to \mathbb{R}$ such that $|f(x)| \leq c, \forall x \in \mathcal{X}$, we have:*

$$|\mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{x\sim Q}[f(x)]| \leq c\|P - Q\|_1. \quad (9.111)$$

Proof.

$$\begin{aligned}
|\mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{x\sim Q}[f(x)]| &= |\sum_{x\in\mathcal{X}} P(x)f(x) - Q(x)f(x)| \\
&\leq \sum_x |P(x)f(x) - Q(x)f(x)| \leq \sum_x |f(x)||P(x) - Q(x)| \\
&\leq c\sum_x |P(x) - Q(x)| = c\|P - Q\|_1.
\end{aligned} \quad (9.112)$$

□

9.6.2 Proof of Theorem 23

Recall that we denote $d_\pi \pi$ as the joint state-action distribution under policy π. To prove Theorem 23, we will use Lemma 1.2 presented in the Appendix from Ross and Bagnell [2012] to prove the following claim:

Lemma 38. *Suppose we learned a approximate model $\hat{P}$ and obtain the optimal policy η_n with respect to the objective function $J(\pi)$ under $\hat{P}$ and the trust-region constraint $\mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi,\pi_n) \leq \alpha$, then compare to π_n^*, we have:*

$$J(\eta_n) - J(\eta_n^*) \leq \frac{\gamma}{2(1-\gamma)}\left(\mathbb{E}_{(s,a)\sim d_{\eta_n}\eta_n}\left[\|\hat{P}_{s,a}-P_{s,a}\|_1\right] + \mathbb{E}_{(s,a)\sim d_{\eta_n^*}\eta_n^*}\left[\|\hat{P}_{s,a}-P_{s,a}\|_1\right]\right). \tag{9.113}$$

Proof. Denote $\hat{V}^{\pi}$ as the value function of policy π under the approximate model $\hat{P}$. From Lemma 1.2 and Corollary 1.2 in Ross and Bagnell [2012], we know that for any two policies π_1 and π_2, we have:

$$\begin{aligned} J(\pi_1) - J(\pi_2) &= \mathbb{E}_{s\sim\rho_0}[\hat{V}^{\pi_1}(s) - \hat{V}^{\pi_2}(s)] \\ &\quad + \frac{\gamma}{2(1-\gamma)}\left(\mathbb{E}_{(s,a)\sim d_{\pi_1}\pi_1}\left[\|\hat{P}_{s,a}-P_{s,a}\|_1\right] + \mathbb{E}_{(s,a)\sim d_{\pi_2}\pi_2}\left[\|\hat{P}_{s,a}-P_{s,a}\|_1\right]\right). \end{aligned} \tag{9.114}$$

Now replace π_1 with η_n and π_2 with η_n^*. Note that both η_n and η_n^* are in the trust region constraint $\mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi,\pi_n) \leq \alpha$ by definition. As η_n is the optimal control under the approximate model $\hat{P}$ (i.e., the optimal solution to Eq. 7.4), we must have $\mathbb{E}_{s\sim\rho_0}[\hat{V}^{\eta_n}(s) - \hat{V}^{\eta_n^*}(s)] \leq 0$. Substitute it back to Eq. 9.114, we immediately prove the above lemma. □

The above lemma shows that the performance gap between η_n and η_n^* is measured under the state-action distributions measured from η_n and η_n^* while our model $\hat{P}$ is only accurate under the state-action distribution from π_n. Luckily due to the trust-region constraint $\mathbb{E}_{s\sim d_{\pi_n}} D_{TV}(\pi,\pi_n)$ and the fact that η_n and η_n^* are both in the trust-region, we can show that $d_{\eta_n}\eta_n$, $d_{\pi_n^*}\pi_n^*$ are not that far from $d_{\pi_n}\pi_n$ using Lemma 36:

$$\begin{aligned} &\|d_{\eta_n}\eta_n - d_{\pi_n}\pi_n\|_1 \leq \|d_{\eta_n}\eta_n - d_{\pi_n}\eta_n\|_1 + \|d_{\pi_n}\eta_n - d_{\pi_n}\pi_n\|_1 \\ &\leq \|d_{\eta_n} - d_{\pi_n}\|_1 + \mathbb{E}_{s\sim d_{\pi_n}}[\|\eta_n(\cdot|s) - \pi_n(\cdot|s)\|_1] \leq \frac{2\alpha}{1-\gamma} + 2\alpha \leq \frac{4\alpha}{1-\gamma}. \end{aligned} \tag{9.115}$$

similarly, for π_n^* we have:

$$\|d_{\eta_n^*}\eta_n^* - d_{\pi_n}\pi_n\|_1 \leq \frac{4\alpha}{1-\gamma}. \tag{9.116}$$

Go back to Eq. 9.113, let us replace $\mathbb{E}_{d_{\eta_n}\eta_n}$ and $\mathbb{E}_{d_{\eta_n^*}\eta_n^*}$ by $\mathbb{E}_{d_{\pi_n}\pi_n}$ and using Lemma 37, we will have:

$$\begin{aligned} &|\mathbb{E}_{(s,a)\sim d_{\eta_n}\eta_n}[\|\hat{P}_{s,a}-P_{s,a}\|_1] - \mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n}[\|\hat{P}_{s,a}-P_{s,a}\|_1]| \leq 2\|d_{\eta_n}\eta_n - d_{\pi_n}\pi_n\|_1 \leq \frac{8\alpha}{1-\gamma} \\ &\Rightarrow \mathbb{E}_{(s,a)\sim d_{\eta_n}\eta_n}[\|\hat{P}_{s,a}-P_{s,a}\|_1] \leq \mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n}[\|\hat{P}_{s,a}-P_{s,a}\|_1] + \frac{8\alpha}{(1-\gamma)}, \end{aligned} \tag{9.117}$$

and similarly,

$$\mathbb{E}_{(s,a)\sim d_{\eta_n^*}\eta_n^*}[\|\hat{P}_{s,a}-P_{s,a}\|_1] \leq \mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n}[\|\hat{P}_{s,a}-P_{s,a}\|_1] + \frac{8\alpha}{(1-\gamma)}. \tag{9.118}$$

Combine Eqs. 9.117 and 9.118, we have:

$$\begin{aligned} J(\eta_n) - J(\eta_n^*) &\leq \frac{\gamma}{2(1-\gamma)}\left(2\mathbb{E}_{(s,a)\sim d_{\pi_n}\pi_n}[\|\hat{P}_{s,a} - P_{s,a}\|_1] + 16\alpha/(1-\gamma)\right) \\ &= \frac{\gamma\delta}{1-\gamma} + \frac{8\gamma\alpha}{(1-\gamma)^2} = O\left(\frac{\gamma\delta}{1-\gamma}\right) + O\left(\frac{\gamma\alpha}{(1-\gamma)^2}\right). \end{aligned} \tag{9.119}$$

Using the definition of $\Delta(\alpha)$, adding $J(\pi_n)$ and subtracting $J(\pi_n)$ on the LHS of the above inequality, we prove the theorem.

9.6.3 Proof of Theorem 24

The definition of π_{n+1} implies that $\mathbb{E}_{s\sim d_{\pi_n}}[D_{TV}(\pi_{n+1}(\cdot|s), \pi_n(\cdot|s))] \leq \beta$. Using Lemma 36, we will have that the total variation distance between $d^t_{\pi_{n+1}}$ and $d^t_{\pi_n}$ is:

$$\|d_{\pi_{n+1}} - d_{\pi_n}\|_1 \leq \frac{2\beta}{1-\gamma}. \tag{9.120}$$

Now we can compute the performance improvement of π_{n+1} over η_n as follows:

$$\begin{aligned} &(1-\gamma)(J(\pi_{n+1}) - J(\eta_n)) = \mathbb{E}_{s\sim d_{\pi_{n+1}}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] \\ &= \mathbb{E}_{s\sim d_{\pi_{n+1}}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] - \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] + \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] \\ &\leq \left|\mathbb{E}_{s\sim d_{\pi_{n+1}}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] - \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right]\right| + \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] \\ &\leq \frac{2\varepsilon\beta}{1-\gamma} + \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]\right] \\ &= \frac{2\varepsilon\beta}{1-\gamma} + \mathbb{A}_n(\pi_{n+1}) \\ &= \frac{2\varepsilon\beta}{1-\gamma} - |\mathbb{A}_n(\pi_{n+1})| \end{aligned} \tag{9.121}$$

Finally, to bound $J(\pi_{n+1}) - J(\pi_n)$, we can simply do:

$$\begin{aligned} J(\pi_{n+1}) - J(\pi_n) &= J(\pi_{n+1}) - J(\eta_n) + J(\eta_n) - J(\pi_n) \\ &\leq \frac{\beta\varepsilon}{(1-\gamma)^2} - \frac{|\mathbb{A}_n(\pi_{n+1})|}{1-\gamma} - \Delta(\alpha). \end{aligned} \tag{9.122}$$

9.6.4 Global Performance Guarantee for DPI

When $|\Delta_n(\alpha)|$ and $|\mathbb{A}_n(\pi_{n+1})|$ are small, say $|\Delta_n(\alpha)| \leq \xi$, $|\mathbb{A}_n(\pi_{n+1})| \leq \xi$, then we can guarantee that η_n and π_n are good policies, if our initial distribution ρ happens to be a good distribution (e.g., close to d_{π^*}), and the *realizable assumption* holds: $\min_{\pi\in\Pi}\mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\eta_n}(s,a)]\right] = \mathbb{E}_{s\sim d_{\pi_n}}\left[\min_{a\sim\mathcal{A}}[A^{\eta_n}(s,a)]\right]$. We call a policy class Π *closed under its convex hull* if for any sequence of policies $\{\pi_i\}_i, \pi_i \in \Pi$, the convex combination $\sum_i w_i\pi_i$, for any w such that $w_i \geq 0$ and $\sum_i w_i = 1$, also belongs to Π.

Theorem 31. *Under the realizable assumption and the assumption of* Π *is closed under its convex hull, and* $\max\{|\mathbb{A}_n(\pi_{n+1})|, \Delta(\alpha)\} \leq \xi \in \mathbb{R}^+$*, then for* η_n*, we have:*

$$J(\eta_n) - J(\pi^*) \leq \left(\max_s\left(\frac{d_{\pi^*}(s)}{\rho_0(s)}\right)\right)\left(\frac{\xi}{\beta(1-\gamma)^2} + \frac{\xi}{\beta(1-\gamma)}\right).$$

The term $(\max_s (d_{\pi^*}(s)/\rho_0(s)))$ measures the distribution mismatch between the initial state distribution ρ_0 and the optimal policy π^*, and appears in some of previous API algorithms–CPI Kakade and Langford [2002] and PSDP Bagnell et al. [2004].[5]

Proof. Recall the average advantage of π_{n+1} over π_n is defined as $\mathbb{A}_{\pi_n}(\pi_{n+1}) = \mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_{n+1}(\cdot|s)}[A^{\eta_n}(s,a)]]$. Also recall that the conservative update where we first compute $\pi_n^* = \arg\min_{\pi\in\Pi}\mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi}A^{\eta_n}(s,a)]$, and then compute the new policy $\pi'_{n+1} = (1-\beta)\pi_n + \beta\pi_n^*$. Note that under the assumption that the policy class Π is closed under its convex hull, we have that $\pi'_{n+1}\in\Pi$. As we showed that π'_{n+1} satisfies the trust-region constraint defined in Eq. 7.5, we must have:

$$\mathbb{A}_{\pi_n}(\pi_{n+1}) = \mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_{n+1}(\cdot|s)}[A^{\eta_n}(s,a)]] \leq \mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{s\sim\pi'_{n+1}}[A^{\eta_n}(s,a)]], \tag{9.123}$$

due to the fact that π_{n+1} is the optimal solution of the optimization problem shown in Eq. 7.5 subject to the trust region constraint. Hence if $\mathbb{A}_{\pi}(\pi_{n+1}) \geq -\xi$, we must have $\mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{s\sim\pi'_{n+1}}[A^{\eta_n}(s,a)]] \geq -\xi$, which means that:

$$\begin{aligned}
&\mathbb{E}_{s\sim d_{\pi_n}}\Big[(1-\beta)\mathbb{E}_{s\sim d_{\pi_n}}A^{\eta_n}(s,a) + \beta\mathbb{E}_{s\sim d_{\pi_n^*}}A^{\eta_n}(s,a)\Big] \\
&= (1-\beta)(1-\gamma)(J(\pi_n) - J(\eta_n)) + \beta\mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_n^*}A^{\eta_n}(s,a)] \geq -\xi, \\
&\Rightarrow \mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_n^*}A^{\eta_n}(s,a)] \geq -\frac{\xi}{\beta} - \frac{1-\beta}{\beta}(1-\gamma)\Delta(\alpha) \geq -\frac{\xi}{\beta} - \frac{1-\gamma}{\beta}\Delta(\alpha).
\end{aligned} \tag{9.124}$$

Recall the realizable assumption: $\mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_n^*}A^{\eta_n}(s,a)] = \mathbb{E}_{s\sim d_{\pi_n}}[\min_a A^{\eta_n}(s,a)]$, we have:

$$\begin{aligned}
&-\frac{\xi}{\beta} - \frac{1-\gamma}{\beta}\Delta(\alpha) \leq \sum_s d_{\pi_n}(s)\min_a A^{\eta_n}(s,a) = \sum_s \frac{d_{\pi_n}(s)}{d_{\pi^*}(s)} d_{\pi^*}(s)\min_a A^{\eta_n}(s,a) \\
&\leq \min_s\left(\frac{d_{\pi_n}(s)}{d_{\pi^*}(s)}\right)\sum_s d_{\pi^*}(s)\min_a A^{\eta_n}(s,a) \\
&\leq \min_s\left(\frac{d_{\pi_n}(s)}{d_{\pi^*}(s)}\right)\sum_s d_{\pi^*}(s)\sum_a \pi^*(a|s)A^{\eta_n}(s,a) \\
&= \min_s\left(\frac{d_{\pi_n}(s)}{d_{\pi^*}(s)}\right)(1-\gamma)(J(\pi^*) - J(\eta_n)).
\end{aligned} \tag{9.125}$$

Rearrange, we get:

$$\begin{aligned}
J(\eta_n) - J(\pi^*) &\leq \left(\max_s\left(\frac{d_{\pi^*}(s)}{d_{\pi_n}(s)}\right)\right)\left(\frac{\xi}{\beta(1-\gamma)} + \frac{\Delta(\alpha)}{\beta}\right) \\
&\leq \left(\max_s\left(\frac{d_{\pi^*}(s)}{\rho(s)}\right)\right)\left(\frac{\xi}{\beta(1-\gamma)^2} + \frac{\xi}{\beta(1-\gamma)}\right)
\end{aligned} \tag{9.126}$$

□

9.6.5 Analysis on Using DAgger for Updating π_n

Note that in ExIt, once an intermediate expert is constructed, DAgger Ross et al. [2011] is used as the imitation learning algorithm to improvement the reactive policy. DAgger does

[5]While CPI considers a different setting where a good reset distribution ν (different from ρ_0) is accessible, DPI can utilize such reset distribution by replacing ρ_0 by ν during training.

not directly optimize the ultimate objective function—the expected total cost, but instead trying minimizes the number of mismatches between the learner and the expert. Here we used more advanced, cost-aware IL algorithms, AggreVaTe Ross and Bagnell [2014] and AggreVaTeD Sun et al. [2017c], which directly reason about the expected total cost, and guarantee to learn a policy that achieves one-step deviation improvement of the expert policy.

Below we analyze the update of π using DAgger.

To analyze the update of π using DAgger, we consider deterministic policy here: we assume π_n and η are both deterministic and the action space $\mathcal{A}$ is discrete. We consider the following update procedure for π:

$$\begin{aligned} &\min_{\pi\in\Pi} \mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{1}(a\neq \arg\min_a A^{\eta_n}(s,a))\right], \\ &\quad s.t., \mathbb{E}_{s\sim d_{\pi_n}}[\|\pi(\cdot|s)-\pi_n(\cdot|s)\|_1]\leq\beta. \end{aligned} \tag{9.127}$$

Namely we simply convert the cost vector defined by the disadvantage function by a "one-hot" encoded cost vector, where all entries are 1, except the entry corresponding to $\arg\min_a A^{\eta_n}(s,a)$ has cost 0. Ignoring the updates on the "expert" η_n, running the above update step with respect to π can be regarded as running online gradient descent with a local metric defined by the trust-region constraint. Recall that η_n may from a different policy class than Π.

Assume that we learn a policy π_{n+1} that achieves ϵ_n prediction error:

$$\mathbb{E}_{s\sim d_{\pi_n}}\left[\mathbb{E}_{a\sim\pi_{n+1}(\cdot|s)}\left[\mathbb{1}(a\neq\arg\min_a A^{\eta_n}(s,a))\right]\right]\leq\epsilon_n. \tag{9.128}$$

Namely we assume that we learn a policy π_{n+1} such that the average probability of mismatch to η_n is at most ϵ_n.

Using Lemma 36, we will have that the total variation distance between $d_{\pi_{n+1}}$ and d_{π_n} is at most:

$$\|d_{\pi_{n+1}}-d_{\pi_n}\|_1\leq\frac{2\beta}{1-\gamma}. \tag{9.129}$$

Applying PDL, we have:

$$\begin{aligned} &(1-\gamma)(J(\pi_{n+1})-J(\eta_n)) = \mathbb{E}_{s\sim d_{\pi_{n+1}}}[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]] \\ &\leq \mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_{n+1}}[A^{\eta_n}(s,a)]]+\frac{2\beta\varepsilon}{1-\gamma} \\ &= \mathbb{E}_{s\sim d_{\pi_n}}[\sum_{a\neq\arg\min_a A^{\eta_n}(s,a)}\pi(a|s)A^{\eta_n}(s,a)]+\frac{2\beta\varepsilon}{1-\gamma} \\ &\leq (\max_{s,a}A^{\eta_n}(s,a))\mathbb{E}_{s\sim d_{\pi_n}}[\mathbb{E}_{a\sim\pi_{n+1}}\mathbb{1}(a\neq\arg\min_a A^{\eta_n}(s,a))]+\frac{2\beta\varepsilon}{1-\gamma} \\ &\leq \varepsilon'\epsilon_n+\frac{2\beta\varepsilon}{1-\gamma}, \end{aligned} \tag{9.130}$$

where we define $\varepsilon'=\max_{s,a}A^{\eta_n}(s,a)$, which should be at a similar scale as ε. Hence we can show that performance difference between π_{n+1} and π_n as:

$$J(\pi_{n+1})-J(\pi_n)\leq\frac{2\beta\epsilon}{(1-\gamma)^2}+\frac{\varepsilon'\epsilon_n}{1-\gamma}-\Delta(\alpha). \tag{9.131}$$

Now we can compare the above upper bound to the upper bound shown in Theorem 24. Note that even if we assume the policy class is rich and the learning process perfect learns a policy (i.e., $\pi_{n+1} = \eta_n$) that achieves prediction error $\epsilon_n = 0$, we can see that the improvement of π_{n+1} over π_n only consists of the improvement from the local optimal control $\Delta(\alpha)$. While in theorem 24, under the same assumption, except for $\Delta(\alpha)$, the improvement of π_{n+1} over π_n has an extra term $\frac{|\mathbb{A}_n(\pi_{n+1})|}{1-\gamma}$, which basically indicates that we learn a policy π_{n+1} that is one-step deviation improved over η_n by leveraging the cost informed by the disadvantage function. If one uses DAgger, than the best we can hope is to learn a policy that performs as good as the "expert" η_n (i.e., $\epsilon_n = 0$).

9.6.6 Additional Experimental Details

Synthetic Discrete MDPs and Conservative Policy Update Implementation

We follow Scherrer [2014] to randomly create 10 discrete MDPs, each with 1000 states, 5 actions and 2 branches (namely, each state action pair leads to at most 2 different states in the next step). We work in model-free setting: we cannot explicitly compute the distribution d_π and we can only generate samples from d_π by executing π.

We maintain a tabular representation $\hat{P} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$, where each entry $P_{i,j,k}$ records the number of visits of the state-action-next state triple. We represent η as a 2d matrix $\eta \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, where $\eta_{i,j}$ stands for the probability of executing action j at state i. The reactive policy uses the binary encoding of the state id as the feature, which we denote as $\phi(s) \in \mathbb{R}^{d_s}$ (d_s is the dimension of feature space, which is $\log_2(|\mathcal{S}|)$ in our setup). Hence the reactive policy π_n sits in low-dimensional feature space and doesn't scale with respect to the size of the state space S.

For both our approach and CPI, we implement the unconstrained cost-sensitive classification (Eq. 7.5) by the Cost-Sensitive One Against All (CSOAA) classification technique. Specifically, given a set of states $\{s_i\}_i$ sampled from d_{π_n}, and a cost vector $\{A^{\eta_n}(s_i,\cdot) \in \mathbb{R}^{|\mathcal{A}|}\}$ (byproduct of VI), we train a linear regressor $\hat{W} \in \mathbb{R}^{|\mathcal{A}|\times d_s}$ to predict the cost vector: $\hat{W}\phi(s) \approx \mathcal{A}^{\eta_n}(s,\cdot)$. Then π_n^* in Eq. 7.6 is just a classifier that predicts action $\arg\min_i(\hat{W}s)[i]$ corresponding to the smallest predicted cost. We then combine π_n^* with the previous policies as shown in Eq. 7.6 to make sure π_{n+1} satisfies the trust region constraint in Eq. 7.5.

For CPI, we estimate $A^{\pi_n}(s,a)$ by running value iteration using $\hat{P}$ with the original cost matrix. We also experimented estimating $A^{\pi_n}(s,\cdot)$ by empirical rollouts with importance weighting, which did not work well in practice due to high variance resulting from the empirical estimate and importance weight. For our method, we alternately compute η_n using VI with the new cost shown in Eq. 7.10 and $\hat{P}$, and update the Lagrange multiplier μ, under convergence. Hence *the only difference* between our approach and CPI here is simply that we use A^{η_n} while CPI uses A^{π_n}. We tuned the step size β (Eq. 7.6) for CPI. Neither our method nor CPI used line-search trick for β.

Our results indicates that using A^{η_n} converges much faster than using A^{π_n}, although computing η_n is much more time consuming than computing A^{π_n}. But again we emphasize that computing η_n doesn't require extra samples. For real large discrete MDPs, we can easily plug in an approximate VI techniques such as Gorodetsky et al. [2015] to significantly speed up computing η_n.

Details for Updating Lagrange Multiplier μ

Though running gradient ascent on μ is theoretically sound and can work in practice as well, but it converges slow and requires to tune the learning rate as we found experimentally. To speed up convergence, we used the same update procedure used in the practical implementation of Guided Policy Search Finn et al. [2016]. We set up $\mu_{\min}$ and $\mu_{\max}$. Starting from $\mu = \mu_{\min}$, we fix μ and compute η using the new cost c' as shown in Eq. 7.10 under the local dynamics $\hat{P}$ using LQR. We then compare $\mathbb{E}_{s\sim\mu_n} D_{KL}(\eta(\cdot|s), \pi_n(\cdot|s))$ to α. If η violates the constraint, i.e., $\mathbb{E}_{s\sim\mu_n} D_{KL}(\eta(\cdot|s), \pi_n(\cdot|s)) > \alpha$, then it means that μ is too small. In this case, we set $\mu_{\min} = \mu$, and compute new μ as $\mu = \min(\sqrt{\mu_{\min}\mu_{\max}}, 10\mu_{\min})$; On the other hand, if η satisfies the KL constraint, i.e, μ is too big, we set $\mu_{\max} = \mu$, and compute new μ as $\mu = \max(\sqrt{\mu_{\min}\mu_{\max}}, 0.1\mu_{\max})$. We early terminate the process once we find η such that $0.9\alpha \leq \mathbb{E}_{s\sim\mu_n} D_{KL}(\eta(\cdot|s), \pi_n(\cdot|s)) \leq 1.1\alpha$. We then store the most recent Lagrange multiplier μ which will be used as warm start of μ for the next iteration.

Details on the Natural Gradient Update to π

Here we provide details for updating π_θ via imitating η_Θ (Sec. 7.5.2) Recall the objective function $L_n(\theta) = \mathbb{E}_{s\sim d_{\pi_{\theta_n}}}[\mathbb{E}_{a\sim\pi(\cdot|s;\theta)}[A^{\eta_{\Theta_n}}(s,a)]]$, ∇_{θ_n} as $\nabla_\theta L_n(\theta)|_{\theta=\theta_n}$, and F_{θ_n} is the Fisher information matrix (equal to the Hessian of the KL constraint measured at θ_n). To compute $F_\theta^{-1}\nabla_\theta$, in our implementation, we use Conjugate Gradient with the Hessian-vector product trick Schulman et al. [2015a] to directly compute $F^{-1}\nabla$.

Note that the unbiased empirical estimation of ∇_{θ_n} and F_{θ_n} is well-studied and can be computed using samples generated from executing π_{θ_n}. Assume we roll out π_{θ_n} to generate K trajectories $\tau^i = \{s_0^i, a_0^i, ...s_T^i, a_T^i\}, \forall i \in [K]$. The empirical gradient and Fisher matrix can be formed using these samples as $\nabla_{\theta_n} = \sum_{s,a}\left[\nabla_{\theta_n}\left(\ln(\pi(a|s;\theta_n))\right) A^{\eta_{\Theta_n}}(s,a)\right]$ and $F_{\theta_n} = \sum_{s,a}\left[(\nabla\ln(\pi(a|s;\theta_n)))(\nabla_{\theta_n}\ln(\pi(a|s;\theta_n))^T\right]$.

The objective $L_n(\theta)$ could be nonlinear with respect θ, depending on the function approximator used for π_n,. Hence one step of gradient descent may not reduce $L_n(\theta)$ enough. In practice, we can perform k steps ($k > 1$) of NGD shown in Eq. 7.12, with the learning rate shrinking to $\sqrt{(\beta/k)/(\nabla_\theta^T F_{\theta_n}^{-1}\nabla_\theta)}$ to ensure that after k steps, the solution still satisfies the constraint in Eq. 7.11.

Details on the Continuous Control Experiment Setup

The cost function $c(s,a)$ for discrete MDP is uniformly sampled from $[0,1]$. For the continuous control experiments, we designed the cost function $c(s,a)$, which is set to be known to our algorithms. For cartpole and helicopter hover, denote the target state as s^*, the cost function is designed to be exactly quadratic: $c(s,a) = (s-s^*)^T Q(s-s^*) + a^T Ra$, which penalizes the distance to the goal and large control inputs. For Swimmer, Hopper and Half-Cheetah experiment, we set up a target moving forward speed v^*. For any state, denote the velocity component as s_v, the quadratic cost function is designed as $c(s,a) = q(s_v - v^*)^2 + a^T Ra$, which encourages the agent to move forward in a constant speed while avoiding using large control inputs.

For reactive policies, we simply used two-layer feedforward neural network as the parameterized policies–the same structures used in the implementation of Schulman et al. [2015a].

For local linear model fit under d_π, given a dataset in the format of $\{s_i, a_i, s_i'\}_{i=1}^N$, where $s_i \sim d_\pi$, $a_i \sim \pi(\cdot|s)$, and $s_i' \sim P_{s_i,a_i}$, we perform Ridge linear regression:

$$A, B, c = \arg\min_{A,B,c} \frac{1}{N}\sum_{i=1}^N \|As_i + Ba_i + c - s_i'\|_2^2 + \lambda(\|A\|_F + \|B\|_F + \|c\|_2), \tag{9.132}$$

where regularization λ is pre-fixed to be a small number for all experiments. With A, B, c, we estimate Σ as $\Sigma = \frac{1}{N}\sum_{i=1}^N e_i e_i^T$, where $e_i = As_i + Ba_i + c - s_i'$.

Bibliography

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Conference on Learning Theory*, 2011.

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1. ACM, 2004.

Pieter Abbeel, Varun Ganapathi, and Andrew Y Ng. Learning vehicular dynamics, with application to modeling helicopters. In *NIPS*, pages 1–8, 2005.

Eric W Aboaf, Steven M Drucker, and Christopher G Atkeson. Task-level robot learning: Juggling a tennis ball more accurately. In *IEEE International Conference on Robotics and Automation*, 1989.

Alekh Agarwal, Ofer Dekel, and Lin Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40. Citeseer, 2010.

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646, 2014.

Eric Allender, Sanjeev Arora, Michael Kearns, Cristopher Moore, and Alexander Russell. A note on the representational incompatibility of function approximation and factored dynamics. In *Advances in Neural Information Processing Systems*, 2003.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *arXiv preprint arXiv:1705.08439*, 2017.

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 2008.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017.

Christopher G Atkeson. Using local trajectory optimizers to speed up global optimization in dynamic programming. In *Advances in neural information processing systems*, pages 663–670, 1994.

Christopher G Atkeson. Efficient robust policy optimization. In *American Control Conference (ACC), 2012*, pages 5220–5227. IEEE, 2012.

Christopher G Atkeson and Jun Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. In *Advances in neural information processing systems*, pages 1643–1650, 2003.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. *ICML*, 2017.

J Andrew Bagnell and Jeff Schneider. Covariant policy search. IJCAI, 2003.

J Andrew Bagnell and Jeff G Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1615–1620. IEEE, 2001.

J Andrew Bagnell, Sham M Kakade, Jeff G Schneider, and Andrew Y Ng. Policy search by dynamic programming. In *Advances in neural information processing systems*, pages 831–838, 2004.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)*, pages 30–37, 1995.

Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015.

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 49–56. ACM, 2005.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016a.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016b.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 2012.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 2015.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *arXiv preprint arXiv:1710.11606*, 2017a.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points ii: First-order methods. *arXiv preprint arXiv:1711.00841*, 2017b.

Kai-Wei Chang, He He, Hal Daumé III, and John Langford. Learning to search for dependencies. *arXiv preprint arXiv:1503.05615*, 2015a.

Kai-wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. Learning to search better than your teacher. In *ICML*, 2015b.

Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, and Debadeepta Dey. Learning to gather information via imitation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 908–915. IEEE, 2017a.

Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Adaptive information gathering via imitation learning. *RSS*, 2017b.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, pages 144–151, 2008.

Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

Christoph Dann, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. On oracle-efficient PAC reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*, 2018.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 2009.

Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, 2018.

Marc P Deisenroth, Carl E Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems*, 2011.

Luc Devroye and Gábor Lugosi. *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012.

Huaian Diao, Zhao Song, Wen Sun, and David P Woodruff. Sketching for kronecker product regression and p-splines. *arXiv preprint arXiv:1712.09473*, 2017.

Carlos Diuk, Lihong Li, and Bethany R Leffler. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 249–256. ACM, 2009.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, May 2015. ISSN 0018-9448. doi: 10.1109/TIT.2015.2409256.

James A Duyck and Geoffrey J Gordon. Predicting structure in handwritten algebra data from low level features. *Data Analysis Project Report, MLD, CMU*, 2015.

Ashley D Edwards, Himanshu Sahni, Yannick Schroeker, and Charles L Isbell. Imitating latent policies from observation. *arXiv preprint arXiv:1805.07914*, 2018.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with Gaussian processes. *Proceedings of the 22nd international conference on Machine learning*, 2005.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pages 1486–1494, 2017.

C. Finn, M. Zhang, J. Fu, X. Tan, Z. McCarthy, E. Scharff, and S. Levine. Guided policy search code implementation, 2016. URL `http://rll.berkeley.edu/gps`. Software available from rll.berkeley.edu/gps.

Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics, 2005.

Kenji Fukumizu, Arthur Gretton, Gert R Lanckriet, Bernhard Schölkopf, and Bharath K Sriperumbudur. Kernel choice and classifiability for rkhs embeddings of probability distributions. In *Advances in neural information processing systems*, pages 1750–1758, 2009.

Chao Gao, Jiyi Liu, Yuan Yao, and Weizhi Zhu. Robust estimation and generative adversarial nets. *arXiv:1810.02030*, 2018.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015. doi: 10.15607/RSS.2015.XI.015.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *JMLR*, 2004.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Aprtificial Intelligence Research*, 2003.

Zhaohan Daniel Guo and Emma Brunskill. Sample efficient feature selection for factored MDPs. *arXiv preprint arXiv:1703.03454*, 2017.

Elad Hazan and Satyen Kale. Projection-free Online Learning. *29th International Conference on Machine Learning (ICML 2012)*, pages 521–528, 2012.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the 19th annual conference on Computa- tional Learning Theory (COLT 2006)*, pages 169–192, 2006.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent predictive state policy networks. *arXiv preprint arXiv:1803.01489*, 2018.

Verena Heidrich-Meisner and Christian Igel. Evolution strategies for direct policy search. In *International Conference on Parallel Problem Solving from Nature*, pages 428–437. Springer, 2008.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Hanzhang Hu, Wen Sun, Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Gradient boosting on stochastic data streams. *arXiv preprint arXiv:1703.00377*, 2017.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *JMLR*, 2010.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. *arXiv preprint arXiv:1610.09512*, 2016.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, 2017.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *arXiv preprint arXiv:1807.03765*, 2018.

Sham Kakade. A natural policy gradient. *NIPS*, 2002.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Sham Kakade, Michael J Kearns, and John Langford. Exploration in metric state spaces. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 306–312, 2003.

Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *IJCAI*, volume 16, pages 740–747, 1999.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Pac reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*, pages 1840–1848, 2016.

Brian Kulis, Peter L Bartlett, Bartlett Eecs, and Berkeley Edu. Implicit Online Learning. In *Proceedings of the 27th international conference on Machine learning (ICML 2010)*, pages 575–582, 2010.

Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-Interscience New York, 1972.

Kailasam Lakshmanan, Ronald Ortner, and Daniil Ryabko. Improved regret bounds for undiscounted continuous reinforcement learning. In *International Conference on Machine Learning*, pages 524–532, 2015.

Tor Lattimore, Marcus Hutter, Peter Sunehag, et al. The sample-complexity of general reinforcement learning. In *Proceedings of the 30th International Conference on Machine Learning*. Journal of Machine Learning Research, 2013.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of a classification-based policy iteration algorithm. In *ICML-27th International Conference on Machine Learning*, pages 607–614. Omnipress, 2010.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of classification-based policy iteration algorithms. *The Journal of Machine Learning Research*, 17(1):583–612, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Lihong Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, pages 560–567, 2008.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Shie Mannor, Reuven Y Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

William Montgomery, Anurag Ajay, Chelsea Finn, Pieter Abbeel, and Sergey Levine. Reset-free guided policy search: efficient deep reinforcement learning with stochastic initial states. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3373–3380. IEEE, 2017.

William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pages 4008–4016, 2016.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

KR Müller, AJ Smola, and G Rätsch. Predicting time series with support vector machines. *Artificial Neural Networks âĂŽÃĎÃő ICANN'9*, 1327:999–1004, 1997. doi: 10.1007/BFb0020283. URL `http://link.springer.com/chapter/10.1007/BFb0020283`.

Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 2000.

Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2146–2153. IEEE, 2017.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Trung Nguyen, Zhuoru Li, Tomi Silander, and Tze Yun Leong. Online feature selection for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 498–506, 2013.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Jungseul Ok, Alexandre Proutiere, and Damianos Tranos. Exploration in structured reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pages 8888–8896, 2018.

Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1763–1771, 2012.

Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. In *Advances in Neural Information Processing Systems*, 2014a.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. In *Advances in Neural Information Processing Systems*, pages 604–612, 2014b.

Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania*, 2018.

Jason Pazis and Ronald Parr. PAC Optimal Exploration in Continuous Space Markov Decision Processes. In *AAAI*, 2013.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *arXiv preprint arXiv:1804.02717*, 2018.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Aloke Phatak and Frank de Hoog. Exploiting the connection between pls, lanczos methods and conjugate gradients: alternative proofs of some properties of pls. *Journal of Chemometrics*, 2002.

Doina Precup and Richard S. Sutton. Exponentiated Gradient Methods for Reinforcement Learning. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, 1997.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *ICLR 2016*, 2015.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *ICML*, 2006.

Nicholas Rhinehart, Jiaji Zhou, Martial Hebert, and J Andrew Bagnell. Visual chunking: A list prediction framework for region-based object detection. In *ICRA*. IEEE, 2015.

M Robards, Peter Sunehag, Scott Sanner, and B Marthi. Sparse Kernel-SARSA(lambda) with an Eligibility Trace. *ECML PKDD*, 2011.

Stephane Ross and Drew Bagnell. Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1703–1710, 2012.

Stéphane Ross and J. Andrew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, pages 661–668, 2010a.

Stéphane Ross and J. Andrew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, pages 661–668, 2010b.

Stephane Ross and J. Andrew Bagnell. Stability Conditions for Online Learnability. *arXiv:1108.3154*, 2011.

Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey J Gordon, and J.Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Stephane Ross, Jiaji Zhou, Yisong Yue, Debadeepta Dey, and Drew Bagnell. Learning policies for contextual submodular prediction. In *ICML*, 2013.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering, 1994.

Ankan Saha, Prateek Jain, and Ambuj Tewari. The Interplay Between Stability and Regret in Online Learning. *arXiv preprint arXiv:1211.6158*, pages 1–19, 2012.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Robert E. Schapire and Manfred K. Warmuth. On the worst-case analysis of temporal-difference learning algorithms. *Machine Learning*, 22(1):95–121, 1996. ISSN 0885-6125. doi: 10.1007/BF00114725.

Bruno Scherrer. Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. *International Conference on Machine Learning (ICML 2010)*, 2010.

Bruno Scherrer. Approximate policy iteration schemes: a comparison. In *International Conference on Machine Learning*, pages 1314–1322, 2014.

Ralf Schoknecht and Artur Merke. TD(0) Converges Provably Faster than the Residual Gradient Algorithm. In *International Conference on Machine Learning (ICML 2003)*, pages 680–687, 2003.

Bernhard Scholkopf and Alexander J. Smola. In *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press Cambridge, MA, USA, 2001.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. *Foundations and TrendsÂő in Machine Learning*, 4(2):107–194, 2011.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 2012.

Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory*, pages 3–24, 2013.

Ohad Shamir. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research*, 18(52):1–11, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Zhao Song and Wen Sun. Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*, 2019.

Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, Gert RG Lanckriet, et al. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

Wen Sun and J Andrew Bagnell. Online bellman residual and temporal difference algorithms with predictive error guarantees. 2016.

Wen Sun and J. Andrew (Drew) Bagnell. Online Bellman Residual Algorithms with Predictive Error Guarantees. In *The 31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*, July 2015.

Wen Sun, Roberto Capobianco, Geoffrey J Gordon, J Andrew Bagnell, and Byron Boots. Learning to smooth with bidirectional predictive state inference machines. In *UAI*, 2016a.

Wen Sun, Jur van den Berg, and Ron Alterovitz. Stochastic extended lqr for optimization-based motion planning under uncertainty. *IEEE Transactions on Automation Science and Engineering*, 13(2):437–447, 2016b.

Wen Sun, Arun Venkatraman, Byron Boots, and J Andrew Bagnell. Learning to filter with predictive state inference machines. In *ICML*, 2016c.

Wen Sun, Debadeepta Dey, and Ashish Kapoor. Safety-aware algorithms for adversarial contextual bandit. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3280–3288. JMLR. org, 2017a.

Wen Sun, Niteesh Sood, Debadeepta Dey, Gireeja Ranade, Siddharth Prakash, and Ashish Kapoor. No-regret replanning under uncertainty. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6420–6427. IEEE, 2017b.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *ICML*, 2017c.

Wen Sun, James Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning and imitation learning. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR)*, 2018a.

Wen Sun, Alina Beygelzimer, Hal Daumé III, John Langford, and Paul Mineiro. Contextual memory trees. *arXiv preprint arXiv:1807.06473*, 2018b.

Wen Sun, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Dual policy iteration. *arXiv preprint arXiv:1805.10755*, 2018c.

Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based reinforcement learning in contextual decision processes. *arXiv preprint arXiv:1811.08540*, 2018d.

Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *ICML*, 2019.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998a.

Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998b.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

RichardS. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. In *Advances in neural information processing systems*, pages 2253–2261, 2010.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *ICML*, 2008.

István Szita and András Lörincz. Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941, 2006.

István Szita and Csaba Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038, 2010.

Aviv Tamar, Panos Toulis, Shie Mannor, and Edoardo M. Airoldi. Implicit Temporal Differences. *arXiv:1412.6734*, pages 1–6, 2014.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. *Deep Reinforcement Learning Workshop, NeurIPS*, 2018.

Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1069–1074. IEEE, 2011.

Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306. IEEE, 2005.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Stephen Tu and Benjamin Recht. The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. *arXiv preprint arXiv:1812.03565*, 2018.

Anirudh Vemula, Wen Sun, and J Andrew Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. *arXiv preprint arXiv:1901.11503*, 2019.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. *AAAI*, 2015.

Arun Venkatraman, Wen Sun, Martial Hebert, J Andrew Bagnell, and Byron Boots. Online instrumental variable regression with applications to online linear system identification. In *AAAI*, pages 2101–2107, 2016a.

Arun Venkatraman, Wen Sun, Martial Hebert, Byron Boots, and J Andrew Bagnell. Inference machines for nonparametric filter learning. 2016b.

Arun Venkatraman, Nicholas Rhinehart, Wen Sun, Lerrel Pinto, Martial Hebert, Byron Boots, Kris Kitani, and J Bagnell. Predictive-state decoders: Encoding the future into recurrent networks. In *Advances in Neural Information Processing Systems*, pages 1172–1183, 2017.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems*, pages 3021–3029, 2013.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, pages 262–270, 2011.

Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*, 2003.