Title
Efficient Reinforcement Learning in Various Environments: from the Idealized to the Realistic

Author
Feng, Fei

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Efficient Reinforcement Learning in Various Environments: from the Idealized to the Realistic

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Fei Feng

2021

ABSTRACT OF THE DISSERTATION

Efficient Reinforcement Learning in Various Environments:
from the Idealized to the Realistic

by

Fei Feng
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2021
Professor Wotao Yin, Co-Chair
Professor Lin Yang, Co-Chair

How to achieve efficient reinforcement learning in various training environments is a central challenge in artificial intelligence. This thesis investigates this question on the spectrum of environments from the most idealized type to a fairly realistic one. We use two characteristics to describe the complexity of an environment: 1. how many observations it contains; 2. how difficult it is to capture high rewards. Based on these two scales, we study four types of environments: 1. finite (a small number of) observations plus a generative model (one of the most idealized sample oracles); 2. finite observations plus an approximate model; 3. rich (possibly infinitely many) but structured observations with an online simulation model; 4. general rich observations with an online simulation model. From the first to the last, the problem becomes more and more difficult and significant to solve. This thesis provides novel algorithms/analyses for each setting to improve both statistical and computational efficiency upon prior work.

The dissertation of Fei Feng is approved.

Deanna Needell

Lieven Vandenberghe

Luminita Vese

Lin Yang, Committee Co-Chair

Wotao Yin, Committee Co-Chair

University of California, Los Angeles

2021

To my parents and Bjoern

for their unconditional love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deep gratitude to both of my advisors Wotao Yin and Lin Yang. Wotao led me into research when I was still an undergraduate. He gave me advise on all perspectives: how to propose interesting questions, how to formulate ideas, how to give talks, how to write papers, etc. He trained me as a whole-package eligible researcher and gave me the freedom to pursue my interests. Lin provided me the greatest instruction on my research of reinforcement learning. He guided me towards the central challenges and worked with me to solve them step by step. He is brilliant and persistent on attacking problems. He showed me the real meaning of keep-thinking and never-give-up. Without Wotao and Lin, I would never obtain my Ph.D. this smoothly. They both become fathers recently. I believe their daughters will be proud of them.

I also wanna give warm thanks to all my collaborators: Alekh Agarwal, Simon Du, Robert Hannah, Yanli Liu, Ruosong Wang, and Yibo Zeng. All of my works contain their talents and efforts. Many thanks to my committee members: Deanna Needell, Lieven Vandenberghe, and Luminita Vese, who have provided extremely useful discussion. I am also grateful to my internship mentors: Tony Qin at DiDi and Hongyi Zhang at Bytedance. They stimulated my interests towards practical algorithms for real-world applications.

Many thanks to my colleagues at UCLA: Hanqin Cai, Robert Hannah, Bumsu Kim, Daniel Mckenzie, Howard Heaton, Qiuwei Li, Jialin Liu, Siting Liu, Yanli Liu, Ernest Ryu, Yuejiao Sun, and Tianyu Wu. Our weekly group meeting opened my eyes to different areas in Mathematics.

I want to express my love to my family and all my dear friends for their company and support in the past five years. Especially, I want to thank Bjoern Bringmann. He is my best friend and partner. We started and finished our Ph.D. together. We share both happiness and frustration along the way of growing-up. My journey through graduate school would not have been as much of a pleasure without him.

VITA

2016 B.S. (Mathematics), Fudan University.

2018 Candidate in Philosophy in Mathematics, University of California, Los Angeles.

2020 Girsky Fellowship, Department of Mathematics, University of California, Los Angeles.

PUBLICATIONS

Yibo Zeng, Fei Feng, and Wotao Yin. AsyncQVI: Asynchronous-Parallel Q-Value Iteration for Discounted Markov Decision Processes with Near-Optimal Sample Complexity. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR 108:713-723, 2020.*

Fei Feng, Wotao Yin, and Lin F. Yang. How Does An Approximate Model Help in Reinforcement Learning? *arXiv preprint arXiv:1912.02986*

Fei Feng, Ruosong Wang, Wotao Yin, Simon S. Du, and Lin F. Yang. Provably Efficient Exploration for Reinforcement Learning Using Unsupervised Learning. In *Advances in Neural Information Processing Systems, Volumn 33, 2020. Accepted as Spotlight.*

Fei Feng, Wotao Yin, Alekh Agarwal, and Lin F. Yang. Provably Correct Optimization and Exploration with Non-linear Policies. *arXiv preprint arXiv:2103.11559*

CHAPTER 1

Introduction

In machine learning field, there are three learning paradigms: supervised learning, unsupervised learning, and reinforcement learning (RL). Supervised learning and unsupervised learning focus on making predictions from a given batch of data where all samples are typically assumed to be i.i.d. generated from some fixed unknown distribution. The learning process and the output predictor will not change the sample distribution. While RL is about learning through interacting with the environment. The learner starts with no prior knowledge of the environment but can explore it with various actions. For every time step, the learner observes the current state of the environment. Based on that observation, it selects an action to act back on the environment. As a result, the environment transitions to a next state and returns a reward signal to the learner. The value of a reward reflects how favorable the corresponding action is at that state (e.g., high rewards mean achieving some tasks). Through trial-and-error, the learner gathers information of the environment and gradually constructs wise decision-making strategies to collect high rewards. The major difference between RL and the other two paradigms is that the learner's behavior influences the distribution of the observations (since the transition is action-related). In other words, it learns a world in a way that learning changes how the world is presented.

RL is a powerful and general framework since it does not require prior knowledge or human supervision. It summarizes many machine learning algorithms and captures a broad spectrum of topics, including health care, traffic control, and experimental design [SBW92, ERR19, SW01, Wie00]. Recently, RL achieves phenomenal empirical successes, e.g.,

AlphaGo [SHM16] defeated the best human player in Go after being trained with an RL algorithm ; OpenAI used RL to precisely and robustly control a robotic arm [AWR17]; agents are trained to defeat world champions in a video game Dota 2 [BBC19]. These achievements (although in games) showcase the potential of RL in replacing or even surpassing human expertise in real-world application.

However, some serious challenges lead to high sample and computational complexities in RL training and therefore, prohibit it being widely and successfully applied. For example, both the AlphaGo and OpenAI arm took nearly a thousand years of human-equivalent experiences to obtain good performances. The reasons of such pains are mainly two-fold:

- *the richness of observations*, i.e., there are too (possibly infinitely) many observations the environment contains. For instance, in the game of Go, there are in total 3^{361} states of the board (361 positions and each position can be none/white/black). The space could be even larger with e.g., high-dimensional images. How to efficiently distinguish among states and process their individual information is challenging.

- *the hardness of knowledge accessing*, i.e., it can be very difficult to capture high rewards. In real-world applications, high rewards can hide in very deep levels. For example, in navigation problems, the learner may need to take a long path to reach the goal; in the game of Go, the learner needs to make a long sequence of correct actions to win eventually. Things could be worse if the reward function is sparse, which provides no useful information to guide the learner towards the final goal. Whenever this occurs, one solution is general wide exploration, i.e., the learner should visit as many states as possible so as to identify high rewards. This is the well-known *exploration problem*. However, wide exploration is non-trivial to realize since: 1. as we mentioned before, the state distribution is influenced by previous actions and a few bad actions can result in trapping in a local area (e.g., keeps going against the wall in maze) or even a complete failure of the task (e.g., crashes into a car in self-driving); 2. naive random exploration

strategy (i.e., for each step, with a small probability the learner uniformly randomly selects an action) might have an exponential sample complexity, which is infeasible for practical use. Fortunately, if the state space is small, the problem is readily solved by logging the number of visitations N_{visit} to each state in memory and designing artificial rewards $\propto \frac{1}{\sqrt{N_{\text{visit}}}}$. This approach is referred to as *exploration with a tabular implementation* and is proved to have minimax-optimal sample complexity (see e.g. [KS02a, BT02, JOA10, JAB18]). However, how to achieve efficient exploration when there are numerous observations is still a central challenge.

This thesis aims for providing novel RL algorithms with better statistical and computational efficiency than prior work for various environments ranging from the most idealized setting to a realistic scenario. Specifically, we study four types:

1. **finite observations plus a generative model**. In this setting, the environment only contains a small number of observations such that maintaining information for all states with a tabular implementation is possible. A generative model (formal definition can be found in Chapter 2) is provided such that the learner can conveniently access the environment and are exempt from the exploration challenge. Under this setting, prior work has already established minimax optimal sample complexity [AGK12, SWW18b, AKY19]. Thus we are mainly interested in how to achieve better computational efficiency.

2. **finite observations plus an approximate model**. Compared with the first setting, we remove the access to a generative model but replace it with an approximate model. So the learner needs to design good decision-making strategies from approximately correct prior knowledge without interacting with the real environment. This setting abstracts scenarios when agents are trained with simulators before deploying to the real world. A fundamental question is how much does an approximate help? We provide systematic answers to this question.

3. **rich but structured observations with an online simulation model**. In the first two settings, we deal with simple observations and no exploration is required. For the third case, we escalate the problem difficulty to directly attacking exploration with rich observations. To make it solvable, we assume the observation space has some intrinsic low-dimensional structure but is not directly given to the learner. An unsupervised learning oracle and an online simulation model are provided to the agent to learn the low-dimensional structure and interact with the environment, respectively.

4. **general rich observations with an online simulation model.** In the last case, we resolve exploration with general rich observations where no structure assumption is imposed. This is the most challenging task among the four but is also the closest to real-world applications.

We want to emphasize that all four types can occur in practice. Our works complete all scenarios and practitioners can select the most suitable approach case by case.

1.1 Main Contributions

From Chapter 2 to Chapter 5, each chapter presents a work studying one setting. We summarize the main contributions as below.

1.1.1 Parallel RL with a Generative Model

In the first setting, we propose an asynchronous-parallel RL algorithm, AsyncQVI. AsyncQVI is the *first* asynchronous-parallel RL algorithm with an explicit sample complexity result which is near-optimal. Our algorithm not only enjoys parallel acceleration but also has a memory complexity that is much smaller than related work. These characteristics make AsyncQVI computational efficient and therefore, suitable for large-scale applications. In our numerical tests, AsyncQVI achieves a $\sim 10\times$ speedup compared with peer methods.

1.1.2 Transfer RL with an Approximate Model

In this work, we provide the *first* systematic answer towards how much an approximate model can help. We use TV-distance as a measure of similarity among models and prove both upper and lower sample complexity bounds. These results bring significant insights about the benefit and limitation of an approximate model. For example, we show that if the given approximate model is sufficiently accurate, it helps with removing sub-optimal actions and scale down the decision selection space; otherwise, it might be of no use for a near-optimal policy. These results provide theoretical guidance in the practice of transfer learning.

1.1.3 RL Exploration with Unsupervised Learning

In this work we study exploration with rich observations generated from a small number of latent states. We use an unsupervised learning subroutine to learn the low-dimensional structure. Our contributions include:

- We propose a new algorithmic framework for the Block Markov Decision Process (BMDP) model [DKJ19]. We combine an unsupervised learning oracle and a tabular RL algorithm in an organic way to find a near-optimal policy for a BMDP. The unsupervised learning oracle is an abstraction of methods used in [THF17, BSO16] and widely used statistical generative models. Notably, our framework can take almost *any* unsupervised learning algorithms and tabular RL algorithms as subroutines.

- Theoretically, we prove that as long as the unsupervised learning oracle and the tabular RL algorithm each has a polynomial sample complexity guarantee, our framework finds a near-optimal policy with sample complexity polynomial in the number of latent states, which is significantly smaller than the number of possible observations (cf. Theorem 4.3.1). To our knowledge, this is the *first* provably efficient method for RL problems with huge observation spaces that uses unsupervised learning for exploration. Furthermore, our result does not require additional assumptions on transition dynamics

as used in [DKJ19]. Our result theoretically sheds light on the success of the empirical paradigms used in [THF17, BSO16].

- We instantiate our framework with particular unsupervised learning algorithms and tabular RL algorithms on hard exploration environments with rich observations studied in [DKJ19], and compare with other methods tested in [DKJ19]. Our experiments demonstrate our method can significantly outperform existing methods on these environments.

1.1.4 RL Exploration with General Function Approximation

In the last setting, we adopt policy optimization methods together with general (possibly nonlinear) function approximation to address exploration problem with rich observations. Our contributions are summarized as below:

- We design ENIAC, an actor-critic method that allows non-linear function approximation in the critic with a statistical guarantee. This is the *first* theoretical justification for the utilization of general function approximation in policy optimization to solve the exploration problem. Our method is robust to model misspecification and strictly extends existing works on linear function approximation.

- We also develop some computational optimizations of our approach with slightly worse sample complexity, and an empirical adaptation building on existing deep RL tools.

- We empirically evaluate this adaptation, and show that it outperforms prior heuristics inspired by linear methods, establishing the value in correctly reasoning about the agent's uncertainty under non-linear function approximation.

1.2 Background

In this section, we provide background knowledge of RL including the underlying mathematical framework: Markov Decision Processes (MDPs) and a notion to measure statistical efficiency of algorithms: PAC-learnability. The purpose of this section is to give a brief introduction of the model and statistical tools that we will use throughout the entire thesis. Each chapter is self-contained with more detailed definition of variants that are tailored to specific problems considered therein.

Notation Given a set $\mathcal{A}$, we denote by $|\mathcal{A}|$ its cardinality and $\Delta(\mathcal{A})$ the set of all probability distributions over $\mathcal{A}$. For a positive integer H, we use $[H]$ for the set $\{1, 2, \ldots, H\}$. Let $\mathcal{E}$ be an event, we denote by $\Pr(\mathcal{E})$ the probability that $\mathcal{E}$ occurs. Let X be a random variable, we denote by $X \sim q(\cdot)$ if X follows the density $q(\cdot)$ and $\mathbb{E}[X]$ and $\mathbb{V}[X]$ the expectation and variance of X, respectively. We use O, Ω, and Θ to denote leading orders in upper, lower, and minimax bounds, respectively; and we use $\widetilde{O}, \widetilde{\Omega}$ and $\widetilde{\Theta}$ to hide the polylog factors.

1.2.1 Markov Decision Processes

Finite-horizon MDPs A finite-horizon MDP is described as $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, H, \rho_0)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P} := \{p_h(\cdot|s,a) \in \Delta(\mathcal{S}), (s,a) \in \mathcal{S}\times\mathcal{A}, h \in [H]\}$ specifies transition probabilities for each step, $\mathcal{R} := \{r_h(s,a) \in [0,1], (s,a) \in \mathcal{S} \times \mathcal{A}, h \in [H]\}$ specifies reward functions for each step, H is the length of horizon, and ρ_0 denotes the initial distribution.

The agent interacts with the environment in episodes. For each episode, the agent starts from an initial state $s_1 \sim \rho_0(\cdot)$ and takes H steps to terminate. For each step $h \in [H]$, the agent observes the state of the environment $s_h \in \mathcal{S}$ and selects an action $a_h \in \mathcal{A}$. As a result, the environment evolves into a new state $s_{h+1} \sim p_h(\cdot|s_h, a_h)$ and the agent receives an instant reward $r_h(s_h, a_h)$.

The above process describes sequential decision-making. How the agent selects actions can be formulated as a policy $\pi : [H] \times \mathcal{S} \to \Delta(\mathcal{A})$, i.e. a mapping that based on the step index maps every state to a probability distribution over the action space. Given a policy π, we can define its value functions for every step:

$$V_h^\pi(s) := \mathbb{E}^\pi \left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \;\middle|\; s_h = s \right], \quad h \in [H],$$

i.e., the expected cumulative rewards starting from that step till the end of the episode following π. Similarly, we can define the action-value functions:

$$Q_h^\pi(s, a) := \mathbb{E}^\pi \left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \;\middle|\; s_h = s, a_h = a \right], \quad h \in [H].$$

In particular, we have $V_h^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \cdot Q_h^\pi(s, a)$. The objective of RL is to

$$\underset{\pi \in \Pi}{\text{maximize}} \;\; \mathbb{E}_{s_1 \sim \rho_0(\cdot)} V_1^\pi(s_1)$$

without full knowledge of $\mathcal{P}$ and $\mathcal{R}$. We denote by π^* an optimal policy and V^* the optimal value, i.e., $V^* := \max_{\pi \in \Pi} \mathbb{E}_{s_1 \sim \rho_0(\cdot)} V_1^\pi(s_1)$. A policy π is ε-optimal if $\mathbb{E}_{s_1 \sim \rho_0(\cdot)} V_1^\pi(s_1) \geq V^* - \varepsilon$ for $\varepsilon > 0$.

Infinite-horizon Discounted MDPs An infinite-horizon discounted MDP is described as $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p(s'|s, a)$ is the transition kernel, $r(s, a) \in [0, 1]$ is the reward function, and $\gamma \in (0, 1)$ is a discount factor.

At step t, the agent observes the current state of the environment s_t and selects an action a_t according to some policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$[1]. As a result, the environment transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$ and returns an instant reward $r(s_t, a_t)$ to the agent.

Given a policy π, we define its value function as:

$$V^\pi(s) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 = s \right], \tag{1.1}$$

[1]In infinite-horizon setting, policy is normally independent of the step t.

i.e., the expected discounted cumulative rewards following π. Similarly, the action-value function is defined as

$$Q^\pi(s,a) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a \right] \tag{1.2}$$

In particular, we have

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \cdot Q^\pi(s,a), \quad \forall s \in \mathcal{S} \\ Q^\pi(s,a) &= \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \big(r(s,a) + \gamma V^\pi(s') \big), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

In this model, the objective of RL is to

$$\underset{\pi \in \Pi}{\text{maximize}} \; V^\pi(s), \quad \forall s \in \mathcal{S},$$

without full knowledge of p and r. We denote by π^* an optimal policy and V^* and Q^* the optimal value and action-value functions respectively. It can be shown that

$$\begin{aligned} V^*(s) &\geq V^\pi(s), \;\; \forall s \in \mathcal{S}, \pi \in \Pi, \\ Q^*(s,a) &\geq Q^\pi(s,a), \;\; \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \pi \in \Pi. \end{aligned}$$

For $\varepsilon > 0$, a policy π is ε-optimal if $V^\pi(s) > V^*(s) - \varepsilon, \; \forall s \in \mathcal{S}$.

Policy-induced Distribution In infinite-horizon discounted MDPs, every policy corresponds to a distribution over the state-action space. Specifically, given π we define:

$$d^\pi_{\tilde{s}}(s,a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = a | s_0 = \tilde{s}),$$

where $\mathrm{Pr}^\pi(s_t = s, a_t = a | s_0 = \tilde{s})$ denotes the probability of reaching (s,a) at the t_{th} step starting from $\tilde{s}$ following π. Similarly, we define $d^\pi_{\tilde{s},\tilde{a}}(s,a)$ if the agent starts from the state-action pair $(\tilde{s},\tilde{a})$ and follows π thereafter. For any distribution $\nu \in \Delta(\mathcal{S} \times \mathcal{A})$, we denote by $d^\pi_\nu(s,a) := \mathbb{E}_{(\tilde{s},\tilde{a}) \sim \nu} \left[d^\pi_{(\tilde{s},\tilde{a})}(s,a) \right]$. Based on the policy-induced distribution, we can rewrite the

Algorithm 1 Value Estimators

Routine: V^π-`ESTIMATOR`

  Input: starting state s.

  Execute π from s; at any step t with (s_t, a_t), terminate with probability $1-\gamma$.

  Return: $\hat{V}^\pi(s) = \sum_{i=0}^{t} r(s_i, a_i)$, where $s_0 = s$.

Routine: Q^π-`ESTIMATOR`

  Input: starting state-action (s, a).

  Execute π from (s, a); at any step t with (s_t, a_t), terminate with probability $1-\gamma$.

  Return: $\hat{Q}^\pi(s, a) = \sum_{i=0}^{t} r(s_i, a_i)$, where $(s_0, a_0) = (s, a)$.

Algorithm 2 d^π Sampler

Routine: d_ν^π-`SAMPLER`

  Input: $\nu \in \Delta(\mathcal{S} \times \mathcal{A}), \pi$.

  Sample $s_0, a_0 \sim \nu$;

  Execute π from s_0, a_0; at any step t with (s_t, a_t), terminate with probability $1-\gamma$.

  Return: (s_t, a_t).

value functions as:

$$V^\pi(\tilde{s}) = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} d^\pi_{\tilde{s}}(s, a) \cdot r(s, a)/(1-\gamma), \tag{1.3}$$
$$Q^\pi(\tilde{s}, \tilde{a}) = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} d^\pi_{\tilde{s},\tilde{a}}(s, a) \cdot r(s, a)/(1-\gamma).$$

Comparing formulations (1.1) and (1.2) with (1.3), one can see the difference is where the summation over time steps is taken: either in rewards or in probabilities. We provide sample oracles for V^π, Q^π, and d^π_ν in Algorithm 1 and 2.

1.2.2 Dynamic Programming

If the transitions and the rewards are known to us, then an optimal policy of an MDP (both infinite and finite-horizon) can be derived by dynamic programming. In the sequel, we introduce three approaches. One is referred to e.g., [Put14, SB18] for more details. It can be summarized that RL algorithms are simulations of these three approaches with stochastic estimation and function approximation.

Value Iteration In finite-horizon MDPs, the value iteration starts from the last step and propagates forward. Let $Q_H^*(s,a) := r_H(s,a)$. The value iteration proceeds as:

$$\begin{aligned} V_h^*(s) &= \max_{a\in\mathcal{A}} Q_h^*(s,a) \\ Q_{h-1}^*(s,a) &= r_{h-1}(s,a) + \sum_{s'\in\mathcal{S}} p_{h-1}(s'|s,a)\cdot V_h^*(s'), \quad h = H, H-1, \ldots, 2. \end{aligned}$$

Then we can construct an optimal policy as $\pi_h^*(s) \in \operatorname{argmax}_{a\in\mathcal{A}} Q_h^*(s,a), h \in [H]$.

For infinite-horizon MDPs, the value iteration can be defined using the Bellman operator $\mathcal{T} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$. Specifically, starting from an arbitrary vector $V_0 \in \mathbb{R}^{|\mathcal{S}|}$,

$$V_n(s) = (\mathcal{T}V_{n-1})(s) := \max_{a\in\mathcal{A}} \Big(r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)\cdot V_{n-1}(s') \Big), \ \forall s \in \mathcal{S}. \tag{1.4}$$

Recall that $\gamma \in (0,1)$. $\mathcal{T}$ is indeed a γ-contraction under $\|\cdot\|_\infty$. Therefore, there exists a unique fixed point V^*. From V^* we can construct an optimal policy π^* by letting

$$\pi^*(s) := \operatorname*{argmax}_{a\in\mathcal{A}} \Big(r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)\cdot V^*(s') \Big).$$

It can be easily shown that $V^*(s) = V^{\pi^*}(s)$ for all $s \in \mathcal{S}$, i.e., the fixed point of the Bellman operator is the optimal value function.

For both models, there can be multiple optimal policies but the optimal value/action-value function is unique.

Policy Iteration For infinite-horizon MDPs, one can also use policy iteration to construct an optimal policy[2]. In contrast to value iteration which iterates over value functions, policy iteration works by updating policies. Starting from an arbitrary policy $\pi_0 : \mathcal{S} \to \Delta(\mathcal{A})$, we perform the following update:

$$\pi_{n+1}(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot V^{\pi_n}(s') \Big), \ \forall s \in \mathcal{S}. \tag{1.5}$$

To obtain V^{π_n}, we can run the following iteration:

$$V_n(s) := \sum_{a \in \mathcal{A}} \pi_n(a|s) \cdot \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot V_{n-1}(s') \Big), \ \forall s \in \mathcal{S}. \tag{1.6}$$

As in value iteration, the update in Equation (1.6) is also a γ-contraction under $\| \cdot \|_\infty$ and V^{π_n} is the fixed point. One can compute V^{π_n} either by running the above iteration for enough many steps or can plug in V^{π_n} on both side of (1.6) and solve as a linear system. It can be proved that in Equation (1.5), π_n converges to an optimal policy.

Remark 1. *Notice that value iteration and policy iteration proceed differently. Value iteration works in the function space without forming intermediate policies. V_n in Equation* (1.4) *is not necessarily the value of some policy but $\lim_{n\to\infty} V_n = V^*$. While in policy iteration, policies are explicitly constructed and improved and V^{π_n} converges to V^*.*

Linear Programming The third way of solving an infinite-horizon MDP is linear programming[3]. It can be shown that if $q(s) > 0$ for all $s \in \mathcal{S}$, the optimal value function V^* is the solution of the following problem:

$$\begin{aligned} &\operatorname*{minimize}_{V \in \mathbb{R}^{|\mathcal{S}|}} \ \sum_{s \in \mathcal{S}} q(s) \cdot V(s) \\ &\text{subject to } V(s) \geq r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot V(s'), \ \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned} \tag{1.7}$$

[2]Policy iteration for finite-horizon MDPs is almost the same as value iteration.

[3]Linear programming is not suggested for finite-horizon MDPs since its value function is step-index related, which makes the formulation complicated in a linear programming formulation.

The dual problem of (1.7) is

$$\begin{aligned} \underset{\mu\in\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}}{\text{maximize}} \quad & \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mu(s,a)\cdot r(s,a) \\ \text{subject to} \quad & \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \gamma\cdot\mu(s',a')\cdot p(s|s',a') + q(s) = \sum_{a'\in\mathcal{A}} \mu(s,a'), \\ & \mu(s,a)\geq 0, \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A}. \end{aligned} \tag{1.8}$$

A fundamental result is that there exists a one-to-one and onto mapping between the feasible set of (1.8) and the policy space [Put14, Cor. 6.9.2] and that a maximizer μ^* corresponds to an optimal policy π^* given the rewards are uniformly bounded [Put14, Thm. 6.9.4].

Remark 2. *All dynamic programming methods require full knowledge of the transitions and the rewards. In RL, since no full knowledge is available, some sample oracle is provided as information source for the agent to grasp knowledge and learn. In each coming chapter, we will specify the detailed working mechanism of a required sample oracle.*

1.2.3 PAC-learnability

In this section, we introduce the notion of *Probably Approximately Correct* (PAC) learning, a generally adopted framework to measure the statistical efficiency of machine learning algorithms, which was proposed in [Val84]. In RL, the PAC-learnability of an algorithm $\mathscr{A}$ is reflected via the following question:

How many transition samples does $\mathscr{A}$ take to learn an ε-optimal policy with probability at least $1-\delta$?

Here the high probability reasoning is due to the randomness in both the training data and the algorithm. Throughout this thesis, we measure the statistical efficiency of algorithms via answering the above question. In the sequel, we list some often-used concentration inequalities to establish sample complexity results.

Theorem 1.2.1 (Hoeffding's Inequality [Hoe63]). *Let $X_1, \cdots, X_n$ be mean-zero independent real-valued random variables with $X_j \in [a_j, b_j]$ and $Y := \frac{1}{n}\sum_{j=1}^n X_j$. For any $\epsilon \geq 0$,*

$$\Pr\Big(\big|Y\big| \geq \epsilon\Big) \leq 2\exp\Big(\frac{-2n^2\epsilon^2}{\sum_{j=1}^n (b_j - a_j)^2}\Big).$$

Hoeffding's inequality is built upon value bounds on random variables. If we also know the variance of each random variable, the concentration can be established via Bernstein's inequality.

Theorem 1.2.2 (Bernstein's Inequality). *Let $X_1, \cdots, X_n$ be mean-zero independent real-valued random variables with $|X_i| \leq c$ for all i and $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \mathbb{V}[X_i]$. Denote by $Y := \frac{1}{n}\sum_{i=1}^n X_i$. For any $\epsilon \geq 0$,*

$$\Pr\Big(\big|Y\big| \geq \epsilon\Big) \leq 2\exp\Big(\frac{-2n\epsilon^2}{2\sigma^2 + 2c\epsilon/3}\Big).$$

When variances are small, Bernstein's inequality is sharper than Hoeffding's inequality.

The above inequalities are about independent random variables. In the following, we have a concentration result for Martingales.

Theorem 1.2.3 (Azuma-Hoeffding Inequality). *Suppose $X_0, X_1, \ldots, X_k, \ldots$ is a Martingale and $|X_k - X_{k-1}| \leq c_k$ almost surely. Then for any positive integers N and any $\epsilon > 0$, it holds that*

$$\Pr\Big(\big|X_N - X_0\big| \geq \epsilon\Big) \leq 2\exp\Big(\frac{-\epsilon^2}{2\sum_{k=1}^N c_k^2}\Big).$$

We will also use Union Bound throughout the thesis. Specifically, given a countable set of events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_i, \ldots$, we have

$$\Pr(\cup_{i=1}^\infty \mathcal{E}_i) \leq \sum_{i=1}^\infty \Pr(\mathcal{E}_i).$$

CHAPTER 2

Parallel RL with a Generative Model

In this chapter, we study how to achieve better computational efficiency given an ideal sample oracle in simple observation spaces. We utilize asynchronous parallel technique to accelerate training. Our algorithm, AsyncQVI, not only enjoys parallel speedup but also is the first asynchronous-parallel RL algorithm with a near-optimal sample complexity result and has a memory complexity which is independent of the number of actions. These characteristics make AsyncQVI suitable for large-scale applications. In numerical tests, we compare AsyncQVI with four related methods. The results show that our algorithm is highly efficient and achieves linear parallel speedup.

The contributions in this chapter were first presented in the joint work with Yibo Zeng and Wotao Yin and was published in AISTATS 2020 [ZFY20].

2.1 Introduction

Markov Decision Processes (MDPs) are a fundamental model to encapsulate sequential decision making under uncertainty. They have been intensively studied and successfully applied to many fields, especially Reinforcement Learning (RL). As a rapidly developing area of artificial intelligence, RL is being flourishingly combined with deep neural network [MKS15, MBM16, Li17] and used in many domains including games [MKS15, SHM16], robotics [KBP13], natural language processing [YHP18], finance [DBK16], healthcare [KM15] and so on. With the advent of big-data applications, computational costs have increased significantly. Therefore, parallel

computing techniques have been applied to reduce RL solving time [GK08, NSB15]. Recently, asynchronous (async) parallel algorithms have been widely researched in RL and gained empirical successes [MBM16, BFT16, GHL17, SA18, ZCT19]. Compared to synchronous (sync) parallel algorithms, where the agents must wait for the slowest agent to finish its task before they can all proceed to the next one, async-parallel algorithms allow agents to run continuously with little idling. Hence, async-parallel algorithms complete more tasks than their synchronous counterparts (though information delays and inconsistencies may negatively affect the task quality). Async-parallel algorithms have other advantages [BT91]: the system is more tolerant of computing faults and communications glitches; it is also easy to incorporate new agents.

In contrast to promising empirical results in async-parallel RL, its theoretical property has not been fully understood. In this paper, we try to mitigate the gap between theory and practice. Specifically, we assume the underlying model is a Discounted Infinite-Horizon Markov Decision Processes (DMDPs) and study the sample complexity of RL with async-parallel acceleration. A DMDP is described by a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \gamma)$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, P contains the transition probabilities, r is the collection of instant rewards, and $\gamma \in (0,1)$ is a discounted factor. At time step t, the controller or the decision maker observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to a policy π, where π maps a state to an action. The action leads the environment to a next state s_{t+1} with probability $p^{a_t}_{s_t s_{t+1}}$. Meanwhile, the controller receives an instant reward $r^{a_t}_{s_t s_{t+1}}$. Here, $r^{a_t}_{s_t s_{t+1}}$ is a deterministic value given the transitional instance (s_t, a_t, s_{t+1}). If only s_t and a_t are specified, $r^{a_t}_{s_t}$ is a random variable and $r^{a_t}_{s_t} = r^{a_t}_{s_t s_{t+1}}$ with probability $p^{a_t}_{s_t s_{t+1}}$. We use notation $\mathbf{p}^a_i := [p^a_{i1}, p^a_{i2}, \cdots, p^a_{i|\mathcal{S}|}]^\top$ and $\bar{r}^a_i := \sum_{j\in\mathcal{S}} p^a_{ij} r^a_{ij}$ and assume, without loss of generality, $r^a_{ij} \in [0,1]$, $\forall\, i, j \in \mathcal{S}, a \in \mathcal{A}$. Given a policy $\pi : \mathcal{S} \to \mathcal{A}$, we denote $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ the state-value vector of π. Specifically,

$$\mathbf{v}^\pi := [v^\pi_1, v^\pi_2, \cdots, v^\pi_{|\mathcal{S}|}]^\top,\ v^\pi_i := \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r^{a_t}_{s_t s_{t+1}} | s_0 = i\big],$$

where the expectation is taken over the trajectory $(s_0, a_0, s_1, a_1, \ldots, s_t, a_t \ldots)$ following π,

i.e. $a_t = \pi_{s_t}$. The objective is to seek for an optimal policy π^* such that $\mathbf{v}^\pi$ is maximized component-wisely. We let $\mathbf{v}^*$ denote the optimal value vector associated with an optimal policy π^*. A policy π is ε-optimal if $\|\mathbf{v}^* - \mathbf{v}^\pi\|_\infty \leq \varepsilon$.

Since no full knowledge of P or r is available, we assume access to a *generative model*. A generative model takes any state-action pair (i, a) as input and outputs a next state j with probability p_{ij}^a and the corresponding instant reward r_{ij}^a. Through drawing samples from a generative model, we collect information of the model and learn a good policy.

In this paper, we propose an algorithm Asynchronous-Parallel Q-Value Iteration (AsyncQVI), which is the first async-parallel RL algorithm with a sample complexity result. AsyncQVI returns an ε-optimal policy with probability at least $1 - \delta$ using

$$\widetilde{O}\Big(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^5\varepsilon^2}\Big)$$

samples, provided that each coordinate is updated at least once within $O(|\mathcal{S}||\mathcal{A}|)$ time and the async delay is bounded by $O(|\mathcal{S}||\mathcal{A}|)$. [SWW18b] established the sample complexity lower bound as $\Omega\Big(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\varepsilon^2}\Big)$.Our result nearly matches the lower bound up to $(1-\gamma)^2$ and logarithmic factors. Besides, AsyncQVI requires only $O(|\mathcal{S}|)$ memory, which is the minimal in tabular implementation. With a near-optimal sample complexity, the minimal memory requirement, and asynchronous-parallel acceleration, AsyncQVI is a competitive RL algorithm.

Notation We write a scalar in *italic* type, a vector or a matrix in **boldface**, and their components with subscripts. For example, $\mathbf{v}$ and v_i are a vector and its ith component, respectively.

2.1.1 Related Work

AsyncQVI is not the first attempt to solve DMDP problems with asynchronous parallel. As early as in [BT89], the authors proposed async-parallel dynamic programming methods.

Table 2.1: Related Async-parallel Methods For DMDPs.

Algorithms	**Assumption**	**Delay**	**Sample**	**Memory**
Total-async QVI [BT89]	Full knowledge	Unbounded	N/A	$O(\|\mathcal{S}\|\|\mathcal{A}\|)$
Partial-async QVI [BT89]	Full knowledge	Bounded	N/A	$O(\|\mathcal{S}\|\|\mathcal{A}\|)$
Async Q-learning [Tsi94]	Online simulation	Unbounded	–	$O(\|\mathcal{S}\|\|\mathcal{A}\|)$
AsyncQVI	Generative model	Bounded	$\surd$	$O(\|\mathcal{S}\|)$

They established and analyzed fundamental asynchronous models, which are characterized by coordinate update and asynchronous delay. This seminal work inspires the later study of async-parallel algorithms for DMDPs that are not fully known beforehand and can only be accessed by samples.

[Tsi94] adapted Q-learning to async-parallel setting in online simulation[1] and provided an almost sure convergence guarantee. However, although several works have established sample complexity results for single-threaded cases [KMN02, EM03, AMG11, AMK13, KBJ14, SWW18b, SWW18c, AKY19], there has been no such a result for async-parallel algorithms. Considering the latent huge cost of sampling, an explicit complexity result is more and more concerned and serves as an important reference for algorithm efficiency.

One may notice that to achieve promising complexity results, several works adopt the *generative model*, e.g., [KMN02, AMG11, AMK13, KBJ14, SWW18b, SWW18c]. This model is proposed by [KMN02]. It is a powerful sample oracle which takes any state-action pair (i, a) as input and returns a next state j with probability p_{ij}^a and the corresponding instant reward r_{ij}^a. Our algorithm adopts the generative model and provides the first explicit sample complexity result for async-parallel RL.

We list related async-parallel methods for DMDPs in Table 2.1 and the generative

[1]Online simulation is a weaker sample oracle than a generative model. In online simulation, the agent is constrained to start from (normally) a few states and proceeds in the form of trajectories.

Table 2.2: Related Algorithms with A Generative Model.

Algorithms	**Parallel**	**Sample**	**Memory**
VRVI [SWW18c]	$\times$	$\widetilde{O}\big(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^4\varepsilon^2}\log(\frac{1}{\delta})\big)$	$O(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$
VRQVI [SWW18b]	$\times$	$\widetilde{O}\big(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^3\varepsilon^2}\log(\frac{1}{\delta})\big)$	$O(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$
AsyncQVI	$\surd$	$\widetilde{O}\big(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta})\big)$	$O(\lvert\mathcal{S}\rvert)$

model methods in Table 2.2. Note that some papers [EM03, AMG11, KBJ14] use the word "asynchronous" for single-threaded coordinate update methods. In constrast, our algorithm is not only multi-threaded, but also allows stale information and async delay. The lower sample complexities achieved by [SWW18b, SWW18c] rely on the *variance reduction* technique, which requires periodic synchronization and $O(|\mathcal{S}||\mathcal{A}|)$ memory footprint to update and store a basis, say $\mathbf{p}_i^{a\top}\mathbf{v}_0$, $\forall\, i \in \mathcal{S}, a \in \mathcal{A}$. In order to take advantage of fully async-parallel structure and achieve the minimal memory complexity $O(|\mathcal{S}|)$, we do not use variance reduction and obtain a slightly higher sample complexity.

The last thing to mention is that there are some other nice async-parallel works about fixed point problems in a Hilbert space, e.g. [PXY16, HY18], while our algorithm is based on a contraction with respect to the ℓ_∞ norm.

2.2 Preliminaries

In this section, we review several key results on Q-value iteration and async-parallel algorithms.

[1]Under the assumption: $\forall i, j$, $\lim_{t\to\infty}\tau_j^i(t) = \infty$ holds with probability 1.

2.2.1 Q-value Iteration

Given a DMDP $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \gamma)$ and a policy π, we define the action-value vector $\mathbf{Q}^\pi$ with entries

$$Q^\pi_{i,a} = \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t r^{a_t}_{s_t s_{t+1}} \big|\ s_0 = i, a_0 = a].$$

For an optimal policy π^*, we let $\mathbf{Q}^*$ denote the corresponding optimal action-value vector. From $\mathbf{Q}^*$, we can obtain $\forall\ i \in \mathcal{S}$, $\pi^*_i = \arg\max_a Q^*_{i,a}$, $v^*_i = \max_a Q^*_{i,a}$. Hence, to derive an optimal policy π^*, it suffices to compute the optimal action-value vector $\mathbf{Q}^*$. To reach this end, we first define an operator $T : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ as

$$[T\mathbf{Q}]_{i,a} = \underbrace{\sum_{j\in\mathcal{S}} p^a_{ij} r^a_{ij}}_{\text{expected instant reward}} + \underbrace{\gamma \sum_{j\in\mathcal{S}} p^a_{ij} \max_{a'} Q_{j,a'}}_{\text{expected discounted future reward}}, \tag{2.1}$$

where $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $[T\mathbf{Q}]_{i,a}$ is the $((i-1)\times|\mathcal{A}| + a)$th component of $T\mathbf{Q}$ with $1 \le i \le |\mathcal{S}|, 1 \le a \le |\mathcal{A}|$. Actually, T is the well-known Bellman operator. It is an $\gamma-$contraction under ℓ_∞ norm and $\mathbf{Q}^*$ is the unique fixed point (see e.g. [Put14]). Therefore, one can apply fixed-point iterations of T to recover $\mathbf{Q}^*$. Next, we introduce the async-parallel coordinate update fashion of fixed-point iterations.

2.2.2 Asynchronous-Parallel Coordinate Updates

Given an ℓ_∞ $\gamma-$ contraction $G : \mathbb{R}^n \to \mathbb{R}^n$, the fixed-point iteration $\mathbf{x}(t+1) = G(\mathbf{x}(t))$, $t \ge 0$ converges linearly. Rewriting $G\mathbf{x}$ as $(G_1\mathbf{x}, \ldots, G_n\mathbf{x})$, we call

$$x_i(t+1) = \begin{cases} G_i(\mathbf{x}(t)), & t \in \mathscr{T}^i; \\ x_i(t), & t \notin \mathscr{T}^i, \end{cases} \tag{2.2}$$

the coordinate update of $G\mathbf{x}$, where $x_i(t)$ is the ith coordinate of $\mathbf{x}$ at iteration t and

$$\mathscr{T}^i := \{t \ge 0 : \text{coordinate } i \text{ is updated at iteration } t\}$$

is the set of iterations at which x_i is updated.

Algorithm 3 A Generic Framework of Async-parallel Coordinate Update

1: **Shared variables:** $\mathbf{x}^0$**,** $L > 0$**,** $t \leftarrow 0$;

2: **Local variable:** $\hat{\mathbf{x}}$;

3: **while** $t < L$, every agent asynchronously **do**

4: Select $i \in \{1, 2, \cdots, n\}$ according to some criterion;

5: Copy (required) shared variable to local memory: $\hat{\mathbf{x}} \leftarrow \mathbf{x}$;

6: Perform an update: $x_i \leftarrow G_i(\hat{\mathbf{x}})$;

7: Increment the global counter: $t \leftarrow t + 1$;

8: **end while**

We use a set of computing agents to perform coordinate update (2.2) in an async-parallel fashion. Unlike the typical parallel implementation where all the agents must wait for the slowest one to finish an update, async-parallel algorithms allow each agent to use the (possibly stale) information it has and complete more iterations within the same period of time, which is preferable for cases where the computing capacity is highly heterogeneous or the workload is far from balanced. See more discussions in [HY17].

We summarize a shared-memory async-parallel coordinate-update framework in Algorithm 3, where each agent first chooses one coordinate to update, then reads necessary information from global memory to the local cache, and finally updates its computed result to the shared memory.

By Line 6 in Algorithm 3, the tth update can be written as

$$x_i(t+1) = \begin{cases} G_i(\hat{\mathbf{x}}(t)), & t \in \mathscr{T}^i; \\ x_i(t), & t \notin \mathscr{T}^i. \end{cases} \tag{2.3}$$

Here, $\hat{\mathbf{x}}(t) := [x_1(\tau_1(t)), \ldots, x_n(\tau_n(t))]^\top$ represents the possibly stale information, where $x_j(\tau_j(t))$ is the most recent version of x_j available at time t that is used to compute $x_i(t+1)$. We have that $0 \leq \tau_j(t) \leq t$. The difference $t - \tau_j(t)$ is called the *delay*. In this paper, we

assume *partial asynchronism* [BT89]:

Assumption 2.2.1 (Partial Asynchronism[2]). *For the async-parallel algorithm, there exists two positive integers B_1, B_2 (asynchronism measure) such that:*

1. *For every i and for every $t \geq 0$, at least one of the elements of the set $\{t, t+1, \ldots, t+B_1-1\}$ belongs to $\mathscr{T}^i$;*
2. *There holds $t - B_2 < \tau_j(t) \leq t$, for all j and all $t \geq 0$.*

Assumption 2.2.1 (a) ensures that the time interval between consecutive updates to each coordinate is uniformly bounded by B_1 and (b) ensures that the communication delays are uniformly bounded by B_2. Note that when $B_1 = B_2 = 1$, the algorithm becomes synchronous. Convergence under this assumption was established in [FJ14].

Proposition 1. *[FJ14, Theorem 2] Consider the iterations Equation* (2.3) *under Assumption 2.2.1. Suppose that G is a γ-contraction under ℓ_∞ norm and $\mathbf{x}^*$ is the fixed point of G. Then $\|\mathbf{x}(t) - \mathbf{x}^*\|_\infty \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t-2B_1}$ for all $t \geq B_1$, where $\rho = \gamma^{\frac{1}{B_1+B_2-1}}$.*

We prove the above proposition by showing a stronger result and the proposition holds as a direct consequence. We first sort the updating index set $\mathscr{T}^i$ as in Equation (2.3) into a sequence $(t_k^i)_{k\geq 0}$, where t_0^i is the first element of $\mathscr{T}^i$ and t_k^i is the $(k+1)$th. Then Theorem 2.2.1 bounds $|x_i(t) - x_i^*|$ in a staircase decreasing way: $|x_i(t) - x_i^*|$ will contract when $t \in \mathscr{T}^i$, or equivalently, $t = t_k^i$ for some k.

Theorem 2.2.1 (Staircase Decreasing). *Consider the iteration* (2.3) *under Assumption 2.2.1. Suppose that G is γ-contractive under infinity norm and $\mathbf{x}^*$ is the fixed-point of G. For each $t \geq B_1$ and $i \in \{1, 2, \cdots, n\}$, if $t \in (t_k^i, t_{k+1}^i]$ for some k, then $x_i(t)$ satisfies*

$$|x_i(t) - x_i^*| \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t_k^i - B_1}, \tag{2.4}$$

[2]Assumption 1.1 in [BT89, Section 7.1] uses B for both B_1 and B_2. Because B_1 and B_2 are different in practice, we keep them separate to derive a tighter bound. Further, we have dropped assumption (c) there to make our algorithm easier to implement.

where $\rho := \gamma^{\frac{1}{B_1+B_2-1}}$.

Proof. We first claim that for each $t \geq B_1$ and $i \in \{1, 2, \cdots, n\}$, there exists some $k \geq 0$ such that $t \in (t_k^i, t_{k+1}^i]$. This follows from Assumption 2.2.1 (a), where $t_0^i \leq B_1 - 1$, $\forall\ i$.

Now we prove Eq. (2.4) by induction. One could check

$$\|\mathbf{x}(t) - \mathbf{x}^*\|_\infty \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty,\ \forall\ t \geq 0$$

as a corollary of [FJ14, Theorem 2] or by another induction. We skip the details here. Thus for the basic case,

$$\max_{0\leq t\leq B_1} \left\{\|\mathbf{x}(t) - \mathbf{x}^*\|_\infty \rho^{-t}\right\} \leq \max_{0\leq t\leq B_1} \left\{\|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{-t}\right\} \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{-B_1},$$

which gives that for each $t \leq B_1$ and $i \in \{1, 2, \cdots, n\}$,

$$|x_i(t) - x_i^*| \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t-B_1}.$$

Since ρ^t is decreasing, we can further obtain that

$$|x_i(B_1) - x_i^*| \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t_k^i - B_1},$$

if $B_1 \in (t_k^i, t_{k+1}^i]$ for some k.

For the induction step, we assume that Eq. (2.4) holds for all $t \geq B_1$ up to some t'. For a fixed $i \in \{1, 2, \cdots, n\}$, supposing that $t' \in (t_{k'}^i, t_{k'+1}^i]$ for some k', then we analyze the scenario at $(t'+1)$ as two cases.

Case 1: $t' \notin \mathscr{T}^i$, i.e., we do not update coordinate i at iteration t'. Hence, $x_i(t'+1) = x_i(t')$ and $t'+1 \in (t_{k'}^i, t_{k'+1}^i]$. Then Eq. (2.4) follows directly.

Case 2: $t' \in \mathscr{T}^i$, i.e., the ith coordinate is updated at iteration t' and $t' = t_{k'+1}^i$. Since G is γ-contractive under infinity norm, we have

$$\begin{aligned} |x_i(t'+1) - x_i^*| = |G_i(\hat{\mathbf{x}}(t)) - x_i^*| &\leq \|G(\hat{\mathbf{x}}(t)) - \mathbf{x}^*\|_\infty \\ &\leq \gamma \max_j \left|x_j(\tau_j(t')) - x_j^*\right|. \end{aligned} \tag{2.5}$$

For a fixed $j \in \{1, 2, \cdots, n\}$, suppose that $\tau_j(t') \in (t^j_{k_\tau}, t^j_{k_\tau+1}]$ for some k_τ. Then the induction hypothesis gives $\left|x_j(\tau_j(t')) - x^*_j\right| \leq \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t^j_{k_\tau} - B_1}$. Since $\tau_j(t') \leq t^j_{k_\tau} + B_1$ by Assumption 2.2.1 (a) and $\tau_j(t') \geq t' - B_2 + 1$ by Assumption 2.2.1 (b), we obtain

$$\begin{aligned}
\gamma\left|x_j(\tau_j(t')) - x^*_j\right| &\leq \gamma\|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t^j_{k_\tau} - B_1} \\
&\leq \gamma\|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{\tau_j(t') - 2B_1} \\
&\leq \gamma\|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t' - 2B_1 - B_2 - 1} \\
&= \|\mathbf{x}(0) - \mathbf{x}^*\|_\infty \rho^{t^i_{k'+1} - B_1}, \qquad (2.6)
\end{aligned}$$

where the equality holds since $\gamma = \rho^{B_1+B_2-1}$ by definition and $t' = t^i_{k'+1}$. Notice that $t'+1 \in (t^i_{k'+1}, t^i_{k'+2}]$. Inserting Eq. (2.6) back into Eq. (2.5) yields the desired result. ■

One may note that if $t \in (t^i_k, t^i_{k+1}]$, then $t^i_k + B_1 \geq t$ by Assumption 2.2.1 (a). Hence, Proposition 1 is a direct consequence of Theorem 2.2.1.

In many DMDPs, the transition probabilities P are sparse. So for any state-action pair (i, a), the possible next states form a tiny subset of $\mathcal{S}$. Hence, to apply async-parallel coordinate updates to Equation (2.1), very few components are required and we only need to bound async delay over a smaller subset. Therefore, we usually have $B_2 \ll B_1$, where $B_1 \geq |\mathcal{S}||\mathcal{A}|$. Hence, the convergence rate $\gamma^{\frac{1}{B_1+B_2-1}}$ we obtain is significantly better than $\gamma^{\frac{1}{2(B_1 \vee B_2)-1}}$ from [FJ14, Theorem. 2].

Remark 3 (Total Asynchronism)**.** *Here we do not adopt the total asynchronism notion [BT89, Section 6.1]. To start with, one cannot derive convergence rate results under total asynchronism since it allows arbitrarily long delays and no improvement can be said for finite iterations. On the contrary, partial asynchronism can avoid this case and be practically enforced [BT89, Section 7.1].*

Algorithm 4 AsyncQVI: Asynchronous-Parallel Q-value Iteration

Input: $\varepsilon \in (0, (1-\gamma)^{-1})$, $\delta \in (0,1)$, L, K;

Shared variables: $\mathbf{v} \leftarrow \mathbf{0}$, $\pi \leftarrow \mathbf{0}$, $t \leftarrow 0$;

Private variables: $\hat{\mathbf{v}}, r, S, q$;

while $t < L$, every agent asynchronously **do**

  Select state $i_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$

  Copy shared variable to local memory : $\hat{\mathbf{v}} \leftarrow \mathbf{v}$;

  Call $\texttt{GM}(s_t, a_t)$ K times and collect samples $s'_1, \ldots, s'_K$ and $r_1, \ldots, r_K$;

  Let $q \leftarrow \frac{1}{K}\sum_{k=1}^{K} r_k + \gamma \frac{1}{K}\sum_{k=1}^{K} \hat{v}_{s'_k} - \frac{(1-\gamma)\varepsilon}{4}$

  if $q > v_{i_t}$ **then**

    mutex lock;

    $v_{i_t} \leftarrow q$, $\pi_{i_t} \leftarrow a_t$;

    mutex unlock;

    increment the global counter: $t \leftarrow t+1$

  end if

end while

Return: π

2.3 AsyncQVI: Asynchronous-Parallel Q-value Iteration

In this section, we present AsyncQVI in Algorithm 4 and its convergence analysis. AsyncQVI is an asynchronous stochastic version of Equation (2.1). To develop AsyncQVI, we first apply the asynchronous framework (Algorithm 3) to Equation (2.1), obtaining

$$Q_{i,a}(t+1) = \begin{cases} \sum_j p_{ij}^a r_{ij}^a + \gamma \sum_j p_{ij}^a \max_{a'} \hat{Q}_{j,a'}(t), & t \in \mathscr{T}^{i,a}; \\ Q_{i,a}(t), & t \notin \mathscr{T}^{i,a}. \end{cases} \tag{2.7}$$

Since there is no knowledge of the transition probability, we approximate the expectations $\sum_j p_{ij}^a \cdot$ by random sampling (Lines 7 and 8, Algorithm 4). So instead of (2.7), we substitute $\sum_j p_{ij}^a r_{ij}^a$ and $\sum_j p_{ij}^a \max_{a'} \hat{Q}_{j,a'}(t)$ by their empirical means, i.e., $r(t) := \frac{1}{K}\sum_k r_{i_t s'_k}^{a_t}$ and $S(\hat{\mathbf{Q}}(t)) := \frac{1}{K}\sum_k \max_{a'} \hat{Q}_{s'_k,a'}(t)$, respectively. For the purpose of analysis, we also tune the update slightly by subtracting a small constant $(1-\gamma)\varepsilon/4$. Consequently, AsyncQVI is equivalent to

$$Q_{i,a}(t+1) = \begin{cases} r(t) + \gamma S(\hat{\mathbf{Q}}(t)) - (1-\gamma)\varepsilon/4 & t \in \mathscr{T}^{i,a}; \\ Q_{i,a}(t), & t \notin \mathscr{T}^{i,a}. \end{cases} \tag{2.8}$$

For memory efficiency, we do not form $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. Instead, since only the values $\max_{a'} Q_{i,a'}$ are used for update, we maintain two vectors $\mathbf{v}, \pi \in \mathbb{R}^{|\mathcal{S}|}$; at each iteration t, we ensure $v_i(t) = \max_a Q_{i,a}(t)$, $\pi_i(t) = \arg\max_a Q_{i,a}(t)$ and $\hat{v}_j(t) = \max_{a'} \hat{Q}_{j,a'}(t)$. By this means, we reduce the memory complexity from $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ to $\mathcal{O}(|\mathcal{S}|)$, which is of a great advantage in real applications.

Remark 4 (Coordinate Selection)**.** *To guarantee convergence, the coordinate should be selected to satisfy Assumption 2.2.1. In practice, however, if all agents have similar powers, one can simply apply either uniformly random or globally cyclic selections.*

Remark 5 (Parallel Overhead)**.** *In AsyncQVI, overhead can only occur during copying variable from global memory to the local memory (Line 6) and where a memory lock is implemented (Lines 9-12). For the former case, the time complexity is $\mathcal{O}(|\mathcal{S}|)$, which is negligible when $\mathcal{O}(|\mathcal{S}|)$ is small or the process of querying samples is much slower. Otherwise, one can consider copying in a less frequent fashion, i.e., updating $\hat{\mathbf{v}}$ every l_0 iterations. Although it will increase B_2 by l_0, the sample complexity is still near-optimal as long as $B_1 + B_2 = \mathcal{O}(|\mathcal{S}||\mathcal{A}|)$. For the latter case, a memory lock (e.g. mutex) ensures that v_i and π_i are indeed the maximum value and a maximizer of the vector $\mathbf{Q}_i$, respectively. Since only two scalars are accessed and altered, the collision is rare.*

2.3.1 Convergence Analysis

Next, we establish convergence for AsyncQVI. To distinguish different sequences, we let $(\mathbf{Q}^{\mathbb{E}}(t))$ denote the asynchronous coordinate update sequence generated through Equation (2.7), where the superscript represents the updates with real expectations. Specifically, if AsyncQVI produces a sequence according to Equation (2.8) with $\hat{\mathbf{Q}}(t) = [Q_{1,1}(\tau_{1,1}(t)), \ldots, Q_{|\mathcal{S}|,|\mathcal{A}|}(\tau_{|\mathcal{S}|,|\mathcal{A}|}(t))]^\top$, then

$$Q^{\mathbb{E}}_{i,a}(t+1) = \begin{cases} \bar{r}^a_i + \gamma \sum_j p^a_{ij} \max_{a'} \hat{Q}^{\mathbb{E}}_{j,a'}(t), & t \in \mathscr{T}^{i,a}; \\ Q^{\mathbb{E}}_{i,a}(t), & t \notin \mathscr{T}^{i,a}, \end{cases} \tag{2.9}$$

where $\hat{\mathbf{Q}}^{\mathbb{E}}(t) = [Q^{\mathbb{E}}_{1,1}(\tau_{1,1}(t)), \ldots, Q^{\mathbb{E}}_{|\mathcal{S}|,|\mathcal{A}|}(\tau_{|\mathcal{S}|,|\mathcal{A}|}(t))]^\top$. There are two things to notice:

1. $(\mathbf{Q}^{\mathbb{E}}(t))^L_{t=0}$ and $(\mathbf{Q}(t))^L_{t=0}$ have the same initial point;
2. at any iteration, $(\mathbf{Q}^{\mathbb{E}}(t))^L_{t=0}$ shares exactly the same choice of coordinate (i_t, a_t) and the same asynchronous delay with $(\mathbf{Q}(t))^L_{t=0}$.

These properties are important to our analysis. Recall that we assume partial asynchronism (Assumption 2.2.1) for AsyncQVI. Then Equation (2.9) also meets Assumption 2.2.1. Hence, Equation (2.9) converges following the fact that T is a $\gamma-$contraction and Proposition 1. Since Equation (2.8) is an approximation of Equation (2.9), we can leverage the convergence of Equation (2.9) to establish the convergence of AsyncQVI. To this end, we first use Hoeffding's Inequality [Hoe63] to analyze the sampling error. Specifically, if we take enough samples per iteration, then the error can be controlled with high probability.

Proposition 2 (Sample Concentration)**.** *With* $K = \left\lceil \frac{8}{(1-\gamma)^4\varepsilon^2} \log\left(\frac{4L}{\delta}\right) \right\rceil$, *AsyncQVI generates a sequence* $(r(t), S(\hat{\mathbf{Q}}(t)))^{L-1}_{t=0}$ *that satisfies* $\left| r(t) + \gamma S(\hat{\mathbf{Q}}(t)) - \bar{r}^{a_t}_{i_t} - \gamma \mathbf{p}^{a_t\top}_{i_t} \hat{\mathbf{v}}(t) \right| \leq \frac{(1-\gamma)\varepsilon}{4}$, $\forall\ 0 \leq t \leq L-1$, *with probability at least* $1-\delta$.

Proof. We first show that for a constant L, with $K = \left\lceil \frac{8}{(1-\gamma)^4\varepsilon^2} \log(\frac{4L}{\delta}) \right\rceil$ samples, AsyncQVI

returns $r(t)$ and $S(\hat{\mathbf{Q}}(t))$ satisfying

$$|r(t) - \bar{r}_{i_t}^{a_t}| \leq \frac{(1-\gamma)^2\varepsilon}{4}, \qquad |S(\hat{\mathbf{Q}}(t)) - \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)| \leq \frac{(1-\gamma)\varepsilon}{4}$$

with probability at least $1 - \delta/L$.

As we explained before, both $r(t)$ and $S(\hat{\mathbf{Q}}(t))$ are averages of K i.i.d. samples with $\mathbb{E}[r(t)] = \sum_j p_{i_tj}^{a_t} r_{i_tj}^{a_t} := \bar{r}_{i_t}^{a_t}$ and $\mathbb{E}[S(\hat{\mathbf{Q}}(t))] = \sum_j p_{i_tj}^{a_t}\hat{v}_j(t) := \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)$. Since we assume $r_{ij}^a \in [0,1]$, it is easy to verify $0 \leq \hat{\mathbf{v}}(t) \leq \frac{1}{1-\gamma}$ by induction. We skip the details here. Then letting $K = \left\lceil \frac{8}{(1-\gamma)^4\varepsilon^2} \log\left(\frac{4L}{\delta}\right)\right\rceil$, we can obtain that

$$\mathbb{P}\Big[|r(t) - \bar{r}_{i_t}^{a_t}| \geq \frac{(1-\gamma)^2\varepsilon}{4}\Big] \leq 2e^{\frac{-2K^2(1-\gamma)^4\varepsilon^2}{16K}} \leq \frac{\delta}{2L};$$
$$\mathbb{P}\Big[|S(\hat{\mathbf{Q}}(t)) - \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)| \geq \frac{(1-\gamma)\varepsilon}{4}\Big] \leq 2e^{\frac{-2K^2(1-\gamma)^4\varepsilon^2}{16K}} \leq \frac{\delta}{2L}.$$

Based on the above result, for a fixed iteration t, we have

$$\left|r(t) + \gamma S(\hat{\mathbf{Q}}(t)) - \bar{r}_{i_t}^{a_t} - \gamma\mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)\right| \leq |r(t) - \bar{r}_{i_t}^{a_t}| + \gamma|S(\hat{\mathbf{Q}}(t)) - \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)| \leq \frac{(1-\gamma)\varepsilon}{4}$$

holds with probability at least $1 - \frac{\delta}{L}$. Taking a union bound over all $0 \leq t \leq L-1$ iterations gives the desired result. ■

Proposition 2 establishes a control over a one-step approximation error between Equation (2.8) and Equation (2.9) provided that $\hat{\mathbf{Q}} = \hat{\mathbf{Q}}^{\mathbb{E}}$. However, for the two sequences $(\mathbf{Q}(t))$ and $(\mathbf{Q}^{\mathbb{E}}(t))$ that only share the same initial point, the error can accumulate. To tackle this issue, we further utilize the γ-contraction property to weaken previously cumulative error. More specifically, if the newly made error and the previously accumulated error keep the ratio $(1-\gamma):1$ for each iteration, the overall error remains $(1-\gamma)\varepsilon + \gamma\varepsilon = \varepsilon$. By this means, we can control the difference between $(\mathbf{Q}(t))$ and $(\mathbf{Q}^{\mathbb{E}}(t))$ by induction.

Proposition 3. *Given the total iteration number L, accuracy parameters ε and δ, with $K = \left\lceil \frac{8}{(1-\gamma)^4\varepsilon^2} \log\left(\frac{4L}{\delta}\right)\right\rceil$, AsyncQVI can generate a sequence $(\mathbf{Q}(t))_{t=1}^L$ satisfying $\|\mathbf{Q}(t) - \mathbf{Q}^{\mathbb{E}}(t)\|_\infty \leq \varepsilon/2,\ \forall\ 1 \leq t \leq L$ with probability at least $1-\delta$.*

Proof of Proposition 3. We denote by $\mathcal{E}_1$ the event

$$\Big\{\big|r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\bar{r}_{i_t}^{a_t}-\gamma \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)\big|\le\frac{(1-\gamma)\varepsilon}{4},\ \forall\ 0\le t\le L-1\Big\}.$$

By Proposition 2, $\mathcal{E}_1$ occurs with probability at least $1-\delta$. Next, we condition on $\mathcal{E}_1$ and prove

$$\|\mathbf{Q}(t)-\mathbf{Q}^{\mathbb{E}}(t)\|_\infty\le\frac{\varepsilon}{2},\ \forall\ 1\le t\le L$$

by induction. The basic case is trivial. For the induction step, we analyze the scenario at $t+1$ as two cases. When $t\notin\mathscr{T}^{i,a}$, $|Q_{i,a}(t+1)-Q^{\mathbb{E}}_{i,a}(t+1)|\le\varepsilon/2$ follows from the hypothesis, since Eqs. (2.8) and (2.9) give that

$$Q_{i,a}(t+1)-Q^{\mathbb{E}}_{i,a}(t+1)=Q_{i,a}(t)-Q^{\mathbb{E}}_{i,a}(t).$$

When $t\in\mathscr{T}^{i,a}$, by Eq. (2.8), Eq. (2.9) and triangle inequality, we have that

$$\begin{aligned}
&\big|Q_{i,a}(t+1)-Q^{\mathbb{E}}_{i,a}(t+1)\big|\\
=&\Big|r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\frac{(1-\gamma)\varepsilon}{4}-\bar{r}_i^a-\gamma\sum_j p_{ij}^a\max_{a'}\hat{Q}^{\mathbb{E}}_{j,a'}(t)\Big|\\
\le&\Big|r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\bar{r}_i^a-\gamma\mathbf{p}_i^{a\top}\hat{\mathbf{v}}(t)-\frac{(1-\gamma)\varepsilon}{4}\Big|+\Big|\gamma\mathbf{p}_i^{a\top}\hat{\mathbf{v}}(t)-\gamma\sum_j p_{ij}^a\max_{a'}\hat{Q}^{\mathbb{E}}_{j,a'}(t)\Big|\\
\le&\Big|r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\bar{r}_i^a-\gamma\mathbf{p}_i^{a\top}\hat{\mathbf{v}}(t)\Big|+\frac{(1-\gamma)\varepsilon}{4}+\gamma\sum_j p_{ij}^a\big|\max_{a'}\hat{Q}_{j,a'}(t)-\max_{a'}\hat{Q}^{\mathbb{E}}_{j,a'}(t)\big|.
\end{aligned}$$

By definition of $\mathcal{E}_1$ and the induction hypothesis, we further obtain that

$$|Q_{i,a}(t+1)-Q^{\mathbb{E}}_{i,a}(t+1)|\le\frac{(1-\gamma)\varepsilon}{4}+\frac{(1-\gamma)\varepsilon}{4}+\gamma\frac{\varepsilon}{2}=\frac{\varepsilon}{2},$$

which completes the proof. ■

Since $(\mathbf{Q}^{\mathbb{E}}(t))$ converges to $\mathbf{Q}^*$ linearly, combining Propositions 1 and 3 gives the desired result.

Theorem 2.3.1 (Linear Convergence)**.** *Under Assumption 2.2.1, given accuracy parameters ε and δ, with $L=\lceil 2B_1+\frac{B_1+B_2-1}{1-\gamma}\log\big(\frac{2}{(1-\gamma)\varepsilon}\big)\rceil$ and $K=\lceil\frac{8}{(1-\gamma)^4\varepsilon^2}\log\big(\frac{4L}{\delta}\big)\rceil$, AsyncQVI can*

produce $\mathbf{Q}(L) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ *and* $\mathbf{v}(L) \in \mathbb{R}^{|\mathcal{S}|}$ *satisfying* $\|\mathbf{Q}^* - \mathbf{Q}(L)\|_\infty \leq \varepsilon$ *and* $\|\mathbf{v}^* - \mathbf{v}(L)\|_\infty \leq \varepsilon$ *with probability at least* $1 - \delta$.

Proof. By Proposition 1,

$$\|\mathbf{Q}^* - \mathbf{Q}^{\mathbb{E}}(L)\|_\infty (1-\gamma)^{-1} \rho^{L-2B_1} = (1-\gamma)^{-1} \gamma^{\frac{L-2B_1}{B_1+B_2-1}}.$$

Notice that $\gamma = (1-(1-\gamma)) \leq e^{-(1-\gamma)}$. We have that

$$\|\mathbf{Q}^* - \mathbf{Q}^{\mathbb{E}}(L)\|_\infty \leq (1-\gamma)^{-1} e^{-(1-\gamma)\frac{L-2B_1}{B_1+B_2-1}} \leq \frac{\varepsilon}{2}, \tag{2.10}$$

where the last inequality holds with $L = \left\lceil 2B_1 + \frac{B_1+B_2-1}{1-\gamma} \log\left(\frac{2}{(1-\gamma)\varepsilon}\right) \right\rceil$. Then, by Proposition 3, with probability at least $1-\delta$,

$$\|\mathbf{Q}^{\mathbb{E}}(L) - \mathbf{Q}(L)\|_\infty \leq \frac{\varepsilon}{2}. \tag{2.11}$$

Inserting Equation (2.11) back into Equation (2.10) gives the desired result

$$\|\mathbf{Q}^* - \mathbf{Q}(L)\|_\infty \leq \|\mathbf{Q}^* - \mathbf{Q}^{\mathbb{E}}(L)\|_\infty + \|\mathbf{Q}^{\mathbb{E}}(L) - \mathbf{Q}(L)\|_\infty \leq \varepsilon.$$

Then one can check $\|\mathbf{v}^* - \mathbf{v}(L)\|_\infty \leq \varepsilon$ at ease. ■

In the following theorem, we show that the vector π maintained through the iterations is an ε-optimal policy. Using this theorem, we shall present the sample complexity of AsyncQVI in Corollary 1.

Theorem 2.3.2. *Under Assumption 2.2.1, given accuracy parameters* ε *and* δ*, with* $L = \left\lceil 2B_1 + \frac{B_1+B_2-1}{1-\gamma} \log\left(\frac{2}{(1-\gamma)\varepsilon}\right) \right\rceil$ *and* $K = \left\lceil \frac{8}{(1-\gamma)^4\varepsilon^2} \log\left(\frac{4L}{\delta}\right) \right\rceil$*, AsyncQVI returns an* ε*-optimal policy* π *with probability at least* $1-\delta$.

To prove the above result, we first define a policy operator.

Definition 1 (Policy Operator)**.** *Given a policy* π *and a vector* $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$*, the policy operator* T_π*:* $\mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ *is defined as*

$$[T_\pi \mathbf{v}]_i = \bar{r}_i^{\pi_i} + \gamma \mathbf{p}_i^{\pi_i \top} \mathbf{v} = r_i^{\pi_i} + \gamma \sum_{j \in \mathcal{S}} p_{ij}^{\pi_i} v_j.$$

There are some properties of T_π.

Proposition 4 (T_π's Properties)**.** *Given a policy π, for any vectors $\mathbf{v}$, $\mathbf{v}' \in \mathbb{R}^{|\mathcal{S}|}$,*

1. ***Monotonicity:*** *if $\mathbf{v} \leq \mathbf{v}'$, then $T_\pi \mathbf{v} \leq T_\pi \mathbf{v}'$.*
2. ***γ-Contraction:*** *$\|T_\pi \mathbf{v} - T_\pi \mathbf{v}'\|_\infty \leq \gamma \|\mathbf{v} - \mathbf{v}'\|_\infty$.*
3. *$\mathbf{v}^\pi$ is the fixed-point of T_π.*

The proof is straightforward following the definition. We skip the details here. Next, we have the following result.

Lemma 2.3.1. [SWW18c] *Given a policy π, for any vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$, if there exists a $\mathbf{v}' \in \mathbb{R}^{|\mathcal{S}|}$ such that $\mathbf{v}' \leq \mathbf{v} \leq T_\pi \mathbf{v}'$, then $\mathbf{v} \leq \mathbf{v}^\pi$.*

Proof. By Proposition 4 (a) and $\mathbf{v}' \leq \mathbf{v}$, we first have $T_\pi \mathbf{v}' \leq T_\pi \mathbf{v}$. Combining with $\mathbf{v} \leq T_\pi \mathbf{v}'$, we further obtain $\mathbf{v} \leq T_\pi \mathbf{v}$. By induction, one can check $\mathbf{v} \leq T_\pi^n \mathbf{v}$, $\forall\ n \in \mathbb{N}$. Moreover, since T_π is a γ-contraction, $\mathbf{v}^\pi = \lim_{n\to\infty} T_\pi^n \mathbf{v}$. Hence, $\mathbf{v} \leq \lim_{n\to\infty} T_\pi^n \mathbf{v} = \mathbf{v}^\pi$. ■

Next, we consider the special case that $\mathbf{v}(L)$ and $\pi(L)$ are both derived from AsyncQVI with

$$\pi_i(L) = \arg\max_a Q_{i,a}(L), \quad v_i(L) = \max_a Q_{i,a}(L),\ \forall i \in \mathcal{S}.$$

If $\|\mathbf{v}^* - \mathbf{v}^\pi\|_\infty \leq \varepsilon$, then π is ε-optimal. To achieve this, we first show that $\mathbf{v}(L)$ satisfies Lemma 2.3.1 (see Lemma 2.3.2). Then with Theorem 2.3.1, $\|\mathbf{v}^* - \mathbf{v}^\pi\|_\infty \leq \|\mathbf{v}^* - \mathbf{v}(L)\|_\infty \leq \varepsilon$.

Lemma 2.3.2. *Under Assumption 2.2.1, AsyncQVI generates a sequence of $\{\mathbf{v}(t)\}_{t=1}^L$ and $\{\pi(t)\}_{t=1}^L$ satisfying*

$$\mathbf{v}(t-1) \leq \mathbf{v}(t) \leq T_{\pi(t)} \mathbf{v}(t-1),\ \forall\ 1 \leq t \leq L \tag{2.12}$$

with probability at least $1 - \delta$.

Proof. By Proposition 2,

$$\left|r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\bar{r}_{i_t}^{a_t}-\gamma \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t)\right| \le \frac{(1-\gamma)\varepsilon}{4},\ \forall\ 0\le t\le L-1$$

holds with probability at least $1-\delta$. Denote by $\mathcal{E}_2$ the event

$$\left\{r(t)+\gamma S(\hat{\mathbf{Q}}(t))-\frac{(1-\gamma)\varepsilon}{4} \le \bar{r}_{i_t}^{a_t}+\gamma \mathbf{p}_{i_t}^{a_t\top}\hat{\mathbf{v}}(t),\ \forall\ 0\le t\le L-1\right\}.$$

Then $\mathcal{E}_2$ occurs with probability at least $1-\delta$.

Now we condition on $\mathcal{E}_2$ and prove Eq. (2.12) by induction. For simplicity, we let $\mathbf{v}(-1)=\mathbf{v}(0)=\mathbf{0}$ and start our proof from $t=0$. Then the basic case holds. For the induction step, suppose that Eq. (2.12) is true for all t up to some t'. Recall that in AsyncQVI, for each iteration, whether v_i or π_i will be updates depends on the value of $Q_{i,a}$. We hence analyze the scenario at $(t'+1)$ as two cases.

Case 1: $Q_{i_{t'},a_{t'}}(t'+1)\le v_{i_{t'}}(t')$. Then $\mathbf{v}$ and π will not be updated, i.e., $\mathbf{v}(t'+1)=\mathbf{v}(t')$ and $\pi(t'+1)=\pi(t')$. In this case, the inequality $\mathbf{v}(t')\le\mathbf{v}(t'+1)$ follows directly. For the other part, by induction hypothesis we have

$$\mathbf{v}(t'+1)=\mathbf{v}(t')\le T_{\pi(t')}\mathbf{v}(t'-1)=T_{\pi(t'+1)}\mathbf{v}(t'-1)\le T_{\pi(t'+1)}\mathbf{v}(t'),$$

where the last inequality comes from $\mathbf{v}(t'-1)\le\mathbf{v}(t')$ and the monotonicity of $T_{\pi(t'+1)}$.

Case 2: $Q_{i_{t'},a_{t'}}(t'+1)>v_{i_{t'}}(t')$. Then $\forall\ i\in\mathcal{S}$,

- **Case 2.1:** $i\ne i_{t'}$. In this case, $v_i(t'+1)=v_i(t')$ and $\pi_i(t'+1)=\pi_i(t')$. Hence, once again by induction hypothesis and T_π's monotonicity, we obtain

$$v_i(t'+1)=v_i(t')\le\left[T_{\pi(t')}\mathbf{v}(t'-1)\right]_i=\left[T_{\pi(t'+1)}\mathbf{v}(t'-1)\right]_i\le\left[T_{\pi(t'+1)}\mathbf{v}(t')\right]_i.$$

- **Case 2.2:** $i=i_{t'}$. According to Lines 9 and 11 of Algorithm 4, the ith coordinate of $\mathbf{v}$ is updated at iteration t' and the former inequality follows directly. For the latter inequality, by Line 8 of Algorithm 4 we have

$$v_i(t'+1)=Q_{i,a_{t'}}(t'+1)=r(t')+\gamma S(\hat{\mathbf{Q}}(t'))-\frac{(1-\gamma)\varepsilon}{4}.$$

By definition of $\mathcal{E}_2$ and $\pi_i(t'+1) = a_{t'}$, we obtain

$$v_i(t'+1) \leq \bar{r}_i^{a_{t'}} + \gamma \mathbf{p}_i^{a_{t'}\top} \hat{\mathbf{v}}(t') = \left[T_{\pi(t'+1)} \hat{\mathbf{v}}(t')\right]_i.$$

Owing to $\hat{\mathbf{v}}(t') \leq \mathbf{v}(t')$ by induction hypothesis and the monotonicity of $T_{\pi(t'+1)}$, we can complete our proof by

$$v_i(t'+1) \leq \left[T_{\pi(t'+1)} \hat{\mathbf{v}}(t')\right]_i \leq \left[T_{\pi(t'+1)} \mathbf{v}(t')\right]_i.$$

■

Finally, combining the results of Lemma 2.3.1, Lemma 2.3.2 and Theorem 2.3.1, we can establish Theorem 2.3.2 at ease.

Corollary 1 (Near-Optimal Sample Complexity)**.** *Under Assumption 2.2.1, AsyncQVI returns an ε-optimal policy π with probability at least $1-\delta$ at the sample complexity $\tilde{\mathcal{O}}\big(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta})\big)$, provided that $B_1 + B_2 = \mathcal{O}(|\mathcal{S}||\mathcal{A}|)$.*

Moreover, given the complete knowledge of transition P and reward r, we can also solve it asynchronous parallelly. To utilize AsyncQVI, one can build a *generative model* in $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}|)$ prepossessing time [Wan17], and the `GM` produces a sample in $\tilde{O}(1)$ arithmetic operations. In this sense, AsyncQVI also has the following computational complexity results.

Corollary 2 (Computational Complexity)**.** *Given a DMDP $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \gamma)$, under Assumption 2.2.1 AsyncQVI returns an ε-optimal policy with probability at least $1-\delta$ at the computational complexity*

$$\tilde{\mathcal{O}}\big(|\mathcal{S}|^2|\mathcal{A}| + \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta})\big),$$

provided that $B_1 + B_2 = \mathcal{O}(|\mathcal{S}||\mathcal{A}|)$.

2.4 Numerical Experiments

Environment To investigate the performance of AsyncQVI, we solve the sailing problem from [Van96] on a 100×100 grid with 80000 states and 8 actions. Each state contains

the sailor's current position (x, y) and the wind direction. Each action is one of the eight directions $\{(0,1), (0,-1), (1,0), (-1,0), (1,1), (1,-1), (-1,1), (-1,-1)\}$. The goal is to reach the target position $(50, 50)$ at the lowest cost. Different from the original settings, we add more randomness to the system. Under the action (δ_x, δ_y), the sailor will be further affected by two drift noises: a mild wind noise $\mathcal{N}(0, \sigma_1^2)$ which occurs with probability (w.p.) 1 and a big vortex noise $\mathcal{N}(0, \sigma_2^2)$ which occurs with a fairly small probability p. So, the next position is

$$\begin{aligned}
&(x + \delta_x + \mathcal{N}(0, \sigma_1^2),\ y + \delta_y + \mathcal{N}(0, \sigma_1^2)), \quad \text{with probability } 1 - p;\\
&(x + \delta_x + \mathcal{N}(0, \sigma_1^2 + \sigma_2^2),\ y + \delta_y + \mathcal{N}(0, \sigma_1^2 + \sigma_2^2)), \quad \text{with probability } p.
\end{aligned}$$

The wind direction at next time maintains its current direction w.p. 0.3, changes 45 degrees to either direction w.p. 0.2 each direction, changes 90 degrees to either direction w.p. 0.1 each, changes 135 degrees to either direction w.p. 0.04 each, and reverses direction w.p. 0.02. We set the instant reward as $d \times |\frac{\text{angle between wind and action directions}}{45^\circ}|$, where d is a constant hyperparameter. When the reward is lower, we can take it as a higher cost. If the sailor reaches the target position, the reward is 1.

Algorithms and Implementation We compare five algorithms with a sample oracle ($\mathcal{SO}$): AsyncQVI, Asynchronous-Parallel Q-learning with constant step size (AQLC), Asynchronous-Parallel Q-learning with diminishing step size (AQLD)[Tsi94], Variance-reduced Value Iteration (VRVI)[SWW18c], and Variance-reduced Q-value Iteration (VRQVI)[SWW18b]. All algorithms and the $\mathcal{SO}$ are implemented in C++11. We use the `thread` class and `pthread.h` for parallel computing.

The tests were performed with 20 threads running on two 2.5GHz 10-core Intel Xeon E5-2670v2 processors. We chose the optimal sample method (uniformly random, cyclic, Markovian sampling) and optimal hyperparameters (sample number, iteration number, learning rate, exploration rate) for each algorithm individually. The learning rate of AQLD

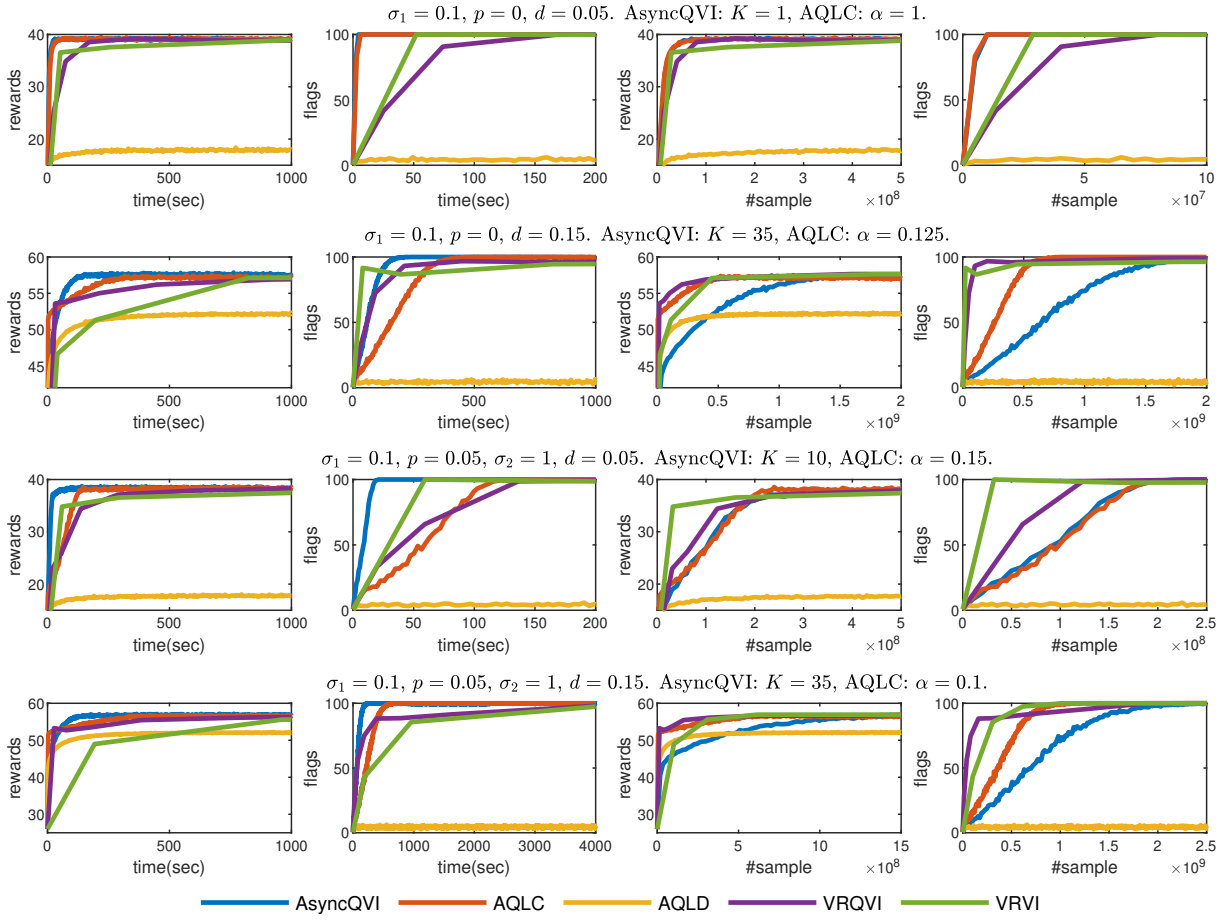

Figure 2.1: Parallel RL vs. single-threaded RL under various transitions.

was set as $1/t^{0.51}$ according to its theoretical analysis, where t is the iteration number. Our code is available in `https://github.com/uclaopt/AsyncQVI`.

To evaluate a policy, we let the agent start from a random initial state and take actions following the policy for 200 steps. Then, we evaluate the policy by recording the total discounted rewards ($\gamma = 0.99$) and whether the agent reaches the target position (flag $= 1$ if so). We repeat 100 episodes of this process and calculate the average total discounted rewards and total flags. A policy with higher rewards and more flags is preferred.

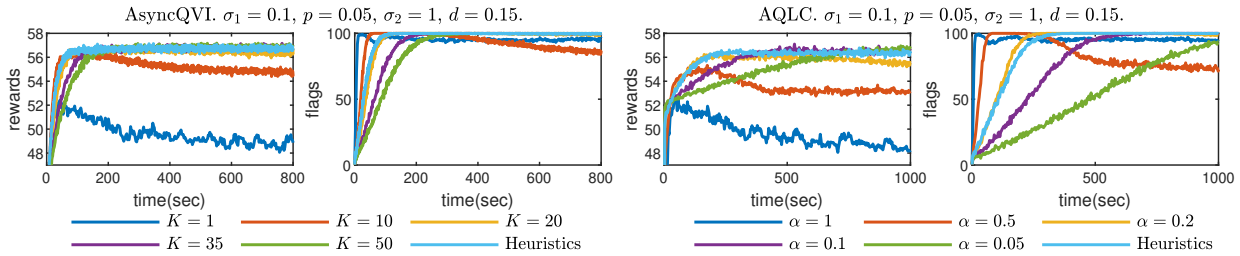

Figure 2.2: AsyncQVI vs. AQLC.

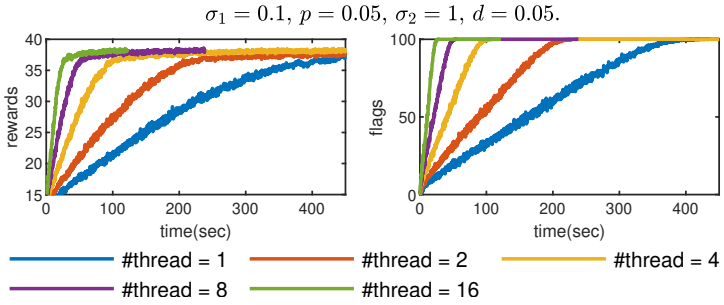

Figure 2.3: Parallel performance of AsyncQVI.

Results We test with different randomness and rewards which represent various MDP settings (see Figure 2.1). In the first test, one-step transition rewards are dominated by rewards for reaching the target ($d = 0.05$ is very small compared with 1) and only the wind noise is considered in positioning. The agent mainly aims at finding the target, which is relatively easy with minor noises. This leads to a fast convergence of policies with low sampling request and bold learning rate. In the second test, with increasing transition rewards ($d = 0.15$), the agent needs to take a more economical way to reach the goal. This prolongs the learning process with more samples and more prudent learning rate. The next two tests make the situation more complicated with a big vortex noise, which gives rise to higher sampling numbers and more conservative learning rates. This phenomenon occurs in VRVI and VRQVI as well. We skip the detailed parameters here.

In these four tests, AsyncQVI and AQLC are almost equivalently outstanding in terms of time and achieve an at least 10× speedup compared to VRQVI and VRVI with 20 threads running parallel. Further, VRQVI and VRVI have lower sample complexities, especially on

complicated cases. The testing results verify our theory. In the sequel, we further analyze the performance of AsyncQVI and AQLC and provide heuristics on how to set sample number and learning rate.

Performance Analysis and Heuristics Recall that AsyncQVI derives from the Q-value operator T (see Equation (2.1)). Let $T_\alpha := (1-\alpha)I + \alpha T$, where α is the learning rate. One can get AQLC through the same approach. What's special is, AQLC only takes one sample each time. This seems to be a very inaccurate approximation and might cause devastating error. However, note that when applying T_α, sample range is also discounted by α. For fixed δ and ϵ, the requested sample number m decreases quadratically with respect to α, since $m \geq C\frac{\alpha^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right)$ by Hoeffding's Inequality [Hoe63]. Hence, when α is smaller, AQLC converges more stably. On the other hand, a tiny learning rate also leads to slow progress, since T_α's contractive factor $(1-\alpha+\alpha\gamma)$ approaches 1. Similarly, for AsyncQVI, when the sample number K is larger, it converges more stably but also more slowly. Therefore, we propose a trade-off heuristic of adaptively increasing the sample number or decreasing the learning rate. Specifically, in our test, we set $K_t = \min(\lfloor t^{0.175} \rfloor, 35)$ for AsyncQVI and $\alpha_t = \max(t^{-0.1}, 0.1)$ for AQLC, where t is the iteration number. The results are depicted in Figure 2.2. The above interpretation also shows that AQLC is a special case of AsyncQVI (with T_α and $K = 1$), which explains the similarity in their optimal performances. However, since AsyncQVI takes $\frac{1}{|\mathcal{A}|}\times$ memory of AQLC, our algorithm is still preferable for high dimensional applications.

Parallel Performance We also test the parallel speedup performance of AsyncQVI using 1, 2, 4, 8, and 16 threads (see Figure 2.3). The result shows an ideal linear speedup.

Summary AsyncQVI and AQLC have similar numerical performance, and they are faster than VRQVI, VRVI and AQLD. In general, async algorithms speed and scale up very well as the number of threads increases, and AsyncQVI is not an exception. On the other hand,

AsyncQVI requires only $\mathcal{O}(|\mathcal{S}|)$ memory, which is much less than the $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ memory of the other three; recall Table 2.1. Therefore, AsyncQVI can solve much larger problem instances.

2.5 Conclusion

In this chapter, with finite observations and a generative model, we propose an async-parallel algorithm AsyncQVI to improve the computational efficiency of RL. Theoretically, under mild asynchronism conditions, our algorithm achieves near-optimal sample complexity and minimal memory requirement. Empirically, AsyncQVI shows an apparent speedup compared with related single-threaded methods. We expect this method as an efficient alternative for large-scale applications.

CHAPTER 3

Transfer RL with an Approximate Model

One of the key approaches to save samples in reinforcement learning (RL) is to use knowledge from an approximate model such as its simulator. However, *how much does an approximate model help to learn a near-optimal policy of the true unknown model*? Despite numerous empirical studies of transfer reinforcement learning, an answer to this question is still elusive. In this chapter, we study the sample complexity of RL while an approximate model of the environment is provided. For an unknown Markov decision process (MDP), we show that the approximate model can effectively reduce the complexity by eliminating sub-optimal actions from the policy searching space. In particular, we provide an algorithm that uses $\widetilde{O}(N/(1-\gamma)^3/\varepsilon^2)$ samples in a generative model to learn an ε-optimal policy, where γ is the discount factor and N is the number of near optimal actions in the approximate model. This can be much smaller than the learning-from-scratch complexity $\widetilde{\Theta}(SA/(1-\gamma)^3/\varepsilon^2)$, where S and A are the sizes of state and action spaces respectively. We also provide a lower bound showing that the above upper bound is nearly-tight if the value gap between near optimal actions and sub-optimal actions in the approximate model is sufficiently large. Our results provide a very precise characterization of how an approximate model helps reinforcement learning when no additional assumption on the model is posed.

The contributions in this chapter were first presented in the joint work with Wotao Yin and Lin Yang [FYY19].

3.1 Introduction

Reinforcement learning (RL) is the framework of learning to control an unknown system through trial and error. Recently, RL achieves phenomenal empirical successes, e.g, AlphaGo [SHM16] defeated the best human player in Go, and OpenAI used RL to precisely and robustly control a robotic arm [AWR17]. The RL framework is general enough such that it can capture a broad spectrum of topics, including health care, traffic control, and experimental design [SBW92, ERR19, SW01, Wie00]. However, *successful* applications of RL in these domains are still rare. The major obstacle that prevents RL being widely used is its high sample complexity: both the AlphaGo and OpenAI arm took nearly a thousand years of human-equivalent experiences to achieve good performances.

One way to reduce the number of training samples is to mimic how human beings learn – borrow knowledge from previous experiences. In robotics research, a robot may need to accomplish different tasks at different times. Instead of learning every task from scratch, a more ideal situation is that the robot can utilize the similarity among the underlying models of these tasks and adapt to future new jobs quickly. Another example is that RL agents are often trained in simulators and then applied to the real world [NCD06, Its95, DRC17]. It is desirable that agents learned from simulators (approximate models) can adapt to the real world (true model) faster than knowing nothing. Both examples lead to a natural question:

How does an approximate model help in RL?

This paper focuses on answering the above question. Suppose the true unknown model is a Markov decision process (MDP) $\mathcal{M}$ and the agent is provided with a prior model $\mathcal{M}_0$ with the same state and action spaces as $\mathcal{M}$ but different transition and reward functions. In particular, we assume

$$\text{dist}(\mathcal{M}_0, \mathcal{M}) \leq \beta,$$

where $\text{dist}(\cdot, \cdot)$ is a statistical distance and β is a small positive scalar. We would like to

study the sample complexity of learning an ε-optimal policy π for $\mathcal{M}$[1].

In this paper, we consider one of the most natural choices for $\mathrm{dist}(\cdot,\cdot)$, the total-variation (TV) distance between the transition kernels of $\mathcal{M}_0$ and $\mathcal{M}$ (see the formal definition in Equation (3.2)). Under such a distance, we establish both upper and lower sample complexity bounds. We utilize the fact that if two models are close under TV-distance, their optimal action-values are close under $\|\cdot\|_\infty$. Thus, given $\mathcal{M}_0$, we can build an interval estimation of the optimal action-value of $\mathcal{M}$. Based on this estimation, for every state, we construct two action sets: an *ε-sufficient set* and an *ε-necessary set*, where the former contains actions that are sufficient to construct an ε-optimal policy for any $\mathcal{M} \in \mathcal{B}_{\mathrm{TV}}(\mathcal{M}_0, \beta)$; while the latter contains actions we must explore or we fail to learn an ε-optimal policy for some $\mathcal{M} \in \mathcal{B}_{\mathrm{TV}}(\mathcal{M}_0, \beta)$. We use the notion of c-action set (see definition in Section 3.2) to define ε-sufficient and ε-necessary sets in a unified way. In particular, their relation is as depicted in Figure 3.1, which will be more revealed in Section 3.3. We establish the sample complexity results via conducting RL on these sets.

To build the upper bound, we develop a transfer RL algorithm that first constructs ε-sufficient sets using the prior knowledge, then focuses RL on them. Our algorithm takes $\widetilde{O}\big(\frac{\overline{N}}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\big)$ samples to learn an ε-optimal policy for any $\mathcal{M} \in \mathcal{B}_{\mathrm{TV}}(\mathcal{M}_0, \beta)$, where $\overline{N}$ is the sum of cardinality of ε-sufficient sets over all states. When $\overline{N}$ is smaller than the cardinality of the original action space, transfer learning outperforms learning-from-scratch sample-wisely. To build the lower bound, we leverage techniques for proving hardness in the bandit literature (e.g. [MT04]) and RL (e.g. [AMK13]) and obtain the result $\Omega\big(\frac{\underline{N}}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\big)$, where $\underline{N}$ is the sum of cardinality of ε-necessary action sets over all states. Both $\overline{N}$ and $\underline{N}$ depends on β, ε, and the optimal action-value of $\mathcal{M}_0$. In particular, $\overline{N} \approx \underline{N}$ when there is a large discrepancy in the action-value function. The lower bound shows that our upper bound is nearly tight under some condition. The reader is referred to Section 3.5.2 for more discussion.

[1] A policy is ε-optimal if the difference between the its value and the optimal value is at most ε.

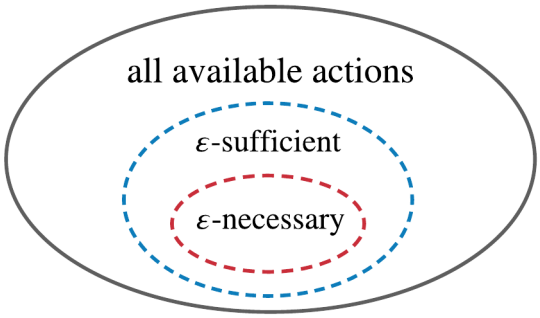

Figure 3.1: The Venn diagram for the ϵ-sufficient action set and the ϵ-necessary action set.

Our results do not rely on additional structure assumptions but only the intrinsic value property of MDPs. To the best of our knowledge, this is the first systematic theoretical answer to the aforementioned question in this setting.

3.1.1 Related Work

Transfer learning is an important strategy to reduce sample complexity in RL. There are many different learning regimes, e.g, multi-task RL [WFR07, LG10, BL13, AER14, CLR14], lifelong RL [TY97, BL14, AJG18, ZAT17], and meta-RL [SD03, ABB18, GML18, SHD18, RZF19]. Please also see surveys in [TS09, Laz12]; and [YZD20].

In the above settings, often more than one prior model (task) is considered. These models are assumed to share structural similarity with the to-be-learned model, or they are all generated from a common distribution. For instance, in [BL13], all models are assumed to be drawn from a finite set of MDPs; in [AJG18], all models share the same transition dynamics but reward functions change with a hidden distribution; in [MJT19] and [AJS20], every model's transition kernel and reward function lie in the linear span of several known base models.

In terms of how one approximate model can help, there are several theoretical works. In [Jia18], the authors use the number of incorrect state-action pairs to characterize the difference

between two models, which is another interesting direction to look at besides the TV-distance as we adopt. However, the statistical distance from such a model to the true model can be arbitrarily large. As the authors show, even if the difference is only one state-action pair, the approximate model could still be information-theoretically useless. Additional conditions are needed to achieve positive transfer. In [MC12], the authors analyzed action-value transfer via inter-task mappings to guide exploration. They do not measure the difference between models but show that as long as the optimal action function of the source task is (almost) larger than that of the target task component-wisely, one can use the former as an initialization for value-based RL on the true model and achieve a smaller sample complexity compared compared with learning-from-scratch. Their working mechanism is similar to ours: eliminate sub-optimal state-action pairs from later exploration. The difference is that we directly identify and remove sub-optimal actions while they achieve this implicitly through value functions. No lower bound is established there.

Compared with the aforementioned works, our results do not rely on additional assumptions and are more complete with both upper and lower bounds. In particular, our upper and lower bounds enjoy a unifying structure that is explicitly characterized by the value function, the TV-distance parameter, and the learning accuracy parameters.

Notation Given an integer K, we use $[K]$ to represent the set $\{1, 2, \ldots, K\}$. We use $|\cdot|$ to denote the cardinality of a set. We denote by O, Ω or Θ the leading order in upper, lower, or minimax optimal bounds respectively; and use $\widetilde{O}, \widetilde{\Omega}$ and $\widetilde{\Theta}$ to hide the polylog factors.

3.2 Setting

Markov decision process We consider an infinite-horizon discounted Markov decision process (MDP), $\mathcal{M} := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p, r, \gamma)$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}^s$ is the set of available actions for state s, $p(s'|s, a)$ is the transition kernel, $r(s, a, s') \in [0, 1]$ is the reward

function, and $\gamma \in (0,1)$ is a discount factor. We denote by $\mathcal{S}'$ the set of states with more than one available action.

At time step t, the controller observes a state s_t and selects an action $a_t \in \mathcal{A}^{s_t}$ according to a *policy* π, which maps a state to one of its available action. The environment then transitions to a new state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$ and the controller receives an instant reward $r(s_t, a_t, s_{t+1})$. Given a policy π, we define its value function as:

$$V^\pi(s) := \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s\big],$$

where the expectation is taken over the trajectory following π. The objective of RL is to learn a policy π^* that can maximize the value function, i.e., $\forall\ \pi, s \in \mathcal{S}:\ V^* := V^{\pi^*}(s) \geq V^\pi(s)$. A policy π is said to be *ε-optimal* if $V^\pi(s) \geq V^*(s) - \varepsilon$ for all $s \in \mathcal{S}$. The action-value function (or Q-function) of a policy is defined as

$$Q^\pi(s,a) := \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a\big].$$

Following the definition of V^π, we naturally have

$$Q^\pi(s,a) = \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \big(r(s,a,s') + \gamma V^\pi(s')\big).$$

The optimal Q-function is denoted by $Q^* := Q^{\pi^*}$. By Bellman Optimality Equation, we have

$$V^*(s) = \max_{a \in \mathcal{A}^s} Q^*(s,a) = \max_{a \in \mathcal{A}^s} \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \big(r(s,a,s') + \gamma V^*(s')\big).$$

c-action set Given an MDP $\mathcal{M}$ and a constant c, for each state s, we define the following set

$$\mathcal{A}^s_{\mathcal{M}}(c) := \begin{cases} \{a \mid V^*_{\mathcal{M}}(s) - Q^*_{\mathcal{M}}(s,a) < c\}, & c > 0; \\ \operatorname{argmax}_a Q^*_{\mathcal{M}}(s,a), & c \leq 0, \end{cases} \tag{3.1}$$

where $V^*_{\mathcal{M}}$ and $Q^*_{\mathcal{M}}$ denote the optimal value and Q-function of $\mathcal{M}$. For $\mathcal{A}^s_{\mathcal{M}}(c)$, we have the following properties:

- *Non-emptiness.* For any $c \in \mathbb{R}$, $\mathcal{A}^s_{\mathcal{M}}(c) \supseteq \operatorname{argmax}_a Q^*_{\mathcal{M}}(s,a)$;
- *Monotonicity.* If $c_1 > c_2$, then $\mathcal{A}^s_{\mathcal{M}}(c_1) \supseteq \mathcal{A}^s_{\mathcal{M}}(c_2)$;
- *Boundedness.* If $c > 1/(1-\gamma)$, $\mathcal{A}^s_{\mathcal{M}}(c) = \mathcal{A}^s_{\mathcal{M}}$, i.e., all available actions for s in $\mathcal{M}$.

TV-distance for MDPs Consider two MDPs $\mathcal{M}_0 := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p_0, r_0, \gamma)$ and $\mathcal{M} := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p, r, \gamma)$. We define

$$d_{\mathrm{TV}}(\mathcal{M}_0, \mathcal{M}) = \max\Big\{ \max_{s\in\mathcal{S}, a\in\mathcal{A}^s} \|p_0(\cdot|s,a) - p(\cdot|s,a)\|_1,\ \|r_0 - r\|_\infty \Big\}. \tag{3.2}$$

$d_{\mathrm{TV}}(\cdot,\cdot)$ is a metric between MDPs with the same state/action space and the discount factor. The name TV comes from that $\|p_0(\cdot|s,a) - p(\cdot|s,a)\|_1$ is equal to the *total variation* distance between distributions $p_0(\cdot|s,a)$ and $p(\cdot|s,a)$. We denote by $\mathcal{M} \in \mathcal{B}_{\mathrm{TV}}(\mathcal{M}_0, \beta)$ if $d_{\mathrm{TV}}(\mathcal{M}_0, \mathcal{M}) \le \beta$.

Generative model The generative model of $\mathcal{M} := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p, r, \gamma)$ is a special sample oracle. It allows any state-action pair $(s,a), a \in \mathcal{A}^s$ as input and outputs $(s', r(s,a,s'))$ with probability $p(s'|s,a)$.

Near-optimal RL algorithm An RL algorithm is near-optimal if without any prior knowledge, it returns an ε-optimal policy for an MDP $\mathcal{M} := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p, r, \gamma)$ with probability at least $1-\delta$ using $\widetilde{O}\left(\frac{\sum_{s\in\mathcal{S}}|\mathcal{A}^s|}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\right)$ samples. Near-optimal RL algorithms often require a generative model of $\mathcal{M}$. Examples can be found in [AMK13] and [SWW18a].

3.3 Main Results

We formalize our problem of knowledge transfer as below.

Problem 1. *Suppose the target unknown model is $\mathcal{M} := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p, r, \gamma)$ and an agent is provided with the full knowledge of an approximate model $\mathcal{M}_0 := (\mathcal{S}, \{\mathcal{A}^s\}_{s\in\mathcal{S}}, p_0, r_0, \gamma)$*

satisfying $\mathcal{M} \in B_{\mathrm{TV}}(\mathcal{M}_0, \beta)$, where $\beta > 0$ is a known constant. How many samples does it take to learn an ε-optimal policy for $\mathcal{M}$ with probability at least $1 - \delta$?

Without additional structure assumption, we answer Problem 1 by exploiting value functions. As mentioned before, we utilize the fact that if $d_{\mathrm{TV}}(\mathcal{M}, \mathcal{M}_0) \leq \beta$, then their optimal Q-functions are apart for at most $\beta/(1-\gamma)^2$ (see Lemma 3.4.1). Given full knowledge of $\mathcal{M}_0$, we can obtain interval estimation of the optimal Q-function of $\mathcal{M}$, which depicts how optimal and sub-optimal an action could be in $\mathcal{M}$.

To establish the upper bound, we select all actions with sufficiently large value estimation upper bounds and form ε-sufficient sets for all states. We prove in Section 3.4 that there exists at least one action in each ε-sufficient set that can construct an ε-optimal policy for any $\mathcal{M} \in \mathcal{B}_{\mathrm{TV}}(\mathcal{M}_0, \beta)$. At the learning stage, we apply an RL algorithm only on ε-sufficient sets. The whole process is summarized in Algorithm 5.

In Algorithm 5, we have an explicit definition of ε-sufficient sets: $\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$, for all $s \in \mathcal{S}$. From Equation (3.1), one can see that $\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$ contains actions whose values in $\mathcal{M}_0$ are only $\overline{C}$ away from the optimal. Such a constraint guarantees that these actions have a chance to be optimal in $\mathcal{M}$ since their value estimation upper bounds in $\mathcal{M}$ is larger than the lower bound of $\mathcal{M}_0$'s optimal actions. Hence, in the following we form a new MDP $\mathcal{M}^c$ with an action space $\{\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})\}_{s\in\mathcal{S}}$ (Line 5) and focus RL on $\mathcal{M}^c$ (Line 6). Notice that the transition kernel and reward function maintain, i.e.,

$$p^c(s'|s,a) = p(s'|s,a), \quad r^c(s,a,s') = r(s,a,s), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}^s_{\mathcal{M}_0}(\overline{C}), s' \in \mathcal{S}.$$

In Theorem 3.3.1, we show that the output policy is ε-optimal for $\mathcal{M}$ with high probability and the sample complexity therein is our upper bound result to Problem 1.

Theorem 3.3.1 (Main Result – Upper Bound)**.** *Given $\varepsilon > 0$ and $\delta \in (0,1)$, with probability at least $1 - \delta$, Algorithm 5 returns an ε-optimal policy for $\mathcal{M}$ using at most*

$$\widetilde{O}\left(\frac{\sum_{s\in\mathcal{S}'} |\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})|}{(1-\gamma)^3\varepsilon^2} \log\left(\frac{1}{\delta}\right)\right)$$

Algorithm 5 Transfer RL from an Approximate Model

1: **Input:** full knowledge of $\mathcal{M}_0$; a generative model of $\mathcal{M}$; a near-optimal RL algorithm $\mathscr{A}_{\text{opt}}$; $\varepsilon > 0$, $\delta \in (0,1)$.

2: **Define:** $\overline{C} := \min\{2/(1-\gamma), 2\beta/(1-\gamma)^2\} - \varepsilon(1-\gamma)/2$.

3: Apply any planning algorithm on $\mathcal{M}_0$ to get Q_0^*;

4: Construct $\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$ with Q_0^* (see Equation (3.1));

5: Form a new MDP $\mathcal{M}^c := (\mathcal{S}, \{\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})\}_{s\in\mathcal{S}}, p^c, r^c, \gamma)$;

6: Apply $\mathscr{A}_{\text{opt}}$ to learn an $\varepsilon/2$-optimal policy π of $\mathcal{M}^c$ with probability at least $1-\delta$.

7: **Output:** π.

samples, where $\overline{C} = \min\{2/(1-\gamma), 2\beta/(1-\gamma)^2\} - \varepsilon(1-\gamma)/2$.[2]

To establish the lower bound, we construct a hard case and use information theory to prove the necessity. The details are displayed in Subsection 3.4.2. The result is summarized in the following theorem.

Theorem 3.3.2 (Main Result – Lower Bound)**.** *Let* $\varepsilon \in (0, \varepsilon_0)$ *and* $\delta \in (0, \delta_0)$*. The sample complexity for Problem 1 is*

$$\Omega\bigg(\frac{\sum_{s\in\mathcal{S}'} |\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)|}{(1-\gamma)^3\varepsilon^2} \log\Big(\frac{1}{\delta}\Big)\bigg),$$

where $\varepsilon_0 = \beta\gamma \cdot \min(\frac{\beta}{32}, \frac{1-\gamma}{48\gamma}) \cdot \big(\min_{s\in\mathcal{S}} V^*(s)\big)^2$, $\delta_0 = 1/40$, *and*

$$\underline{C}_s = \begin{cases} V^*(s) - \frac{9}{12(1-\gamma)-64(1-\gamma)^2\varepsilon+4.5\beta\gamma}, & \text{if } \beta/2 + \frac{4\gamma-1}{3\gamma} \geq 1; \\ V^*(s) - \frac{V^*(s)^2}{V^*(s)+\beta\gamma V^*(s)^2+4\varepsilon(1+\gamma\beta V^*(s)/2)^2}, & o.w. \end{cases}$$

In Theorem 3.3.2, $\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)$ is our official definition of the ε-necessary set for each state $s \in \mathcal{S}$.

[2] *The complexity is summed over* $\mathcal{S}'$ *since no action exploration is needed for single-action states.*

Remark 6. *Both upper and lower bounds share a unified structure via the notion of a c-action set. When the problem parameters meet the conditions in Theorem 3.3.2,* $\underline{C}_s \leq \overline{C}$. *By the monotonicity property of c-action set (see Section 3.2), the relation as depicted in Figure 3.1 is revealed. More discussions on* $\overline{C}$ *and* $\underline{C}_s$ *can be found in Subsection 3.5.2.*

Note that the sample complexity of learning-from-scratch is $\widetilde{\Theta}\Big(\frac{\sum_{s\in\mathcal{S}'}|\mathcal{A}^s|}{\varepsilon^2(1-\gamma)^3}\log\big(\frac{1}{\delta}\big)\Big)$ (see [AMK13, SWW18b]). For the upper bound, if $|\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})| \ll |\mathcal{A}^s|$, transfer RL achieves a significantly smaller sample complexity than learning-from-scratch. The best case, as in Corollary 3, is when $|\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})| = 1$ for all $s \in \mathcal{S}$ and therefore, the optimal policy is transferable. For the lower bound, if $|\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)|$ is close to $|\mathcal{A}^s|$, the prior knowledge does not offer much help. The worst case, as in Corollary 4, is when $|\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)| = |\mathcal{A}^s|$, then with or without the prior knowledge, it has the same order of complexity. In particular, if $|\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})| \approx |\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)|$, the upper bound is tight up to a log factor.

Corollary 3 (The Best Scenario)**.** *Suppose for every state* $s \in \mathcal{S}'$, $|\operatorname{argmax}_a Q^*_{\mathcal{M}_0}(s,a)| = 1$ *and*

$$\big|V^*_{\mathcal{M}}(s) - \max\big\{Q^*_{\mathcal{M}}(s,a) \mid Q^*_{\mathcal{M}}(s,a) < V^*_{\mathcal{M}}(s), a \in \mathcal{A}^s\big\}\big| \geq 2\beta/(1-\gamma)^2 - \varepsilon/(1-\gamma).$$

Then $|\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})| = 1,\ \forall s \in \mathcal{S}$ *and the optimal policy for* $\mathcal{M}_0$ *is also optimal for* $\mathcal{M}$.

Corollary 4 (The Worst Scenario)**.** *Let* $\varepsilon \in \big(0, \varepsilon_0\big)$ *and* $\delta \in (0, \delta_0)$, *where* ε_0 *and* δ_0 *are as defined in Theorem 3.3.2. Suppose for every state* $s \in \mathcal{S}'$, $\operatorname{argmax}_a Q^*_{\mathcal{M}_0}(s,a) = \mathcal{A}^s$. *Then the sample complexity for Problem 1 has the same order as that of learning-from-scratch.*

3.4 Analysis

In this section, we prove the main results.

3.4.1 Proof of the Upper Bound

We follow the notations used in Problem 1 and always use Q_0^* and Q^* for the optimal Q-functions of $\mathcal{M}_0$ and $\mathcal{M}$, respectively. Our first result is

Lemma 3.4.1. $\|Q_0^* - Q^*\|_\infty \leq \min\{1/(1-\gamma), \beta/(1-\gamma)^2\}$.

Proof. Since $R(s,a,s') \in [0,1]$, the bound $1/(1-\gamma)$ is a direct result. For any policy π, denote by Q_0^π and Q^π the action-value functions and V_0^π and V^π the state-value functions of running π in $\mathcal{M}_0$ and $\mathcal{M}$, respectively. Then, by definition, for every $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
\left|Q_0^\pi(s,a) - Q^\pi(s,a)\right| &= \left|r_0(s,a) + \gamma p_0(\cdot|s,a)^\top V_0^\pi - r(s,a) - \gamma p(\cdot|s,a)^\top V^\pi\right| \\
&\leq \left|r_0(s,a) - r(s,a)\right| + \gamma\left|p_0(\cdot|s,a)^\top V_0^\pi - p(\cdot|s,a)^\top V_0^\pi\right| \\
&\quad + \gamma \cdot \left|p(\cdot|s,a)^\top V_0^\pi - p(\cdot|s,a)^\top V^\pi\right| \\
&\leq \beta + \gamma \cdot \left\|p_0(\cdot|s,a) - p(\cdot|s,a)\right\|_1 \cdot \left\|V_0^\pi\right\|_\infty + \gamma \cdot \left\|V_0^\pi - V^\pi\right\|_\infty \\
&\leq \beta + \gamma\beta/(1-\gamma) + \gamma\left\|Q_0^\pi - Q^\pi\right\|_\infty.
\end{aligned}$$

Thus, $\left\|Q_0^\pi - Q^\pi\right\|_\infty \leq \beta/(1-\gamma) + \gamma \cdot \left\|Q_0^\pi - Q^\pi\right\|_\infty$ and $\left\|Q_0^\pi - Q^\pi\right\|_\infty \leq \beta/(1-\gamma)^2$. Denote by π_0^* an optimal policy of $\mathcal{M}_0$ and π^* an optimal policy of $\mathcal{M}$. Then

$$Q_0^{\pi^*} - Q^{\pi^*} \leq Q_0^* - Q^* \leq Q_0^{\pi_0^*} - Q^{\pi_0^*}.$$

Thus, $\left\|Q_0^* - Q^*\right\|_\infty \leq \max\left\{\left\|Q_0^{\pi^*} - Q^{\pi^*}\right\|_\infty, \left\|Q_0^{\pi_0^*} - Q^{\pi_0^*}\right\|_\infty\right\} \leq \min\left\{\frac{1}{1-\gamma}, \frac{\beta}{(1-\gamma)^2}\right\}$. ■

Next, we provide a sufficient condition for constructing an ϵ-optimal policy of $\mathcal{M}$.

Lemma 3.4.2. *Given $\epsilon > 0$, for every state $s \in \mathcal{S}$, we denote by a^s an action in $\mathcal{A}^s$ such that $Q^*(s,a^s) \geq \max_a Q^*(s,a) - \epsilon(1-\gamma)$. Then $\pi(s) = a^s$ is an ϵ-optimal policy for $\mathcal{M}$.*

Proof. Since $\pi(s) = a^s$ for all $s \in \mathcal{S}$, we have

$$\begin{aligned} V^*(s) - V^\pi(s) &= \max_{a \in \mathcal{A}^s} Q^*(s,a) - Q^\pi(s,a^s) \\ &= \max_{a \in \mathcal{A}^s} Q^*(s,a) - Q^*(s,a^s) + Q^*(s,a^s) - Q^\pi(s,a^s) \\ &\le \epsilon(1-\gamma) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a^s)(V^*(s') - V^\pi(s')) \\ &\le \epsilon(1-\gamma) + \gamma \cdot \epsilon(1-\gamma) + \cdots \le \sum_{n=0}^{\infty} \epsilon(1-\gamma)\gamma^n = \epsilon, \end{aligned}$$

where the last line is an induction step. Note that the above inequality holds for all states. Thus, π is an ϵ-optimal policy for $\mathcal{M}$. ■

Lemma 3.4.2 shows that instead of searching over the whole action space we only need to focus on actions that satisfy the condition therein. Based on this, we construct the ε-sufficient sets as in the next lemma.

Lemma 3.4.3. *Given $\beta > 0$ and $\epsilon > 0$, let $C(\beta,\epsilon) := 2 \cdot \min\{1/(1-\gamma), \beta/(1-\gamma)^2\} - \epsilon(1-\gamma)$. For every state $s \in \mathcal{S}$, there exists an action in $\mathcal{A}^s_{\mathcal{M}_0}\big(C(\beta,\epsilon)\big)$ that satisfies the condition in Lemma 3.4.2.*

Proof of Lemma 3.4.3. Following Lemma 3.4.2, for every state s we need to find an action $a^s \in \mathcal{A}^s$ such that

$$Q^*(s,a^s) \ge \max_a Q^*(s,a) - \epsilon(1-\gamma). \tag{3.3}$$

Denote by $c_0 := \min\{1/(1-\gamma), \beta/(1-\gamma)^2\}$. If

$$\max_{a \in \mathcal{A}^s} Q^*(s,a) - \epsilon(1-\gamma) \le \max_{a' \in \mathcal{A}^s} Q_0^*(s,a') - c_0,$$

then by Lemma 3.4.1, any optimal action for state s in $\mathcal{M}_0$ satisfies Equation (3.3). Since $\mathcal{A}^s_{\mathcal{M}_0}(C(\beta,\epsilon))$ contains all these optimal actions, it is a valid searching set. If

$$\max_{a \in \mathcal{A}^s} Q^*(s,a) - \epsilon(1-\gamma) > \max_{a' \in \mathcal{A}^s} Q_0^*(s,a') - c_0,$$

then denoting by a^* an action such that $Q^*(s,a^*) = \max_a Q^*(s,a)$, we have

$$Q_0^*(s,a^*) \geq Q^*(s,a^*) - c_0 > \max_{a'} Q_0^*(s,a') - 2c_0 + \epsilon(1-\gamma),$$

i.e., $a^* \in \mathcal{A}^s_{\mathcal{M}_0}(C(\beta,\epsilon))$. Combining the above results, there exists an $\epsilon(1-\gamma)$-optimal action for state s in $\mathcal{M}$ in the set $\mathcal{A}^s_{\mathcal{M}_0}(C(\beta,\epsilon))$. ■

In Lemma 3.4.3, if we let $\epsilon = \varepsilon/2$, then $\mathcal{A}^s_{\mathcal{M}_0}\big(C(\beta,\epsilon)\big)$ is exactly the ε-sufficient set as used in Algorithm 5. Thus, combined with Lemma 3.4.2 and the fact that $\mathcal{M}^c$ shares the same transition and reward on the contracted action sets, an optimal policy of $\mathcal{M}^c$ is at least $\varepsilon/2$-optimal for $\mathcal{M}$. Since we output an $\varepsilon/2$-optimal policy for $\mathcal{M}^c$ in Algorithm 5, it is ε-optimal for $\mathcal{M}$. Theorem 3.3.1 is obtained.

3.4.2 Proof of the Lower Bound

Next, we prove Theorem 3.3.2. We first give a definition about the correctness of RL algorithms.

Definition 2. *($(\mathcal{M}_0,\beta,\varepsilon,\delta)$-correctness) Given $\beta > 0$ and a prior model $\mathcal{M}_0$, we say that an RL algorithm $\mathscr{A}$ is $(\mathcal{M}_0,\beta,\varepsilon,\delta)$-correct if for every $\mathcal{M} \in B_{\mathrm{TV}}(\mathcal{M}_0,\beta)$, $\mathscr{A}$ can output an ε-optimal policy with probability at least $1-\delta$.*

Next, we construct a class of hard MDPs. We will then select one model $\mathcal{M}_0$ from the class as prior and show that if an RL algorithm $\mathscr{A}$ learns with samples significantly fewer than the lower bound, there would always exist an MDP $\mathcal{M} \in B_{\mathrm{TV}}(\mathcal{M}_0,\beta)$ such that $\mathscr{A}$ cannot be $(\mathcal{M}_0,\beta,\varepsilon,\delta)$-correct.

Construction of the Hard Case We define a family of MDPs $\mathbb{M}$ with a structure as in Figure 3.2. The state space $\mathcal{S}$ consists of three disjoint subsets $\mathcal{X}$ (gray nodes), $\mathcal{Y}_1$ (green nodes), and $\mathcal{Y}_2$ (blue nodes). The set $\mathcal{X}$ includes K states $\{x_1, x_2, \ldots, x_K\}$ and each of them has L available actions $\{a_1, a_2, \ldots, a_L\} =: \mathcal{A}$. States in $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are all of single-action. For

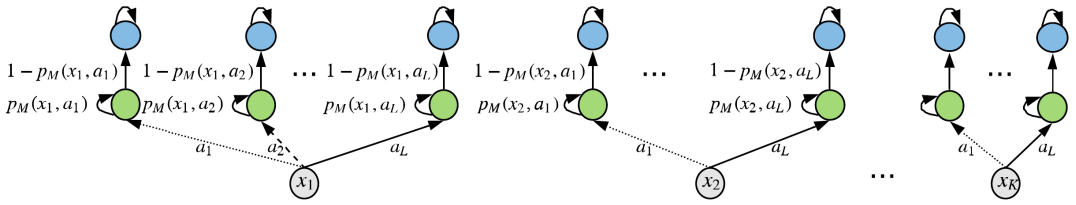

Figure 3.2: The class of MDPs considered to prove the lower bound in Theorem 3.3.2.

state $x \in \mathcal{X}$, by taking action $a \in \mathcal{A}$, it transitions to a state $y_1(x,a) \in \mathcal{Y}_1$ with probability 1. Note that such a mapping is one-to-one from $\mathcal{X} \times \mathcal{A}$ to $\mathcal{Y}_1$. For state $y_1(x,a) \in \mathcal{Y}_1$, it transitions to itself with probability $p_{\mathcal{M}}(x,a) \in (0,1)$ and to a corresponding state $y_2(y_1) \in \mathcal{Y}_2$ with probability $1 - p_{\mathcal{M}}(x,a)$. $p_{\mathcal{M}}(x,a)$ can be different for different models. All states in $\mathcal{Y}_2$ are absorbing. The reward function $R(s,a,s') = 1$ if $s' \in \mathcal{Y}_1$; otherwise 0.

$\mathbb{M}$ is a generalization of a multi-armed bandit problem used in [MT04] to prove a lower bound on bandit learning. A similar example is also shown in [AMK13]. For an MDP $\mathcal{M} \in \mathbb{M}$, it is fully determined by the parameter set $\{p_{\mathcal{M}}(x_k,a_l), k \in [K], l \in [L]\}$. And its Q-function is $Q_{\mathcal{M}}(x,a) = 1/(1-\gamma p_{\mathcal{M}}(x,a)), \forall\ (x,a) \in \mathcal{X} \times \mathcal{A}$.

In the sequel, we restrict $\beta \in (0,2)$ (by definition in Equation 3.2, $d_{\mathrm{TV}}(\cdot,\cdot) \leq 2$). Fixing β, we further restrict the discount factor $\gamma \in \big(\max\{0.4, 1-10\beta\}, 1\big)$.

Prior Model $\mathcal{M}_0$ and Hypotheses of $\mathcal{M}$ For $\mathcal{M}_0$, we simplify the notation $p_{\mathcal{M}_0}(x_k,a_l)$ as $p_0(x_k,a_l)$. Without loss of generality, we assume $1 > p_0(x_k,a_1) \geq p_0(x_k,a_2) \geq \cdots \geq p_0(x_k,a_L) \geq 0$, for every $x_k \in \mathcal{X}$. Thus, in $\mathcal{M}_0$, the Q-values from a_1 to a_L are monotonically non-increasing. Given β and γ, we require $p_0(x_k,a_1) \in \big(\frac{4\gamma-1}{3\gamma}, 1\big)$ for every $x_k \in \mathcal{X}$.

After selecting a prior model $\mathcal{M}_0$ satisfying the above conditions, we define $p_0^k := \max\{p_0(x_k,a_1) - \beta/2, \frac{4\gamma-1}{3\gamma}\}$ for every $k \in [K]$. Then let $\varepsilon_0 := \min_{k\in[K]} \big\{\beta\gamma(1-p_0^k)/(16(1-\gamma p_0^k)^2)\big\}$. Fixing $\varepsilon \in (0,\varepsilon_0)$, we define two numbers α_1^k and $\alpha_2^k := 4(1-\gamma p_0^k)^2\varepsilon/\gamma$ such that

$$\frac{1}{1-\gamma(p_0^k+\alpha_1^k)} - \frac{1}{1-\gamma p_0^k} = 2\varepsilon \quad \text{and} \quad \frac{1}{1-\gamma(p_0^k+\alpha_2^k)} - \frac{1}{1-\gamma(p_0^k+\alpha_1^k)} \geq 2\varepsilon. \quad (3.4)$$

Now, we are ready to define possibilities of $\mathcal{M}$. Given K integers $\{L_k \in [L]\}_{k\in[K]}$, let

$$\mathcal{M}_1 : \text{for every } k \in [K], \begin{cases} p_{\mathcal{M}_1}(x_k, a_1) = p_0^k + \alpha_1^k, \\ p_{\mathcal{M}_1}(x_k, a_l) = p_0^k, & 2 \le l \le L_k, \\ p_{\mathcal{M}_1}(x_k, a_l) = p_0(x_k, a_l), & l > L_k; \end{cases} \tag{3.5}$$

$$\text{for every } k \in [K], 1 < l \le L_k,\ \mathcal{M}_{k,l} : \begin{cases} p_{\mathcal{M}_{k,l}}(x_k, a_l) = p_0^k + \alpha_2^k \\ p_{\mathcal{M}_{k,l}}(x_{k'}, a_{l'}) = p_{\mathcal{M}_1}(x_{k'}, a_{l'}), \quad \forall\ (k', l') \neq (k, l). \end{cases}$$

Due to our specific selection of p_0^k, α_1^k and α_2^k, we have the following lemma.

Lemma 3.4.4. *Let* $L_k := \left|\left\{l \in [L] \;\middle|\; | \ p_0^k + \alpha_2^k - p_0(x_k, a_l) \ | \le \beta/2\right\}\right|$ *for every* $k \in [K]$. *All possibilities defined in* (3.5) *lie in* $B_{TV}(\mathcal{M}_0, \beta)$.

To prove Lemma 3.4.4, we first introduce the following lemma.

Lemma 3.4.5. *Following the notation in Lemma 3.4.4, we have*

$$\left\{l \in [L] \;\middle|\; | \ p_0^k + \alpha_2^k - p_0(x_k, a_l) \ | \le \beta/2\right\} = [L_k],\ \forall k \in [K].$$

Proof of Lemma 3.4.5. Since $p_0(x_k, a_1) \ge p_0(x_k, a_2) \ge \cdots \ge p_0(x_k, a_l)$, the set $\left\{l \in [L] \;\middle|\; | \ p_0^k + \alpha_2^k - p_0(x_k, a_l) \ | \le \beta/2\right\}$ contains consecutive integers. Now we only need to show that $l = 1$ is contained in the set. By definition of p_0^k, it holds that $p_0(x_k, a_1) - \beta/2 \le p_0^k < p_0(x_k, a_1)$. When $\varepsilon \in (0, \varepsilon_0)$, we have $0 < \alpha_2^k < \beta/2$. Thus, $l = 1$ is in the set. ■

Based on Lemma 3.4.5, we can prove Lemma 3.4.4.

Proof of Lemma 3.4.4. We first verify $\mathcal{M}_1 \in B_{\mathrm{TV}}(\mathcal{M}_0, \beta)$. When $\varepsilon \in (0, \varepsilon_0)$, by definition, we have $0 < \alpha_1^k < \alpha_2^k < \beta/2$ and $p_0(x_k, a_1) - \beta/2 \le p_0^k < p_0(x_k, a_1)$. Then it holds that

$$p_0(x_k, a_1) - \beta/2 < p_0^k + \alpha_1^k < p_0(x_k, a_1) + \alpha_1^k < p_0(x_k, a_1) + \beta/2.$$

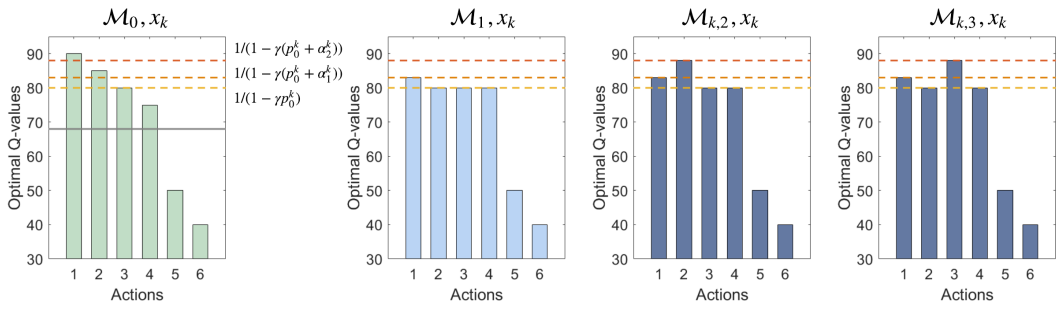

Figure 3.3: The optimal Q-values of $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_{k,2}$, and $\mathcal{M}_{k,3}$ at state x_k with 6 actions. The dashed lines indicate the values: $\frac{1}{1-\gamma p_0^k}$, $\frac{1}{1-\gamma(p_0^k+\alpha_1^k)}$, and $\frac{1}{1-\gamma(p_0^k+\alpha_2^k)}$, respectively. Actions above the grey line in $\mathcal{M}_0$ are in the set for the definition of L_k in Lemma 3.4.4. Note that for states $x_{k'}$ $(k' \neq k)$, $\mathcal{M}_{k,2}$ and $\mathcal{M}_{k,3}$ have the same shape as $\mathcal{M}_1$.

Thus, $|p_0^k + \alpha_1^k - p_0(x_k, a_1)| \leq \beta/2$. For $2 \leq l \leq L_k$, if $p_0(x_k, a_l) \geq p_0^k$, then $p_0(x_k, a_l) - p_0^k \leq p_0(x_k, a_1) - p_0^k \leq \beta/2$; otherwise, $p_0^k - p_0(x_k, a_l) \leq p_0^k + \alpha_2^k - p_0(x_k, a_l) \leq \beta/2$ (Lemma 3.4.5). Hence, $\mathcal{M}_1 \in B_{\mathrm{TV}}(\mathcal{M}_0, \beta)$. For each $\mathcal{M}_{k,l}$, the validity directly follows Lemma 3.4.5 and $\mathcal{M}_1 \in B_{\mathrm{TV}}(\mathcal{M}_0, \beta)$. ■

We refer to the models in Equation (3.5) as *hypotheses* of $\mathcal{M}$. There are in total $1 + \sum_{k\in[K]}(L_k - 1)$ of them. In $\mathcal{M}_1$, for every $x_k \in \mathcal{X}$, a_1 is the optimal action and a_2 to a_{L_k} are the second-best actions with a value only 2ε less than the optimal (Equation (3.4)). $\{\mathcal{M}_{k,l}\}$ are built on top of $\mathcal{M}_1$ by raising up the value of the action a_l at state x_k by at least 4ε (Equation (3.4)). Thus, in $\mathcal{M}_{k,l}$, the best action for x_k is a_l and the best action for $x_{k'}$ $(k' \neq k)$ is still a_1. See Figure 3.3 for illustration. Every hypothesis gives a probability measure over the same sample space. We denote by $\mathbb{E}_1$, $\mathbb{P}_1$ and $\mathbb{E}_{k,l}$, $\mathbb{P}_{k,l}$ the expectation and probability under hypothesis $\mathcal{M}_1$ and $\mathcal{M}_{k,l}$, respectively. These probability measures capture both randomness in the MDP and the randomization in an RL algorithm like its sampling strategy.

We fix $\varepsilon \in (0, \varepsilon_0)$ and $\delta \in (0, 1/40)$. Let $\mathscr{A}$ be an $(\mathcal{M}_0, \beta, \varepsilon, \delta)$-correct RL algorithm. We denote by $T_{k,l}$ the number of samples that algorithm $\mathscr{A}$ calls from the generative model with

input state $y_1(x_k, a_l)$ till $\mathscr{A}$ stops (these sample calls are not necessarily consecutive). For every $k \in [K], 1 < l \leq L_k$, we define an event $E_{k,l} = \{\mathscr{A}$ outputs a policy π with $\pi(x_k) = a_1\}$. Then we have the following key lemma.

Lemma 3.4.6. *For any $k \in [K], 1 < l \leq L_k$, if $\mathbb{E}_1[T_{k,l}] < \frac{c_1}{(1-\gamma)^3\varepsilon^2} \log\left(\frac{1}{4\delta}\right)$, $\mathbb{P}_{k,l}(E_{k,l}) > \delta$, where $c_1 > 0$ is some large constant.*

To prove Lemma 3.4.6, we need some definitions. Let

$$t^* = \frac{c_1}{(1-\gamma)^3\varepsilon^2} \log\left(\frac{1}{4\delta}\right),$$

where $c_1 > 0$ is to be determined later. We denote by $T_{k,l}$ the number of samples that algorithm $\mathscr{A}$ calls from the generative model with input state $y_1(x_k, a_l)$ till $\mathscr{A}$ stops (these sample calls are not necessarily consecutive). For every $k \in [K], 1 < l \leq L_k$, we define the following events:

$$A_{k,l} = \{T_{k,l} \leq 4t^*\}, \quad E_{k,l} = \{\mathscr{A} \text{ outputs a policy } \pi \text{ with } \pi(x_k) = a_1\},$$
$$C_{k,l} = \Big\{ \max_{1 \leq T_{k,l} \leq 4t^*} \left|p_0^k \cdot T_{k,l} - S_{k,l}(T_{k,l})\right| \leq \sqrt{16t^* \cdot p_0^k \cdot (1-p_0^k)\log(1/4\delta)} \Big\},$$

where $S_{k,l}(T_{k,l})$ is the sum of rewards (non-discounted) by calling the generative model $T_{k,l}$ times with input state $y_1(x_k, a_l)$. For these events, we have the following lemmas.

Lemma 3.4.7. *For any $k \in [K], 1 < l \leq L_k$, if $\mathbb{E}_1[T_{k,l}] \leq t^*$, $\mathbb{P}_1(A_{k,l}) > 3/4$.*

Proof.

$$t^* \geq \mathbb{E}_1[T_{k,l}] > 4t^*\mathbb{P}_1(T_{k,l} > 4t^*) = 4t^*(1 - \mathbb{P}_1(T_{k,l} \leq 4t^*)).$$

Thus, $\mathbb{P}_1(A_{k,l}) > 3/4$. ■

Lemma 3.4.8. *For any $k \in [K], 1 < l \leq L_k$, $\mathbb{P}_1(C_{k,l}) > 3/4$.*

Proof. Let $\epsilon := \sqrt{16t^* \cdot p_0^k \cdot (1-p_0^k)\log(1/4\delta)}$. When $1 < l \leq L_k$, under hypothesis $\mathcal{M}_1$, $p_{\mathcal{M}_1}(x_k, a_l) = p_0^k$. By definition, the instant rewards from state $y_1(x_k, a_l)$ are i.i.d. Bernoulli(p_0^k)

random variables and $p_0^k \cdot T_{k,l} - S_{k,l}(T_{k,l})$ is a martingale. Using Doob's inequality ([Dur19, Theorem 4.4.2]), we have the following bound:

$$\mathbb{P}_1\Big(\max_{1\le T_{k,l}\le 4t^*} \big|p_0^k T_{k,l} - S_{k,l}(T_{k,l})\big| \ge \sqrt{16t^* p_0^k(1-p_0^k)\log\frac{1}{4\delta}}\Big) \le \frac{\mathbb{E}_1\Big[\big(4t^*\cdot p_0^k - S_{k,l}(4t^*)\big)^2\Big]}{16t^*\cdot p_0^k(1-p_0^k)\log(1/4\delta)}.$$

Since $\mathbb{E}_1[(4t^*\cdot p_0^k - S_{k,l}(4t^*))^2] = 4t^* p_0^k(1-p_0^k)$ and $\delta < 1/40$, we obtain that

$$\mathbb{P}_1(C_{k,l}) \ge 1 - 1/(4\log(1/4\delta)) > 3/4.$$

■

Since $\mathscr{A}$ is $(\mathcal{M}_0, \beta, \varepsilon, \delta)$-correct, it should return a policy π such that when $\mathcal{M} = \mathcal{M}_1$, $\pi(x_k) = a_1$ for every $k \in [K]$ with probability at least $1-\delta$, i.e. $\mathbb{P}_1\big(\ E_{k,l},\ \text{for all } k\in[K], 1 < l \le L_k\ \big) \ge 1-\delta > 3/4$. We define the event $\mathcal{E}_{k,l} := A_{k,l} \cap E_{k,l} \cap C_{k,l}$. Combining the results above, it holds that

$$\mathbb{P}_1(\mathcal{E}_{k,l}) > 1 - 3/4 = 1/4, \quad \forall\ k\in[K],\ 1 < l \le L_k.$$

Based on the above results, we show the correctness of Lemma 3.4.6.

Proof of Lemma 3.4.6. Given $k \in [K]$ and $1 < l \le L_k$, we denote by W the length-$T_{k,l}$ random sequence of the instant rewards by calling the generative model $T_{k,l}$ times with the input state $y_1(x_k, a_l)$. If $\mathcal{M} = \mathcal{M}_1$, this is an i.i.d. Bernoulli(p_0^k) sequence; if $\mathcal{M} = \mathcal{M}_{k,l}$, this is an i.i.d Bernoulli$(p_0^k + \alpha_2^k)$ sequence. We define the likelihood function $L_{k,l}$ as

$$L_{k,l}(w) = \mathbb{P}_{k,l}(W = w)$$

for every possible realization w. We simplify the previous notation $S_{k,l}(T_{k,l})$ as $S_{k,l}$. Then we compute the following likelihood ratio

$$\begin{aligned}\frac{L_{k,l}(W)}{L_1(W)} =& \frac{(p_0^k+\alpha_2^k)^{S_{k,l}}(1-p_0^k-\alpha_2^k)^{T_{k,l}-S_{k,l}}}{(p_0^k)^{S_{k,l}}(1-p_0^k)^{T_{k,l}-S_{k,l}}} = \left(1+\frac{\alpha_2^k}{p_0^k}\right)^{S_{k,l}}\left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{T_{k,l}-S_{k,l}} \\ =& \left(1+\frac{\alpha_2^k}{p_0^k}\right)^{S_{k,l}}\left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{S_{k,l}\frac{1-p_0^k}{p_0^k}}\left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{T_{k,l}-S_{k,l}/p_0^k}.\end{aligned}$$

Since $\gamma > 0.4$ and $p_0^k \geq \frac{4\gamma-1}{3\gamma}$, we have $p_0^k > 1/2$. By our choice of α_2^k and ε, it holds that $\alpha_2^k/(1-p_0^k) \leq \beta/4 \in (0, 1/2]$ and $\alpha_2^k/p_0^k \leq \beta(1-p_0^k)/(4p_0^k) \in (0, 1/2)$. With the fact that $\log(1-u) \geq -u - u^2$ for $u \in [0, 1/2]$ and $\exp(-u) \geq 1-u$ for $u \in [0, 1]$, we have that

$$\begin{aligned}\left(1 - \frac{\alpha_2^k}{1-p_0^k}\right)^{\frac{1-p_0^k}{p_0^k}} &\geq \exp\left(\frac{1-p_0^k}{p_0^k}\left(-\frac{\alpha_2^k}{1-p_0^k} - \left(\frac{\alpha_2^k}{1-p_0^k}\right)^2\right)\right) \\ &\geq \left(1 - \frac{\alpha_2^k}{p_0^k}\right)\left(1 - \frac{(\alpha_2^k)^2}{p_0^k(1-p_0^k)}\right).\end{aligned}$$

Thus,

$$\begin{aligned}\frac{L_{k,l}(W)}{L_1(W)} &\geq \left(1 - \frac{(\alpha_2^k)^2}{(p_0^k)^2}\right)^{S_{k,l}} \left(1 - \frac{(\alpha_2^k)^2}{p_0^k \cdot (1-p_0^k)}\right)^{S_{k,l}} \left(1 - \frac{\alpha_2^k}{1-p_0^k}\right)^{T_{k,l} - S_{k,l}/p_0^k} \\ &\geq \left(1 - \frac{(\alpha_2^k)^2}{(p_0^k)^2}\right)^{T_{k,l}} \left(1 - \frac{(\alpha_2^k)^2}{p_0^k \cdot (1-p_0^k)}\right)^{T_{k,l}} \left(1 - \frac{\alpha_2^k}{1-p_0^k}\right)^{T_{k,l} - S_{k,l}/p_0^k}\end{aligned}$$

due to $S_{k,l} \leq T_{k,l}$. Next, we proceed on the event $\mathcal{E}_{k,l}$. By definition, if $\mathcal{E}_{k,l}$ occurs, $A_{k,l}$ also occurs. Using $\log(1-u) \geq -2u$ for $u \in [0, 1/2]$, it follows that

$$\begin{aligned}\left(1 - \frac{(\alpha_2^k)^2}{(p_0^k)^2}\right)^{T_{k,l}} &\geq \left(1 - \frac{(\alpha_2^k)^2}{(p_0^k)^2}\right)^{4t^*} \geq \exp\left(-8t^* \frac{(\alpha_2^k)^2}{(p_0^k)^2}\right) \\ &= \exp\left(\frac{-8c_1 \log(1/4\delta)}{(1-\gamma)^3\varepsilon^2} \cdot \frac{16(1-\gamma p_0^k)^4 \varepsilon^2}{\gamma^2 (p_0^k)^2}\right) \\ &\geq \exp\Big(-128c_1 \log(1/4\delta) \frac{(1 - \frac{4\gamma-1}{3})^4}{(1-\gamma)^3\gamma^2(\frac{4\gamma-1}{3\gamma})^2}\Big) \\ &= \exp\Big(-128c_1 \log(1/4\delta) \frac{256(1-\gamma)}{9(4\gamma-1)^2}\Big) \\ &\geq \exp\Big(-128c_1 \log(1/4\delta) \frac{256}{9 * 0.6}\Big) \geq (4\delta)^{6100c_1},\end{aligned}$$

where the second line follows $p_0^k \geq \frac{4\gamma-1}{3\gamma}$. Using $\log(1-u) \geq -2u$ for $u \in [0, 1/2]$, we also

obtain

$$\begin{aligned}\left(1-\frac{(\alpha_2^k)^2}{p_0^k\cdot(1-p_0^k)}\right)^{T_{k,l}} &\geq \left(1-\frac{(\alpha_2^k)^2}{p_0^k\cdot(1-p_0^k)}\right)^{4t^*} \geq \exp\left(-8t^*\frac{(\alpha_2^k)^2}{p_0^k(1-p_0^k)}\right)\\ &= \exp\left(-8\frac{c_1}{(1-\gamma)^3\varepsilon^2}\log(1/4\delta)\frac{16(1-\gamma p_0^k)^4\varepsilon^2}{\gamma^2 p_0^k(1-p_0^k)}\right)\\ &\geq \exp\Big(-128c_1\log(1/4\delta)\frac{(1-\frac{4\gamma-1}{3})^4}{(1-\gamma)^3\gamma^2(\frac{4\gamma-1}{3\gamma})\min\{\frac{1-\gamma}{3\gamma},\beta/2\}}\Big)\\ &= \exp\Big(-128c_1\log(1/4\delta)\frac{256}{27\gamma(4\gamma-1)}\cdot\frac{1-\gamma}{\min\{\frac{1-\gamma}{3\gamma},\beta/2\}}\Big)\\ &\geq \exp\Big(-128c_1\log(1/4\delta)\frac{256\cdot 20}{27\gamma(4\gamma-1)}\Big)\\ &\geq \exp\Big(-128c_1\log(1/4\delta)\frac{5120}{27*0.4*0.6}\Big) \geq (4\delta)^{102000c_1},\end{aligned}$$

where the third line follows $1-p_0^k \geq \min\{1-\frac{4\gamma-1}{3\gamma}, 1-p_0(x_k,a_1)+\beta/2\} \geq \min\{\frac{1-\gamma}{3\gamma},\beta/2\}$ and the fifth line is due to $\frac{1-\gamma}{\min\{\frac{1-\gamma}{3\gamma},\beta/2\}} \leq \max\{3\gamma,20\} = 20$ (since $\gamma > 1-2\beta$). Further, when $\mathcal{E}_{k,l}$ occurs, $A_{k,l}$ and $C_{k,l}$ both occur. Therefore, following similar steps, we have

$$\begin{aligned}\left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{T_{k,l}-S_{k,l}/p_0^k} &\geq \left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{\max_{1\leq T_{k,l}\leq 4t^*}|T_{k,l}-S_{k,l}/p_0^k|}\\ &\geq \left(1-\frac{\alpha_2^k}{1-p_0^k}\right)^{\sqrt{16t^*\frac{1-p_0^k}{p_0^k}\log(1/4\delta)}}\\ &\geq \exp\left(-\sqrt{64\frac{(\alpha_2^k)^2}{p_0^k(1-p_0^k)}t^*\log(1/4\delta)}\right) \geq (4\delta)^{\sqrt{810000c_1}}.\end{aligned}$$

In total, we have $L_{k,l}(W)/L_1(W) \geq (4\delta)^{108100c_1+\sqrt{810000c_1}}$. By taking c_1 small enough, e.g. $c_1 = 5e^{-7}$, we have $L_{k,l}(W)/L_1(W) > 4\delta$. By a change of measure,

$$\mathbb{P}_{k,l}(E_{k,l}) \geq \mathbb{P}_{k,l}(\mathcal{E}_{k,l}) = \mathbb{E}_{k,l}[\mathbf{1}_{\mathcal{E}_{k,l}}] = \mathbb{E}_1\left[\frac{L_{k,l}(W)}{L_1(W)}\mathbf{1}_{\mathcal{E}_{k,l}}\right] > 4\delta * 1/4 = \delta.$$

■

Based on Lemma 3.4.6, we can prove Theorem 3.3.2. The main idea is that if $\mathscr{A}$ is $(\mathcal{M}_0,\beta,\varepsilon,\delta)$-correct, it should return a policy π such that $\pi(x_k)=a_l$ with probability $\geq 1-\delta$ under hypothesis $\mathcal{M}_{k,l}$, i.e., $\mathbb{P}_{k,l}(E_{k,l}) < \delta$. By Lemma 3.4.6, this requires $\mathbb{E}_1[T_{k,l}] > t^*$ for all

$k \in [K], 1 < l \leq L_k$. Thus, in total we need $\Omega\left(\frac{\sum_{k\in[K]} L_k}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\right)$ samples. Following the definition of L_k, we can recover the ε-necessary sets and obtain Theorem 2. The proof is displayed below.Note that as any online algorithm can be realized in the generative model setting, the lower bound automatically adapts to the online setting.

Proof of Theorem 3.3.2. Since $\mathscr{A}$ is $(\mathcal{M}^0, \beta, \varepsilon, \delta)$-correct, under hypothesis $\mathcal{M}_{k,l}$, $\mathscr{A}$ should produce a policy π such that $\pi(x_k) = a_l$ with probability $\geq 1-\delta$. Thus, we should have $\mathbb{P}_{k,l}(E_{k,l}) < \delta$ for all $k \in [K], 1 < l \leq L_k$. From Lemma 3.4.6, it requires $\mathbb{E}_1[T_{k,l}] > t^*$ for all $k \in [K], 1 < l \leq L_k$. In total, we need $\Omega\left(\frac{\sum_{k\in[K]} L_k}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\right)$ samples. By definition of L_k, for every $k \in [K]$, we have

$$\begin{aligned}\{a_l, l \leq L_k\} &= \mathcal{A}^{x_k}_{\mathcal{M}_0}\Big(\frac{1}{1-\gamma p_0(x_k,a_1)} - \frac{1}{1-\gamma(p_0^k+\alpha_2^k-\beta/2)}\Big)\\ &= \mathcal{A}^{x_k}_{\mathcal{M}_0}\Big(V^*(x_k) - \frac{1}{1-\gamma p_0^k - 4(1-\gamma p_0^k)^2\varepsilon + \beta\gamma/2}\Big).\end{aligned}$$

If $p_0^k = \frac{4\gamma-1}{3\gamma}$, i.e. $p_0(x_k,a_1) - \beta/2 \leq \frac{4\gamma-1}{3\gamma}$, then

$$L_k = \Big|\mathcal{A}^{x_k}_{\mathcal{M}_0}\Big(V^*(x_k) - \frac{9}{12(1-\gamma) - 64(1-\gamma)^2\varepsilon + 4.5\beta\gamma}\Big)\Big|. \tag{3.6}$$

If $p_0^k = p_0(x_k,a_1) - \beta/2$, i.e. $p_0(x_k,a_1) - \beta/2 > \frac{4\gamma-1}{3\gamma}$, then

$$\begin{aligned}L_k &= \Big|\mathcal{A}^{x_k}_{\mathcal{M}_0}\Big(V^*(x_k) - \frac{1}{1-\gamma p_0(x_k,a_1) + \gamma\beta - 4\varepsilon(1-\gamma p_0(x_k,a_1)+\gamma\beta/2)^2}\Big)\Big| \qquad (3.7)\\ &= \Big|\mathcal{A}^{x_k}_{\mathcal{M}_0}\Big(V^*(x_k) - \frac{V^*(x_k)^2}{V^*(x_k) + \gamma\beta(V^*(x_k))^2 - 4\varepsilon(1+\gamma\beta V^*(x_k)/2)^2}\Big)\Big|\end{aligned}$$

Combining Equation (3.6) and (3.7), we have

$$\Omega\left(\frac{\sum_{k\in[K]} L_k}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\right) = \Omega\left(\frac{\sum_{s\in\mathcal{S}'}|\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)|}{(1-\gamma)^3\varepsilon^2}\log(1/\delta)\right),$$

which concludes our proof of the lower bound result in Theorem 3.3.2. ■

Note that as any online algorithm can be realized in the generative model setting, the lower bound automatically adapts to the online setting. In Corollary 4, we have $L_k = |\mathcal{A}^s|$, which happens since there is no gap between values in $\mathcal{M}_0$. Thus, for any $c > 0$, $|\mathcal{A}^s_{\mathcal{M}_0}(c)| = |\mathcal{A}^s|$.

3.5 Further Discussion

In this section, we look into the main results in a graphical way. In 3.5.1, we depict the parameter relations in the upper and lower bounds; in 3.5.2, we compare the values of $\overline{C}$ and $\underline{C}_s$ and discuss when can the upper bound be nearly tight; in 3.5.3, we illustrate the scenario when an approximate model does not help.

3.5.1 Parameter Relation

In our upper and lower bounds, there are several counterforces and constraints among the parameters: 1. the upper bound is affected by the ratio $\beta/(1-\gamma)$; 2. given $\beta \in (0,2)$, we restrict $\gamma \in (\max\{0.4, 1-10\beta\}, 1)$ to establish the hard case for the lower bound; 3. we select the value of p_0^k in the hard case based on whether $p_0(x_k, a_1) \leq \beta/2 + \frac{4\gamma-1}{3\gamma}$; 4. ε is required to be smaller than ε_0 in the hard case. We depict these relations in Figure 3.4.

In the left graph of Figure 3.4, one can see that only in the area below the orange line, i.e., $\beta < (1-\gamma)$, the upper bound in Theorem 3.3.1, is determined by β; otherwise, the upper bound is trivial. For the lower bound, when $\beta < 1$, with a properly chosen γ (the blue area below the blue solid line), we can realize $p_0^k = p_0(x_k, a_1) - \beta/2$ for some properly chosen $p_0(x_k, a_1)$; otherwise, p_0^k can only be $\frac{4\gamma-1}{3\gamma}$. In the right graph of Figure 3.4, we plot ε_0 in terms of β and γ with $p_0^k = \frac{4\gamma-1}{3\gamma}$. We can see that for a fixed γ, $\varepsilon_0 \approx c\beta$ and the larger γ is, the larger c is.

3.5.2 $\overline{C}$ and $\underline{C}_s$

We plot the values of $\overline{C}$ and $\underline{C}_s$ (the one in the hard case) in Figure 3.5. We can see that $\underline{C}_s$ matches $\overline{C}$ up to a constant factor when β is away from 0. When β is small, the upper bound provides useful information on how much an approximate model can help. As β increases, the upper bound becomes trivial and the lower bound is informative on the limitation of an approximate model. This change is not surprising: the larger the distance between two

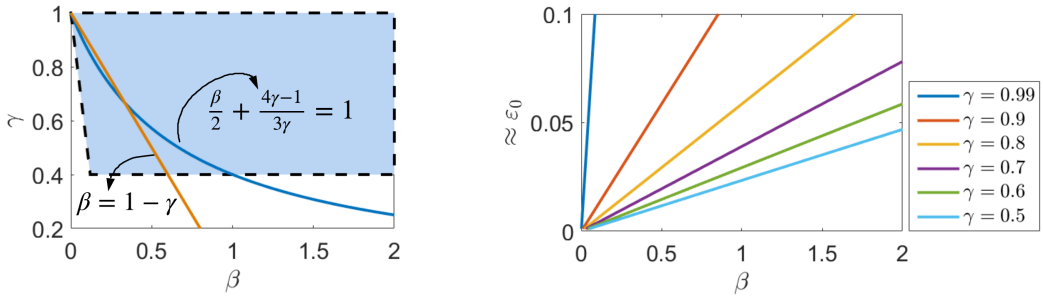

Figure 3.4: Relations between parameters. [Left]: the blue area is for $\gamma \in (\max\{0.4, 1 - 10\beta\}, 1)$; the orange line depicts whether β takes effect in the upper bound; the blue line showcases different realizations of the hard case. [Right]: the upper bound on ε with various β and γ in the hard case.

models is, the less helpful an approximate model would be. The value of β when the upper bound becomes trivial also depends on γ: the greater γ is, the smaller this value is since the more sensitive the value function becomes in terms of the variation in transition. To have the upper bound meets the lower bound, we need $|\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})| = |\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)|$, which is achievable if the value gap in $\mathcal{M}_0$ is big enough. We illustrate this situation in Figure 3.6.

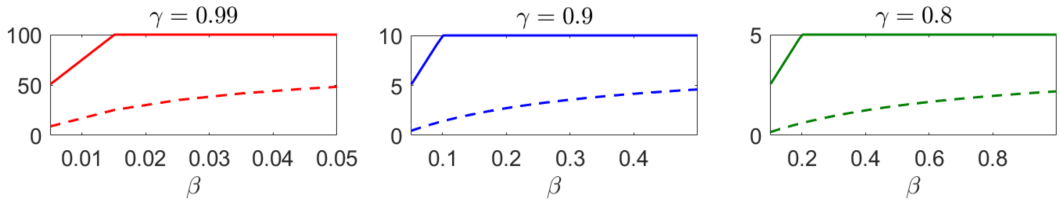

Figure 3.5: $\overline{C}$ (solid lines) and $\underline{C}_s$ (dashed lines) in the hard case with $p_0^k = (4\gamma - 1)/(3\gamma)$.

3.5.3 Empirical Verification of the Worst Case

We further do a numerical demonstration to show when an approximate model does not help. We test on a sailing problem [Van96]. In Figure 3.6, we generate two MDPs $\mathcal{M}_0$ and $\mathcal{M}$ with $\mathcal{M} \in B_{\mathrm{TV}}(\mathcal{M}_0, 0.3)$. We compare the performances of two algorithms: 1. direct Q-learning [WD92] on $\mathcal{M}$ (blue line); 2. use the full knowledge of $\mathcal{M}_0$ to compute $Q^*_{\mathcal{M}_0}$, then use $Q^*_{\mathcal{M}_0}$ as an initialization for Q-learning on $\mathcal{M}$ (red line). One can observe a jump-start

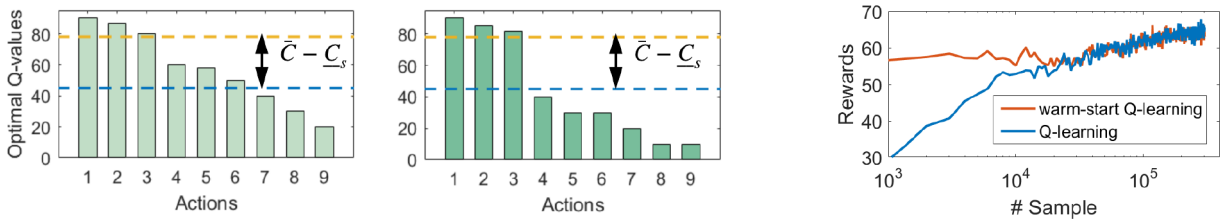

Figure 3.6: [Left]: two types of $\mathcal{M}_0$. Actions with values above the yellow lines form $\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s)$; actions with values above the blue lines form $\mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$. In the left model, $\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s) \subset \mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$; In the right model, due to a large value gap, $\mathcal{A}^s_{\mathcal{M}_0}(\underline{C}_s) = \mathcal{A}^s_{\mathcal{M}_0}(\overline{C})$. [Right]: an empirical verification of the worst case.

improvement. However, to reach higher rewards, it takes the same number of samples with or without the prior knowledge. This verifies our worst case result in Corollary 4: when the accuracy requirement is high, i.e., ε is small, the help of an approximate model can be very limited.

3.6 Conclusion

In this chapter, we study the statistical efficiency of RL when a prior model (under TV-distance) is given. Both sample complexity upper and lower bound are provided. We show that the approximate model can help eliminate sub-optimal actions and reduce the sample complexity of learning a near-optimal policy for the true unknown model. We also show that the help can be rather limited if the value gap in the prior model is small and the precision requirement of the policy is high.

CHAPTER 4

RL Exploration with Unsupervised Learning

Motivated by the prevailing paradigm of using unsupervised learning for efficient exploration in reinforcement learning (RL) problems [THF17, BSO16], we investigate when this paradigm is provably efficient. In this chapter, we study episodic Markov decision processes with rich observations generated from a small number of latent states. We present a general algorithmic framework that is built upon two components: an unsupervised learning algorithm and a no-regret tabular RL algorithm. Theoretically, we prove that as long as the unsupervised learning algorithm enjoys a polynomial sample complexity guarantee, we can find a near-optimal policy with sample complexity polynomial in the number of latent states, which is significantly smaller than the number of observations. Empirically, we instantiate our framework on a class of hard exploration problems to demonstrate the practicality of our theory.

The contributions in this chapter were first presented in the joint work with Ruosong Wang, Wotao Yin, Simon Du, and Lin Yang and was published in NeurIPS 2020 [FWY20].

4.1 Introduction

Reinforcement learning (RL) is the framework of learning to control an unknown system through trial and error. It takes as inputs the observations of the environment and outputs a policy, i.e., a mapping from observations to actions, to maximize the cumulative rewards. To learn a near-optimal policy, it is critical to sufficiently explore the environment and identify all

the opportunities for high rewards. However, modern RL applications often need to deal with huge observation spaces such as those consist of images or texts, which makes it challenging or impossible (if there are infinitely many observations) to fully explore the environment in a direct way. In some work, function approximation scheme is adopted such that essential quantities for policy improvement, e.g. state-action values, can be generalized from limited observed data to the whole observation space. However, the use of function approximation alone does not resolve the exploration problem [DKW20].

To tackle this issue, multiple empirically successful strategies are developed [THF17, BSO16, PAE17, ABA18, LLG18, FAP18, OVW16]. Particularly, in [THF17] and [BSO16], the authors use state abstraction technique to reduce the problem size. They construct a mapping from observations to a small number of hidden states and devise exploration on top of the latent state space rather than the original observation space.

To construct such a state abstraction mapping, practitioners often use *unsupervised learning*. The procedure has the following steps: collect a batch of observation data, apply unsupervised learning to build a mapping, use the mapping to guide exploration and collect more data, and repeat. Empirical study evidences the effectiveness of such an approach at addressing hard exploration problems (e.g., the infamous Montezuma's Revenge). However, it has not been theoretically justified. In this chapter, we aim to answer this question:

Is exploration driven by unsupervised learning in general ***provably efficient***?

The generality includes the choice of unsupervised learning algorithms, reinforcement learning algorithms, and the condition of the problem structure.

We first review some existing theoretical results on provably efficient exploration. More discussion about related work is deferred to appendix. For an RL problem with finitely many states, there are many algorithms with a tabular implementation that learn to control efficiently. These algorithms can learn a near-optimal policy using a number of samples polynomially depending on the size of the state space. However, if we directly apply these

algorithms to rich observations cases by treating each observation as a state, the sample complexities are polynomial in the cardinality of the observation space. Such a dependency is unavoidable without additional structural assumptions [JOA10]. If structural conditions are considered, for example, observations are generated from a small number of latent states [KAL16, JKA17, DJK18, DKJ19], then the sample complexity only scales polynomially with the number of hidden states. Unfortunately, the correctness of these algorithms often requires strict assumptions (e.g., deterministic transitions, reachability) that may not be satisfied in many real applications.

Our Contributions In this chapter we study RL problems with rich observations generated from a small number of latent states for which an unsupervised learning subroutine is used to guide exploration. We summarize our contributions below.

- We propose a new algorithmic framework for the Block Markov Decision Process (BMDP) model [DKJ19]. We combine an unsupervised learning oracle and a tabular RL algorithm in an organic way to find a near-optimal policy for a BMDP. The unsupervised learning oracle is an abstraction of methods used in [THF17, BSO16] and widely used statistical generative models. Notably, our framework can take almost *any* unsupervised learning algorithms and tabular RL algorithms as subroutines.
- Theoretically, we prove that as long as the unsupervised learning oracle and the tabular RL algorithm each has a polynomial sample complexity guarantee, our framework finds a near-optimal policy with sample complexity polynomial in the number of latent states, which is significantly smaller than the number of possible observations (cf. Theorem 4.3.1). To our knowledge, this is the *first* provably efficient method for RL problems with huge observation spaces that uses unsupervised learning for exploration. Furthermore, our result does not require additional assumptions on transition dynamics as used in [DKJ19]. Our result theoretically sheds light on the success of the empirical paradigms used in [THF17, BSO16].

- We instantiate our framework with particular unsupervised learning algorithms and tabular RL algorithms on hard exploration environments with rich observations studied in [DKJ19], and compare with other methods tested in [DKJ19]. Our experiments demonstrate our method can significantly outperform existing methods on these environments.

Main Challenge and Our Technique We assume there is an unsupervised learning oracle (see formal definition in Section 4.3) which can be applied to learn decoding functions and the accuracy of learning increases as more training data are fed. The unsupervised learning algorithm can only guarantee good performance with respect to the input distribution that generates the training data. Unlike standard unsupervised learning where the input distribution is fixed, in our problem, the input distribution depends on our policy. On the other hand, the quality of a policy depends on whether the unsupervised learning oracle has (approximately) decoded the latent states. This inter-dependency is the main challenge we need to tackle in our algorithm design and analysis.

Here we briefly explain our framework. Let $\mathcal{M}$ be the MDP with rich observations. We form an auxiliary MDP $\mathcal{M}'$ whose state space is the latent state space of $\mathcal{M}$. Our idea is to simulate the process of running a no-regret tabular RL algorithm $\mathscr{A}$ directly on $\mathcal{M}'$. For each episode, $\mathscr{A}$ proposes a policy π for $\mathcal{M}'$ and expects a trajectory of running π on $\mathcal{M}'$ for updating and then proceeds. To obtain such a trajectory, we design a policy ϕ for $\mathcal{M}$ as a composite of π and some initial decoding functions. We run ϕ on $\mathcal{M}$ to collect observation trajectories. Although the decoding functions may be inaccurate initially, they can still help us collect observation samples for later refinement. After collecting sufficient observations, we apply the unsupervised learning oracle to retrain decoding functions and then update ϕ as a composite of π and the newly-learned functions and repeat running ϕ on $\mathcal{M}$. After a number of iterations (proportional to the size of the latent state space), with the accumulation of training data, decoding functions are trained to be fairly accurate on recovering latent states, especially those π has large probabilities to visit. This implies that running the latest ϕ

on $\mathcal{M}$ is almost equivalent to running π on $\mathcal{M}'$. Therefore, we can obtain a state-action trajectory with high accuracy as the algorithm $\mathscr{A}$ requires. Since $\mathscr{A}$ is guaranteed to output a near-optimal policy after a polynomial (in the size of the true state-space) number of episodes, our algorithm uses polynomial number of samples as well.

4.1.1 Related Work

In this section, we review related provably efficient RL algorithms. We remark that we focus on environments that require explicit exploration. With certain assumptions of the environment, e.g., the existence of a good exploration policy or the distribution over the initial state is sufficiently diverse, one does not need to explicitly explore [Mun05, ASM08, GSP19, KL02, BKS04, SG14, AKL20b, FWX20, CJ19]. Without these assumptions, the problem can require an exponential number of samples, especially for policy-based methods [DKW20].

Exploration is needed even in the most basic tabular setting. There is a substantial body of work on provably efficient tabular RL [AJ17, JOA10, KWY18, AOM17, KS02b, DLB17, SLW06, JAB18, SJ19, ZB19]. A common strategy is to use UCB (upper-confidence-bound) bonus to encourage exploration in less-visited states and actions. One can also study RL in metric spaces [PP13, SS19, NYW19]. However, in general, this type of algorithms has an exponential dependence on the state dimension.

To deal with huge observation spaces, one might use function approximation. [WV13] proposed an algorithm, optimistic constraint propagation (OCP), which enjoys polynomial sample complexity bounds for a family of Q-function classes, including the linear function class as a special case. But their algorithm can only handle deterministic systems, i.e., both transition dynamics and rewards are deterministic. The setting is recently generalized by [DLW19] to environments with low variance and by [DLM20] to the agnostic setting. [LLW11] proposed a Q-learning algorithm which requires the Know-What-It-Knows oracle. But it is in general unknown how to implement such an oracle.

Our work is closely related to a sequence of works which assumes the transition has certain low-rank structure [KAL16, JKA17, DJK18, SJK19, DKJ19, JYW20a, YW19b]. The most related paper is [DKJ19] which also builds a state abstraction map. Their sample complexity depends on two quantities of the transition probability of the hidden states: *identifiability* and *reachability*, which may not be satisfied in many scenarios. Identifiability assumption requires that the L_1 distance between the posterior distributions (of previous level's hidden state, action pair) given any two different hidden states is strictly larger than some constant (Assumption 3.2 in [DKJ19]). This is an inherent necessary assumption for the method in [DKJ19] as they need to use the posterior distribution to distinguish hidden states. Reachability assumption requires that there exists a constant such that for every hidden state, there exists a policy that reaches the hidden state with probability larger than this constant (Definition 2.1 in [DKJ19]). Conceptually, this assumption is not needed for finding a near-optimal policy because if one hidden state has negligible reaching probability, one can just ignore it. Nevertheless, in [DKJ19], the reachability assumption is also tied with building the abstraction map. Therefore, it may not be removable if one uses the strategy in [DKJ19]. In this paper, we show that given an unsupervised learning oracle, one does not need the identifiability and reachability assumptions for efficient exploration.

4.2 Preliminaries

Notations Given a set $\mathcal{A}$, we denote by $|\mathcal{A}|$ the cardinality of $\mathcal{A}$, $\mathcal{P}(\mathcal{A})$ the set of all probability distributions over $\mathcal{A}$, and $\mathrm{Unif}(\mathcal{A})$ the uniform distribution over $\mathcal{A}$. We use $[h]$ for the set $\{1, 2, \ldots, h\}$ and $f_{[h]}$ for the set of functions $\{f_1, f_2, \ldots, f_h\}$. Given two functions $f : \mathcal{X} \to \mathcal{Y}$ and $g : \mathcal{Y} \to \mathcal{Z}$, their composite is denoted as $g \circ f : \mathcal{X} \to \mathcal{Z}$.

Block Markov Decision Process We consider a Block Markov Decision Process (BMDP), which is first formally introduced in [DKJ19]. A BMDP is described by a tuple $\mathcal{M} :=$

$(\mathcal{S}, \mathcal{A}, \mathcal{X}, \mathcal{P}, r, f_{[H+1]}, H)$. $\mathcal{S}$ is a finite unobservable *latent state space*, $\mathcal{A}$ is a finite action space, and $\mathcal{X}$ is a possibly infinite observable context space. $\mathcal{X}$ can be partitioned into $|\mathcal{S}|$ disjoint blocks $\{\mathcal{X}_s\}_{s\in\mathcal{S}}$, where each block $\mathcal{X}_s$ corresponds to a unique state s. $\mathcal{P}$ is the collection of the *state-transition probability* $p_{[H]}(s'|s,a)$ and the *context-emission distribution* $q(x|s)$ for all $s, s' \in \mathcal{S}, a \in \mathcal{A}, x \in \mathcal{X}$. $r : [H] \times \mathcal{S} \times \mathcal{A} \to [0,1]$ is the reward function. $f_{[H+1]}$ is the set of decoding functions, where f_h maps every observation at level h to its true latent state. Finally, H is the length of horizon. When $\mathcal{X} = \mathcal{S}$, this is the usual MDP setting.

For each episode, the agent starts at level 1 with the initial state s_1 and takes H steps to the final level $H+1$. We denote by $\mathcal{S}_h$ and $\mathcal{X}_h$ the set of possible states and observations at level $h \in [H+1]$, respectively. At each level $h \in [H+1]$, the agent has no access to the true latent state $s_h \in \mathcal{S}_h$ but an observation $x_h \sim q(\cdot|s_h)$. An action a_h is then selected following some policy $\phi : [H] \times \mathcal{X} \to \mathcal{P}(\mathcal{A})$. As a result, the environment evolves into a new state $s_{h+1} \sim p_h(\cdot|s_h, a_h)$ and the agent receives an instant reward $r(h, s_h, a_h)$. A trajectory has such a form: $\{s_1, x_1, a_1, \ldots, s_H, x_H, a_H, s_{H+1}, x_{H+1}\}$, where all state components are unknown.

Policy Given a BMDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{X}, \mathcal{P}, r, f_{[H+1]}, H)$, there is a corresponding MDP $\mathcal{M}' := (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, H)$, which we refer to as *the underlying MDP* in the later context. A policy on $\mathcal{M}$ has a form $\phi : [H] \times \mathcal{X} \to \mathcal{P}(\mathcal{A})$ and a policy on $\mathcal{M}'$ has a form $\pi : [H] \times \mathcal{S} \to \mathcal{P}(\mathcal{A})$. Given a policy π on $\mathcal{M}'$ and a set of functions $\hat{f}_{[H+1]}$ where $\hat{f}_h : \mathcal{X}_h \to \mathcal{S}_h, \forall\, h \in [H+1]$, we can induce a policy on $\mathcal{M}$ as $\pi \circ \hat{f}_{[H+1]} =: \phi$ such that $\phi(h, x_h) = \pi(h, \hat{f}_h(x_h)),\ \forall\, x_h \in \mathcal{X}_h,\ h \in [H]$. If $\hat{f}_{[H+1]} = f_{[H+1]}$, then π and ϕ are equivalent in the sense that they induce the same probability measure over the state-action trajectory space.

Given an MDP, the value of a policy π (starting from s_1) is defined as $V_1^\pi = \mathbb{E}^\pi \left[\sum_{h=1}^H r(h, s_h, a_h) \middle| s_1\right]$, A policy that has the maximal value is an *optimal policy* and the *optimal value* is denoted by V_1^*, i.e., $V_1^* = \max_\pi V_1^\pi$. Given $\varepsilon > 0$, we say π is *ε-optimal* if $V_1^* - V_1^\pi \le \varepsilon$. Similarly, given a BMDP, we define the value of a policy ϕ (starting from s_1) as: $V_1^\phi = \mathbb{E}^\phi \left[\sum_{h=1}^H r(h, s_h, a_h) \middle| s_1\right]$, The notion of optimallity and ε-optimality are similar to MDP.

4.3 A Unified Framework for Unsupervised RL

4.3.1 Unsupervised Learning Oracle and No-regret Tabular MDP Algorithm

In this paper, we consider RL on a BMDP. The goal is to find a near-optimal policy with sample complexity polynomial to the cardinality of the latent state space. We assume no knowledge of $\mathcal{P}$, r, and $f_{[H+1]}$, but the access to an unsupervised learning oracle $\mathcal{ULO}$ and an (ε, δ)-correct episodic no-regret algorithm. We give the definitions below.

Definition 3 (Unsupervised Learning Oracle $\mathcal{ULO}$)**.** *There exists a function $g(n, \delta)$ such that for any fixed $\delta > 0$, $\lim_{n\to\infty} g(n, \delta) = 0$. Given a distribution μ over $\mathcal{S}$, and n samples from $\sum_{s\in\mathcal{S}} q(\cdot|s)\mu(s)$ such that with probability at least $1 - \delta$ over n training data, we can find a function $\hat{f} : \mathcal{X} \to \mathcal{S}$ such that*

$$\mathbb{P}_{s\sim\mu, x\sim q(\cdot|s)}\big(\hat{f}(x) = \alpha(s)\big) \geq 1 - g(n, \delta)$$

for some unknown permutation $\alpha : \mathcal{S} \to \mathcal{S}$.

In Definition 3, we assume f is the true decoding function i.e., $\mathbb{P}_{s\sim\mu, x\sim q(\cdot|s)}\big(f(x) = s\big) = 1$ and we call the permutation α as a *good permutation* between f and $\hat{f}$. For the function $g(n, \delta)$, we introduce a corresponding inverse function:

$$g^{-1}(\epsilon, \delta) := \min\{N \mid \text{for all } n > N, g(n, \delta) < \epsilon\}.$$

Since $\lim_{n\to\infty} g(n, \delta) = 0$, $g^{-1}(\epsilon, \delta)$ is well-defined. We assume that $g^{-1}(\epsilon, \delta)$ is a polynomial in terms of $1/\epsilon, \log(\delta^{-1})$ and possibly problem-dependent parameters.

This definition is motivated by [THF17] in which authors use auto-encoder and SimHash [Cha02] to construct the decoding function and they use this UCB-based approach on top of the decoding function to guide exploration. It is still an open problem to obtain a sample complexity analysis for auto-encoder. Let alone the composition with SimHash. Nevertheless, in Section 4.4, we give several examples of $\mathcal{ULO}$ with theoretical guarantees. Furthermore, once we have an analysis of auto-encoder and we can plug-in that into our framework effortlessly.

Definition 4 ((ε, δ)-correct Episodic No-regret Algorithm)**.** *Let $\varepsilon > 0$ and $\delta > 0$. $\mathscr{A}$ is an (ε, δ)-correct episodic no-regret algorithm if for any MDP $\mathcal{M}' := (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, H)$ with the initial state s_1, $\mathscr{A}$*

- *runs for at most $C(\varepsilon, \delta) = \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta^{-1}))$ episodes (which we call as the sample complexity of $\mathscr{A}$);*
- *proposes a policy π^k at the beginning of episode k and collects a sample trajectory of $\mathcal{M}'$ following π^k;*
- *outputs a policy π at the end such that with probability at least $1 - \delta$, π is ε-optimal.*

Definition 4 simply describes tabular RL algorithms that have polynomial sample complexity guarantees for episodic MDPs. Instances are vivid in literature (see Section 4.1.1).

4.3.2 A Unified Framework

With a $\mathcal{ULO}$ and an (ε, δ)-correct episodic no-regret algorithm $\mathscr{A}$, we propose a unified framework in Algorithm 6 to solve an unsupervised RL problem on a BMDP. The main challenge of this problem is that the true states are unobservable. Hence $\mathscr{A}$ cannot be run on the underlying MDP. If one resorts to directly viewing the BMDP as a giant MDP with state space $\mathcal{X}$, then the dimension of the problem is significantly large. To circumvent this issue, our solution is to unsupervised learn decoding functions $\hat{f}_{[H+1]}$ with observation samples. Specifically, for each episode, we use the policy π proposed by $\mathscr{A}$ for the underlying MDP together with certain decoding functions $\hat{f}_{[H+1]}$ to generate a policy $\pi \circ \hat{f}_{[H+1]}$ for the BMDP. Then we collect observation samples using $\pi \circ \hat{f}_{[H+1]}$ and all previously generated policies over BMDP. As more samples are collected, we refine the decoding functions using the $\mathcal{ULO}$. As long as we collect sufficient samples, we can simulate a trajectory as if using the true decoding functions (up to some permutations) and therefore, as if running the policy π directly on the underlying MDP as $\mathscr{A}$ required. We continue this procedure until the algorithm $\mathscr{A}$ halts. Note that this procedure is essentially what practitioners use [THF17, BSO16], as we have

Algorithm 6 A Unified Framework for Unsupervised RL

Input: BMDP $\mathcal{M}; \mathcal{ULO}$; (ε, δ)-correct episodic no-regret algorithm $\mathscr{A}$; batch size $B > 0$; $\varepsilon \in (0,1)$; $\delta \in (0,1)$; $N := \lceil \log(2/\delta)/2 \rceil$; $L := \lceil 9H^2/(2\varepsilon^2) \log(2N/\delta) \rceil$.
for $n = 1$ **to** N **do**
    Clear the memory of $\mathscr{A}$ and restart;
    for episode $k = 1$ **to** K **do**
        Obtain π^k from $\mathscr{A}$;
        Obtain a trajectory: $\tau^k, f^k_{[H+1]} \leftarrow \texttt{TSR}(\mathcal{ULO}, \pi^k, B)$;
        Update the algorithm: $\mathscr{A} \leftarrow \tau^k$;
    end for
    Obtain π^{K+1} from $\mathscr{A}$;
    Finalize the decoding functions: $\tau^{K+1}, f^{K+1}_{[H+1]} \leftarrow \texttt{TSR}(\mathcal{ULO}, \pi^{K+1}, B)$;
    Construct a policy for $\mathcal{M}$: $\phi^n \leftarrow \pi^{K+1} \circ f^{K+1}_{[H+1]}$.
end for
Run each ϕ^n $(n \in [N])$ for L episodes and get the average rewards per episode $\bar{V}_1^{\phi^n}$.
Output a policy $\phi \in \operatorname{argmax}_{\phi \in \phi^{[N]}} \bar{V}_1^{\phi}$.

discussed in Section 5.1.

We now describe in more detail our algorithm. Suppose the algorithm $\mathscr{A}$ runs for K episodes. At the beginning of each episode $k \in [K]$, $\mathscr{A}$ proposes a policy $\pi^k : [H] \times \mathcal{S} \to \mathcal{A}$ for the underlying MDP. We use the Trajectory Sampling Routine `TSR` to generate a trajectory τ^k given π^k and then feed τ^k to $\mathscr{A}$. After K episodes, we obtain a policy π^{K+1} from $\mathscr{A}$ and a set of decoding functions $f^{K+1}_{[H+1]}$ from `TSR`. We then construct a policy for the BMDP as $\pi^{K+1} \circ f^{K+1}_{[H+1]}$. We repeat this process for N times for making sure our algorithm succeeds with high probability.

The detailed description of `TSR` is displayed in Algorithm 7. We here briefly explain the

Algorithm 7 Trajectory Sampling Routine TSR $(\mathcal{ULO}, \pi, B)$

1: **Input:** $\mathcal{ULO}$; a policy $\pi : [H] \times \mathcal{S} \to \mathcal{A}$; episode index k; batch size $B > 0$; $\epsilon \in (0, 1)$; $\delta_1 \in (0, 1)$; $J := (H + 1)|\mathcal{S}| + 1$.

2: **Data:**

- a policy set Π;
- label standard data $\mathcal{Z} := \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots \mathcal{Z}_{H+1}\}$, $\mathcal{Z}_h := \{\mathcal{D}_{h,s_1}, \mathcal{D}_{h,s_2}, \ldots\}$;
- present decoding functions $f^0_{[H+1]}$;

3: **for** $i = 1$ **to** J **do**

4: Combine policy: $\Pi \leftarrow \Pi \cup \{\pi \circ f^{i-1}_{[H+1]}\}$;

5: Generate $((k-1)J + i) \cdot B$ trajectories of training data $\mathcal{D}$ with Unif(Π);

6: Generate B trajectories of testing data $\mathcal{D}''$ with $\pi \circ f^{i-1}_{[H+1]}$.

7: Train with $\mathcal{ULO}$: $\tilde{f}^i_{[H+1]} \leftarrow \mathcal{ULO}(\mathcal{D})$;

8: Match labels: $f^i_{[H+1]} \leftarrow \text{FixLabel}(\tilde{f}^i_{[H+1]}, \mathcal{Z})$;

9: **for** $h \in [H + 1]$ **do**

10: Let $\mathcal{D}''_{h,s} := \{x \in \mathcal{D}''_h : f^i_h(x) = s, s \in \mathcal{S}_h\}$;

11: Update label standard set: if $\mathcal{D}_{h,s} \notin \mathcal{Z}_h$ and $|\mathcal{D}''_{h,s}| \geq 3\epsilon \cdot B \log(\delta_1^{-1})$, then let $\mathcal{Z}_h \leftarrow \mathcal{Z}_h \cup \{\mathcal{D}''_{h,s}\}$

12: **end for**

13: **end for**

14: Run $\pi \circ f^J_{[H+1]}$ to obtain a trajectory τ;

15: Renew $f^0_{[H+1]} \leftarrow f^J_{[H+1]}$;

16: **Output:** $\tau, f^J_{[H+1]}$.

idea. To distinguish between episodes, with input policy π^k (Line 6 Algorithm 6), we add the episode index k as superscripts to π and $f_{[H+1]}$ in TSR. We maintain a policy set in memory and initialize it as an empty set at the beginning of Algorithm 6. Note that, at each episode,

Algorithm 8 FixLabel($\tilde{f}_{[H+1]}, \mathcal{Z}$)

Input: a set of decoding functions $\tilde{f}_{[H+1]}$; a set of label standard data $\mathcal{Z} := \{\mathcal{Z}_1, \mathcal{Z}_2, \cdots, \mathcal{Z}_{H+1}\}$, $\mathcal{Z}_h := \{\mathcal{D}_{h,s_1}, \mathcal{D}_{h,s_2}, \ldots\}$.
for $h \in [H+1]$ **do**
  for $\mathcal{D}_{h,s} \in \mathcal{Z}_h$ **do**
    if $s \in \mathcal{S}_h$ and $|\{x \in \mathcal{D}_{h,s} : \tilde{f}_h(x) = s'\}| > 3/5|\mathcal{D}_{h,s}|$ **then**
      Swap the output of s' with s in $\tilde{f}_h$;
    end if
  end for
end for
Output: $\tilde{f}_{[H+1]}$

our goal is to simulate a trajectory of π running on the underlying MDP. `TSR` achieves this in an iterative fashion: it starts with the input policy π^k and the latest-learned decoding functions $f^{k,0}_{[H+1]} := f^{k-1,J}_{[H+1]}$; for each iteration i, it first adds the policy $\pi^k \circ f^{k,i-1}_{[H+1]}$ in Π and then plays Unif(Π) to collect a set of observation trajectories (i.e., each trajectory is generated by first uniformly randomly selecting a policy from Π and then running it in the BMDP);[1] then updates $f^{k,i-1}_{[H+1]}$ to $\tilde{f}^{k,i}_{[H+1]}$ by running $\mathcal{ULO}$ on these collected observations. Note that $\mathcal{ULO}$ may output labels inconsistent with previously trained decoding functions. We further match labels of $\tilde{f}^{k,i}_{[H+1]}$ with the former ones by calling the FixLabel routine (Algorithm 8) at Line 8 of Algorithm 7. To accomplish the label matching process, we cache a set $\mathcal{Z}$ in memory which stores observation examples $\mathcal{D}_{h,s}$ for each state s and each level h. $\mathcal{Z}$ is initialized as an empty set and gradually grows. Whenever we confirm a new label, we add the corresponding observation examples to $\mathcal{Z}$ (Line 11 Algorithm 7). Then for later learned decoding functions, they can use this standard set to correspondingly swap their labels and match with previous

[1]This resampling over all previous policies is mainly for the convenience of analysis. It can be replaced using previous data but requires more refined analysis.

functions. After the matching step, we get $f^{k,i}_{[H+1]}$. Continuously running for J iterations, we stop and use $\pi^k \circ f^{k,J}_{[H+1]}$ to obtain a trajectory. Note that with $f^{k,J}_{[H+1]}$, this final trajectory is translated to be over hidden states rather than observations.

We now present our main theoretical result.

Theorem 4.3.1. *Suppose in Definition 3,* $g^{-1}(\epsilon, \delta_1) = \text{poly}(|\mathcal{S}|, 1/\epsilon, \log(\delta_1^{-1}))$ *for any* $\epsilon, \delta_1 \in (0,1)$ *and* $\mathscr{A}$ *is* (ε, δ_2)*-correct with sample complexity* $\text{poly}\left(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log\left(\delta_2^{-1}\right)\right)$ *for any* $\varepsilon, \delta_2 \in (0,1)$*. Then Algorithm 6 outputs a policy* ϕ *such that with probability at least* $1-\delta$*,* ϕ *is an* ε*-optimal policy for the BMDP, using at most* $\text{poly}\left(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta^{-1})\right)$ *trajectories.*

Theorem 4.3.1 formally justifies what we claimed in Section 5.1 that as long as the sample complexity of $\mathcal{ULO}$ is polynomial and $\mathscr{A}$ is a no-regret tabular RL algorithm, polynomial number of trajectories suffices to find a near-optimal policy. To our knowledge, this is the first result that proves unsupervised learning can guide exploration in RL problems with a huge observation space and therefore, we theoretically justify the empirical paradigm used in [THF17, BSO16]. In the following, we give a formal proof of our main result.

4.3.3 Proofs

We first give a sketch of the proof. Note that if `TSR` always correctly simulates a trajectory of π^k on the underlying MDP, then by the correctness of $\mathscr{A}$, the output policy of $\mathscr{A}$ in the end is near-optimal with high probability. If in `TSR`, $f^{k,J}_{[H+1]}$ decodes states correctly (up to a fixed permutation, with high probability) for every observation generated by playing $\pi^k \circ f^{k,J}_{[H+1]}$, then the obtained trajectory (on $\mathcal{S}$) is as if obtained with $\pi^k \circ f_{[H+1]}$ which is essentially equal to playing π^k on the underlying MDP. Let us now consider $\pi^k \circ f^{k,i}_{[H+1]}$ for some intermediate iteration $i \in [J]$. If there are many observations from a previously unseen state, s, then $\mathcal{ULO}$ guarantees that all the decoding functions in future iterations will be correct with high probability of identifying observations of s. Since there are at most $|\mathcal{S}|$ states to reach for each level following π^k, after $(H+1)|\mathcal{S}|$ iterations, `TSR` is guaranteed to output a set of

decoding functions that are with high probability correct under policy π^k. With this set of decoding functions, we can simulate a trajectory for $\mathscr{A}$ as if we know the true latent states.

For episode k, we denote the training dataset $\mathcal{D}$ generated by running Unif(Π) as $\{\mathcal{D}_{k,i,h}\}_{h=1}^{H+1}$ (Line 5) and the testing dataset $\mathcal{D}''$ generated by $\pi^k \circ f_{[H+1]}^{k,i-1}$ as $\{\mathcal{D}''_{k,i,h}\}_{h=1}^{H+1}$ (Line 6). The subscript h represents the level of the observations. Furthermore, we denote by $\mu_{k,i,h}(\cdot)$ the distribution over hidden states at level h induced by $\pi^k \circ f_{[H+1]}^{k,i-1}$. To formally prove the correctness of our framework, we first present the following lemma, showing that whenever some policy π with some decoding functions visits a state s with relatively high probability, all the decoding functions of later iterations will correctly decode the observations from s with high probability.

Lemma 4.3.1. *Suppose for some $s^* \in \mathcal{S}_h$, (k,i) is the earliest pair such that $\big|\{x \in \mathcal{D}''_{k,i,h} : f_h^{k,i}(x) = \alpha_h(s^*)\}| \geq 3\epsilon \cdot B \log(\delta_1^{-1})$ and $\{x \in \mathcal{D}''_{k,i,h} : f_h^{k,i}(x) = \alpha_h(s^*)\}$ is added into $\mathcal{Z}_h$ as $\mathcal{D}_{h,\alpha_h(s^*)}$ at line 11 Algorithm 7, where α_h is a good permutation between $f_h^{k,i}$ and f_h. Then for each $(k',i') > (k,i)$ (in lexical order), with probability at least $1 - \mathcal{O}(\delta_1)$,*

$$\Pr_{x \sim q(\cdot|s^*)} \big[f_h^{k',i'}(x) \neq \alpha_h^*(s^*)\big] \leq \epsilon$$

provided $0 < \epsilon \log(\delta_1^{-1}) \leq 0.1$ and $B \geq B_0$. Here B_0 is some constant to be determined later and α_h^* is some fixed permutation on $\mathcal{S}_h$.

Proof. For iterations $(k',i') \geq (k,i)$, the function $\tilde{f}_h^{k',i'}$ is obtained by applying $\mathcal{ULO}$ on the dataset generated by

$$\mu' := \text{Unif}(\{\mu_{k'',i'',h}\}_{(k'',i'')<(k',i')})$$

and the dataset has size $\big((k'-1)\cdot J + i'\big) \cdot B = \Theta(k'JB)$. Thus, with probability at least $1-\delta_1$, for some permutation α'_h,

$$\Pr_{s \sim \mu', x \sim q(\cdot|s)} \big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big] \leq g\big(\Theta(k'JB), \delta_1\big). \tag{4.1}$$

By taking

$$B_0 := \Theta\Big(\frac{g^{-1}(\epsilon^2/(K\cdot J), \delta_1)}{K \cdot J}\Big), \tag{4.2}$$

we have when $B \geq B_0$, $g\big(\Theta(k'JB), \delta_1\big) \leq \epsilon^2/(K \cdot J)$ for all $k' \in [K]$. Later, in Proposition 5, we will show that $B_0 = \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon)$. Now we consider $f_h^{k,i}$. Since the FixLabel routine (Algorithm 8) does not change the accuracy ratio, from Equation (4.1), it holds with probability at least $1 - \delta_1$ that

$$\Pr_{s\sim\mu_{k,i,h}, x\sim q(\cdot|s)}[f_h^{k,i}(x) \neq \alpha_h \circ f_h(x)] \leq k \cdot J \cdot g\big(\Theta(kJB), \delta_1\big) \leq \epsilon.$$

Therefore, by Chernoff bound, with probability at least $1 - \mathcal{O}(\delta_1)$,

$$\big|\{x \in \mathcal{D}''_{k,i,h} : f_h(x) \neq s \text{ and } f_h^{k,i}(x) = \alpha_h(s)\}\big| < \ \epsilon \cdot B \log(\delta_1^{-1}).$$

Since $\big|\{x \in \mathcal{D}''_{k,i,h} : f_h^{k,i}(x) = \alpha_h(s^*)\}| \geq 3\epsilon \cdot B \log(\delta_1^{-1})$, we have that

$$\begin{aligned}\big|\{x \in \mathcal{D}''_{k,i,h} : \ f_h(x) = s^* \text{ and } f_h^{k,i}(x) = \alpha_h(s^*)\}| &> \frac{2}{3} \cdot \big|\{x \in \mathcal{D}''_{k,i,h} : f_h^{k,i}(x) = \alpha_h(s^*)\}\big| \\ &\geq 2\epsilon \cdot B \log(\delta_1^{-1}).\end{aligned} \tag{4.3}$$

Thus, by Chernoff bound, with probability at least $1 - \mathcal{O}(\delta_1)$, $\mu_{k,i,h}(s^*) \geq \epsilon \cdot \log(\delta_1^{-1})$. Also note that $f_h^{k,i}$ is the first function that has confirmed on s^* (i.e., no $\mathcal{D}_{h,\alpha_h(s^*)}$ exists in $\mathcal{Z}_h$ of line 8 at iteration (k,i)). By Line 10 and Line 11, for later iterations, in $\mathcal{Z}_h$, $\mathcal{D}_{h,\alpha_h(s^*)} = \{x \in \mathcal{D}''_{k,i,h} : f_h^{k,i}(x) = \alpha_h(s^*)\}$.

Next, for another $(k', i') > (k, i)$, we let the corresponding permutation be α'_h for $\tilde{f}_h^{k',i'}$. Since $\mu'(s') \geq \mu_{k,i,h}(s')/(k' \cdot J)$, with probability at least $1 - \delta_1$,

$$\Pr_{s\sim\mu_{k,i,h}, x\sim q(\cdot|s)}\big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big] \leq \ k' \cdot J \cdot g(\Theta(k'JB), \delta_1).$$

Notice that

$$\begin{aligned}\Pr_{s\sim\mu_{k,i,h}, x\sim q(\cdot|s)}\big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big] &= \sum_{s'\in\mathcal{S}_h} \mu_{k,i,h}(s') \Pr_{x\sim q(\cdot|s')}\big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big] \\ &\geq \mu_{k,i,h}(s^*) \Pr_{x\sim q(\cdot|s^*)}\big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big] \\ &\geq \epsilon \cdot \log(\delta^{-1}) \Pr_{x\sim q(\cdot|s^*)}\big[\tilde{f}_h^{k',i'}(x) \neq \alpha'_h \circ f_h(x)\big].\end{aligned}$$

Thus, with probability at least $1-\delta_1$,

$$\Pr_{x\sim q(\cdot|s^*)}\left[\tilde{f}_h^{k',i'}(x)\neq \alpha'_h\circ f_h(x)\right]\leq \frac{k'\cdot J\cdot g(\Theta(k'JB),\delta_1)}{\epsilon\cdot\log(\delta_1^{-1})}\leq\epsilon$$

with $B\geq B_0$ and B_0 as defined in Equation (4.2). Let $s':=\alpha'_h(s^*)$. Conditioning on $\mathcal{ULO}$ being correct on $\tilde{f}^{k',i'}_{[H+1]}$ and $f^{k,i}_{[H+1]}$, by Chernoff bound and Equation (4.3), with probability at least $1-\mathcal{O}(\delta_1)$, we have

$$\begin{aligned}\left|\{x\in\mathcal{D}_{h,\alpha_h(s^*)}:\tilde{f}_h^{k',i'}(x)=s'\}\right|&\geq\left|\{x\in\mathcal{D}_{h,\alpha_h(s^*)}:f_h(x)=s^*,\tilde{f}_h^{k',i'}(x)=s'\}\right|\\&\geq(1-\epsilon\cdot\log(\delta_1^{-1}))\cdot\frac{2}{3}\cdot\left|\mathcal{D}_{h,\alpha_h(s^*)}\right|>\frac{3}{5}\left|\mathcal{D}_{h,\alpha_h(s^*)}\right|,\end{aligned}$$

where the fraction $\frac{2}{3}$ follows from Equation (4.3) and we use the fact that $\mathcal{D}''_{k,i,h}$ are independent from the training dataset. By our label fixing procedure, we find a permutation that swaps s' with s for $\tilde{f}_h^{k',i'}$ to obtain $f_h^{k',i'}$. By the above analysis, with probability at least $1-\mathcal{O}(\delta_1)$, $\Pr_{x\sim q(\cdot|s^*)}\left[f_h^{k',i'}(x)\neq\alpha_h(s^*)\right]\leq\epsilon$ as desired. Consequently, we let $\alpha_h^*(s^*)=\alpha_h(s^*)$, which satisfies the requirement of the lemma. ■

Next, by the definition of our procedure of updating the label standard dataset (Line 11, Algorithm 7), we have the following corollary.

Corollary 5. *Consider Algorithm 7. Let $\mathcal{Z}_{k,i,h}$ be the label standard dataset at episode k before iteration i for $\mathcal{S}_h$. Then, with probability at least $1-\mathcal{O}(H|\mathcal{S}|\delta_1)$,*

$$\textit{for all } k,i \textit{ and } \mathcal{D}_{h,s}\in\mathcal{Z}_{k,i,h},|\{x\in\mathcal{D}_{h,s}:\alpha_h^*\circ f_h(x)=s,s\in\mathcal{S}_h\}|>2/3|\mathcal{D}_{h,s}|.$$

At episode k and iteration i of the algorithm `TSR`, let $\mathcal{E}_{k,i}$ be the event that for all $h\in[H+1],\mathcal{D}_{h,s}\in\mathcal{Z}_{k,i,h}$, $\Pr_{x\sim q(\cdot|s)}\left[f_h^{k,i}(x)\neq\alpha_h^*\circ f_h(x)\right]\leq\epsilon$. We have the following corollary as a consequence of Lemma 4.3.1 by taking the union bound over all states.

Corollary 6. $\forall k,i:\quad\Pr\left[\mathcal{E}_{k,i}\right]\geq 1-\mathcal{O}(H|\mathcal{S}|\delta_1)$.

The next lemma shows that after $(H+1)|\mathcal{S}|+1$ iterations of the `TSR` subroutine, the algorithm outputs a trajectory for the algorithm $\mathscr{A}$ as if it knows the true mapping $f_{[H+1]}$.

Lemma 4.3.2. *Suppose in an episode k, we are running algorithm* `TSR`*. Then after $J = (H+1)|\mathcal{S}|+1$ iterations, we have, for every $j \geq J$, with probability at least $1 - \mathcal{O}(H|\mathcal{S}|\delta_1)$,*

$$\text{for all } h \in [H+1], \Pr_{s\sim\mu_{k,j+1,h}, x\sim q(\cdot|s)}\left[f_h^{k,j}(x) \neq \alpha_h^* \circ f_h(x)\right] \leq \epsilon'$$

for some small enough ϵ and $50H \cdot \epsilon \cdot |\mathcal{S}| \cdot \log(\delta_1^{-1}) < \epsilon' < 1/2$, provided $B \geq B_0$ as defined in Lemma 4.3.1.

Proof. For $i < J$, there are two cases:

1. there exists an $h \in [H+1]$ such that $\Pr_{s\sim\mu_{k,i+1,h}, x\sim q(\cdot|s)}\left[f_h^{k,i}(x) \neq \alpha_h \circ f_h(x)\right] > \epsilon'/(2H)$;
2. for all $h \in [H+1]$, $\Pr_{s\sim\mu_{k,i+1,h}, x\sim q(\cdot|s)}\left[f_h^{k,i}(x) \neq \alpha_h \circ f_h(x)\right] \leq \epsilon'/(2H)$,

where α_h is some good permutations between $f_h^{k,i}$ and f_h. If case 1 happens, then there exists a state $s^* \in \mathcal{S}_h$ such that

$$\Pr_{x\sim q(\cdot|s^*)}\left[f_h^{k,i}(x) \neq \alpha_h \circ f_h(x)\right] \cdot \mu_{k,i+1,h}(s^*) > \frac{\epsilon'}{2H|\mathcal{S}|}. \tag{4.4}$$

If $\mathcal{D}_{h,\alpha_h(s^*)} \in \mathcal{Z}_{k,i,h}$, where $\mathcal{Z}_{k,i,h}$ is defined as in Corollary 5, by Lemma 4.3.1, with probability at least $1 - \mathcal{O}(\delta_1)$,

$$\Pr_{x\sim q(\cdot|s^*)}[f_h^{k,i}(x) \neq \alpha_h^* \circ f_h(x)] \leq \epsilon$$

and $\alpha_h^*(s^*) = \alpha_h(s^*)$. Thus, $\mu_{k,i+1,h}(s^*) > \frac{\epsilon'}{2H|\mathcal{S}|}/\epsilon > 1$, a contradiction with $\mu_{k,i+1,h}(s^*) \leq 1$. Therefore, there is no $\mathcal{D}_{h,\alpha_h(s^*)}$ in $\mathcal{Z}_{k,i,h}$. Then, due to $\Pr_{x\sim q(\cdot|s^*)}\left[f_h^{k,i}(x) \neq \alpha_h \circ f_h(x)\right] \leq 1$, by Equation (4.4), we have

$$\mu_{k,i+1,h}(s^*) > \frac{\epsilon'}{2H|\mathcal{S}|}. \tag{4.5}$$

Since $f_h^{k,i+1}$ is trained on $\text{Unif}(\{\mu_{k',i',h}\}_{(k',i')<(k,i+1)})$, by Definition of $\mathcal{ULCO}$, with probability at least $1 - \delta_1$,

$$\Pr_{s\sim\mu_{k,i+1,h}, x\sim q(\cdot|s)}\left[f_h^{k,i+1}(x) \neq \alpha_h'(s)\right] \leq k \cdot J \cdot g(\Theta(kJB), \delta_1) \leq \epsilon^2,$$

with $B \geq B_0$ (B_0 is defined in Equation (4.2)) and α'_h is some good permutation between $f_h^{k,i+1}$ and f_h. Thus, by Equation (4.5) and the choice of ϵ and ϵ', we have

$$\Pr_{x\sim q(\cdot|s^*)}\left[f_h^{k,i+1}(x) \neq \alpha'_h(s^*)\right] < \epsilon/25.$$

Thus,

$$\mu_{k,i+1,h}(s^*) \cdot \Pr_{x\sim q(\cdot|s^*)}\left[f_h^{k,i+1}(x) = \alpha'_h(s^*)\right] > \frac{\epsilon'}{2H|\mathcal{S}|} \cdot (1-\epsilon/25) > 24\epsilon \cdot \log(\delta_1^{-1}),$$

where the last inequality is due to $\epsilon < \epsilon' < 1$. By Chernoff bound, with probability at least $1 - \mathcal{O}(\delta_1)$,

$$|\{x \in \mathcal{D}''_{k,i+1,h} : f_h^{k,i+1}(x) = \alpha'_h(s^*)\}| \geq 3\epsilon \cdot B\log(\delta_1^{-1}).$$

Therefore, if case 1 happens, one state s will be confirmed in iteration $i+1$ and $\alpha_h^*(s^*) = \alpha'_h(s^*)$ is defined.

To analyze case 2, we first define sets $\{\mathcal{G}_{k,i+1,h}\}_{h=1}^{H+1}$ with $\mathcal{G}_{k,i+1,h} := \{s \in \mathcal{S}_h \mid \mathcal{D}_{h,s} \in \mathcal{Z}_{k,i+1,h}\}$, i.e., $\mathcal{G}_{k,i+1,h}$ contains all confirmed states of level h before iteration $i+1$ at episode k. If case 2 happens, we further divide the situation into two subcases:

a) for all $h \in [H+1]$, for all $s \in \mathcal{G}^c_{k,i+1,h}$, $\mu_{k,i+1,h}(s) \leq \epsilon'/(8H|\mathcal{S}|)$;

b) there exists an $h \in [H+1]$ and a state $s^* \in \mathcal{G}^c_{k,i+1,h}$ such that $\mu_{k,i+1,h}(s^*) \geq \epsilon'/(8H|\mathcal{S}|)$,

First notice that for every $h \in [H+1]$ and $j > i$, since $f_h^{k,j}$ is trained on $\text{Unif}(\{\mu_{k',i',h}\}_{(k',i')\leq(k,j)})$, by Definition of $\mathcal{ULCO}$ and our choice of B in Equation (4.2), with probability at least $1-\delta_1$, we have

$$\Pr_{s\sim\mu_{k,i+1,h},x\sim q(\cdot|s)}[f_h^{k,j} \neq \alpha'_h(s)] \leq \epsilon^2, \tag{4.6}$$

$$\Rightarrow \sum_{s\in\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f_h^{k,j}(x) \neq \alpha'_h(s)] + \sum_{s\notin\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f_h^{k,j}(x) \neq \alpha'_h(s)] \leq \epsilon^2,$$

where α'_h is some good permutation between $f_h^{k,j}$ and f_h.

If subcase a) happens, note that for $s \in \mathcal{G}_{k,i+1,h}$, due to the FixLabel routine (Algorithm 8), $\alpha'_h(s) = \alpha^*_h(s)$, for $f^{k,j}_h (j > i)$ we have

$$\begin{aligned}
&\sum_{s\in\mathcal{S}_h} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f^{k,j}_h(x) \neq \alpha^*_h(s)] \\
&= \sum_{s\in\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f^{k,j}_h(x) \neq \alpha^*_h(s)] + \sum_{s\notin\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f^{k,j}_h(x) \neq \alpha^*_h(s)] \\
&= \sum_{s\in\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f^{k,j}_h(x) \neq \alpha'_h(s)] + \sum_{s\notin\mathcal{G}_{k,i+1,h}} \mu_{k,i+1,h}(s) \Pr_{x\sim q(\cdot|s)}[f^{k,j}_h(x) \neq \alpha^*_h(s)] \\
&\leq \epsilon^2 + \epsilon'/(8H) < \epsilon'/(4H).
\end{aligned}$$

Taking a union bound over all $f^{k,j}_{[H+1]}$, we have that for any $h \in [H+1]$, with probability at least $1 - \mathcal{O}(H\delta_1)$,

$$\begin{aligned}
&\Pr_{s\sim\mu_{k,j+1,h},x\sim q(\cdot|s)}[f^{k,j}_h(x) = \alpha^*_h(s)] \geq \Pr_{s\sim\mu_{k,j+1,h},x\sim q(\cdot|s)}[f^{k,j}_h(x) = \alpha^*_h(s) = f^{k,i}_h(x)] \\
&\geq \Pr_{\text{for all } h'\in[h],s_{h'}\sim\mu_{k,j+1,h'},x_{h'}\sim q(\cdot|s_{h'})}[\text{ for all } h' \in [h], f^{k,j}_{h'}(x_{h'}) = \alpha^*_{h'}(s) = f^{k,i}_{h'}(x_{h'})] \\
&= \Pr_{\text{for all } h'\in[h],s_{h'}\sim\mu_{k,i+1,h'},x_{h'}\sim q(\cdot|s_{h'})}[\text{ for all } h' \in [h], f^{k,j}_{h'}(x_{h'}) = \alpha^*_{h'}(s) = f^{k,i}_{h'}(x_{h'})] \\
&\geq 1 - (\epsilon'/(2H) + \epsilon'/(4H)) \cdot H \geq 1 - \epsilon'.
\end{aligned}$$

Therefore, if case 2 and subcase a) happens, the desired result is obtained.

If subcase b) happens, we consider the function $f^{k,i+1}_h$. By Equation (4.6),

$$\begin{aligned}
&\mu_{k,i+1,h}(s^*) \cdot \Pr_{x\sim q(\cdot|s^*)}[f^{k,i+1}_h(x) \neq \alpha'_h(s^*)] \leq \epsilon^2 \\
\Rightarrow &\Pr_{x\sim q(\cdot|s^*)}[f^{k,i+1}_h(x) \neq \alpha'_h(s^*)] \leq \epsilon^2/(\epsilon'/(8H|\mathcal{S}|)) \leq \epsilon,
\end{aligned}$$

where α'_h here is some good permutation between $f^{k,i+1}_h$ and f_h. Thus,

$$\mu_{k,i+1,h}(s^*) \cdot \Pr_{x\sim q(\cdot|s^*)}\left[f^{k,i+1}_h(x) = \alpha'_h(s^*)\right] > \frac{\epsilon'}{8H|\mathcal{S}|} \cdot (1-\epsilon) > 6\epsilon \cdot \log(\delta_1^{-1}).$$

By Chernoff bound, with probability at least $1 - \mathcal{O}(\delta_1)$,

$$|\{x \in \mathcal{D}''_{k,i+1,h} : f^{k,i+1}_h(x) = \alpha'_h(s^*)\}| \geq 3\epsilon \cdot B \log(\delta_1^{-1}).$$

Therefore, the state s^* will be confirmed in iteration $i+1$ and $\alpha_h^*(s^*) = \alpha_h'(s^*)$ is defined.

In conclusion, for each iteration, there are two scenarios, either the desired result in Lemma 4.3.2 holds already or a new state will be confirmed for the next iteration. Since there are in total $\sum_{h=1}^{H+1} |\mathcal{S}_h| \leq (H+1)|\mathcal{S}|$ states, after $J := (H+1)|\mathcal{S}| + 1$ iterations, by Lemma 4.3.1, with probability at least $1 - \mathcal{O}(H|\mathcal{S}|\delta_1)$, for every $j \geq J$, for all $h \in [H+1]$ and all $s \in \mathcal{S}_h$, we have $\Pr_{x\sim q(\cdot|s)}[f_h^{k,j}(x) \neq \alpha_h^*(s)] \leq \epsilon$. Therefore, it holds that for

$$\Pr_{s\sim\mu_{k,j+1,h}, x\sim q(\cdot|s)}(f_h^{k,j}(x) \neq \alpha_h^*(s)) \leq \epsilon < \epsilon'.$$

■

Proposition 5. *Suppose in Definition 3,* $g^{-1}(\epsilon, \delta_1) = \text{poly}(1/\epsilon, \log(\delta_1^{-1}))$ *for any* $\epsilon, \delta_1 \in (0,1)$ *and* $\mathscr{A}$ *is* (ε, δ_2)*-correct with sample complexity* $\text{poly}\left(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log\left(\delta_2^{-1}\right)\right)$ *for any* $\varepsilon, \delta_2 \in (0,1)$*. Then for each iteration of the outer loop of Algorithm 6, the policy* ϕ^n *is an* $\varepsilon/3$*-optimal policy for the BMDP with probability at least* 0.99*, using at most* $\text{poly}\left(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon\right)$ *trajectories.*

Proof. We first show that the trajectory obtained by running π^k with the learned decoding functions $f_{[H+1]}^{k,J}$ matches, with high probability, that from running π^k with $\alpha_{[H+1]}^* \circ f_{[H+1]}$. Let $K = C(\varepsilon/4, \delta_2)$ be the total number of episodes played by $\mathscr{A}$ to learn an $\varepsilon/4$-optimal policy with probability at least $1 - \delta_2$. For each episode $k \in [K]$, let the trajectory of observations be $\{x_h^k\}_{h=1}^{H+1}$. We define event

$$\mathcal{E}_k := \{\forall h \in [H+1], f_h^{k,J}(x_h^k) = \alpha_h^*(f_h(x_h^k))\},$$

where $J = (H+1)|\mathcal{S}| + 1$. Note that on $\mathcal{E}_k$, the trajectory of running $\pi^k \circ \alpha_{[H+1]}^* \circ f_{[H+1]}$ equals running $\pi^k \circ f_{[H+1]}^{k,J}$. We also let the event $\mathcal{F}$ be that $\mathcal{ULCO}$ succeeds on every iteration (satisfies Lemma 4.3.2). Thus,

$$\Pr[\mathcal{F}] \geq 1 - K \cdot J \cdot \delta_1 = 1 - \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta_1^{-1})) \cdot \delta_1.$$

Furthermore, each $x_{k,h}$ is obtained by the distribution $\sum_s \mu_{k,J+1,h}(s)q(\cdot|s)$. On $\mathcal{F}$, by Lemma 4.3.2, we have

$$\Pr[f_h^{k,J}(x_h^k) = \alpha_h^*(f_h(x_h^k))] \leq \epsilon'$$

by the choice of B. Therefore,

$$\Pr[\mathcal{E}_k|\mathcal{F}] \geq 1 - (H+1)\epsilon'.$$

Overall, we have

$$\Pr\left[\mathcal{E}_k,\ \forall k \in [K]\middle|\mathcal{F}\right] \geq 1 - K(H+1)\epsilon'.$$

Thus, with probability at least $1 - \delta_2 - \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta_1^{-1})) \cdot (\epsilon' + \delta_1)$, $\mathcal{A}$ outputs a policy π, that is $\varepsilon/4$-optimal for the underlying MDP with state sets $\{\mathcal{S}_h\}_{h=1}^{H+1}$ permutated by $\alpha_{[H+1]}^*$, which we denote as event $\mathcal{E}'$. Conditioning on $\mathcal{E}'$, since on a high probability event $\mathcal{E}''$ with $\Pr[\mathcal{E}''] \geq 1 - (H+1)\epsilon'$, $\pi \circ f_{[H+1]}^{K,J}$ and $\pi \circ \alpha_{[H+1]}^* \circ f_{[H+1]}$ have the same trajectory, the value achieved by $\pi \circ f_{[H+1]}^{K,J}$ and $\pi \circ \alpha_{[H+1]}^* \circ f_{[H+1]}$ differ by at most $(H+1)^2\epsilon'$. Thus, with probability at least $1 - \delta_2 - \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta_1^{-1})) \cdot (\epsilon' + \delta_1)$, the output policy $\pi \circ f_{[H+1]}^{K,J}$ is at least $\varepsilon/4 + \mathcal{O}(H^2\epsilon')$ accurate, i.e.,

$$V_1^* - V_1^{\pi \circ f_{[H+1]}^{K,J}} \leq V_1^* - V_1^{\pi \circ \alpha_{[H+1]}^* \circ f_{[H+1]}} + \ \mathcal{O}(H^2\epsilon') \leq \ \varepsilon/4 \ + \ \mathcal{O}(H^2\epsilon').$$

Setting ϵ', δ_1, and δ_2 properly, $V_1^* - V_1^{\pi \circ f_{[H+1]}^{K,J}} \leq \varepsilon/3$ with probability at least 0.99. Since $1/\delta_1 = \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon)$ and $1/\epsilon = \text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta_1^{-1}))$, B_0 in Lemma 4.3.1 and Lemma 4.3.2 is $\text{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon)$. The desired result is obtained. ■

Finally, based on Proposition 5, we establish Theorem 4.3.1.

Proof of Theorem 4.3.1. By Proposition 5 and taking $N = \lceil \log(2/\delta)/2 \rceil$, with probability at least $1 - \delta/2$, there exists a policy in $\{\phi^n\}_{n=1}^N$ that is $\varepsilon/3$-optimal for the BMDP. For each policy ϕ^n, we take $L := \lceil 9H^2/(2\varepsilon^2) \log(2N/\delta) \rceil$ episodes to evaluate its value. Then by Hoeffding's inequality, with probability at least $1 - \delta/(2N)$,

$$|\bar{V}_1^{\phi^n} - V_1^{\phi^n}| \leq \varepsilon/3.$$

By taking the union bound and selecting the policy $\phi \in \operatorname{argmax}_{\phi \in \phi^{[N]}} \bar{V}_1^{\phi}$, with probability at least $1-\delta$, it is ε-optimal for the BMDP. In total, the number of needed trajectories is $N \cdot \sum_{k=1}^{K} \sum_{i=1}^{J} \big((k-1)J+i+1\big)B+N \cdot L = \mathcal{O}(N \cdot K^2 \cdot J^2 \cdot B+N \cdot L) = \operatorname{poly}(|\mathcal{S}|, |\mathcal{A}|, H, 1/\varepsilon, \log(\delta^{-1}))$. We complete the proof. ■

4.4 Examples of Unsupervised Learning Oracle

In this section, we give some examples of $\mathcal{ULO}$. First notice that the generation process of $\mathcal{ULO}$ is termed as the mixture model in statistics [MB88, MP04], which has a wide range of applications (see e.g., [BF20]). We list examples of mixture models and some algorithms as candidates of $\mathcal{ULO}$.

Gaussian Mixture Models (GMM) In GMM, $q(\cdot|s) = \mathcal{N}(s, \sigma_s^2)$, i.e., observations are hidden states plus Gaussian noise.[2] When the noises are (truncated) Gaussian, under certain conditions, e.g. states are well-separated, we are able to identify the latent states with high accuracy. A series of works [AK01, VW04, AM05, DS00, RV17] proposed algorithms that can be served as $\mathcal{ULO}$.

Bernoulli Mixture Models (BMM) BMM is considered in binary image processing [JV04] and texts classification [JV02]. In BMM, every observation is a point in $\{0,1\}^d$. A true state determines a frequency vector. In [NMR20], the authors proposed a reliable clustering algorithm for BMM data with polynomial sample complexity guarantee.

Subspace Clustering In some applications, each state is a set of vectors and observations lie in the spanned subspace. Suppose for different states, the basis vectors differ under certain metric, then recovering the latent state is equivalent to subspace clustering. Subspace

[2]To make the model satisfy the disjoint block assumption in the definition of BMDP, we need some truncation of the Gaussian noise so that each observation only corresponds to a unique hidden state.

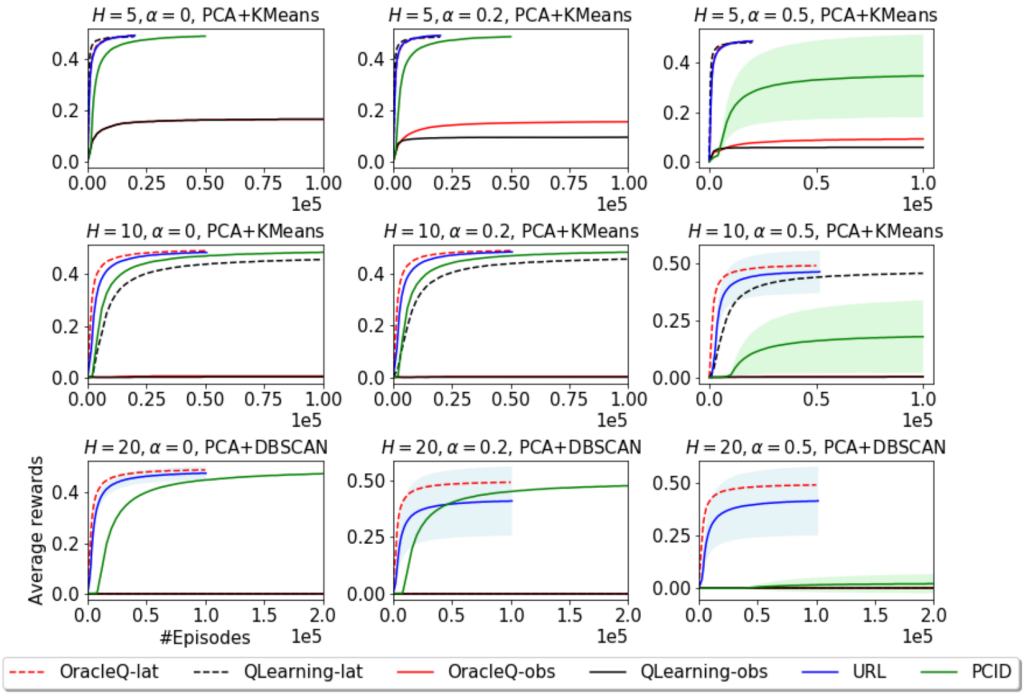

Figure 4.1: Performances for LockBernoulli.

clustering has a variety of applications include face clustering, community clustering, and DNA sequence analysis [WSW15, Vid11, EV13]. Proper algorithms for $\mathcal{ULO}$ can be found in e.g., [WXL13, SEC14].

In addition to the aforementioned models, other reasonable settings are Categorical Mixture Models [BT13], Poisson Mixture Models [LZ06], Dirichlet Mixture Models [Dah06] and so on.

4.5 Numerical Experiments

In this section we conduct experiments to demonstrate the effectiveness of our framework. Our code is available at `https://github.com/FlorenceFeng/StateDecoding`.

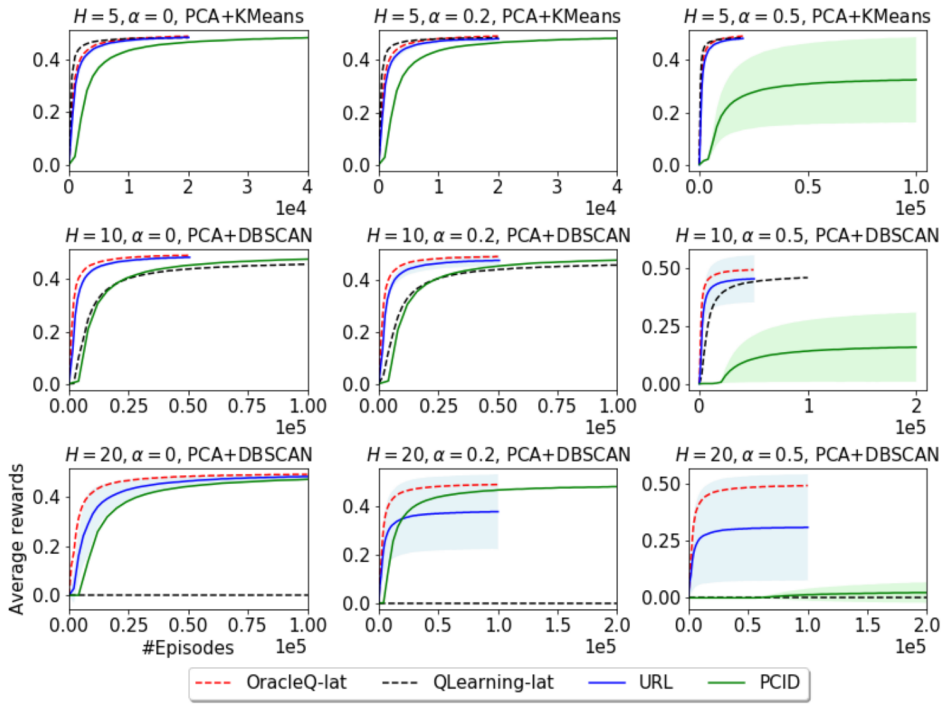

Figure 4.2: Performances for LockGaussian, $\sigma = 0.1$. All lines are mean values of 50 tests and the shaded areas depict the one standard deviations. OracleQ-lat and QLearning-lat have direct access to the latent states, which are not for practical use. URL and PCID only have access to the observations. OracleQ-obs and QLearning-obs are omitted due to infinitely many observations.

Environments We conduct experiments in two environments: LockBernoulli and LockGaussian. These environments are also studied in [DKJ19], which are designed to be hard for exploration. Both environments have the same latent state structure with H levels, 3 states per level and 4 actions. At level h, from states $s_{1,h}$ and $s_{2,h}$ one action leads with probability $1-\alpha$ to $s_{1,h+1}$ and with probability α to $s_{2,h+1}$, another has the flipped behavior, and the remaining two lead to $s_{3,h+1}$. All actions from $s_{3,h}$ lead to $s_{3,h+1}$. Non-zero reward is only achievable if the agent can reach $s_{1,H+1}$ or $s_{2,H+1}$ and the reward follows Bernoulli(0.5). Action labels are randomly assigned at the beginning of each time of training. We consider

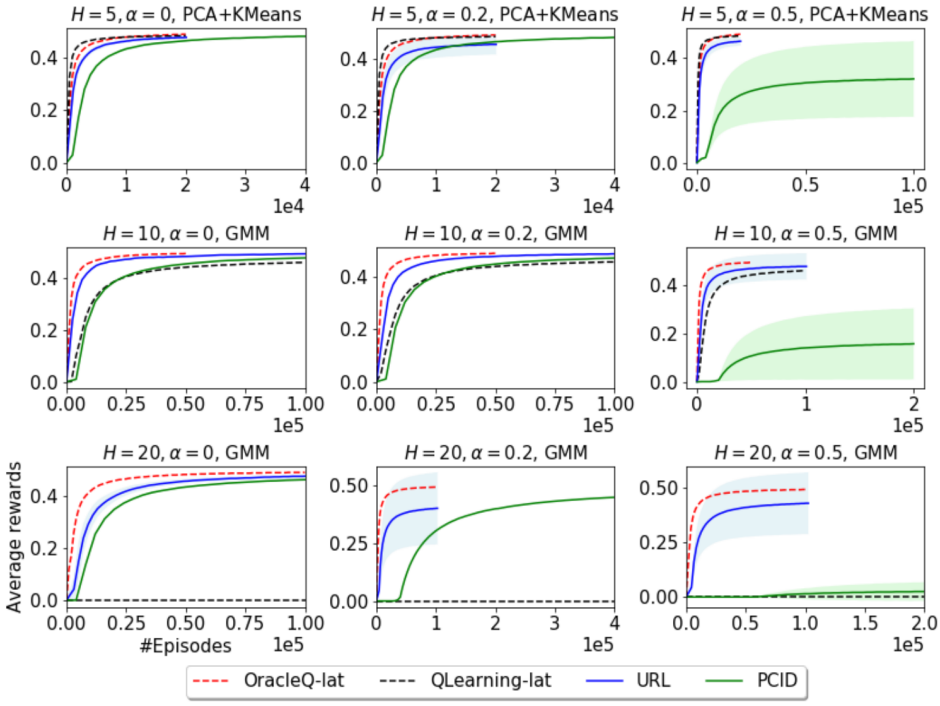

Figure 4.3: Performances for LockGaussian, $\sigma = 0.2$. All lines are mean values of 50 tests and the shaded areas depict the one standard deviations. OracleQ-lat and QLearning-lat have direct access to the latent states, which are not for practical use. URL and PCID only have access to the observations. OracleQ-obs and QLearning-obs are omitted due to infinitely many observations.

three values of α: 0, 0.2, and 0.5.

In LockBernoulli, the observation space is $\{0, 1\}^{H+3}$ where the first 3 coordinates are reserved for the one-hot encoding of the latent state and the last H coordinates are drawn i.i.d from Bernoulli(0.5). LockBernoulli meets our requirements as a BMDP. In LockGaussian, the observation space is $\mathbb{R}^{H+3}$. Every observation is constructed by first letting the first three coordinates be the one-hot encoding of the latent state, then adding i.i.d Gaussian noises $\mathcal{N}(0, \sigma^2)$ to all $H + 3$ coordinates. We consider $\sigma = 0.1$ and 0.2. LockGaussian is not a BMDP. We use this environment to evaluate the robustness of our method to violated

assumptions.

The environments are designed to be hard for exploration. There are in total 4^H choices of actions of one episode, but only 2^H of them lead to non-zero reward in the end. So random exploration requires exponentially many trajectories. Also, with a larger H, the difficulty of learning accurate decoding functions increases and makes exploration with observations a more challenging task.

Algorithms and Hyperparameters We compare 4 algorithms: OracleQ [JAB18]; QLearning, the tabular Q-Learning with ϵ-greedy exploration; URL, our method; and PCID [DKJ19]. For OracleQ and QLearning, there are two implementations: 1. they directly see the latent states (OracleQ-lat and QLearning-lat); 2. only see observations (OracleQ-obs and QLearning-obs). For URL and PCID, only observations are available. OracleQ-lat and QLearning-lat are served as a near-optimal skyline and a sanity-check baseline to measure the efficiency of observation-only algorithms. OracleQ-obs and QLearning-obs are only tested in LockBernoulli since there are infinitely many observations in LockGaussian. For URL, we use OracleQ as the tabular RL algorithm.

For OracleQ, we tune the learning rate and a confidence parameter; for QLearning, we tune the learning rate and the exploration parameter ϵ; for PCID, we follow the code provided in [DKJ19], tune the number of clusters for k-means and the number of trajectories n to collect in each outer iteration, and finally select the better result between linear function and neural network implementation.

In our method, we use OracleQ as the tabular RL algorithm to operate on the decoded state space and try three unsupervised learning approaches: 1. first conduct principle component analysis (PCA) on the observations and then use k-means (KMeans) to cluster; 2. first apply PCA, then use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) for clustering, and finally use support vector machine to fit a classifier; 3. employ Gaussian Mixture Model (GMM) to fit the observation data then generate a label predictor.

We call the python library `sklearn` for all these methods. During unsupervised learning, we do not separate observations by levels but add level information in decoded states. Besides the hyperparameters for OracleQ and the unsupervised learning oracle, we also tune the batch size B adaptively in Algorithm 7. In our tests, instead of resampling over all previous policies as Line 5 Algorithm 7, we use previous data. Specifically, we maintain a training dataset $\mathcal{D}$ in memory and for iteration i, generate B training trajectories following $\pi \circ f_{[H+1]}^{i-1}$ and merge them into $\mathcal{D}$ to train $\mathcal{ULO}$. Also, we stop training decoding functions once they become stable, which takes 100 training trajectories when $H = 5$, $500 \sim 1000$ trajectories when $H = 10$, and $1000 \sim 2500$ trajectories when $H = 20$. Since this process stops very quickly, we also skip the label matching steps (Line 8 to Line 12 Algorithm 7) and the final decoding function leads to a near-optimal performance as shown in the results.

Results The results are presented in Figure 4.1, 4.2, and 4.3. x-axis is the number of training trajectories and y-axis is average reward. All lines are mean values of 50 tests and the shaded areas depict the one standard deviations. The title for each subfigure records problem parameters and the unsupervised learning method we apply for URL. In LockBernoulli, OracleQ-obs and QLearning-obs are far from being optimal even for small-horizon cases. URL is mostly as good as the skyline (OracleQ-lat) and much better than the baseline (QLearning-lat) especially when $H = 20$. URL outperforms PCID in most cases. When $H = 20$, we observe a probability of 80% that URL returns near-optimal values for $\alpha = 0.2$ and 0.5. In LockGaussian, for $H = 20$, we observe a probability of $> 75\%$ that URL returns a near-optimal policy for $\alpha = 0.2$ and 0.5.

4.6 Conclusion

In this chapter, we gave a general framework that turns an unsupervised learning algorithm and a no-regret tabular RL algorithm into an algorithm for RL problems with huge observation

spaces. We provided theoretical analysis to show it is provably efficient. We also conducted numerical experiments to show the effectiveness of our framework in practice. An interesting future theoretical direction is to characterize the optimal sample complexity under our assumptions.

CHAPTER 5

RL Exploration with General Function Approximation

In this chapter, we provide an algorithm for efficient exploration with general rich observation spaces. Specifically, we apply policy optimization methods. Policy optimization methods remain a powerful workhorse in empirical Reinforcement Learning (RL), with a focus on neural policies that can easily reason over complex and continuous state and/or action spaces. Theoretical understanding of strategic exploration in policy-based methods with non-linear function approximation, however, is largely missing. In this chapter, we address this question by designing ENIAC, an actor-critic method that allows non-linear function approximation in the critic. We show that under certain assumptions, e.g., a bounded eluder dimension d for the critic class, the learner finds to a near-optimal policy in $\widetilde{O}(\text{poly}(d))$ exploration rounds. The method is robust to model misspecification and strictly extends existing works on linear function approximation. We also develop some computational optimizations of our approach with slightly worse statistical guarantees, and an empirical adaptation building on existing deep RL tools. We empirically evaluate this adaptation, and show that it outperforms prior heuristics inspired by linear methods, establishing the value in correctly reasoning about the agent's uncertainty under non-linear function approximation.

The contributions in this chapter were first presented in the joint work with Wotao Yin, Alekh Agarwal, and Lin Yang [FYA21].

5.1 Introduction

The success of reinforcement learning (RL) in many empirical domains largely relies on developing policy gradient methods with deep neural networks [SLA15, SWD17, HZA18]. The techniques have a long history in RL [Wil92, SMS99, KT00]. A number of theoretical results study their convergence properties [KL02, SG14, GSP19, ABB19, AKL20a, BR19] when the agent has access to a distribution over states which is sufficiently exploratory, such as in a generative model. However, unlike their value- or model-based counterparts, the number of policy-based approaches which actively explore and provably find a near-optimal policy remains relatively limited, and restricted to tabular [SER20] and linear function approximation [CYJ20, AHK20] settings. Given this gap between theory and the empirical literature, it is natural to ask how we can design provably sample-efficient policy-based methods for RL that allow the use of general function approximation, such as via neural networks.

In this chapter we design an actor-critic method with general function approximation: *Exploratory Non-linear Incremental Actor Critic (ENIAC)*. Our method follows a similar high-level framework as [AHK20], but with a very different bonus function in order to reason about the uncertainty of our non-linear critic. In each iteration, we use the bonus to learn an optimistic critic, so that optimizing the actor with it results in exploration of the previously unseen parts of the environment. Unlike [AHK20], we allow non-linear function approximation in the critic, which further parameterizes a non-linear policy class through Soft Policy Iteration (SPI) [EKM09, HZA18, GSP19, ABB19, AHK20] or Natural Policy Gradient (NPG) [Kak01, PS08, AKL20a] updates. Theoretically, we show that if the critic function class has a bounded *eluder dimension* [RV13] d, then our algorithm outputs a near-optimal policy in poly(d) number of interactions, with high probability, for both SPI and NPG methods.

Unlike the linear setting studied in [AHK20], whose bonus functions can be computed in

closed form, the bonus function for a general function class is considerably more complex. Following the recent work on non-linear value-based methods by [WSY20], the bonus function is based on the *range of values* (or the width function) predicted at a particular state-action pair by the critic function which accurately predicts the observed returns. Hence, this function characterizes how uncertain we are about a state-action pair given the past observations. The value-based method in [WSY20] relies on solving the value iteration problem using the experience, which introduces dependence issues across different stages of the algorithm. But, we directly use the width function as our exploration bonus and have a simpler sub-sampling design that that in [WSY20]. Under mild assumptions, our bonus function can be computed in a time polynomially depending on the size of the current dataset. We also provide a heuristic method to compute the bonus functions for neural networks. Furthermore, all our results are robust to model misspecification and do not require an explicit specification about the transition dynamics as used in [WSY20].

In order to further improve the efficiency, we develop variants of our methods that require no bonus computation in the execution of the actor. The key idea is to replace certain conditional exploration steps triggered by the bonus with a small uniform exploration. Note that this uniform exploration is in addition to the optimistic reasoning, thus different from vanilla ϵ-greedy methods. The bonus is later incorporated while updating the critic, which is a significant optimization in settings where the actor runs in real-time with resource constrained hardware such as robotic platforms [PCS18], and plays well with existing asynchronous actor-critic updates [MBM16].

We complement our theoretical analysis with empirical evaluation on a continuous control domain requiring non-linear function approximation, and show the benefit of using a bonus systematically derived for this setting over prior heuristics from both theoretical and empirical literature.

5.1.1 Related Work

The rich literature on exploration in RL primarily deals with tabular [KS02a, BT02, JOA10, JAB18] and linear [YW19a, JYW20b] settings with value- or model-based methods. Recent papers [SER20, CYJ20, AHK20] have developed policy-based methods also in the same settings. Of these, our work directly builds upon that of [AHK20], extending it to non-linear settings.

For general non-linear function approximation, a series of papers provide statistical guarantees under structural assumptions [JKA17, SJK19, DJK18], but these do not lend themselves to computationally practical versions. Other works [DKJ19, MHK20, AKK20] study various latent variable models for non-linear function approximation in model-based settings. The notion of eluder dimension [RV13] used in our theory has been previously used to study RL in deterministic settings [WV13]. Most related to our work are the recent value-based technique of [WSY20], which describes a UCB-VI style algorithm with statistical guarantees scaling with eluder dimension and the model-based policy optimization of [CYS21], which incorporates optimism into policy evaluation via building confidence sets of the transition model and uses eluder dimension to define model capacity. In this chapter, we instead study model-free policy-based methods, which provide better robustness to misspecification in theory and are more amenable to practical implementation.

Notation Given a set $\mathcal{A}$, we denote by $|\mathcal{A}|$ the cardinality of $\mathcal{A}$, $\Delta(\mathcal{A})$ the set of all distributions over $\mathcal{A}$, and $\text{Unif}(\mathcal{A})$ the uniform distribution over $\mathcal{A}$. We use $[n]$ for the index set $\{1, \ldots, n\}$. Let $a, b \in \mathbb{R}^n$. We denote by $a^\top b$ the inner product between a and b and $\|a\|_2$ the Euclidean norm of a. Given a matrix A, we use $\|A\|_2$ for the spectral norm of A. Given a function $f : \mathcal{X} \to \mathbb{R}$ and a finite dataset $\mathcal{Z} \subset \mathcal{X}$, we define $\|f\|_{\mathcal{Z}} := \sqrt{\sum_{x \in \mathcal{Z}} f(x)^2}$. We abbreviate Kullback-Leibler divergence to **KL** and use $\mathcal{O}$ for leading orders in asymptotic upper bounds and $\widetilde{\mathcal{O}}$ to hide the polylog factors.

5.2 Setting

Markov Decision Process In this paper, we focus on the discounted Markov Decision Process (MDP) with an infinite horizon. We use $\mathcal{M}$ to represent an MDP. Each MDP is described as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is a possibly infinite state space, $\mathcal{A}$ is a finite action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ specifies a transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a reward function, and $\gamma \in (0, 1)$ is a discount factor.

At each time step, the agent observes a state $s \in \mathcal{S}$ and selects an action $a \in \mathcal{A}$ according to a *policy* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The environment then transitions to a new state s' with probability $P(s'|s, a)$ and the agent receives an instant reward $r(s, a)$.

For a policy π, its Q-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$Q^\pi(s, a, r) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right],$$

where the expectation is taken over the trajectory following π. And the value function is $V^\pi(s, r) := \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s, a, r)]$. From V^π and Q^π, the advantage function of π is: $A^\pi(s, a, r) = Q^\pi(s, a, r) - V^\pi(s, r), \forall s \in \mathcal{S}, a \in \mathcal{A}$. We ignore r in V, Q or A, if it is clear from the context.

Besides value, we are also interested in the distribution induced by a policy. Specifically, we define the discounted state-action distribution $d^\pi_{\tilde{s}}(s, a)$ induced by π as:

$$d^\pi_{\tilde{s}}(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = a | s_0 = \tilde{s}),$$

where $\mathrm{Pr}^\pi(s_t = s, a_t = a|s_0 = \tilde{s})$ is the probability of reaching (s, a) at the t_{th} step starting from $\tilde{s}$ following π. Similarly, we define $d^\pi_{\tilde{s},\tilde{a}}(s, a)$ if the agent starts from state $\tilde{s}$ followed by action $\tilde{a}$ and follows π thereafter. For any distribution $\nu \in \Delta(\mathcal{S} \times \mathcal{A})$, we denote by $d^\pi_\nu(s, a) := \mathbb{E}_{(\tilde{s},\tilde{a})\sim\nu} [d^\pi_{(\tilde{s},\tilde{a})}(s, a)]$ and $d^\pi_\nu(s) := \sum_a d^\pi_\nu(s, a)$.

Given an initial distribution $\rho \in \Delta(\mathcal{S})$, we define $V^\pi_\rho := \mathbb{E}_{s_0\sim\rho}[V^\pi(s_0)]$. Similarly, if $\nu \in \Delta(\mathcal{S} \times \mathcal{A})$, we define $V^\pi_\nu := \mathbb{E}_{(s_0,a_0)\sim\nu}[Q^\pi(s_0, a_0)]$. The goal of RL is to find a policy in

some policy space Π such that its value with respect to an initial distribution ρ_0 is maximized, i.e.,

$$\underset{\pi \in \Pi}{\text{maximize}}\, V_{\rho_0}^{\pi}.$$

Without loss of generality, we consider the RL problems starting from the unique initial state s_0 in the later context. All the results straightforwardly apply to arbitrary ρ_0.

Function Class and Policy Space Let $\mathcal{F} := \{f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}\}$ be a general function class, e.g., neural networks. We denote by $\Pi_{\mathcal{F}} := \{\pi_f, f \in \mathcal{F}\}$ a policy space induced by applying the softmax transform to functions in $\mathcal{F}$, i.e.,

$$\pi_f(a|s) = \frac{\exp(f(s,a))}{\sum_{a' \in \mathcal{A}} \exp(f(s,a'))}.$$

For the ease of presentation, we assume there exists a function $f \in \mathcal{F}$ such that, for all s, $\pi_f(\cdot|s)$ is a uniform distribution[1] on $\mathcal{A}$. Given $\mathcal{F}$, we define its *function-difference* class $\Delta\mathcal{F} := \{\Delta f \mid \Delta f = f - f',\ f, f' \in \mathcal{F}\}$ and the width function on $\Delta\mathcal{F}$ as:

$$w(\Delta\mathcal{F}, s, a) := \sup_{\Delta f \in \Delta\mathcal{F}} \Delta f(s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{5.1}$$

Note that our width is defined on the function difference class instead of the original function class $\mathcal{F}$, where the latter is adopted in [RV13] and [WSY20]. These two formulations are essentially equivalent.

If $\mathcal{F}$ can be smoothly parameterized by $\theta \in \mathbb{R}^d$, we further introduce the (centered) *tangent class* of $\mathcal{F}_\theta$ as:

$$\mathcal{G}_{\mathcal{F}} := \{g_\theta^u \mid g_\theta^u(s,a) := u^\top \nabla_\theta \log \pi_{f_\theta}(s,a), u \in \mathcal{U}, f_\theta \in \mathcal{F}\}, \tag{5.2}$$

where $\mathcal{U} \subset \mathbb{R}^d$ is some bounded parameter space. We define the function-difference class $\Delta\mathcal{G}_{\mathcal{F}}$ and the width function $w(\Delta\mathcal{G}_{\mathcal{F}}, s, a)$ for $\mathcal{G}_{\mathcal{F}}$ accordingly.

[1]This requirement is not strict, our algorithms and analysis apply for any distribution that are supported on all actions.

Algorithm 9 Exploratory Non-Linear Incremental Actor Critic (ENIAC)

Input: Function class $\mathcal{F}$.

Hyperparameters: $N > 0$, $K > 0$, $\beta > 0$, $\alpha \in (0, 1)$.

For all $s \in \mathcal{S}$, initialize $\pi^1(\cdot|s) = \text{Unif}(\mathcal{A})$.

Let experience buffer $\mathcal{Z}^0 = \emptyset$.

for $n = 1$ **to** N **do**

    Generate K samples: $\{s_i, a_i\}_{i=1}^K \sim d_{s_0}^{\pi^n}$;

    Merge training set: $\mathcal{Z}^n \leftarrow \mathcal{Z}^{n-1} \cup \{s_i, a_i\}_{i=1}^K$;

    Let $\rho_{\text{cov}}^n := \text{Unif}(d_{s_0}^{\pi^1}, \ldots, d_{s_0}^{\pi^n})$;

    Define a bonus function b^n using (5.12) or (5.13);

    Update the policy using Algorithm 10: $\pi^{n+1} \leftarrow \text{Policy Update}(\rho_{\text{cov}}^n, b^n, \alpha)$.

end for

Output: $\text{Unif}(\pi_2, \pi_3, \ldots, \pi_{N+1})$

Next, given a function class $\mathcal{F}$, we consider RL on the induced policy space $\Pi_{\mathcal{F}}$. If $\mathcal{F}$ is non-smooth, we apply SPI as the policy optimization routine while approximating Q-values with $\mathcal{F}$; if $\mathcal{F}$ is smoothly parameterized by θ, we can alternatively apply NPG for policy optimization and use $\mathcal{G}_{\mathcal{F}}$ to approximate advantage functions. The corresponding function-difference classes are used to design bonus functions and guide exploration.

5.3 Algorithms

In this section, we describe our algorithm, *Exploratory Non-Linear Incremental Actor Critic* (ENIAC), which takes a function class $\mathcal{F}$ and interacts with an RL environment to learn a good policy. The formal pseudo-code is presented in Algorithm 9. We explain the high-level design and steps in the algorithm in this section, before giving our main results in the next section.

Algorithm 10 Policy Update

Input: Fitting distribution ρ, bonus function b, α.

Hyperparameters: $T > 0$, $M > 0$, $\eta > 0$.

Initialize π_0 using (5.3) or (5.4).

for $t = 0$ **to** $T - 1$ **do**

    Generate M samples from ρ using (5.5) or (5.9);

    Fit critic to the M samples using (5.6) or (5.10);

    Actor update using (5.7), (5.8), or (5.11) to obtain π_{t+1};

end for

Output: $\text{Unif}(\pi_0, \pi_1, \ldots, \pi_{T-1})$

5.3.1 High-level Framework

At a high-level, ENIAC solves a series of policy optimization problems in a sequence of carefully designed MDPs. Each MDP is based on the original MDP, but differs in the choice of an *initial state distribution* and a *reward bonus*. We use them to induce optimistic bias to encourage exploration. Through the steps of the algorithm, the initial distribution gains coverage, while the bonus shrinks so that good policies in the modified MDPs eventually yield good policies in the original MDP as well.

A key challenge in large state spaces is to quantify the notion of *state coverage*, which we define using the function class $\mathcal{F}$. We say a distribution ρ_{cov} provides a good coverage if any function $f \in \mathcal{F}$ that has a small prediction error on data sampled from ρ_{cov} also has a small prediction error under the state distribution d^π for any other policy π. In tabular settings, this requires ρ_{cov} to visit each state, while coverage in the feature space suffices for linear MDPs [JYW20b, YW19a].

In ENIAC, we construct such a covering distribution ρ_{cov} iteratively, starting from the state distribution of a uniform policy and augmenting it gradually as new policies visit

previously unexplored parts of the MDP. Concretely, we maintain a *policy cover* $\{\pi^1, \pi^2, \ldots\}$, which initially contains only a random policy, π^1, (Line 3 of Algorithm 9). At iteration n, the algorithm lets ρ^n_{cov} be a uniform mixture of $\{d^{\pi^1}_{s_0}, d^{\pi^2}_{s_0}, \ldots, d^{\pi^n}_{s_0}\}$ (line 8).

Having obtained the cover, we move on to induce the reward bonus by collecting a dataset of trajectories from ρ^n_{cov} (line 6).[2] These collected trajectories are used to identify a set $\mathcal{K}^n$ of state-action pairs covered by ρ^n_{cov}: any functions $f, g \in \mathcal{F}$ that are close under ρ^n_{cov} also approximately agree with each other for all $s, a \in \mathcal{K}^n$. We then create a *reward bonus*, b^n (Line 9, formally defined later), toward encouraging explorations outside the set $\mathcal{K}^n$.

Finally, taking ρ^n_{cov} as the initial distribution and the bonus augmented reward $r + b^n$ as the reward function, we find a policy π that approximately maximizes $V^{\pi}_{\rho^n_{\text{cov}}}(r + b^n)$ (line 10). It can be shown that this policy either *explores* by reaching new parts of the MDP or *exploits* toward identifying a near optimal policy. We then add this policy to our cover and proceed to the next epoch of the algorithm .

Within this high-level framework, different choices of the policy update and corresponding bonus functions induce different concrete variants of Algorithm 9. We describe these choices below.

5.3.2 Policy Optimization

In this section, we describe our policy optimization approach, given a policy cover ρ and a reward bonus b. We drop the dependence on epoch n for brevity, and recall that the goal is to optimize $V^{\pi}_{\rho}(r + b)$. We present two different actor critic style optimization approaches: Soft Policy Iteration (SPI) and Natural Policy Gradient (NPG), which offer differing tradeoffs in generality and practical implementation. SPI is amenable to arbitrary class $\mathcal{F}$, while NPG requires second-order smoothness. On the other hand, NPG induces fully convex critic

[2] In the Algorithm 9, only π^n is rolled out as the samples can be combined with historical data to form samples from ρ^n_{cov}.

objective for any class $\mathcal{F}$, and is closer to popular optimization methods like TRPO, PPO and SAC. Our presentation of both these methods is adapted from [AKL20a], and we describe the overall outline of these approaches in Algorithm 10, with the specific update rules included in the rest of this section.

For each approach, we provide a *sample-friendly* version and a *computation-friendly* version for updating the policy. The two versions of updating methods only differ in the initialization and actor updating steps. The computation-friendly version provides a policy that can be executed efficiently while being played. The sample-friendly version requires to compute the bonus function during policy execution but saves samples up to poly($|\mathcal{A}|$) factors. We now describe these procedures in more details.

5.3.2.1 Policy Initialization

For both SPI and NPG approaches, we use the following methods to initialize the policy.

Sample-friendly initialization. Given bonus b, we define $\mathcal{K} := \{(s,a) \mid b(s,a) = 0\}$. We abuse the notation $s \in \mathcal{K}$ if $b(s,a) = 0,\ \forall a \in \mathcal{A}$. We initialize the policy as follows.

$$\pi_0(\cdot|s) = \begin{cases} \text{Unif}(\mathcal{A}) & s \in \mathcal{K}; \\ \text{Unif}(\{a \in \mathcal{A} : (s,a) \notin \mathcal{K}\}) & \text{o.w.} \end{cases} \tag{5.3}$$

Here the policy selects actions uniformly for states where all actions have been well-explored under ρ and only plays actions that are not well-covered in other states. Note that such a policy can be represented by b and a function $f \in \mathcal{F}$.

Computation-friendly initialization. The computation-friendly method does not recompute the set $\mathcal{K}$ and initialize the policy to be purely random, i.e.,

$$\pi_0(\cdot|s) = \text{Unif}(\mathcal{A}),\ \forall s \in \mathcal{S}. \tag{5.4}$$

5.3.2.2 SPI Policy Update

For each iteration, t, we first generate M (some parameter to be determined) Q-value samples with the input distribution ρ as the initial distribution:

$$\{s_i, a_i, \widehat{Q}^{\pi_t}(s_i, a_i, r+b)\}_{i=1}^{M},\ (s_i, a_i) \sim \rho, \tag{5.5}$$

where $\widehat{Q}^{\pi_t}$ is an unbiased estimator of Q^{π_t} (see, e.g., Algorithm 1 in Chapter 1). Then we fit a critic to the above samples by setting f_t as a solution of :

$$\underset{f\in\mathcal{F}}{\text{minimize}} \sum_{i=1}^{M} \left(\widehat{Q}^{\pi_t}(s_i, a_i, r+b) - b(s_i, a_i) - f(s_i, a_i)\right)^2. \tag{5.6}$$

Here we offset the fitting with the initial bonus to maintain consistency with linear function approximation results, where a non-linear bonus introduces an approximation error [JYW20b, AHK20]. Note that for the SPI, we do not require f to be differentiable.

Based on the critic, we update the actor to a new policy. There are two update versions: one is more sample-efficient, the other is more computational-convenient.

Sample-friendly version. For this version, we only update the policy on states $s \in \mathcal{K}$ since our critic is unreliable elsewhere. For $s \notin \mathcal{K}$, we keep exploring previously unknown actions by simply sticking to the initial policy. Then the policy update rule is:

$$\pi_{t+1}(a|s) \propto \pi_t(a|s) \exp\left(\eta f_t(s,a) \mathbf{1}\{s \in \mathcal{K}\}\right), \tag{5.7}$$

where $\eta > 0$ is a step size to be specified. Note that since $b(s,a) \equiv 0$ for $s \in \mathcal{K}$, Equation (5.7) is equivalent to $\pi_{t+1}(a|s) \propto \pi_t(a|s) \exp\left(\eta(f_t(s,a) + b(s,a))\mathbf{1}\{s \in \mathcal{K}\}\right)$ where the initial bonus is added back.

Computation-friendly version. For this version, we remove the indicator function while allowing some probability of uniform exploration:

$$\pi'_{t+1}(a|s) \propto \pi'_t(a|s) \exp\left(\eta f_t(s,a)\right), \quad \pi_{t+1}(\cdot|s) = (1-\alpha) \cdot \pi'_{t+1} + \alpha \cdot \text{Unif}(\mathcal{A}). \tag{5.8}$$

Above, $\{\pi'_t\}$ is an auxiliary sequence of policies initialized as π_0 and $\alpha > 0$. Note that for $s \in \mathcal{K}$, since $b(s,a) \equiv 0$ we still have $\pi'_{t+1}(a|s) \propto \pi'_t(a|s) \exp\big(\eta(f_t(s,a) + b(s,a))\big)$, i.e., the offset initial bonus is added back. Thus, compared with Equation (5.7), Equation (5.8) differs at: 1. α-probability random exploration for $s \in \mathcal{K}$; 2. update policy for $s \notin \mathcal{K}$ with a possibly not correct value (if $b(s,a) \neq 0$) but guarantees at least α-probability random exploration. Such a change creates a polynomial scaling with $|\mathcal{A}|$ in the sample complexity but saves us from computing bonuses during policy execution which is required by the sample-friendly version.

5.3.2.3 NPG Policy Update

NPG update shares the same structure as that for SPI. Recall that now the function class $\mathcal{F}$ is smoothly parameterized by θ. At each iteration t, we first generate M (some parameter to be determined) advantage samples from the input distribution ρ,

$$\{s_i, a_i, \widehat{A}^{\pi_t}(s_i, a_i, r+b)\}_{i=1}^{M}, \ (s_i, a_i) \sim \rho \tag{5.9}$$

where $\widehat{A}^{\pi_t}$ is an unbiased estimator of A^{π_t} (using Algorithm 1). We define $\bar{b}_t(s,a) := b(s,a) - \mathbb{E}_{a\sim\pi_t(\cdot|s)}[b(s,a)]$ as a centered version of the original bonus and $g_t(s,a) := \nabla_\theta \log \pi_{f_{\theta_t}}(s,a)$ to be the tangent features at θ_t. We then fit a critic to the bonus offset target $\widehat{A}^{\pi_t} - \bar{b}_t$ by setting u_t as a solution of:

$$\underset{u\in\mathcal{U}}{\text{minimize}} \sum_{i=1}^{M} \big(\widehat{A}^{\pi_t}(s_i, a_i, r+b) - \bar{b}_t(s_i, a_i) - u^\top g_t(s_i, a_i)\big)^2. \tag{5.10}$$

Compared to SPI, a big advantage is that the above critic objective is a linear regression problem, for which any off-the-shelf solver can be used, even with a large number of samples in high dimensions.

With the critic, we update the actor to generate a new policy as below.

Sample-friendly version. Similar to the sample-friendly version of SPI, we only update

the policy on $s \in \mathcal{K}$ as:

$$\theta_{t+1} = \theta_t + \eta u_t, \quad \pi_{t+1}(a|s) \propto \exp(f_{\theta_{t+1}}(s,a)\mathbf{1}\{s \in \mathcal{K}\}), \tag{5.11}$$

where $\eta > 0$ is a step size to be specified.

We omit the details of the computation-friendly version, which is obtained similar to the counterpart in SPI.

5.3.3 Bonus Function

In this section, we describe the bonus computation given a dataset $\mathcal{Z}^n$ generated from some covering distribution ρ_{cov}. As described in previous subsections, the bonus assigns value 0 to state-action pairs that are well-covered by ρ_{cov} and a large value elsewhere. To measure the coverage, we use a width function (defined in Equation (5.1)) dependent on $\mathcal{Z}^n$. The bonus differs slightly for the SPI and NPG updates since SPI uses $\mathcal{F}$ for critic fit while NPG use $\mathcal{G}_{\mathcal{F}}$. Specifically, for the sample-friendly version, we take the following bonus function

$$b^n(s,a) = \mathbf{1}\{w(\widetilde{\mathcal{F}}^n, s, a) \geq \beta\} \cdot \frac{1}{1-\gamma}, \tag{5.12}$$

where for SPI,

$$\widetilde{\mathcal{F}}^n := \{\Delta f \in \Delta\mathcal{F} \mid \|\Delta f\|_{\mathcal{Z}^n} \leq \epsilon\}$$

and for NPG,

$$\widetilde{\mathcal{F}}^n := \{\Delta g \in \Delta\mathcal{G}_{\mathcal{F}} \mid \|\Delta g\|_{\mathcal{Z}^n} \leq \epsilon\}$$

with $\mathcal{G}_{\mathcal{F}}$ being the tangent class defined in Equation (5.2). Here β, ϵ are positive parameters to be determined. For the computation-friendly version, we scale up the bonus by a factor of $|\mathcal{A}|/\alpha$ to encourage more exploration, i.e.,

$$b^n(s,a) := \mathbf{1}\{w(\widetilde{\mathcal{F}}^n, s, a) \geq \beta\} \cdot \frac{|\mathcal{A}|}{(1-\gamma)\alpha}. \tag{5.13}$$

Remark 7. *The bonus can be computed efficiently by reducing the width computation to regression [FAD18]. We can additionally improve the computational efficiency using the sensitivity sampling technique developed in [WSY20], which significantly subsamples the dataset $\mathcal{Z}$. We omit the details for brevity. For neural networks, we provide a heuristic to approximate the bonus in Section 5.6.*

5.3.4 Algorithm Name Conventions

Since Algorithm 9 provides different options for sub-routines, we specify different names for them as below.

- ENIAC-SPI-SAMPLE (ENIAC with sample-friendly SPI update): initialize with (5.3), collect data with (5.5), fit critic using (5.6), and update actor using (5.7);
- ENIAC-SPI-COMPUTE (ENIAC with computation-friendly SPI update): initialize with (5.4), collect data with (5.5), fit critic using (5.6), and update actor using (5.8);
- ENIAC-NPG-SAMPLE (ENIAC with sample-friendly NPG update): initialize with (5.3), collect data with (5.9), fit critic using (5.10), and update actor using (5.11);
- ENIAC-NPG-COMPUTE (ENIAC with computation-friendly NPG update): initialize with (5.4), collect data with (5.9), fit critic using (5.10), and update actor using a similar fashion as (5.8) modified from (5.11).

5.4 Theory

In this section, we provide convergence results of ENIAC with both the SPI and NPG options in the update rule. We use superscript n for the n-th epoch in Algorithm 9 and the subscript t for the t-th iteration in Algorithm 10. For example, π_t^n is the output policy of the t-th iteration in the n-th epoch.

The sample complexities of our algorithms depend on the complexity of the function class for critic fit (and also the policy, implicitly). To measure the latter, we adopt the notion of

eluder dimension which is first introduced in [RV13].

Definition 5 (Eluder Dimension)**.** *Given a class $\mathcal{F}$, $\epsilon \geq 0$, and $\mathcal{Z} := \{(s_i, a_i)\}_{i=1}^n$ be a sequence of state-action pairs.*

- *A state-action pair (s, a) is ϵ-dependent on $\mathcal{Z}$ with respect to $\mathcal{F}$ if any $f, f' \in \mathcal{F}$ satisfying $\|f - f'\|_{\mathcal{Z}} := \sqrt{\sum_{(s',a') \in \mathcal{Z}} (f(s', a') - f'(s', a'))^2} \leq \epsilon$ also satisfy $|f(s, a) - f'(s, a)| \leq \epsilon$.*
- *An (s, a) is ϵ-independent of $\mathcal{Z}$ with respect to $\mathcal{F}$ if (s, a) is not ϵ-dependent on $\mathcal{Z}$.*
- *The ϵ-eluder dimension $dim_E(\mathcal{F}, \epsilon)$ of a function class $\mathcal{F}$ is the length of the longest sequence of elements in $\mathcal{S} \times \mathcal{A}$ such that, for some $\epsilon' \geq \epsilon$, every element is ϵ'-independent of its predecessors.*

It is well known [RV13] that if $f(z) = g(w^T z)$, where $z \in \mathbb{R}^d$, and g is a smooth and strongly monotone link function, then the eluder dimension of $\mathcal{F}$ is $O(d)$, where the additional constants depend on the properties of g. In particular, it is at most d for linear functions, and hence provides a strict generalization of results for linear function approximation.

Based on this measure, we now present our main results for the SPI and NPG in the following subsections. For the sake of presentation, we provide the complexity bounds for ENIAC-SPI-SAMPLE and ENIAC-NPG-SAMPLE.

5.4.1 Main Results for ENIAC-SPI

At a high-level, there are two main sources of suboptimality. First is the error in the critic fitting, which further consists of both the *estimation error* due to fitting with finite samples, as well as an *approximation error* due to approximating the Q function from a restricted function class $\mathcal{F}$. Second, we have the suboptimality of the policy in solving the induced optimistic MDPs at each step. The latter is handled using standard arguments from the policy optimization literature (e.g. [ABB19, AKL20a]), while the former necessitates certain assumptions on the representability of the class $\mathcal{F}$. To this end, we begin with a closedness

assumption on $\mathcal{F}$. For brevity, given a policy π we denote by

$$\mathcal{T}^{\pi} f(s,a) := \mathbb{E}^{\pi}[r(s,a) + \gamma f(s',a')|s,a]. \tag{5.14}$$

Assumption 5.4.1 ($\mathcal{F}$-closedness)**.** *For all $\pi \in \{\mathcal{S} \to \Delta(\mathcal{A})\}$ and $g : \mathcal{S} \times \mathcal{A} \to [0, \frac{2}{(1-\gamma)^2}]$, we have $\mathcal{T}^{\pi} g \in \mathcal{F}$.*

Assumption 5.4.1 is a policy evaluation analog of a similar assumption in [WSY20]. For linear f, the assumption always holds if the MDP is a linear MDP [JYW20b] under the same features. We also impose regularity and finite cover assumptions on $\mathcal{F}$.

Assumption 5.4.2 (Regularity)**.** *We assume that* $\max(\sup_{f \in \mathcal{F}} \|f\|_{\infty}, \frac{1}{1-\gamma}) \leq W$.

Assumption 5.4.3 (ϵ-cover)**.** *For any $\epsilon > 0$, there exists an ϵ-cover $\mathcal{C}(\mathcal{F}, \epsilon) \subseteq \mathcal{F}$ with size $|\mathcal{C}(\mathcal{F}, \epsilon)| \leq \mathcal{N}(\mathcal{F}, \epsilon)$ such that for any $f \in \mathcal{F}$, there exists $f' \in \mathcal{C}(\mathcal{F}, \epsilon)$ with $\|f - f'\|_{\infty} \leq \epsilon$.*

With the above assumptions, we have the following sample complexity result for ENIAC-SPI-SAMPLE.

Theorem 5.4.1 (Sample Complexity of ENIAC-SPI-SAMPLE)**.** *Let $\delta \in (0,1)$ and $\varepsilon \in (0, 1/(1-\gamma))$. Suppose Assumptions 5.4.1, 5.4.2, and 5.4.3 hold. With proper hyperparameters, ENIAC-SPI-SAMPLE returns a policy π satisfying $V^{\pi} \geq V^{\pi^\star} - \varepsilon$ with probability at least $1 - \delta$ after taking at most*

$$\tilde{\mathcal{O}}\Big(\frac{W^8 \cdot \big(dim_E(\mathcal{F}, \beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{F}, \epsilon'))\big)^2}{\varepsilon^8 (1-\gamma)^8}\Big)$$

samples, where $\beta = \varepsilon(1-\gamma)/2$ and $\epsilon' = \mathrm{poly}(\varepsilon, \gamma, 1/W, 1/dim_E(\mathcal{F}, \beta))$[3]*.*

One of the technical challenges of proving this theorem is to establish an eluder dimension upper bound on the sum of the error sequence. Unlike that in [RV13] and [WSY20], who apply the eluder dimension argument directly to a sequence of data points, we prove a new

[3] *The formal definition of ϵ' can be found in Theorem 5.5.1*

bound that applies to the sum of expectations over a sequence of distributions. This bound is then carefully combined with the augmented MDP argument in [AHK20] to establish our exploration guarantee. The proof details are displayed in Section 5.5. We now make a few remarks about the result.

Linear case. When $f(s,a) = u^T\phi(s,a)$ with $u, \phi(s,a) \in \mathbb{R}^d$, $\dim_E(\mathcal{F}, \beta) = O(d)$. Our result improves that of [AHK20] by using Bernstein concentration inequality to bound the generalization error. If Hoeffding inequality is used instead, our complexity will match that of [AHK20], thereby strictly generalizing their work to the non-linear setting.

Model misspecification Like the linear case, ENIAC-SPI (both SAMPLE and COMPUTE) is robust to the failure of Assumption 5.4.1. In Section 5.5.3, we provide a *bounded transfer error* assumption, similar to that of [AHK20], under which our guarantees hold up to an approximation error term. Informally, this condition demands that for any policy π_t, the best value function estimator f_t^* computed from *on-policy samples* also achieves a small approximation error for Q^{π_t} under the distribution $d_{s_0}^{\pi^*}$. A formal version is presented in Section 5.5.3.

Comparison to value-based methods. Like the comparison between LSVI-UCB and PC-PG in the linear case, our results have a poorer scaling with problem and accuracy parameters than the related work of [WSY20]. However, they are robust to a milder notion of model misspecification as stated above and readily lend themselves to practical implementations as our experiments demonstrate.

Sample complexity of ENIAC-SPI-COMPUTE. As remarked earlier, a key computational bottleneck in our approach is the need to compute the bonus while executing our policies. In Section 5.5 we analyze ENIAC-SPI-COMPUTE, which avoids this overhead and

admits a

$$\tilde{\mathcal{O}}\Big(\frac{W^{10}\cdot|\mathcal{A}|^2\cdot\big(\dim_E(\mathcal{F},\beta)\big)^2\cdot\big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^{10}(1-\gamma)^{10}}\Big)$$

sample complexity under the same assumptions. The worse sample complexity of ENIAC-SPI-COMPUTE arises from: 1. the uniform sampling over all actions instead of targeted randomization only over unknown actions for exploration; 2. α-probability uniform exploration even on known states.

5.4.2 Main Results for ENIAC-NPG

The results for ENIAC-NPG are qualitatively similar to those for ENIAC-SPI. However, there are differences in details as we fit the advantage function using the *tangent class* $\mathcal{G}_{\mathcal{F}}$ now, and this also necessitates some changes to the underlying assumptions regarding closure for Bellman operators and other regularity assumptions. We start with the former, and recall the definition of the tangent class $\mathcal{G}_{\mathcal{F}}$ in Equation (5.2). For a particular function $f\in\mathcal{F}$, we further use $\mathcal{G}_f\subseteq\mathcal{G}_{\mathcal{F}}$ to denote the subset of linear functions induced by the features $\nabla_\theta\log\pi_{f_\theta}$.

Assumption 5.4.4 ($\mathcal{G}_f$-closedness)**.** *For any $f\in\mathcal{F}$, let $\pi_f(a|s)\propto\exp(f(s,a))$. For any measurable set $\mathcal{K}\in\mathcal{S}\times\mathcal{A}$ and $g:\mathcal{S}\times\mathcal{A}\to[0,\frac{4}{(1-\gamma)^2}]$, we have $\mathcal{T}^{\pi_{f,\mathcal{K}}}g-\mathbb{E}_{\pi_{f,\mathcal{K}}}[\mathcal{T}^{\pi_{f,\mathcal{K}}}g]\in\mathcal{G}_f$, where*

$$\pi_{f,\mathcal{K}}(\cdot|s)=\begin{cases}\pi_f(\cdot|s), & \text{if for all } a\in\mathcal{A},(s,a)\in\mathcal{K}\\ \mathrm{Unif}(\{a|(s,a)\notin\mathcal{K}\}), & o.w.\end{cases}$$

and the operator $\mathcal{T}$ is defined in Equation (5.14)*.*

One may notice that the policy $\pi_{f,\mathcal{K}}$ complies with our actor update in (5.11) since $b=0$ for $s\in\mathcal{K}$. We also impose regularity and finite cover assumptions on $\mathcal{G}_{\mathcal{F}}$ as below.

Assumption 5.4.5 (Regularity)**.** *We assume that $\|u\|_2 \leq B$ for all $u \in \mathcal{U} \subset \mathbb{R}^d$, and f_θ is twice differentiable for all $f_\theta \in \mathcal{F}$, and further satisfies:*

$$\|f_\theta\|_\infty \leq W, \quad \|\nabla f_\theta\|_2 \leq G \quad \textit{and} \quad \|\nabla^2 f_\theta\|_2 \leq \Lambda.$$

We denote by $D := \max(BG, 1/(1-\gamma))$.

Assumption 5.4.6 (ϵ-cover)**.** *For the function class $\mathcal{G}_\mathcal{F}$, for any $\epsilon > 0$, there exists an ϵ-cover $\mathcal{C}(\mathcal{G}_\mathcal{F}, \epsilon) \subseteq \mathcal{G}_\mathcal{F}$ with size $|\mathcal{C}(\mathcal{G}_\mathcal{F}, \epsilon)| \leq \mathcal{N}(\mathcal{G}_\mathcal{F}, \epsilon)$ such that for any $g \in \mathcal{G}_\mathcal{F}$, there exists $g' \in \mathcal{C}(\mathcal{G}_\mathcal{F}, \epsilon)$ with $\|g - g'\|_\infty \leq \epsilon$.*

We provide the sample complexity guarantee for ENIAC-NPG-SAMPLE as below.

Theorem 5.4.2 (Sample Complexity of ENIAC-NPG-SAMPLE)**.** *Let $\delta \in (0,1)$ and $\varepsilon \in (0, 1/(1-\gamma))$. Suppose Assumptions 5.4.4, 5.4.5, and 5.4.6 hold. With proper hyperparameters, ENIAC-NPG-SAMPLE returns a policy π satisfying $V^\pi \geq V^{\pi^\star} - \varepsilon$ with probability at least $1-\delta$ after taking at most*

$$\tilde{\mathcal{O}}\Big(\frac{D^6(D^2+\Lambda B^2)\cdot\big(dim_E(\mathcal{G}_\mathcal{F},\beta)\big)^2\cdot\big(\log(\mathcal{N}(\mathcal{G}_\mathcal{F},\epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big)$$

samples, where $\beta = \varepsilon(1-\gamma)/2$ and $\epsilon' = poly(\varepsilon, \gamma, 1/D, 1/dim_E(\mathcal{G}_\mathcal{F}, \beta))$[4]*.*

Notice that the differences between Theorems 5.4.1 and 5.4.2 only arise in the function class complexity terms and the regularity parameters, where the NPG version pays the complexity of the tangent class instead of the class $\mathcal{F}$ as in the SPI case. NPG, however, offers algorithmic benefits as remarked before, and the result here extends to a more general form under a bounded transfer error condition that we present in Section 5.5.4. As with the algorithms, the theorems essentially coincide in the linear case. One interesting question for further investigation is the relationship between the eluder dimensions of the classes $\mathcal{F}$ and $\mathcal{G}_\mathcal{F}$, which might inform statistical preferences between the two approaches.

[4] *The formal definition of ϵ' can be found in Theorem 5.5.3.*

5.5 Proofs

In this section, we provide proofs to the theoretical results in Section 5.4.

5.5.1 Definition and Notation

We denote by $\mathcal{M}$ the original MDP and $\tilde{\pi}$ an arbitrary fixed comparator policy (e.g., an optimal policy). Our target is to show that after N epochs, ENIAC is able to output a policy whose value is larger than $V^{\tilde{\pi}}$ minus some problem-dependent constant. First we describe the construction of some auxiliary MDPs, which is conceptually similar to [AHK20], modulo the difference in the bonus functions.

For each epoch $n \in [N]$, we consider three MDPs: the original MDP $\mathcal{M}$, the bonus-added MDP $\mathcal{M}_{b^n} := (\mathcal{S}, \mathcal{A}, P, r + b^n, \gamma)$, and an auxiliary MDP $\mathcal{M}^n$. $\mathcal{M}^n$ is defined as $(\mathcal{S}, \mathcal{A} \cup \{a^\dagger\}, P^n, r^n, \gamma)$, where $a^\dagger$ is an extra action which is only available for $s \notin \mathcal{K}^n$ (recall that $s \in \mathcal{K}^n$ if and only if $b^n(s,a) \equiv 0$ for all $a \in \mathcal{A}$). For all $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$P^n(\cdot|s,a) = P(\cdot|s,a), \quad r^n(s,a) = r(s,a) + b^n(s,a).$$

For $s \notin \mathcal{K}^n$,

$$P^n(s|s,a^\dagger) = 1, \quad r^n(s,a^\dagger) = 1.$$

Basically, $a^\dagger$ allows the agent to stay in a state $s \notin \mathcal{K}^n$ while accumulating maximum instant rewards.

Given $\mathcal{M}^n$, we further define $\tilde{\pi}^n$ such that $\tilde{\pi}^n(\cdot|s) = \tilde{\pi}(\cdot|s)$ for $s \in \mathcal{K}^n$ and $\tilde{\pi}^n(a^\dagger|s) = 1$ for $s \notin \mathcal{K}^n$. We denote by $\tilde{d}_{\mathcal{M}^n}$ the state-action distribution induced by $\tilde{\pi}^n$ on $\mathcal{M}^n$ and $d^{\tilde{\pi}}$ the state-action distribution induced by $\tilde{\pi}$ on $\mathcal{M}$.

Additional Notations Given a policy π, we denote by $V^\pi_{b^n}, Q^\pi_{b^n}$, and $A^\pi_{b^n}$ the state-value, Q-value, and advantage function of π on $\mathcal{M}_{b^n}$ and $V^\pi_{\mathcal{M}^n}, Q^\pi_{\mathcal{M}^n}$, and $A^\pi_{\mathcal{M}^n}$ for the counterparts

on $\mathcal{M}^n$. For the policy π_t^n, i.e., the policy at the t_{th} iteration in the n_{th} epoch of ENIAC, we further simplify the notation as $V_{b^n}^t, Q_{b^n}^t$, and $A_{b^n}^t$ and also $V_{\mathcal{M}^n}^t, Q_{\mathcal{M}^n}^t$, and $A_{\mathcal{M}^n}^t$.

Remark 8. *Note that only $\tilde{\pi}^n$ can take the action $a^\dagger$ for $s \notin \mathcal{K}^n$. All policies $\{\pi_t^n\}$ is not aware of $a^\dagger$ and therefore, $V_{b^n}^t = V_{\mathcal{M}^n}^t$, $Q_{b^n}^t = Q_{\mathcal{M}^n}^t$, and $A_{b^n}^t = A_{\mathcal{M}^n}^t$.*

Based on the above definitions, we directly have the following two lemmas.

Lemma 5.5.1. *Consider any state $s \in \mathcal{K}^n$, we have:*

$$\tilde{d}_{\mathcal{M}^n}(s,a) \leq d^{\tilde{\pi}}(s,a), \quad \forall a \in \mathcal{A}.$$

Proof. The proof follows that of Lemma B.1. in [AHK20]. We present below for the readers' convenience.

We prove by induction over the time steps along the horizon. Recall $\tilde{d}_{\mathcal{M}^n}$ is the state-action distribution of $\tilde{\pi}^n$ over $\mathcal{M}^n$ and $d^{\tilde{\pi}}$ is the state-action distribution of $\tilde{\pi}$ on both $\mathcal{M}_{b^n}$ and $\mathcal{M}$ as they share the same dynamics. We use another subscript h to indicate the step index, e.g., $\tilde{d}_{\mathcal{M}^n,h}$ is the state-action distribution at the h_{th} step following $\tilde{\pi}^n$ on $\mathcal{M}^n$.

Starting at $h = 0$, if $s_0 \in \mathcal{K}^n$, then $\tilde{\pi}^n(\cdot|s_0) = \tilde{\pi}(\cdot|s_0)$ and we can easily get:

$$\tilde{d}_{\mathcal{M}^n,0}(s_0,a) = d_0^{\tilde{\pi}}(s_0,a), \quad \forall a \in \mathcal{A}.$$

Now we assume that at step h, for all $s \in \mathcal{K}^n$, it holds that

$$\tilde{d}_{\mathcal{M}^n,h}(s,a) \leq d_h^{\tilde{\pi}}(s,a), \ \forall a \in \mathcal{A}.$$

Then, for step $h+1$, by definition we have that for $s \in \mathcal{K}^n$

$$\begin{aligned}
\tilde{d}_{\mathcal{M}^n,h+1}(s) &= \sum_{s',a'} \tilde{d}_{\mathcal{M}^n,h}(s',a') P_{\mathcal{M}^n}(s|s',a') \\
&= \sum_{s',a'} \mathbf{1}\{s' \in \mathcal{K}^n\} \tilde{d}_{\mathcal{M}^n,h}(s',a') P_{\mathcal{M}^n}(s|s',a') \\
&= \sum_{s',a'} \mathbf{1}\{s' \in \mathcal{K}^n\} \tilde{d}_{\mathcal{M}^n,h}(s',a') P(s|s',a'),
\end{aligned}$$

where the second line is due to that if $s' \notin \mathcal{K}^n$, $\tilde{\pi}$ will deterministically pick $a^\dagger$ and $P_{\mathcal{M}^n}(s|s', a^\dagger) = 0$. On the other hand, for $d^{\tilde{\pi}}_{h+1}(s, a)$, it holds that for $s \in \mathcal{K}^n$,

$$\begin{aligned}
d^{\tilde{\pi}}_{h+1}(s) &= \sum_{s',a'} d^{\tilde{\pi}}_h(s', a')P(s|s', a') \\
&= \sum_{s',a'} \mathbf{1}\{s' \in \mathcal{K}^n\} d^{\tilde{\pi}}_h(s', a')P(s|s', a') + \sum_{s',a'} \mathbf{1}\{s' \notin \mathcal{K}^n\} d^{\tilde{\pi}}_h(s', a')P(s|s', a') \\
&\geq \sum_{s',a'} \mathbf{1}\{s' \in \mathcal{K}^n\} d^{\tilde{\pi}}_h(s', a')P(s|s', a') \\
&\geq \sum_{s',a'} \mathbf{1}\{s' \in \mathcal{K}^n\} \tilde{d}_{\mathcal{M}^n,h}(s', a')P(s|s', a') = \tilde{d}_{\mathcal{M}^n,h+1}(s).
\end{aligned}$$

Using the fact that $\tilde{\pi}^n(\cdot|s) = \tilde{\pi}(\cdot|s)$ for $s \in \mathcal{K}^n$, we conclude that the inductive hypothesis holds at $h + 1$ as well. Using the definition of the average state-action distribution, we conclude the proof. ■

Lemma 5.5.2. *For any epoch $n \in [N]$, we have*

$$V^{\tilde{\pi}^n}_{\mathcal{M}^n} \geq V^{\tilde{\pi}}_{\mathcal{M}}.$$

Proof. The result is straightforward since if following $\tilde{\pi}^n$ we run into some $s \notin \mathcal{K}^n$, then by definition, $\tilde{\pi}^n$ is able to collect maximum instant rewards for all steps later. ■

5.5.2 Proof Sketch

We intend to compare the values of the output policy $\pi^N_{\text{ave}} := \text{Unif}(\pi^2, \pi^3, \ldots, \pi^{N+1})$ and the comparator $\tilde{\pi}$. To achieve this, we use two intermediate quantities $V^{\pi^{n+1}}_{b^n}$ and $V^{\tilde{\pi}^n}_{\mathcal{M}^n}$ and build the following inequalities as bridges:

$$V^{\pi^N_{\text{ave}}} = \frac{1}{N}\sum_{n=1}^{N} V^{\pi^{n+1}} \geq \frac{1}{N}\sum_{n=1}^{N} V^{\pi^{n+1}}_{b^n} - A, \quad V^{\pi^{n+1}}_{b^n} = V^{\pi^{n+1}}_{\mathcal{M}^n} \geq V^{\tilde{\pi}^n}_{\mathcal{M}^n} - B, \quad V^{\tilde{\pi}^n}_{\mathcal{M}^n} \geq V^{\tilde{\pi}},$$

where A and B are two terms to be specified. If the above relations all hold, the desired result is natually induced. For these inequalities, we observe that

1. The leftmost inequality is about the value differences of a sequence of policies $(\pi^2, \pi^3, \ldots, \pi^{N+1})$ on two different reward functions (with or without the bonus). Thus, it is bounded by the cumulative bonus, or equivalently, the *expected bonus* over the state-action measure induced by these policies, which we use the eluder dimension of the approximation function class to bound. We present this result for SPI-Sample, SPI-Compute, and NPG-Sample in Lemma 5.5.3, 5.5.7, and 5.5.10, respectively.

2. The rightmost inequality is proved in Lemma 5.5.2.

3. To show the middle inequality, we analyze the convergence of actor-critic updates, leveraging properties of the multiplicative weight updates for a regret bound following the analysis of [AKL20a].

In the sequel, we present sample complexity analysis for ENIAC-SPI-SAMPLE, ENIAC-SPI-COMPUTE, and ENIAC-NPG-SAMPLE. ENIAC-NPG-COMPUTE can be easily adapted with minor changes of the assumptions. In particular, we provide general results considering model misspecification and the theorems in Section 5.4 fall as special cases under Assumption 5.4.1 or 5.4.4.

5.5.3 Proofs of ENIAC-SPI

In this part, we provide analysis for ENIAC-SPI-SAMPLE and ENIAC-SPI-COMPUTE. We start with stating the assumptions which quantifies model misspecification.

Assumption 5.5.1 (Bounded Transfer Error)**.** *Given a target function $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we define the critic loss function $L(f; d, g)$ with $d \in \Delta(\mathcal{S} \times \mathcal{A})$ as:*

$$L(f; d, g) := \mathbb{E}_{(s,a)\sim d}\left[\big(f(s,a) - g(s,a)\big)^2\right].$$

For the fixed comparator policy $\tilde{\pi}$ (defined at the beginning of Section 5.5.1), we define $\tilde{d}(s,a) := d_{s_0}^{\tilde{\pi}}(s) \circ \mathrm{Unif}(\mathcal{A})$. In ENIAC-SPI (both sample and compute versions), for every

epoch $n \in [N]$ and every iteration t inside epoch n, we assume that

$$\inf_{f \in \mathcal{F}_t^n} L(f; \tilde{d}, Q_{b^n}^t - b^n) \leq \epsilon_{bias},$$

where $\mathcal{F}_t^n := \operatorname{argmin}_{f \in \mathcal{F}} L(f; \rho_{cov}^n, Q_{b^n}^t - b^n)$ and $\epsilon_{bias} \geq 0$ is some problem-dependent constant.

ϵ_{bias} measures both approximation error and distribution shift error. In later proof, we select a particular function in $\tilde{f}_t^n \in \mathcal{F}_t^n$ such that

$$L(\tilde{f}_t^n; \tilde{d}, Q_{b^n}^t - b^n) \leq 2\epsilon_{\text{bias}}. \tag{5.15}$$

We establish complexity results by comparing the empirical minimizer f_t^n of (5.6) with this optimal fitter $\tilde{f}_t^n$.

Assumption 5.5.2. *For the same loss L as defined in Assumption 5.5.1 and the fitter $\tilde{f}_t^n$, we assume that there exists some $C \geq 1$ and $\epsilon_0 \geq 0$ such that for any $f \in \mathcal{F}$,*

$$\mathbb{E}_{(s,a)\sim\rho_{cov}^n}\left[\left(f(s,a) - \tilde{f}_t^n(s,a)\right)^2\right] \leq C \cdot \left(L(f; \rho_{cov}^n, Q_{b^n}^t - b^n) - L(\tilde{f}_t^n; \rho_{cov}^n, Q_{b^n}^t - b^n)\right) + \epsilon_0$$

for $n \in [N]$ and $0 \leq t \leq T-1$.

Remark 9. *Under Assumption 5.4.1, $Q_{b^n}^t - b^n = \mathbb{E}^{\pi_t^n}[r(s,a) + \gamma Q_{b^n}^t(s',a')] \in \mathcal{F}$. Thus, ϵ_{bias} can take value 0 and $\tilde{f}_t^n = Q_{b^n}^t - b^n$. Further in Assumption 5.5.2, we have*

$$\mathbb{E}_{(s,a)\sim\rho_{cov}^n}\left[\left(f(s,a) - \tilde{f}_t^n(s,a)\right)^2\right] = L(f; \rho_{cov}^n, Q_{b^n}^t - b^n).$$

Thus, C can take value 1 and $\epsilon_0 = 0$. If $Q_{b^n}^t - b^n$ is not realizable in $\mathcal{F}$, ϵ_{bias} and ϵ_0 could be strictly positive. Hence, the above two assumptions are generalized version of the closedness condition considering model misspecification.

Next we prove the sample complexity of ENIAC-SPI-SAMPLE. We follow the proof steps in Section 5.5.2 and first establish a bonus bound.

Lemma 5.5.3 (SPI-SAMPLE: The Bound of Bonus)**.** *With probability at least $1 - N\delta$, it holds that*

$$\sum_{n=1}^{N}\left(V_{b^n}^{\pi^{n+1}} - V^{\pi^{n+1}}\right) \leq \frac{2\epsilon^2 + 8KW^2 + \beta^2}{(1-\gamma)\beta^2 K} \cdot dim_E(\mathcal{F}, \beta) + \frac{N}{1-\gamma}\sqrt{\frac{\log(2/\delta)}{2K}}.$$

Proof.

$$\sum_{n=1}^{N}\left(V_{b^n}^{\pi^{n+1}}-V^{\pi^{n+1}}\right)\le\sum_{n=1}^{N}\mathbb{E}_{(s,a)\sim d^{n+1}}\mathbf{1}\{(s,a)\notin\mathcal{K}^n\}/(1-\gamma)$$
$$=\sum_{n=1}^{N}\mathbb{E}_{(s,a)\sim d^{n+1}}\mathbf{1}\{w(\tilde{\mathcal{F}}^n,s,a)\ge\beta\}/(1-\gamma),$$

where d^{n+1} denotes the state-action distribution induced by π^{n+1} on $\mathcal{M}$. We denote by $\mathcal{D}^n$ the sampled dataset $\{(s_i,a_i)\}_{i=1}^K\sim d^n$ at the beginning of epoch n. Then $\mathcal{Z}^n=\mathcal{Z}^{n-1}\cup\mathcal{D}^n$. By Hoeffding's inequality, with probability at least $1-\delta$,

$$\mathbb{E}_{(s,a)\sim d^{n+1}}\mathbf{1}\{w(\tilde{\mathcal{F}}^n,s,a)\ge\beta\}\le\frac{1}{K}\sum_{(s,a)\in\mathcal{D}^{n+1}}\mathbf{1}\{w(\tilde{\mathcal{F}}^n,s,a)\ge\beta\}+\sqrt{\frac{\log(2/\delta)}{2K}}.$$

Taking the union bound, with probability at least $1-N\delta$, we have

$$\sum_{n=1}^{N}V_{b^n}^{\pi^{n+1}}-V^{\pi^{n+1}}\le\frac{1}{K(1-\gamma)}\sum_{n=1}^{N}\sum_{(s,a)\in\mathcal{D}^{n+1}}\mathbf{1}\{w(\tilde{\mathcal{F}}^n,s,a)\ge\beta\}+\frac{N}{1-\gamma}\sqrt{\frac{\log(2/\delta)}{2K}}\quad(5.16)$$

Next we bound the first term in Equation (5.16) following a similar process as in [RV13, Proposition 3]. We simplify $w(\tilde{\mathcal{F}}^n,\cdot,\cdot)$ as $w^n(\cdot,\cdot)$ and label all samples in $\mathcal{Z}^n$ in lexical order, e.g., (s_i^{n+1},a_i^{n+1}) denotes the ith sample in $\mathcal{D}^{n+1}$. For every (s_i^{n+1},a_i^{n+1}), we define a sequence S_{i-1}^{n+1} which contains all samples generated before (s_i^{n+1},a_i^{n+1}), i.e.,

$$S_{i-1}^{n+1}:=\big((s_1^1,a_1^1),\dots,(s_K^1,a_K^1),(s_1^2,a_1^2),\cdots(s_K^n,a_K^n),(s_1^{n+1},a_1^{n+1}),\dots,(s_{i-1}^{n+1},a_{i-1}^{n+1})\big)\quad(5.17)$$

Next we show that,

$$\sum_{n=1}^{N}\sum_{(s,a)\in\mathcal{D}^{n+1}}\mathbf{1}\{w^n(s,a)\ge\beta\}\le\left(2\epsilon^2/\beta^2+8W^2K/\beta^2+1\right)\cdot\dim_E(\mathcal{F},\beta).\quad(5.18)$$

For $n\le N$, if $w^n(s_i^{n+1},a_i^{n+1})>\beta$ then (s_i^{n+1},a_i^{n+1}) is β-dependent with respect to $\mathcal{F}$ on fewer than $8(\epsilon)^2/\beta^2+32W^2K/\beta^2$ disjoint subsequences of S_{i-1}^{n+1}. To see this, note that if $w^n(s_i^{n+1},a_i^{n+1})>\beta$, there exists $\bar{f},\underline{f}\in\mathcal{F}$ such that $\bar{f}-\underline{f}\in\tilde{\mathcal{F}}^n$ and $\bar{f}(s_i^{n+1},a_i^{n+1})-\underline{f}(s_i^{n+1},a_i^{n+1})\ge\beta$. By definition, if (s_i^{n+1},a_i^{n+1}) is β-dependent on a subsequence $\big((s_{t_1},a_{t_1}),\dots,(s_{t_k},a_{t_k})\big)$

of S_{i-1}^{n+1}, then $\sum_{j=1}^{k}(\bar{f}(s_{t_j},a_{t_j})-\underline{f}(s_{t_j},a_{t_j}))^2 \geq \beta^2$. It follows that, if (s_i^{n+1},a_i^{n+1}) is β-dependent on L disjoint subsequences of S_{i-1}^{n+1} then $\|\bar{f}-\underline{f}\|_{S_{i-1}^{n+1}}^2 \geq L\beta^2$, where we recall our notation $\|f\|_S=\sqrt{\sum_{x\in S}f(x)^2}$. By the definition of $\tilde{\mathcal{F}}^n$ and $S_{i-1}^{n+1}=\mathcal{Z}^n\cup\{(s_j^{n+1},a_j^{n+1})\}_{j=1}^{i-1}$, we have

$$\|\bar{f}-\underline{f}\|_{S_{i-1}^{n+1}} \leq \|\bar{f}-\underline{f}\|_{\mathcal{Z}^n}+\|\bar{f}-\underline{f}\|_{\{(s_j^{n+1},a_j^{n+1})\}_{j=1}^{i-1}} \leq \epsilon+2W\sqrt{i-1} \leq \epsilon+2W\sqrt{K},$$

where W is an upper bound of $\|f\|_\infty$. Hence, $L<2\epsilon^2/\beta^2+8W^2K/\beta^2$.

Next, we show that in any state-action sequence $((s_1,a_1),\ldots,(s_\tau,a_\tau))$, there is some $j\leq\tau$ such that the element (s_j,a_j) is β-dependent with respect to $\mathcal{F}$ on at least $\tau/d-1$ disjoint subsequences of the subset $((s_1,a_1),\ldots,(s_{j-1},a_{j-1}))$, where $d:=\dim_E(\mathcal{F},\beta)$. Here we assume that $\tau\geq d$ since otherwise the claim is trivially true. To see this, for an integer L safistying $Ld+1\leq\tau\leq(L+1)\cdot d$, we will construct L disjoint subsequences $S_1,\ldots,S_L$ *one element at a time*. First, for each $i\in[L]$ add (s_i,a_i) to the subsequence S_i. Now, if (s_{L+1},a_{L+1}) is β-dependent on all subsequences $S_1,\ldots,S_L$, our claim is established. Otherwise, select a subsequence S_i such that (s_{L+1},a_{L+1}) is β-independent of it and append (s_{L+1},a_{L+1}) to S_i. Repeat this process for elements with indices $j>L+1$ until (s_j,a_j) is β-dependent on all subsequences or $j=\tau$. In the latter scenario, since $\tau-1$ elements have already been put in subsequences, we have that $\sum|S_j|\geq L\cdot d$. However, by the definition of $\dim_E(\mathcal{F},\beta)$, since each element of a subsequence S_j is β-independent of its predecessors, we must have $|S_j|\leq d,\forall j\in[L]$ and therefore, $\sum|S_j|\leq L\cdot d$. In this case, (s_τ,a_τ) must be β-dependent on all subsequences.

Now consider the subsequence $S_\beta:=\big((s_{i_1}^{n_1},a_{i_1}^{n_1}),\ldots,(s_{i_\tau}^{n_\tau},a_{i_\tau}^{n_\tau})\big)$ of S_K^{N+1} which consists of all elements such that $w_n\big((s_i^{n+1},a_i^{n+1})\big)\geq\beta$. With that being said, S_β consists of all sample points where large width occurs from epoch 1 to epoch N. The indices in S_β are in lexical order and $(s_{i_j}^{n_j},a_{i_j}^{n_j})$ denotes the j_{th} element in S_β. As we have established, each $(s_{i_j}^{n_j},a_{i_j}^{n_j})$ is β-dependent on fewer than $2\epsilon^2/\beta^2+8W^2K/\beta^2$ disjoint subsequences of $S_{i_j-1}^{n_j}$ (recall the definition in Equation (5.17)). It follows that each $(s_{i_j}^{n_j},a_{i_j}^{n_j})$ is β-dependent on

fewer than $2\epsilon^2/\beta^2 + 8W^2K/\beta^2$ disjoint subsequences of $((s_{i_1}^{n_1}, a_i^{n_1}), \ldots, (s_{i_{j-1}}^{n_{j-1}}, a_{i_{j-1}}^{n_{j-1}})) \subset S_\beta$, i.e., the elements in S_β before $(s_{i_j}^{n_j}, a_{i_j}^{n_j})$. Combining this with the fact we have established that there exists some $(s_{i_j}^{n_j}, a_{i_j}^{n_j})$ that is β-dependent on at least $\tau/d - 1$ disjoint subsequences of $((s_{i_1}^{n_1}, a_i^{n_1}), \ldots, (s_{i_{j-1}}^{n_{j-1}}, a_{i_{j-1}}^{n_{j-1}}))$, we have $\tau/d - 1 \leq 2\epsilon^2/\beta^2 + 8W^2K/\beta^2$. It follows that $\tau \leq \left(2\epsilon^2/\beta^2 + 8W^2K/\beta^2 + 1\right) \cdot d$, which is Equation (5.18).

Combining all above results, with probability at least $1 - N\delta$,

$$\sum_{n=1}^{N} \left(V_{b^n}^{\pi^{n+1}} - V^{\pi^{n+1}}\right) \leq \frac{2\epsilon^2 + 8KW^2 + \beta^2}{(1-\gamma)\beta^2 K} \cdot \dim_E(\mathcal{F}, \beta) + \frac{N}{1-\gamma}\sqrt{\frac{\log(2/\delta)}{2K}}.$$

■

Next we prove the last step in Section 5.5.2. For notation brevity, we focus on a specific epoch n and drop the dependence on n in the policy and critic functions. We define

$$\widehat{A}_{b^n}^t(s,a) := f_t(s,a) + b^n(s,a) - \mathbb{E}_{a' \sim \pi_t(\cdot|s)}[f_t(s,a') + b^n(s,a')], \tag{5.19}$$

where f_t is the output of the critic fit step at iteration t in epoch n. It can be easily verified that $\mathbb{E}_{a\sim\pi_t(\cdot|s)}\widehat{A}_{b^n}^t(s,a) = 0$ and the SPI-SAMPLE update in Equation (5.7) is equivalent to

$$\pi_{t+1}(\cdot|s) \propto \pi_t(\cdot|s) \exp\left(\eta \widehat{A}_{b^n}^t(s,\cdot)\mathbf{1}\{s \in \mathcal{K}^n\}\right), \quad \forall s \in \mathcal{S}. \tag{5.20}$$

$\widehat{A}_{b^n}^t$ is indeed our approximation to the true advantage function $A_{b^n}^t$. In the sequel, we show that the actor-critic convergence is upper bounded by the approximation error which can further be controlled with sufficient samples under our assumptions.

Lemma 5.5.4 (SPI-SAMPLE: Actor-Critic Convergence)**.** *In ENIAC-SPI-SAMPLE, let $\widehat{A}_{b^n}^t$ be as defined in Equation* (5.19) *and the step size $\eta = \sqrt{\frac{\log(|\mathcal{A}|)}{16W^2T}}$. For any epoch $n \in [N]$, SPI-SAMPLE obtains a sequence of policies $\{\pi_t\}_{t=0}^{T-1}$ such that when comparing to $\tilde{\pi}^n$:*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}(V_{\mathcal{M}^n}^{\tilde{\pi}^n} - V_{b^n}^t) &= \frac{1}{T}\sum_{t=0}^{T-1}(V_{\mathcal{M}^n}^{\tilde{\pi}^n} - V_{\mathcal{M}^n}^t) \\ &\leq \frac{1}{1-\gamma}\Big(8W\sqrt{\frac{\log(|\mathcal{A}|)}{T}} + \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[\left(A_{b^n}^t(s,a) - \widehat{A}_{b^n}^t(s,a)\right)\mathbf{1}\{s \in \mathcal{K}^n\}\right]\Big).\end{aligned}$$

Proof. The equality is mentioned in Remark 8. We first show that $A^t_{\mathcal{M}^n}(s, a^\dagger) \le 0$ for any $s \notin \mathcal{K}^n$. Since π_t uniformly randomly selects an unfamiliar action with bonus $1/(1-\gamma)$ for $s \notin \mathcal{K}^n$, we have $V^t_{\mathcal{M}^n}(s) \ge 1/(1-\gamma)$. Thus,

$$A^t_{\mathcal{M}^n}(s, a^\dagger) = Q^t_{\mathcal{M}^n}(s, a^\dagger) - V^t_{\mathcal{M}^n}(s) = 1 - (1-\gamma) \cdot V^t_{\mathcal{M}^n}(s) \le 0, \quad \forall s \notin \mathcal{K}^n,$$

where $Q^t_{\mathcal{M}^n}(s, a^\dagger) = 1 + \gamma V^t_{\mathcal{M}^n}(s)$ ($a^\dagger$ leads s to s). Based on the above result, we have

$$\begin{aligned}
V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n} &= \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{\mathcal{M}^n}(s, a) \\
&= \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{\mathcal{M}^n}(s, a) \mathbf{1}\{s \in \mathcal{K}^n\} + \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{\mathcal{M}^n}(s, a) \mathbf{1}\{s \notin \mathcal{K}^n\} \\
&= \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{\mathcal{M}^n}(s, a) \mathbf{1}\{s \in \mathcal{K}^n\} + \frac{1}{1-\gamma} \sum_{s} \tilde{d}_{\mathcal{M}^n}(s) A^t_{\mathcal{M}^n}(s, a^\dagger) \mathbf{1}\{s \notin \mathcal{K}^n\} \\
&\le \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{\mathcal{M}^n}(s, a) \mathbf{1}\{s \in \mathcal{K}^n\} \\
&= \frac{1}{1-\gamma} \sum_{(s,a)} \tilde{d}_{\mathcal{M}^n}(s, a) A^t_{b^n}(s, a) \mathbf{1}\{s \in \mathcal{K}^n\} \\
&= \frac{1}{1-\gamma} \left(\mathbb{E}_{(s,a)\sim \tilde{d}_{\mathcal{M}^n}} \left[\widehat{A}^t_{b^n}(s, a) \mathbf{1}\{s \in \mathcal{K}^n\} \right] + \mathbb{E}_{(s,a)\sim \tilde{d}_{\mathcal{M}^n}} \left[(A^t_{b^n}(s, a) - \widehat{A}^t_{b^n}(s, a)) \mathbf{1}\{s \in \mathcal{K}^n\} \right] \right)
\end{aligned} \tag{5.21}$$

where the first line is by the performance difference lemma in [Kak03], the third line is due to that $\tilde{\pi}^n$ deterministically picks $a^\dagger$ for $s \notin \mathcal{K}^n$, and the fifth line follows that π_t never picks $a^\dagger$ so for any action $a \in \mathcal{A}$ we have $A^t_{\mathcal{M}^n} = A^t_{b^n}$.

Next we establish an upper bound of the first term in Equation (5.21). Recall that in SPI-SAMPLE the policy update is equivalent to (5.20). Thus, for $s \in \mathcal{K}^n$, we have

$$\mathbf{KL}\big(\tilde{\pi}^n(\cdot|s), \pi_{t+1}(\cdot|s)\big) - \mathbf{KL}\big(\tilde{\pi}^n(\cdot|s), \pi_t(\cdot|s)\big) = \mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[-\eta \widehat{A}^t_{b^n}(s, a) + \log(z^t(s))],$$

where $z^t(s) := \sum_a \pi_t(a|s) \exp(\eta \widehat{A}^t_{b^n}(s, a))$. Since $|\widehat{A}^t_{b^n}(s, a)| \le 4W$ and when $T > \log(|\mathcal{A}|)$, $\eta < 1/(4W)$, we have $\eta \widehat{A}^t_{b^n}(s, a) \le 1$. By the inequality that $\exp(x) \le 1 + x + x^2$ for $x \le 1$

and $\log(1+x) \le x$ for $x > -1$,

$$\log(z^t(s)) \le \eta \sum_a \pi_t(a|s) \widehat{A}^t_{b^n}(s,a) + 16\eta^2 W^2 = 16\eta^2 W^2.$$

Hence, for $s \in \mathcal{K}^n$,

$$\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_{t+1}(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_t(\cdot|s)) \le -\eta \mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\widehat{A}^t_{b^n}(s,a)] + 16\eta^2 W^2.$$

Adding both sides from $t = 0$ to $T-1$ and taking $\eta = \sqrt{\frac{\log(|\mathcal{A}|)}{16W^2T}}$, we get

$$\begin{aligned}
&\sum_{t=0}^{T-1} \mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}[\widehat{A}^t_{b^n}(s,a)\mathbf{1}\{s \in \mathcal{K}^n\}] \\
=&\sum_{t=0}^{T-1} \frac{1}{\eta} \mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n}} \left[\Big(\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_0(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_T(\cdot|s)) \Big) \mathbf{1}\{s \in \mathcal{K}^n\} \right] + 16\eta T W^2 \\
\le& \log(|\mathcal{A}|)/\eta + 16\eta T W^2 \le 8W\sqrt{\log(|\mathcal{A}|)T},
\end{aligned}$$

where the inequality follows that $\pi_0(\cdot|s) = \text{Unif}(\mathcal{A})$. Lastly, combining with Equation (5.21), the regret on $\mathcal{M}^n$ satisfies

$$\sum_{t=0}^{T-1} (V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n}) \le \frac{1}{1-\gamma} \left(8W\sqrt{\log(|\mathcal{A}|)T} + \sum_{t=1}^{T} \mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}} \left[\big(A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a) \big) \mathbf{1}\{s \in \mathcal{K}^n\} \right] \right).$$

■

Next, we analyze the approximation error and build an upper bound on $A^t_{b^n} - \widehat{A}^t_{b^n}$. Recall that $A^t_{b^n}$ is the true advantage of policy π^n_t in the bonus-added MDP and $\widehat{A}^t_{b^n}$ is an approximation to $A^t_{b^n}$ with the empirical minimizer f_t as defined in (5.19). We still focus on a specific epoch n and simplify the notation $\tilde{f}^n_t$ as defined in (5.15) to f^*_t.

Lemma 5.5.5 (SPI-SAMPLE: Approximation Bound)**.** *At epoch n, assume for all $0 \le t \le T-1$:*

$$L(f_t; \rho^n_{cov}, Q^t_{b^n} - b^n) \le L(f^*_t; \rho^n_{cov}, Q^t_{b^n} - b^n) + \epsilon_{stat}, \tag{5.22}$$

where $\epsilon_{stat} > 0$ is to be determined in the next lemma, and let

$$\epsilon^2 = NK\big(C \cdot \epsilon_{stat} + \epsilon_0 + 16W\epsilon_1\big) + 8W^2 \log(\mathcal{N}(\mathcal{F}, \epsilon_1)/\delta) \cdot \sqrt{NK}, \tag{5.23}$$

where ϵ is used in bonus function (see Section 5.3.3) and C, ϵ_0 are defined in Assumption 5.5.2, and $\epsilon_1 > 0$ denotes the function cover radius which will be determined later. Under Assumption 5.5.1 and 5.5.2, we have that for every $0 \le t \le T-1$, with probability at least $1-\delta$,

$$\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big(A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a)\Big)\mathbf{1}\{s \in \mathcal{K}^n\} \le 4\sqrt{|\mathcal{A}|\epsilon_{bias}} + 2\beta.$$

Proof. To analyze the difference between $A^t_{b^n}$ and $\widehat{A}^t_{b^n}$, we introduce an intermediate variable $A^*_t(s,a) := f^*_t + b^n - \mathbb{E}_{a'\sim\pi_t(\cdot|s)}[f^*_t + b^n]$, i.e., the approximated advantage generated by the selected best on-policy fit. Then

$$\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}(A^t_{b^n} - \widehat{A}^t_{b^n})\mathbf{1}\{s \in \mathcal{K}^n\} = \mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[(A^t_{b^n} - A^*_t)\mathbf{1}\{s \in \mathcal{K}^n\} + (A^*_t - \widehat{A}^t_{b^n})\mathbf{1}\{s \in \mathcal{K}^n\}\right].$$

For the first difference, we have

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big(A^t_{b^n} - A^*_t\Big)\mathbf{1}\{s \in \mathcal{K}^n\} \\
&= \mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big(Q^t_{b^n} - f^*_t - b^n\Big)\mathbf{1}\{s \in \mathcal{K}^n\} - \mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n}, a\sim\pi_t(\cdot|s)}(Q^t_{b^n} - f^*_t - b^n)\mathbf{1}\{s \in \mathcal{K}^n\} \\
&\le \sqrt{\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}(Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} + \sqrt{\mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n}, a\sim\pi_t(\cdot|s)}(Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} \\
&\le \sqrt{\mathbb{E}_{(s,a)\sim d^{\tilde{\pi}}}(Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} + \sqrt{\mathbb{E}_{s\sim d^{\tilde{\pi}}, a\sim\pi_t(\cdot|s)}(Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} \\
&= \sqrt{\mathbb{E}_{(s,a)\sim\tilde{d}} |\mathcal{A}|\tilde{\pi}(a|s) \cdot (Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} + \sqrt{\mathbb{E}_{(s,a)\sim\tilde{d}} |\mathcal{A}|\pi_t(a|s) \cdot (Q^t_{b^n} - f^*_t - b^n)^2\mathbf{1}\{s \in \mathcal{K}^n\}} \\
&< 4\sqrt{|\mathcal{A}|\epsilon_{\text{bias}}},
\end{aligned}$$

where the first inequality is by Cauchy-Schwarz, the second inequality is by Lemma 5.5.1, and the last two lines follow Assumption 5.5.1 and the definition of f^*_t.

For the second difference,

$$\begin{aligned}&\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}(A_t^* - \widehat{A}_{b^n}^t)\mathbf{1}\{s\in\mathcal{K}^n\}\\ =&\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}(f_t^* - f_t)\mathbf{1}\{s\in\mathcal{K}^n\} - \mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n},a\sim\pi_t(\cdot|s)}(f_t^* - f_t)\mathbf{1}\{s\in\mathcal{K}^n\} \qquad (5.24)\end{aligned}$$

Next we show that $\Delta f_t := (f_t^* - f_t) \in \tilde{\mathcal{F}}^n$. Recall that $\tilde{\mathcal{F}}^n := \{\Delta f \in \Delta\mathcal{F} \mid \|\Delta f\|_{\mathcal{Z}^n} \le \epsilon\}$. We only need to show that $\|\Delta f_t\|_{\mathcal{Z}^n} \le \epsilon$. To achieve this, we plan to utilize the fact that f_t is trained with samples generated from $\rho_{\text{cov}}^n := \text{Unif}(d_{s_0}^{\pi^1}, d_{s_0}^{\pi^2}, \ldots, d_{s_0}^{\pi^n})$ while $\mathcal{Z}^n$ is sequentially constructed with samples from $d_{s_0}^{\pi^i}, i \in [n]$. However, such a correlation does not guarantee a trivial concentration bound. We need to deal with the subtle randomness dependency therein: 1. π^i depends on $\pi^{[i-1]}$ thus the samples in $\mathcal{Z}^n$ are not independent; 2. $\mathcal{Z}^n$ determines $\tilde{\mathcal{F}}^n$, $\tilde{\mathcal{F}}^n$ defines the bonus b^n, and Δf_t is obtained based on b^n. So Δf_t and $\mathcal{Z}^n$ are not independent. Nevertheless, we carefully leverage function cover on $\Delta\mathcal{F}$ to establish a martingale convergence on every anchor function in the cover set, then transform to a bound on the realization Δf_t.

Let $\mathcal{C}(\Delta\mathcal{F}, 2\epsilon_1)$ be a cover set of $\Delta\mathcal{F}$. Then for every $\Delta f \in \Delta\mathcal{F}$, there exists a $\Delta g \in \mathcal{C}(\Delta\mathcal{F}, 2\epsilon_1)$ such that $\|\Delta f - \Delta g\|_\infty \le 2\epsilon_1$. We rank the samples in $\mathcal{Z}^n$ in lexical order, i.e., (s_k^i, a_k^i) is the k_{th} sample generated following $d_{s_0}^{\pi^i}$ at the beginning of the i_{th} epoch. There are in total nK samples in $\mathcal{Z}^n$. For every $\Delta g \in \mathcal{C}(\Delta\mathcal{F}, 2\epsilon_1)$, we define nK corresponding random variables:

$$X_{(i,k)}^{\Delta g} := (\Delta g(s_k^i, a_k^i))^2 - \mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}[(\Delta g(s,a))^2], \quad i \in [n], k \in [K]$$

We rank $\{X_{(i,k)}^{\Delta g}\}$ in lexical order and upon which, we define a martingale:

$$Y_{0,0}^{\Delta g} = 0, \quad Y_{(i,k)}^{\Delta g} = \sum_{(i',k')=(1,1)}^{(i,k)} X_{(i',k')}^{\Delta g}, \quad i \in [n], k \in [K].$$

Then by single-sided Azuma-Hoeffding's inequality, with probability at least $1-\delta$, for all $\Delta g \in \mathcal{C}(\Delta\mathcal{F}, 2\epsilon_1)$, it holds that

$$Y_{(n,K)}^{\Delta g} \le \sqrt{32W^4 \cdot nK \cdot \log\Big(\frac{\mathcal{N}(\Delta\mathcal{F}, 2\epsilon_1)}{\delta}\Big)} \le \sqrt{64W^4 \cdot nK \cdot \log\Big(\frac{\mathcal{N}(\mathcal{F}, \epsilon_1)}{\delta}\Big)}, \quad (5.25)$$

where the right inequality is by Lemma 5.5.14. Next, we transform to Δf_t. Since there exists a $\Delta g \in \mathcal{C}(\Delta\mathcal{F}, 2\epsilon_1)$ such that $\|\Delta f_t - \Delta g\|_\infty \le 2\epsilon_1$, we have that for all $i \in [n]$ and $k \in [K]$,

$$\begin{aligned}
&\left|(\Delta f_t(s_k^i, a_k^i))^2 - (\Delta g(s_k^i, a_k^i))^2\right| \\
&= |\Delta f_t(s_k^i, a_k^i) - \Delta g(s_k^i, a_k^i)| \cdot |\Delta f_t(s_k^i, a_k^i) + \Delta g(s_k^i, a_k^i))| \le 8W\epsilon_1
\end{aligned}$$

and

$$\begin{aligned}
&\left|\mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}[(\Delta f_t(s,a))^2] - \mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}[(\Delta g(s,a))^2]\right| \\
&\le \mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}|\Delta f_t(s,a) - \Delta g(s,a)| \cdot |\Delta f_t(s,a) + \Delta g(s,a)| \le 8W\epsilon_1
\end{aligned}$$

Therefore,

$$\begin{aligned}
Y_{(n,K)}^{\Delta f_t} &= \sum_{(i,k)=(1,1)}^{(n,K)} (\Delta f_t(s_k^i, a_k^i))^2 - \mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}[(\Delta f_t(s,a))^2] \qquad (5.26)\\
&\le \sum_{(i,k)=(1,1)}^{(n,K)} (\Delta g(s_k^i, a_k^i))^2 - \mathbb{E}_{(s,a)\sim d_{s_0}^{\pi^i}}[(\Delta g(s,a))^2] + nK \cdot 16W\epsilon_1 \\
&= Y_{(n,K)}^{\Delta g} + nK \cdot 16W\epsilon_1.
\end{aligned}$$

Note that

$$Y_{(n,K)}^{\Delta f_t} = \|\Delta f_t\|_{\mathcal{Z}^n}^2 - \sum_{i=1}^{n} K \cdot \mathbb{E}_{d_{s_0}^{\pi^i}}[(\Delta f_t)^2] = \|\Delta f_t\|_{\mathcal{Z}^n}^2 - nK \cdot \mathbb{E}_{\rho_{\mathrm{cov}}^n}[(\Delta f_t)^2]. \qquad (5.27)$$

Combining (5.25), (5.26), and (5.27), we have that

$$\|\Delta f_t\|_{\mathcal{Z}^n}^2 \le nK \cdot \mathbb{E}_{\rho_{\mathrm{cov}}^n}[(\Delta f_t)^2] + nK \cdot 16W\epsilon_1 + \sqrt{64W^4 \cdot nK \cdot \log\Big(\frac{\mathcal{N}(\mathcal{F}, \epsilon_1)}{\delta}\Big)}.$$

By Assumption 5.5.2,

$$\mathbb{E}_{\rho_{\mathrm{cov}}^n}[(\Delta f_t)^2] = \mathbb{E}_{(s,a)\sim\rho_{\mathrm{cov}}^n}[(f_t^* - f_t)^2] \le C \cdot (L(f_t; \rho_{\mathrm{cov}}^n, Q_{b^n}^t - b^n) - L(f_t^*; \rho_{\mathrm{cov}}^n, Q_{b^n}^t - b^n)) + \epsilon_0$$

$$\le C \cdot \epsilon_{\mathrm{stat}} + \epsilon_0.$$

By the choice of ϵ, $\|\Delta f_t\|_{\mathcal{Z}^n}^2 \leq \epsilon^2$ with probability at least $1-\delta$. Thus, $\Delta f_t \in \tilde{\mathcal{F}}^n$ and for all $(s,a) \in \mathcal{K}^n$, $|f_t^*(s,a) - f_t(s,a)| \leq \beta$. Plugging into (5.24), we have (5.24) $\leq 2\beta$. The desired result is obtained. ■

Next, we give an explicit form of ϵ_{stat} as defined in Equation (5.22).

Lemma 5.5.6. *Following the same notation as in Lemma 5.5.5, it holds with probability at least $1-\delta$ that*

$$L(f_t; \rho_{cov}^n, Q_{b^n}^t - b^n) - L(f_t^*; \rho_{cov}^n, Q_{b^n}^t - b^n) \leq \frac{500C \cdot W^4 \cdot \log\left(\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}\right)}{M} + 13W^2 \cdot \epsilon_2 + \epsilon_0,$$

where C, ϵ_0 are defined in Assumption 5.5.2, and $\epsilon_2 > 0$ denotes the function cover radius which will be determined later.

Proof. First note that in the loss function, the expectation has a nested structure: the outer expectation is taken over $(s,a) \sim \rho_{\text{cov}}^n$ and the inner conditional expectation is $Q_{b^n}^t(s,a) = \mathbb{E}^{\pi_t}[\sum_{h=0}^{\infty} \gamma^h \big(r(s_h,a_h) + b^n(s_h,a_h)\big) | (s_0,a_0) = (s,a)]$ given a sample of $(s,a) \sim \rho_{\text{cov}}^n$. To simplify the notation, we use x to denote (s,a), $y|x$ for an unbiased sample of $Q_{b^n}^t(s,a) - b^n(s,a)$, and ν for ρ_{cov}^n, the marginal distribution over x, then the loss function can be recast as

$$\mathbb{E}_{x\sim\nu}[(f_t(x) - \mathbb{E}[y|x])^2] := L(f_t; \rho_{\text{cov}}^n, Q_{b^n}^t - b^n)$$
$$\mathbb{E}_{x\sim\nu}[(f_t^*(x) - \mathbb{E}[y|x])^2] := L(f_t^*; \rho_{\text{cov}}^n, Q_{b^n}^t - b^n).$$

In particular, f_t can be rewritten as

$$f_t \in \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{i=1}^{M} (f(x_i) - y_i)^2,$$

where (x_i, y_i) are drawn i.i.d.: x_i is generated following the marginal distribution ν and y_i is generated conditioned on x_i. For any function f, we have:

$$\begin{aligned}
& \mathbb{E}_{x,y}[(f_t(x) - y)^2] \\
= \; & \mathbb{E}_{x,y}[(f_t(x) - \mathbb{E}[y|x])^2] + \mathbb{E}_{x,y}[(\mathbb{E}[y|x] - y)^2] + 2\mathbb{E}_{x,y}[(f_t(x) - \mathbb{E}[y|x])(\mathbb{E}[y|x] - y)] \\
= \; & \mathbb{E}_{x,y}[(f_t(x) - \mathbb{E}[y|x])^2] + \mathbb{E}_{x,y}[(\mathbb{E}[y|x] - y)^2],
\end{aligned}$$

where the last step follows from the cross term being zero. Thus we can rewrite the generalization error as

$$\begin{aligned} & \mathbb{E}_x[(f_t(x) - \mathbb{E}[y|x])^2] - \mathbb{E}_x[(f_t^*(x) - \mathbb{E}[y|x])^2] \\ = \; & \mathbb{E}_{x,y}(f_t(x) - y)^2 - \mathbb{E}_{x,y}(f_t^*(x) - y)^2. \end{aligned} \tag{5.28}$$

Next, we establish a concentration bound on f_t. Since f_t depends on the training set $\{(x_i, y_i)\}_{i=1}^M$, as in Assumption 5.5.5, we use a function cover on $\mathcal{F}$ for a uniform convergence argument. We denote by $\mathscr{F}_t^n$ the σ-algebra generated by randomness before epoch n iteration t. Recall that $f_t^* \in \operatorname{argmin}_{f\in\mathcal{F}} L(f; \rho_{\text{cov}}^n, Q_{b^n}^t - b^n)$. Conditioning on $\mathscr{F}_t^n$, ρ_{cov}^n, $Q_{b^n}^t - b^n$, and f_t^* are all deterministic. For any $f \in \mathcal{F}$, we define

$$Z_i(f) := (f(x_i) - y_i)^2 - (f_t^*(x_i) - y_i)^2, \quad i \in [M]$$

Then $Z_1(f), \ldots, Z_M(f)$ are i.i.d. random variables and

$$\begin{aligned} \mathbb{V}[Z_i(f) \mid \mathscr{F}_t^n] &\le \mathbb{E}[Z_i(f)^2 \mid \mathscr{F}_t^n] \\ &= \mathbb{E}\left[\Big((f(x_i) - y_i)^2 - (f_t^*(x_i) - y_i)^2\Big)^2 \mid \mathscr{F}_t^n\right] \\ &= \mathbb{E}\left[\big(f(x_i) - f_t^*(x_i)\big)^2 \cdot \big(f(x_i) + f_t^*(x_i) - 2y_i\big)^2 \mid \mathscr{F}_t^n\right] \\ &\le 36W^4 \cdot \mathbb{E}[\big(f(x_i) - f_t^*(x_i)\big)^2 \mid \mathscr{F}_t^n] \\ &\le 36W^4 \cdot (C \cdot \mathbb{E}[Z_i(f) \mid \mathscr{F}_t^n] + \epsilon_0), \end{aligned}$$

where the last inequality is by Assumption 5.5.2 and Equation (5.28). Next, we apply Bernstein's inequality on the function cover $\mathcal{C}(\mathcal{F}, \epsilon_2)$ and take the union bound. Specifically, with probability at least $1 - \delta$, for all $g \in \mathcal{C}(\mathcal{F}, \epsilon_2)$,

$$\begin{aligned} & \mathbb{E}[Z_i(g) \mid \mathscr{F}_t^n] - \frac{1}{M}\sum_{i=1}^M Z_i(g) \\ \le & \sqrt{\frac{2\mathbb{V}[Z_i(g) \mid \mathscr{F}_t^n] \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}} + \frac{12W^4 \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} \\ \le & \sqrt{\frac{72W^4(C \cdot \mathbb{E}[Z_i(g) \mid \mathscr{F}_t^n] + \epsilon_0) \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}} + \frac{12W^4 \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}. \end{aligned}$$

For f_t, there exists $g \in \mathcal{C}(\mathcal{F}, \epsilon_2)$ such that $\|f_t - g\|_\infty \leq \epsilon_2$ and

$$\begin{aligned}|Z_i(f_t) - Z_i(g)| &= \left|(f_t(x_i) - y_i)^2 - (g(x_i) - y_i)^2\right| \\ &= |f_t(x_i) - g(x_i)| \cdot |f_t(x_i) + g(x_i) - 2y_i| \leq 6W^2\epsilon_2.\end{aligned}$$

Therefore, with probability at least $1 - \delta$,

$$\begin{aligned}&\mathbb{E}[Z_i(f_t) \mid \mathscr{F}_t^n] - \frac{1}{M}\sum_{i=1}^M Z_i(f_t) \\ \leq &\mathbb{E}[Z_i(g) \mid \mathscr{F}_t^n] - \frac{1}{M}\sum_{i=1}^M Z_i(g) + 12W^2\epsilon_2 \\ \leq &\sqrt{\frac{72W^4(C \cdot \mathbb{E}[Z_i(g) \mid \mathscr{F}_t^n] + \epsilon_0)\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}} + \frac{12W^4\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} + 12W^2\epsilon_2 \\ \leq &\sqrt{\frac{72W^4(C \cdot \mathbb{E}[Z_i(f_t) \mid \mathscr{F}_t^n] + 6CW^2\epsilon_2 + \epsilon_0)\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}} + \frac{12W^4\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} + 12W^2\epsilon_2.\end{aligned}$$

Since f_t is an empirical minimizer, we have $\frac{1}{M}\sum_{i=1}^M Z_i(f_t) \leq 0$. Thus,

$$\mathbb{E}[Z_i(f_t) \mid \mathscr{F}_t^n] \leq \sqrt{\frac{72W^4(C \cdot \mathbb{E}[Z_i(f_t) \mid \mathscr{F}_t^n] + 6CW^2\epsilon_2 + \epsilon_0)\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M}} + \frac{12W^4\log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} + 12W^2\epsilon_2.$$

Solving the above inequality with quadratic formula and using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, $\sqrt{ab} \leq a/2 + b/2$ for $a > 0, b > 0$, we obtain

$$\mathbb{E}[Z_i(f_t) \mid \mathscr{F}_t^n] \leq \frac{500C \cdot W^4 \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} + 13W^2 \cdot \epsilon_2 + \epsilon_0.$$

Since the right-hand side is a constant, through taking another expectation, we have

$$\mathbb{E}[Z_i(f_t)] \leq \frac{500C \cdot W^4 \cdot \log\frac{\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta}}{M} + 13W^2 \cdot \epsilon_2 + \epsilon_0.$$

Notice that $\mathbb{E}[Z_i(f_t)] = L(f_t; \rho_{\text{cov}}^n, Q_{b^n}^t - b^n) - L(f_t^*; \rho_{\text{cov}}^n, Q_{b^n}^t - b^n)$. The desired result is obtained. ■

Combining all previous lemmas, we have the following theorem which states the detailed sample complexity of ENIAC-SPI-SAMPLE (a detailed version of Theorem 5.4.1)

Theorem 5.5.1 (Main Result: Sample Complexity of ENIAC-SPI-SAMPLE)**.** *Let* $\delta \in (0,1)$ *and* $\varepsilon \in (0, 1/(1-\gamma))$*. With Assumptions 5.5.1, 5.5.2, 5.4.2, and 5.4.3, we set the hyperparameters as:*

$$\beta = \frac{\varepsilon(1-\gamma)}{2}, \quad T = \frac{64W^2 \cdot \log|\mathcal{A}|}{\varepsilon^2(1-\gamma)^2}, \quad N \geq \frac{32W^2 \cdot dim_E(\mathcal{F},\beta)}{\varepsilon^3(1-\gamma)^3}, \quad \eta = \sqrt{\frac{\log(|\mathcal{A}|)}{16W^2T}}$$
$$\epsilon_1 = \frac{(1-\gamma)^3\varepsilon^3}{128W \cdot dim_E(\mathcal{F},\beta)}, \quad K = \frac{128W^2 \cdot dim_E(\mathcal{F},\beta) \cdot \big(\log(\frac{3NT \cdot \mathcal{N}(\mathcal{F},\epsilon_1)}{\delta})\big)^2 \cdot \log(\frac{6NT}{\delta})}{\varepsilon^3(1-\gamma)^3},$$
$$\epsilon_2 = \frac{(1-\gamma)^3\varepsilon^3}{110C \cdot W^2 \cdot dim_E(\mathcal{F},\beta)}, \quad M = \frac{4000C^2W^4 \cdot dim_E(\mathcal{F},\beta) \cdot \log(\frac{3NT \cdot \mathcal{N}(\mathcal{F},\epsilon_2)}{\delta})}{\varepsilon^3(1-\gamma)^3},$$

and ϵ *satisfies Equation* (5.23) *correspondingly. Then with probability at least* $1-\delta$*, for the average policy* $\pi^N_{ave} := \pi^N_{ave} := \mathrm{Unif}(\pi^2, \ldots, \pi^{N+1})$*, we have*

$$V^{\pi^N_{ave}} \geq V^{\tilde{\pi}} - \frac{4\sqrt{|\mathcal{A}|\epsilon_{bias}}}{1-\gamma} - \epsilon_0 \cdot \frac{16C\, dim_E(\mathcal{F},\beta)}{\varepsilon^2(1-\gamma)^3} - 9\varepsilon$$

for any comparator $\tilde{\pi}$ *with total number of samples:*

$$\tilde{\mathcal{O}}\Big(\frac{C^2W^8 \cdot \big(dim_E(\mathcal{F},\beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big),$$

where $\epsilon' = \min(\epsilon_1, \epsilon_2)$*.*

Proof. By Lemma 5.5.3, we have that with probability at least $1 - N\delta_1$,

$$V^{\pi^N_{\mathrm{ave}}} \geq \frac{1}{N}\sum_{n=1}^{N} V_{b^n}^{\pi^{n+1}} - \frac{2\epsilon^2 + 8KW^2 + \beta^2}{(1-\gamma)\beta^2 NK} \cdot \dim_E(\mathcal{F},\beta) + \frac{1}{1-\gamma}\sqrt{\frac{\log(2/\delta_1)}{2K}}. \quad (5.29)$$

By Lemma 5.5.4, 5.5.5, and 5.5.2, we have that for every $n \in [N]$, with probability at least $1 - 2T\delta_1$,

$$V_{b^n}^{\pi^{n+1}} \geq V^{\tilde{\pi}} - \frac{1}{1-\gamma}\Big(8W\sqrt{\frac{\log(|\mathcal{A}|)}{T}} + 4\sqrt{|\mathcal{A}|\epsilon_{\mathrm{bias}}} + 2\beta\Big). \quad (5.30)$$

Combining inequalities (5.29) and (5.30), we have with probability at least $1 - 3NT\delta_1$,

$$V^{\pi^N_{\mathrm{ave}}} \geq V^{\tilde{\pi}} - \frac{1}{1-\gamma}\Bigg(\frac{2\epsilon^2 + 8KW^2 + \beta^2}{\beta^2 NK} \cdot \dim_E(\mathcal{F},\beta) + \sqrt{\frac{\log(2/\delta_1)}{2K}} + 8W\sqrt{\frac{\log(|\mathcal{A}|)}{T}} + 4\sqrt{|\mathcal{A}|\epsilon_{\mathrm{bias}}} + 2\beta\Bigg). \quad (5.31)$$

We plug in the value of ϵ^2 in Equation (5.23) with the bound on ϵ_{stat} in Lemma 5.5.6 and choose hyperparameters such that every term in (5.31) (except for the ones with ϵ_0 or ϵ_{bias}) is bounded by ε. Finally, we set $\delta_1 = \delta/(3NT)$ and $\epsilon' = \min(\epsilon_1, \epsilon_2)$. In total, the sample complexity is

$$N(K+TM) = \tilde{\mathcal{O}}\Big(\frac{C^2W^8 \cdot \big(\dim_E(\mathcal{F},\beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big).$$

■

Corollary 7. *If Assumption 5.4.1 holds, with proper hyperparameters, the average policy* $\pi^N_{ave} := \text{Unif}(\pi^2, \ldots, \pi^{N+1})$ *of ENIAC-SPI-SAMPLE achieves* $V^{\pi^N_{ave}} \geq V^{\tilde{\pi}} - \varepsilon$ *with probability at least* $1-\delta$ *and the sample complexity is*

$$\tilde{\mathcal{O}}\Big(\frac{W^8 \cdot \big(dim_E(\mathcal{F},\beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big).$$

Proof. The result is straightforward as mentioned in Remark 9 that under Assumption 5.4.1, $\epsilon_{\text{bias}} = 0$, $C = 1$, and $\epsilon_0 = 0$. ■

Next we prove the result for ENIAC-SPI-COMPUTE. SPI-COMPUTE only differs from SPI-SAMPLE at two places: the value of the bonus and the actor update rule. These differences cause changes in the bonus bound result and the convergence analysis while Lemma 5.5.5 and 5.5.6 still hold with the same definition of $\widehat{A}^t_{b^n}$ as in (5.19). In the sequel, we present the bonus bound and the convergence result for SPI-COMPUTE.

Lemma 5.5.7 (SPI-COMPUTE: The Bound of Bonus)**.** *With probability at least* $1 - N\delta$*,*

$$\sum_{n=1}^{N} V^{\pi^{n+1}}_{b^n} - V^{\pi^{n+1}} \leq \frac{|\mathcal{A}|}{(1-\gamma)\alpha} \cdot \frac{2\epsilon^2 + 8W^2K + \beta^2}{\beta^2 K} \cdot dim_E(\mathcal{F},\beta) + \frac{N|\mathcal{A}|}{(1-\gamma)\alpha}\sqrt{\frac{\log(2/\delta)}{2K}}.$$

The proof is similar to Lemma 5.5.3. We only need to revise the bonus value from $\frac{1}{1-\gamma}$ to $\frac{|\mathcal{A}|}{(1-\gamma)\alpha}$.

As for the actor-critic convergence, we focus on a specific epoch n and still define

$$\widehat{A}^t_{b^n}(s,a) := f_t(s,a) + b^n(s,a) - \mathbb{E}_{a' \sim \pi_t(\cdot|s)}[f_t(s,a') + b^n(s,a')]. \tag{5.32}$$

It is easy to verify that $\mathbb{E}_{a\sim\pi_t(\cdot|s)}[\widehat{A}^t_{b^n}] = 0$ and for $s \in \mathcal{K}^n$, the actor update in SPI-COMPUTE is equivalent to

$$\pi'_{t+1}(a|s) \propto \pi'_t(a|s)\exp\big(\eta\widehat{A}^t_{b^n}(s,a)\big),\ \pi_{t+1} = (1-\alpha)\pi'_{t+1} + \alpha\mathrm{Unif}(\mathcal{A})$$

since $b^n(s,\cdot) = 0$ for $s \in \mathcal{K}^n$. As before, we use $\widehat{A}_{b^n}(s,a)$ to approximate the true advantage of π^n_t on $\mathcal{M}_{b^n}$. Then we have the following result.

Lemma 5.5.8 (SPI-COMPUTE: Actor-Critic Convergence)**.** *In ENIAC-SPI-COMPUTE, let $\widehat{A}^t_{b^n}$ be as defined in Equation* (5.32)*, $\eta = \sqrt{\frac{\log(|\mathcal{A}|)}{16W^2T}}$, and $\alpha = \frac{1}{1+\sqrt{T}}$. For any epoch $n \in [N]$, SPI-COMPUTE obtains a sequence of policies $\{\pi_t\}_{t=0}^{T-1}$ such that when comparing to $\tilde{\pi}^n$:*

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}(V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{b^n}) = \frac{1}{T}\sum_{t=0}^{T-1}(V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n}) \\
&\le \frac{1}{1-\gamma}\Big(12W\sqrt{\frac{\log(|\mathcal{A}|)}{T}} + \frac{1}{T}\sum_{t=0}^{T-1}\Big(\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\big(A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a)\big)\mathbf{1}\{s\in\mathcal{K}^n\}\Big).
\end{aligned}$$

Proof of Lemma 5.5.8. Similar to the reasoning in Lemma 5.5.4, we first have that $A^t_{\mathcal{M}^n}(s,a^\dagger) \le 0$ for any $s \notin \mathcal{K}^n$. To see this, note that for $s \notin \mathcal{K}^n$, there exists an action with bonus $b^n = |\mathcal{A}|/\big((1-\gamma)\alpha\big)$ and π_t has probability at least $\alpha/|\mathcal{A}|$ selects that action. Therefore, $V^t_{\mathcal{M}^n}(s) \ge 1/(1-\gamma)$ and

$$A^t_{\mathcal{M}^n}(s,a^\dagger) = Q^t_{\mathcal{M}^n}(s,a^\dagger) - V^t_{\mathcal{M}^n}(s) = 1 - (1-\gamma)\cdot V^t_{\mathcal{M}^n}(s) \le 0,\quad \forall s \notin \mathcal{K}^n.$$

Recall that $\tilde{\pi}^n$ deterministically picks $a^\dagger$ for $s \notin \mathcal{K}^n$. Based on the above inequality, it holds that

$$\begin{aligned}
V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n} &= \frac{1}{1-\gamma}\sum_{(s,a)}\tilde{d}_{\mathcal{M}^n}(s,a)A^t_{\mathcal{M}^n}(s,a) \le \frac{1}{1-\gamma}\sum_{(s,a)}\tilde{d}_{\mathcal{M}^n}(s,a)A^t_{\mathcal{M}^n}(s,a)\mathbf{1}\{s\in\mathcal{K}^n\} \\
&= \frac{1}{1-\gamma}\sum_{(s,a)}\tilde{d}_{\mathcal{M}^n}(s,a)A^t_{b^n}(s,a)\mathbf{1}\{s\in\mathcal{K}^n\}. \qquad (5.33)
\end{aligned}$$

Next we restrict on $s \in \mathcal{K}^n$ and establish the consecutive KL difference on $\{\pi'_t(\cdot|s)\}$. Specifically, since for $s \in \mathcal{K}^n$, $\pi'_{t+1}(\cdot|s) \propto \pi'_t(\cdot|s)\exp(\eta \widehat{A}^t_{b^n}(s,a))$,

$$\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_{t+1}(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_t(\cdot|s)) = \mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[-\eta \widehat{A}^t_{b^n}(s,a) + \log(z^t)],$$

where $z^t := \sum_a \pi'_t(a|s)\exp(\eta \widehat{A}^t_{b^n}(s,a))$. With the assumptions that $|\widehat{A}^t_{b^n}(s,a)| \le 4W$ and $\eta \le 1/(4W)$ when $T > \log(|\mathcal{A}|)$, we have that $\eta \widehat{A}^t_{b^n}(s,a) \le 1$. By the inequality that $\exp(x) \le 1 + x + x^2$ for $x \le 1$, we have that

$$\begin{aligned}
\log(z^t) &\le \log(1 + \eta \sum_a \pi'_t(a|s)\widehat{A}^t_{b^n}(s,a) + 16\eta^2 W^2) \\
&= \log\left(1 + \eta \sum_a \left(\frac{\pi_t(a|s)}{1-\alpha} - \frac{\alpha \cdot \mathrm{Unif}(\mathcal{A})}{1-\alpha}\right) \cdot \widehat{A}^t_{b^n}(s,a) + 16\eta^2 W^2\right) \\
&= \log\left(1 - \frac{\eta\alpha}{(1-\alpha)|\mathcal{A}|}\sum_a \widehat{A}^t_{b^n}(s,a) + 16\eta^2 W^2\right) \\
&\le \log(1 + \eta\frac{4W\alpha}{1-\alpha} + 16\eta^2 W^2) \\
&\le \frac{4W\eta\alpha}{1-\alpha} + 16\eta^2 W^2,
\end{aligned}$$

where the second line follows from that $\pi'_t = \frac{\pi_t}{1-\alpha} - \frac{\alpha \mathrm{Unif}(\mathcal{A})}{1-\alpha}$ and the last line follows that $\log(1+x) \le x$ for $x > 0$. Hence, for $s \in \mathcal{K}^n$,

$$\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_{t+1}(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_t(\cdot|s)) \le -\eta \mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\widehat{A}^t_{b^n}(s,a)] + \frac{4W\eta\alpha}{1-\alpha} + 16\eta^2 W^2.$$

Take $\alpha = \frac{1}{1+\sqrt{T}}$. Adding both sides from $t = 0$ to $T-1$, we get

$$\begin{aligned}
&\sum_{t=0}^{T-1} \mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}[\widehat{A}^t_{b^n}(s,a)\mathbf{1}\{s \in \mathcal{K}^n\}] \\
\le &\frac{1}{\eta}\mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n}}\left[\left(\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_0(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi'_T(\cdot|s))\right)\mathbf{1}\{s \in \mathcal{K}^n\}\right] + 4W\sqrt{T} + 16\eta T W^2 \\
\le &\log(|\mathcal{A}|)/\eta + 4W\sqrt{T} + 16\eta T W^2 \le 12W\sqrt{\log(|\mathcal{A}|)T}.
\end{aligned}$$

Combining with Equation (5.33), the regret on $\mathcal{M}^n$ satisfies

$$\sum_{t=0}^{T-1}(V_{\mathcal{M}^n}^{\tilde{\pi}^n} - V_{\mathcal{M}^n}^t)$$
$$\leq \frac{1}{1-\gamma}\left(\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[\widehat{A}_{b^n}^t(s,a)\mathbf{1}\{s\in\mathcal{K}^n\}\right] + \sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[A_{b^n}^t(s,a) - \widehat{A}_{b^n}^t(s,a))\mathbf{1}\{s\in\mathcal{K}^n\}\right]\right)$$

$$\leq \frac{1}{1-\gamma}\left(12W\sqrt{\log(|\mathcal{A}|)T} + \sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[\left(A_{b^n}^t(s,a) - \widehat{A}_{b^n}^t(s,a)\right)\mathbf{1}\{s\in\mathcal{K}^n\}\right]\right).$$

■

Since the definition of $\widehat{A}_{b^n}^t$ is the same as the one for SPI-SAMPLE, Lemma 5.5.5 and Lemma 5.5.6 are directly applied. In total, we have the following theorem for the sample complexity of ENIAC-SPI-COMPUTE.

Theorem 5.5.2 (Main Result: Sample Complexity of ENIAC-SPI-COMPUTE)**.** *Let $\delta \in (0,1)$ and $\varepsilon \in (0, 1/(1-\gamma))$. With Assumptions 5.5.1, 5.5.2, 5.4.2, and 5.4.3, we set the hyperparameters as:*

$$\beta = \frac{\varepsilon(1-\gamma)}{2},\ T = \frac{144W^2\cdot\log|\mathcal{A}|}{\varepsilon^2(1-\gamma)^2},\ N \geq \frac{384W^3|\mathcal{A}|\log(|\mathcal{A}|)\cdot dim_E(\mathcal{F},\beta)}{\varepsilon^4(1-\gamma)^4},\ \eta = \sqrt{\frac{\log(|\mathcal{A}|)}{16W^2T}},$$

$$\alpha = \frac{1}{1+\sqrt{T}},\ \epsilon_1 = \frac{(1-\gamma)^4\varepsilon^4}{1536W^2|\mathcal{A}|\log(|\mathcal{A}|)\cdot dim_E(\mathcal{F},\beta)},\ \epsilon_2 = \frac{(1-\gamma)^4\varepsilon^4}{1248CW^3|\mathcal{A}|\log(|\mathcal{A}|)dim_E(\mathcal{F},\beta)},$$

$$K = \frac{1536W^3|\mathcal{A}|^2(\log(|\mathcal{A}|))^2\cdot dim_E(\mathcal{F},\beta)\cdot\left(\log(\frac{3NT\cdot\mathcal{N}(\mathcal{F},\epsilon_1)}{\delta})\right)^2\cdot\log(\frac{6NT}{\delta})}{\varepsilon^4(1-\gamma)^4},$$
$$M = \frac{48000C^2W^5|\mathcal{A}|\log(|\mathcal{A}|)dim_E(\mathcal{F},\beta)\log(\frac{3NT\cdot\mathcal{N}(\mathcal{F},\epsilon_2)}{\delta})}{\varepsilon^4(1-\gamma)^4},$$

and ϵ satisfies Equation (5.23) correspondingly. Then with probability at least $1-\delta$, for the average policy $\pi_{ave}^N := \mathrm{Unif}(\pi^2,\ldots,\pi^{N+1})$, we have

$$V^{\pi_{ave}^N} \geq V^{\tilde{\pi}} - \frac{4\sqrt{|\mathcal{A}|\epsilon_{bias}}}{1-\gamma} - \epsilon_0\cdot\frac{200CW\cdot|\mathcal{A}|\log(|\mathcal{A}|)\cdot dim_E(\mathcal{F},\beta)}{\varepsilon^3(1-\gamma)^4} - 9\varepsilon$$

for any comparator $\tilde{\pi}$ with total number of samples:

$$\tilde{\mathcal{O}}\Big(\frac{C^2W^{10}\cdot|\mathcal{A}|^2\cdot\big(dim_E(\mathcal{F},\beta)\big)^2\cdot\big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^{10}(1-\gamma)^{10}}\Big),$$

where $\epsilon'=\min(\epsilon_1,\epsilon_2)$.

Corollary 8. *If Assumption 5.4.1 holds, with proper hyperparameters, the average policy $\pi^N_{ave}:=\mathrm{Unif}(\pi^2,\ldots,\pi^{N+1})$ of ENIAC-SPI-COMPUTE achieves $V^{\pi^N_{ave}}\geq V^{\tilde{\pi}}-\varepsilon$ with probability at least $1-\delta$ and total number of samples:*

$$\tilde{\mathcal{O}}\Big(\frac{W^{10}\cdot|\mathcal{A}|^2\cdot\big(dim_E(\mathcal{F},\beta)\big)^2\cdot\big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^{10}(1-\gamma)^{10}}\Big).$$

5.5.4 Proofs of ENIAC-NPG

In this section, we provide the sample complexity of ENIAC-NPG-SAMPLE. For ENIAC-NPG-COMPUTE, it can be adapted from ENIAC-SPI-COMPUTE and ENIAC-NPG-SAMPLE.

The analysis of ENIAC-NPG-SAMPLE is in parallel to that of ENIAC-SPI-SAMPLE. As before, we provide a general result which considers model misspecification and Theorem 5.4.2 falls as a special case under the closedness Assumption 5.4.4.

We simplify the notation as π_θ for $\pi_{f_\theta}(a|s):=\frac{\exp(f_\theta(s,a))}{\sum_{a'}\exp(f_\theta(s,a'))}$. Then for epoch n iteration t in ENIAC-NPG-SAMPLE,

$$\pi^n_t(\cdot|s)=\begin{cases}\pi_{\theta^n_t}(\cdot|s), & s\in\mathcal{K}^n\\ \mathrm{Unif}(\{a\in\mathcal{A}:(s,a)\notin\mathcal{K}^n\}), & o.w.\end{cases}$$

We state the following assumptions to quantify the misspecification error.

Assumption 5.5.3 (Bounded Transfer Error)**.** *Given a target function $g:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, we define the critic loss function $L(u;d,g,\pi_\theta)$ with $d\in\Delta(\mathcal{S}\times\mathcal{A})$ as:*

$$L(u;d,g,\pi_\theta):=\mathbb{E}_{(s,a)\sim d}\left[(u^\top\nabla_\theta\log\pi_\theta-g)^2\right].$$

For the fixed comparator policy $\tilde{\pi}$ as mentioned in Section 5.5.1, we define a state-action distribution $\tilde{d}(s,a) := d_{s_0}^{\tilde{\pi}}(s) \circ \mathrm{Unif}(\mathcal{A})$. In ENIAC-NPG-SAMPLE, for every epoch $n \in [N]$ and every iteration t inside epoch n, we assume that

$$\inf_{u \in \mathcal{U}_t^n} L(u; \tilde{d}, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) \leq \epsilon_{bias},$$

where $\mathcal{U}_t^n := \mathrm{argmin}_{u \in \mathcal{U}} L(u; \rho_{cov}^n, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n})$ and $\epsilon_{bias} \geq 0$ is a problem-dependent constant.

Recall that $\big(A_{b^n}^t - \bar{b}_t^n\big)(s,a) = Q_{b^n}^t(s,a) - b^n(s,a) - \mathbb{E}_{a \sim \pi_t^n(\cdot|s)}[Q_{b^n}^t(s,a) - b^n(s,a)]$. As before, we denote by $\tilde{u}_t^n$ a particular vector in $\mathcal{U}_t^n$ such that $L(\tilde{u}_t^n; \tilde{d}, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) \leq 2\epsilon_{\text{bias}}$. Note that we use $\nabla_\theta \log \pi_{\theta_t^n}$ as the linear features for critic fit at iteration t epoch n, even though π_t^n is not the same as $\pi_{\theta_t^n}$. Nevertheless, we show later that this choice of features is sufficient for good critic fitting on the known states, where we measure our critic error.

Remark 10. *Under the closedness condition Assumption 5.4.4,*

$$\begin{aligned} A_{b^n}^t(s,a) - \bar{b}^n(s,a) &= Q_{b^n}^t(s,a) - b^n(s,a) - \mathbb{E}_{a' \sim \pi_t^n}(Q_{b^n}^t - b^n(s,a')) \\ &= \mathbb{E}^{\pi_t^n}[r(s,a) + \gamma Q_{b^n}^t(s',a')] - \mathbb{E}_{a' \sim \pi_t^n}[\mathbb{E}^{\pi_t^n}[r(s,a') + \gamma Q_{b^n}^t(s'',a'')]] \\ &\in \mathcal{G}_{f_{\theta_t^n}}, \end{aligned}$$

where the last step follows, since π_t^n can be described as $\pi_{\theta_t^n, \mathcal{K}^n}$ under the notation of Assumption 5.4.4, whence the containment of $\mathcal{G}_{f_{\theta_t^n}}$ follows. Thus, there exists a vector $u \in \mathcal{U}$ such that $u^\top \nabla \log \pi_{f_{\theta_t^n}} = A_{b^n}^t - \bar{b}^n$ everywhere. We can then take ϵ_{bias} as 0 and $\tilde{u}_t^n = u$. Assumption 5.5.3 therefore is a generalized version of the closedness condition.

For NPG, the loss function L is convex in the parameters u since the features are fixed for every individual iteration. As a result, we naturally have an inequality as in Assumption 5.5.2 for SPI. We present it in the lemma below, which essentially follows a similar result for the linear case in [AHK20].

Lemma 5.5.9. *For the same loss function L as defined in Assumption 5.5.3, it holds that*

$$\begin{aligned}&\mathbb{E}_{(s,a)\sim\rho^n_{cov}}\left[\left((u^n_t-\tilde{u}^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}\right)^2\right]\\ \leq& L(u^n_t;\rho^n_{cov},A^t_{b^n}-\bar{b}^n_t,\pi_{\theta^n_t})-L(\tilde{u}^n_t;\rho^n_{cov},A^t_{b^n}-\bar{b}^n_t,\pi_{\theta^n_t}).\end{aligned}$$

Proof. For the left-hand side, we have that

$$\begin{aligned}&\mathbb{E}_{(s,a)\sim\rho^n_{\text{cov}}}\left[\left((u^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}-(\tilde{u}^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}\right)^2\right]\\ =&\mathbb{E}_{(s,a)\sim\rho^n_{\text{cov}}}\left[\left((u^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}+\bar{b}^n_t-A^t_{b^n}\right)^2\right]-\mathbb{E}_{(s,a)\sim\rho^n_{\text{cov}}}\left[\left((\tilde{u}^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}+\bar{b}^n_t-A^t_{b^n}\right)^2\right]\\ &-2\mathbb{E}_{(s,a)\sim\rho^n_{\text{cov}}}\left[\left((u^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}-(\tilde{u}^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}\right)\cdot\left((\tilde{u}^n_t)^\top\nabla_\theta\log\pi_{\theta^n_t}+\bar{b}^n_t-A^t_{b^n}\right)\right]\end{aligned}$$

Since $\tilde{u}^n_t$ is a minimizer. By first-order optimality condition, the cross term is greater or equal to 0. The desired result is obtained. ■

We follow the same steps as listed in 5.5.2 and start with the bonus bound.

Lemma 5.5.10 (NPG-SAMPLE: The Bound of Bonus)**.** *With probability at least $1-N\delta$,*

$$\sum_{n=1}^N V^{\pi^{n+1}}_{b^n}-V^{\pi^{n+1}}\leq\frac{2\epsilon^2+32G^2B^2K+\beta^2}{(1-\gamma)\beta^2K}\cdot dim_E(\mathcal{G}_\mathcal{F},\beta)+\frac{N}{1-\gamma}\sqrt{\frac{\log(2/\delta)}{2K}}.$$

The proof is similar to Lemma 5.5.3. The only thing changed is the function approximation space. Thus we have $\dim_E(\mathcal{G}_\mathcal{F},\beta)$ instead of $\dim_E(\mathcal{F},\beta)$ and $\|g^u_\theta\|_\infty\leq 2GB$, $\forall g^u_\theta\in\mathcal{G}_\mathcal{F}$.

Next, we establish the convergence result of NPG update. We focus on a specific episode n and for each iteration t, we define

$$\widehat{A}^t_{b^n}(s,a):=u_t^\top\nabla f_{\theta_t}(s,a)+b^n-\mathbb{E}_{a'\sim\pi_{\theta_t}(\cdot|s)}[u_t^\top\nabla f_{\theta_t}(s,a')+b^n(s,a')]. \tag{5.34}$$

Since $\pi_t(\cdot|s)=\pi_{\theta_t}(\cdot|s)$ for $s\in\mathcal{K}^n$, $\mathbb{E}_{a'\sim\pi_t(\cdot|s)}[\widehat{A}^t_{b^n}(s,a')]=0$ for $s\in\mathcal{K}^n$.

From the algorithm we can see that $\widehat{A}^t_{b^n}$ is indeed our approximation to the real advantages $A^t_{b^n}$. In contrary to ENIAC-SPI, the actor update in ENIAC-NPG does not use $\widehat{A}^t_{b^n}$ directly

but by modifying the parameter θ. In the next lemma, we show how to link the NPG update to a formula of $\widehat{A}^t_{b^n}$ and eventually are able to bound the policy sub-optimality with function approximation error.

Lemma 5.5.11 (NPG-SAMPLE: Convergence)**.** *In ENIAC-NPG-SAMPLE, let $\widehat{A}^t_{b^n}$ be as defined in Equation* (5.34) *and $\eta = \sqrt{\frac{\log(|\mathcal{A}|)}{(16D^2+\Lambda B^2)T}}$. For any epoch $n \in [N]$, NPG-SAMPLE obtains a sequence of policies $\{\pi_t\}_{t=0}^{T-1}$ such that when comparing to $\tilde{\pi}$:*

$$\frac{1}{T}\sum_{t=0}^{T-1}(V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{b^n}) = \frac{1}{T}\sum_{t=0}^{T-1}(V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n})$$
$$\leq \frac{1}{1-\gamma}\left(2\sqrt{\frac{\log(|\mathcal{A}|)(16D^2+\Lambda B^2)}{T}} + \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\left[\left(A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a)\right)\mathbf{1}\{s\in\mathcal{K}^n\}\right]\right).$$

Proof. For the same reason as in Lemma 5.5.4, we have

$$V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n} \leq \frac{1}{1-\gamma}\sum_{(s,a)}\tilde{d}_{\mathcal{M}^n}(s,a)A^t_{b^n}(s,a)\mathbf{1}\{s\in\mathcal{K}^n\}. \tag{5.35}$$

We focus on on $s \in \mathcal{K}^n$. Then $\pi_t(\cdot|s) \propto \exp(f_{\theta_t}(s,\cdot))$ and $b(s,\cdot) = 0$. It holds that

$$\begin{aligned}
&\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_{t+1}(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_t(\cdot|s))\\
&= -\mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}\left[f_{\theta_{t+1}}(s,a) - f_{\theta_t}(s,a)\right] + \log\frac{\sum_a \exp(f_{\theta_{t+1}}(s,a))}{\sum_a \exp(f_{\theta_t}(s,a))}\\
&\leq -\mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\eta\cdot u_t^\top\nabla_\theta f_{\theta_t} - \eta^2\frac{\Lambda B^2}{2}] + \log\frac{\sum_a \exp(f_{\theta_t}(s,a) + \eta\cdot u_t^\top\nabla_\theta f_{\theta_t} + \eta^2\Lambda B^2/2)}{\sum_a \exp(f_{\theta_t}(s,a))}\\
&= -\eta\cdot\mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\widehat{A}^t_{b^n}(s,a)] - \eta\cdot\mathbb{E}_{a'\sim\pi_t(\cdot|s)}u_t^\top\nabla_\theta f_{\theta_t}(s,a')\\
&\quad + \log\Big(\sum_a \pi_t(s,a)\exp\left(\eta\cdot\widehat{A}^t_{b^n}(s,a) + \eta\cdot\mathbb{E}_{a'\sim\pi_t(\cdot|s)}u_t^\top\nabla_\theta f_{\theta_t}\right)\Big) + \eta^2\Lambda B^2\\
&= -\mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\eta\widehat{A}^t_{b^n}(s,a)] + \log\Big(\sum_a \pi_t(a|s)\exp\left(\eta\widehat{A}^t_{b^n}(s,a)\right)\Big) + \eta^2\Lambda B^2.
\end{aligned}$$

where the inequality is by Taylor expansion and the regularity assumption 5.4.5:

$$f_{\theta_t} + (\theta_{t+1}-\theta_t)^\top\nabla_\theta f_{\theta_t} - \frac{\Lambda}{2}\|\theta_{t+1}-\theta_t\|_2^2 \leq f_{\theta_{t+1}} \leq f_{\theta_t} + (\theta_{t+1}-\theta_t)^\top\nabla_\theta f_{\theta_t} + \frac{\Lambda}{2}\|\theta_{t+1}-\theta_t\|_2^2.$$

Since $|\widehat{A}^t_{b^n}(s,a)| \le 4D$ and $\eta \le 1/(4D)$ when $T > \log(|\mathcal{A}|)$, $\eta\widehat{A}^t_{b^n}(s,a) \le 1$. By the inequality that $\exp(x) \le 1 + x + x^2$ for $x \le 1$, we have that

$$\begin{aligned}
&\log\Big(\sum_a \pi_t(a|s)\exp\big(\eta\widehat{A}^t_{b^n}(s,a)\big)\Big) \\
\le &\log\Big(1 + \mathbb{E}_{a\sim\pi_t(\cdot|s)}[\eta\widehat{A}^t_{b^n}(s,a)] + 16\eta^2D^2\Big) \le 16\eta^2D^2.
\end{aligned}$$

Hence, for $s \in \mathcal{K}^n$,

$$\mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_{t+1}(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s), \pi_t(\cdot|s)) \le -\eta\mathbb{E}_{a\sim\tilde{\pi}^n(\cdot|s)}[\widehat{A}^t_{b^n}(s,a)] + \eta^2(16D^2 + \Lambda B^2).$$

Adding both sides from $t = 0$ to $T-1$ and taking $\eta = \sqrt{\frac{\log(|\mathcal{A}|)}{(16D^2+\Lambda B^2)T}}$, we get

$$\begin{aligned}
&\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big[\widehat{A}^t_{b^n}(s,a)\mathbf{1}\{s\in\mathcal{K}^n\}\Big] \\
\le &\frac{1}{\eta}\mathbb{E}_{s\sim\tilde{d}_{\mathcal{M}^n}}\Big[\big(\mathbf{KL}(\tilde{\pi}^n(\cdot|s),\pi_0(\cdot|s)) - \mathbf{KL}(\tilde{\pi}^n(\cdot|s),\pi_T(\cdot|s))\big)\mathbf{1}\{s\in\mathcal{K}^n\}\Big] + \eta T(16D^2+\Lambda B^2) \\
\le &\log(|\mathcal{A}|)/\eta + \eta T(16D^2+\Lambda B^2) \le 2\sqrt{\log(|\mathcal{A}|)\cdot(16D^2+\Lambda B^2)\cdot T}.
\end{aligned}$$

Combining with Equation (5.35), the regret on $\mathcal{M}^n$ satisfies

$$\begin{aligned}
&\sum_{t=0}^{T-1}(V^{\tilde{\pi}^n}_{\mathcal{M}^n} - V^t_{\mathcal{M}^n}) \\
&\le \frac{1}{1-\gamma}\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big[\widehat{A}^t_{b^n}(s,a)\mathbf{1}\{s\in\mathcal{K}^n\}\Big] + \frac{1}{1-\gamma}\sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big[A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a))\mathbf{1}\{s\in\mathcal{K}^n\Big] \\
&\le \frac{1}{1-\gamma}\Bigg(2\sqrt{\log(|\mathcal{A}|)(16D^2+\Lambda B^2)T} + \sum_{t=0}^{T-1}\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big[\big(A^t_{b^n}(s,a) - \widehat{A}^t_{b^n}(s,a)\big)\mathbf{1}\{s\in\mathcal{K}^n\}\Big]\Bigg).
\end{aligned}$$

■

Next, we establish two lemmas to bound the difference between the true advantage $A^t_{b^n}(s,a)$ and the approximation $\widehat{A}^t_{b^n}(s,a)$.

Lemma 5.5.12 (Approximation Bound)**.** *At epoch n, assume for all $0 \le t \le T-1$,*

$$L(u_t^n; \rho_{cov}^n, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) \le L(\tilde{u}_t^n; \rho_{cov}^n, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) + \epsilon_{stat},$$

where $\epsilon_{stat} > 0$ is to be determined later, and

$$\epsilon^2 = NK\big(\epsilon_{stat} + 16D\epsilon_1\big) + 8D^2 \log(\mathcal{N}(\mathcal{G}_{\mathcal{F}}, \epsilon_1)/\delta) \cdot \sqrt{NK}, \tag{5.36}$$

where ϵ is used in bonus function design (see Section 5.3.3) and ϵ_1 is to be determined. Under Assumption 5.5.3 and 5.4.5, we have that for every $0 \le t \le T-1$, with probability at least $1-(n+1)\delta$,

$$\mathbb{E}_{(s,a)\sim\tilde{d}_{\mathcal{M}^n}}\Big(A_{b^n}^t(s,a) - \widehat{A}_{b^n}^t(s,a)\Big) \le 4\sqrt{|\mathcal{A}|\epsilon_{bias}} + 2\beta.$$

Lemma 5.5.13. *Following the same notation as in Lemma 5.5.12, it holds with probability at least $1-\delta$ that*

$$L(u_t^n; \rho_{cov}^n, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) - L(\tilde{u}_t^n; \rho_{cov}^n, A_{b^n}^t - \bar{b}_t^n, \pi_{\theta_t^n}) \le \frac{500D^4 \cdot d \log\left(\frac{6D}{\epsilon_2\delta}\right)}{M} + 13D^2 \cdot \epsilon_2,$$

where d is the linear dimension of u.

The proofs of the above lemmas can be easily adapted from Lemma 5.5.5 or Lemma 5.5.6 by replacing f_t with $u_t^\top \nabla f_{\theta_t}$, $\tilde{f}_t^n$ with $(\tilde{u}_t^n)^\top \nabla f_{\theta_t}$, and $\mathcal{F}$ with $\mathcal{G}_{\mathcal{F}}$. In particular, for Lemma 5.5.13, since the linear feature is fixed for critic fit at iteration t epoch n, the function cover is defined on the space $\mathcal{G}_{f_{\theta_t^n}}$. By Lemma 5.5.15, the covering number is therefore represented with the linear dimension of u, d.

In the following, we present the detailed form of the sample complexity of NPG-SAMPLE.

Theorem 5.5.3 (Main Result: Sample Complexity of ENIAC-NPG-SAMPLE)**.** *Let $\delta \in (0,1)$*

and $\varepsilon \in (0, 1/(1-\gamma))$. *With Assumptions 5.5.3 and 5.4.5, we set the hyperparameters as:*

$$\beta = \frac{\varepsilon(1-\gamma)}{2}, \quad \eta = \sqrt{\frac{\log(|\mathcal{A}|)}{(16D^2 + \Lambda B^2)T}}$$
$$T = \frac{64(D^2 + \Lambda B^2) \cdot \log|\mathcal{A}|}{\varepsilon^2(1-\gamma)^2}, \quad N \geq \frac{128B^2G^2 \cdot dim_E(\mathcal{G}_\mathcal{F}, \beta)}{\varepsilon^3(1-\gamma)^3},$$
$$\epsilon_1 = \frac{(1-\gamma)^3\varepsilon^3}{128D \cdot dim_E(\mathcal{G}_\mathcal{F}, \beta)}, \quad K = \frac{32D^2 \cdot dim_E(\mathcal{G}_\mathcal{F}, \beta) \cdot \big(\log(\frac{3NT \cdot \mathcal{N}(\mathcal{G}_\mathcal{F}, \epsilon_1)}{\delta})\big)^2 \cdot \log(\frac{6NT}{\delta})}{\varepsilon^3(1-\gamma)^3},$$
$$\epsilon_2 = \frac{(1-\gamma)^3\varepsilon^3}{110D^2 \cdot dim_E(\mathcal{G}_\mathcal{F}, \beta)}, \quad M = \frac{4000D^4 \cdot dim_E(\mathcal{G}_\mathcal{F}, \beta) \cdot d\log(\frac{18DNT}{\epsilon_2\delta})}{\varepsilon^3(1-\gamma)^3},$$

and ϵ *satisfies Equation* (5.36) *correspondingly. Then with probability at least* $1-\delta$, *for the average policy* $\pi^N_{ave} := \mathrm{Unif}(\pi^2, \ldots, \pi^{N+1})$, *we have*

$$V^{\pi^N_{ave}} \geq V^{\tilde{\pi}} - \frac{4\sqrt{|\mathcal{A}|\epsilon_{bias}}}{1-\gamma} - 9\varepsilon$$

for any comparator $\tilde{\pi}$ *with total number of samples:*

$$\tilde{\mathcal{O}}\Big(\frac{D^6(D^2 + \Lambda B^2) \cdot \big(dim_E(\mathcal{G}_\mathcal{F}, \beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{G}_\mathcal{F}, \epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big),$$

where $\epsilon' = \min(\epsilon_1, \epsilon_2)$ *such that* $\log(\mathcal{N}(\mathcal{G}_\mathcal{F}, \epsilon')) = \Omega(d)$.

The proof is similar to that of Theorem 5.5.1. We also have the following result when the closedness assumption is satisfied.

Corollary 9. *If Assumption 5.4.4 holds, with proper hyperparameters, the average policy* $\pi^N_{ave} := \mathrm{Unif}(\pi^2, \ldots, \pi^{N+1})$ *of ENIAC-NPG-SAMPLE achieves* $V^{\pi^N_{ave}} \geq V^{\tilde{\pi}} - \varepsilon$ *with probability at least* $1-\delta$ *and total number of samples:*

$$\tilde{\mathcal{O}}\Big(\frac{D^6(D^2 + \Lambda B^2) \cdot \big(dim_E(\mathcal{G}_\mathcal{F}, \beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{G}_\mathcal{F}, \epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big)$$

Note that under Assumption 5.4.4, as mentioned in Remark 10, $\epsilon_{\text{bias}} = 0$.

5.5.5 Auxiliary Lemmas

Lemma 5.5.14. *Given a function class $\mathcal{F}$, for its covering number, we have $\mathcal{N}(\Delta\mathcal{F}, \epsilon) \leq \mathcal{N}(\mathcal{F}, \epsilon/2)^2$.*

Proof. Let $\Delta\mathcal{C}(\mathcal{F}, \epsilon/2) := \{f - f' | f, f' \in \mathcal{C}(\mathcal{F}, \epsilon/2)\}$. Then $\Delta\mathcal{C}(\mathcal{F}, \epsilon/2)$ is an ϵ-cover for $\Delta\mathcal{F}$ and $|\Delta\mathcal{C}(\mathcal{F}, \epsilon/2)| \leq |\mathcal{C}(\mathcal{F}, \epsilon/2)|^2 \leq \mathcal{N}(\mathcal{F}, \epsilon/2)^2$. ■

Lemma 5.5.15. *Given $f \in \mathcal{F}$, under the regularity Assumption 5.4.5, we have that the covering number of the linear class $\mathcal{G}_f := \{u^\top \nabla_\theta \log \pi_f, u \in \mathcal{U} \subset \mathbb{R}^d, f \in \mathcal{F}\}$ achieves $\mathcal{N}(\mathcal{G}_f, \epsilon) \leq \left(\frac{3D}{\epsilon}\right)^d$.*

Proof. In order to construct a cover set of $\mathcal{G}_f$ with radius ϵ_2, we need that for any $u \in \mathcal{U} \subset \mathbb{R}^d$, there exist a $\tilde{u}$, such that

$$\|u^\top \nabla_\theta \log \pi_f(s, a) - \tilde{u}^\top \nabla_\theta \log \pi_f(s, a)\|_\infty \leq \epsilon_2.$$

where the infinity norm is taken over all $(s, a) \in \mathcal{S} \times \mathcal{A}$. By Cauchy-Schwarz inequality, we have

$$\|u^\top \nabla_\theta \log \pi_f - \tilde{u}^\top \nabla_\theta \log \pi_f\|_\infty = \|(u - \tilde{u})^\top \nabla_\theta \log \pi_f\|_\infty \leq 2G\|u - \tilde{u}\|_2.$$

Thus, it is enough to have $\|u - \tilde{u}\|_2 \leq \epsilon_2/(2G)$, which is equivalent to cover a ball in $\mathbb{R}^d$ with radius B (recall that $\|u\| \leq B$) with small balls of radius $\epsilon_2/(2G)$. The latter has a covering number bounded by $\left(\frac{6BG}{\epsilon_2}\right)^d \leq \left(\frac{6D}{\epsilon_2}\right)^d$[5]. ■

5.6 Experiment

We conduct experiments to testify the effectiveness of ENIAC. Specifically, we aim to show that

[5]The covering number of Euclidean balls can be easily found in literature.

1. ENIAC is competent to solve RL problem which requires exploration.
2. Compared with PC-PG which uses linear feature for bonus design, the idea of width in ENIAC performs better when using complex neural networks.

Our code is available at https://github.com/FlorenceFeng/ENIAC.

5.6.1 Implementation of ENIAC

We implement ENIAC using PPO [SWD17] as the policy update routine and use fully-connected neural networks (FCNN) to parameterize actors and critics. At the beginning of each epoch $n \in [N]$, as in Algorithm 9, we add a policy π^n into the cover set, generate visitation samples following π^n, update the replay buffer to $\mathcal{Z}^n$, and compute an approximate width function $w^n : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Recall that $w^n(s,a)$ is rigorously defined as:

$$\max_{f,f' \in \mathcal{F}} f(s,a) - f'(s,a), \text{ subject to } \|f - f'\|_{\mathcal{Z}^n} \leq \epsilon. \tag{5.37}$$

To approximate this value in a more stable and efficient manner, we make several revisions to (5.37):

1. instead of training both f and f', we fix f' and only train f;
2. due to 1., we change the objective from $f - f'$ to $(f - f')^2$ for symmetry;
3. instead of always retraining f for every query point (s,a), we gather a batch of query points $\mathcal{Z}_Q^n$ and train in a finite-sum formulation.

Specifically, we initialize f as a neural network with the same structure as the critic network (possibly different weights and biases) and initialize f' as a copy of f. Then we fix f' and train f by maximizing the following loss:

$$\sum_{(s,a)\in\mathcal{Z}_Q^n} \frac{\lambda\big(f(s,a) - f'(s,a)\big)^2}{|\mathcal{Z}_Q^n|} - \sum_{(s',a')\in\mathcal{Z}^n} \frac{\big(f(s',a') - f'(s',a')\big)^2}{|\mathcal{Z}^n|} - \sum_{(s,a)\in\mathcal{Z}_Q^n} \frac{\lambda_1\big(f(s,a) - f'(s,a)\big)}{|\mathcal{Z}_Q^n|}, \tag{5.38}$$

where the last term is added to avoid a zero gradient (since f and f' are identical initially). We generate $\mathcal{Z}_Q^n$ by using the current policy-cover as the initial distribution then rolling out with π^n. (5.38) can be roughly regarded as a Lagrangian form of (5.37) with regularization. The intuition is that we want the functions to be close on frequently visited area (the second term) and to be as far as possible on the query part (the first term). If a query point is away from the frequently visited region, then the constraint is loose and the difference between f and f' can be enlarged and a big bonus is granted; otherwise, the constraint becomes a dominant force and the width is fairly small. After training for several steps of stochastic gradient descent, we freeze both f and f' and return $|f(s,a) - f'(s,a)|$ as $w^n(s,a)$. During the experiments, we set bonus as $0.5 \cdot \frac{w^n(s,a)}{\max_{\mathcal{Z}_Q^n} w^n}$ without thresholding for the later actor-critic steps. The width training process is presented in Algorithm 11. To stabilize training, for each iteration we sample a minibatch $\mathcal{D}_Q$ from the query batch, then run several steps of stochastic gradient descent with changing minibatches on $\mathcal{Z}^n$ while fixing $\mathcal{D}_Q$. The hyperparameters for width training are listed in Table 5.1.

We remark that in practice width training can be fairly flexible and customized for different environments. For example, one can design alternative loss functions as long as they follow the intuition of width; f and f' can be initialized with different weights and the loss function plays a pulling-in role instead of a stretching-out force as in our implementation; $\mathcal{Z}_Q^n$ can be generated with various distributions as long as it has a relatively wide cover to ensure the quality of a batch-trained width.

5.6.2 Environment and Baselines

We test on a continuous control task which requires exploration: continuous control MountainCar[6] from OpenAI Gym [BCP16]. This environment has a 2-dimensional continuous state space and a 1-dimensional continuous action space $[-1, 1]$. The agent only receives a

[6]https://gym.openai.com/envs/MountainCarContinuous-v0/

Algorithm 11 Width Training in ENIAC

Input: Replay buffer $\mathcal{Z}^n$, query batch $\mathcal{Z}_Q^n$.
Initialize f with the same network structure as the critic.
Copy f' as f and fix f' during training.
for $i = 1$ **to** I **do**
    Sample a minibatch $\mathcal{D}_Q$ from $\mathcal{Z}_Q^n$
    for $j = 1$ **to** J **do**
        Sample a minibatch $\mathcal{D}_j$ from $\mathcal{Z}^n$
        Do one step of gradient descent on f with loss in Equation (5.38) and $\mathcal{D}_Q$ and $\mathcal{D}_j$.
    end for
end for
Output: $w^n := |f - f'|$

Table 5.1: ENIAC Width Training Hyperparameters

Hyperparameter	2-layer	4-layer	6-layer
λ	0.1	0.1	0.1
λ_1	0.01	0.01	0.01
$\lvert Z_Q \rvert$	20000	20000	20000
Learning Rate	0.001	0.001	0.0015
$\lvert \mathcal{D}_j \rvert$	160	160	160
$\lvert \mathcal{D}_Q \rvert$	20	20	10
Gradient Clippling	5.0	5.0	5.0
I	1000	1000	1000
J	10	10	10

large reward (+100) if it can reach the top of the hill and small negative rewards for any action. A locally optimal policy is to do nothing and avoid action costs. The length of horizon

is 100 and $\gamma = 0.99$.

We compare five algorithms: ENIAC, vanilla PPO, PPO-RND, PC-PG, and ZERO. All algorithms use PPO as their policy update routine and the same FCNN for actors and critics. The vanilla PPO has no bonus; PPO-RND uses RND bonus [BES19] throughout training; PC-PG iteratively constructs policy cover and uses linear features (kernel-based) to compute bonus as in the implementation of [AHK20], which we follow here; ZERO uses policy cover as in PC-PG and the bonus is all-zero. For ENIAC, PC-PG, and ZERO, instead of adding bonuses to extrinsic rewards, we directly take the larger ones, i.e., the agent receives $\max(r, b)$ during exploration[7]. In ENIAC, we use uniform distribution to select policy from the cover set, i.e., $\rho^n_{\text{cov}} = \text{Unif}(d^{\pi^1}, \ldots, d^{\pi^n})$ as in the main algorithm; PC-PG optimizes the selection distribution based on the policy coverage (see [AHK20] for more details). All algorithms were based on the PPO implementation of [Sha18]. The network structure is described in the main body and the last layer outputs the parameters of a 1D Gaussian for action selection. For PC-PG, we follow the same implementation as mentioned in [AHK20]; for PPO-RND, the RND network has the same architecture as the policy network, except that the last linear layer mapping hidden units to actions is removed. We found that tuning the intrinsic reward coefficient was important for getting good performance for RND. The hyperparameters for optimization are listed in Table 5.2 and 5.3.

5.6.3 Results

We evaluate all the methods for varying depths of the critic network: 2-layer stands for (64, 64) hidden units, 4-layer for (64, 128, 128, 64), and 6-layer for (64, 64, 128, 128, 64, 64). Layers are connected with ReLU non-linearities for all networks. In Figure 5.1, we see that ENIAC robustly achieves high performance consistently in all cases. Both PC-PG and ZERO perform well for depth 2, but as we increase the depth, the heuristic kernel-based bonus

[7]This is simply for implementation convenience and does not change the algorithm. One can also adjust bonus as $\max(r, b) - r$.

Table 5.2: ENIAC/PC-PG Optimization Hyperparameters

Hyperparameter	Values Considered	2-layer	4-layer	6-layer
Learning Rate	$e^{-3}, 5e^{-4}, e^{-4}$	$5e^{-4}$	$5e^{-4}$	$5e^{-4}$
τ_{GAE}	0.95	0.95	0.95	0.95
Gradient Clippling	0.5, 1, 2, 5	5.0	5.0	5.0
Entropy Bonus	0.01	0.01	0.01	0.01
PPO Ratio Clip	0.2	0.2	0.2	0.2
PPO Minibatch	160	160	160	160
PPO Optimization Epochs	5	5	5	5
ϵ-greedy sampling	0, 0.01, 0.05	0.05	0.05	0.05

Table 5.3: PPO-RND Hyperparameters

Hyperparameter	Values Considered	2-layer	4-layer	6-layer
Learning Rate	$e^{-3}, 5e^{-4}, e^{-4}$	e^{-4}	e^{-4}	e^{-4}
τ_{GAE}	0.95	0.95	0.95	0.95
Gradient Clippling	5.0	5.0	5.0	5.0
Entropy Bonus	0.01	0.01	0.01	0.01
PPO Ratio Clip	0.2	0.2	0.2	0.2
PPO Minibatch	160	160	160	160
PPO Optimization Epochs	5	5	5	5
Intrinsic Reward Normalization	true, false	false	false	false
Intrinsic Reward Coefficient	0.5, 1, $e, e^2, e^3, 5e^3, e^4$	$5e^3$	e^3	e^3

and the 0-offset bonus do not provide a good representation of the critic's uncertainty and its learning gets increasingly slower and unreliable. PPO and PPO-RND perform poorly, consistent with the results of [AHK20]. One can also regard the excess layers as masks on the true states and turn them into high-dimensional observations. When observations

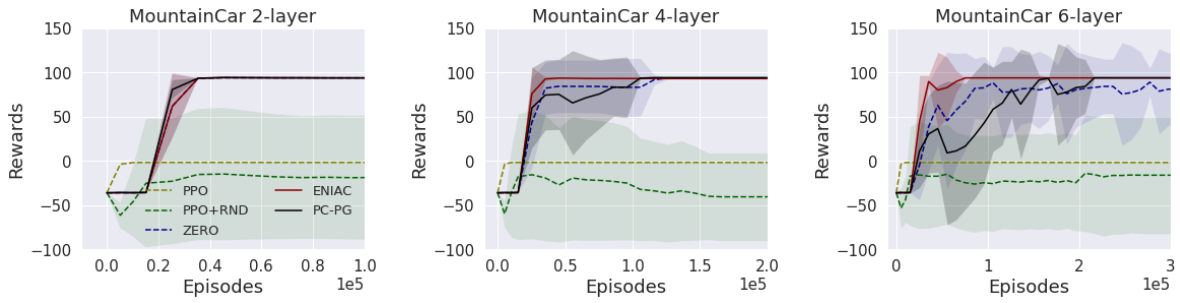

Figure 5.1: Performance of different methods on MountainCar as we vary the netural network depth. The performances are evaluated over 10 random seeds where lines are means and shades represent standard deviations. We stop training once the policy can obtain rewards>93.

become increasingly complicated, more non-linearity is required for information processing and ENIAC is a more appealing choice.

We visualize ENIAC's policies in Figure 5.2, where we plot the state visitations of the exploration policies from the cover, as well as the exploitation policies trained using the cover with just the external reward, for varying number of epochs. We see that ENIAC quickly attains exploration in the vicinity of the optimal state, allowing the exploitation policy to become optimal. Since the bonus in our experiments is smaller than the maximum reward, the effect of the bonus dissipates once we reach the optimal state, even for the exploration policies. We also visualize typical landscapes of bonus functions in ENIAC and PC-PG in Figure 5.3. Both bonuses grant small values on frequently visited area and large values on scarsely visited part. But the bonus in ENIAC changes in a smoother way than the one in PC-PG. This might inspire future study on the shaping of bonuses.

The results testify the competence of ENIAC on the exploration problem. Especially, compared with PC-PG, the usage of width is more suitable for complex function approximation.

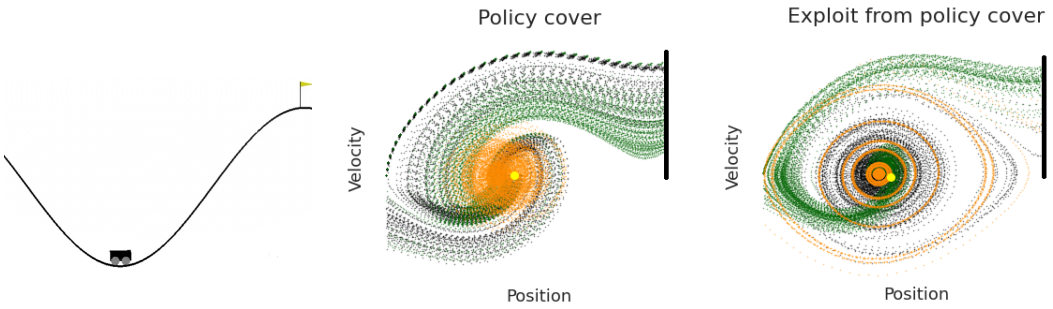

Figure 5.2: The MountainCar environment (left). Trajectories of exploration (middle) and exploitation (right) policies of ENIAC, with colors denoting different epochs: orange for the first policy in the cover set, black for the second, and green for the third. Agent starts from the centric area (near the yellow circle) and the black vertical line on the right represents goal positions.

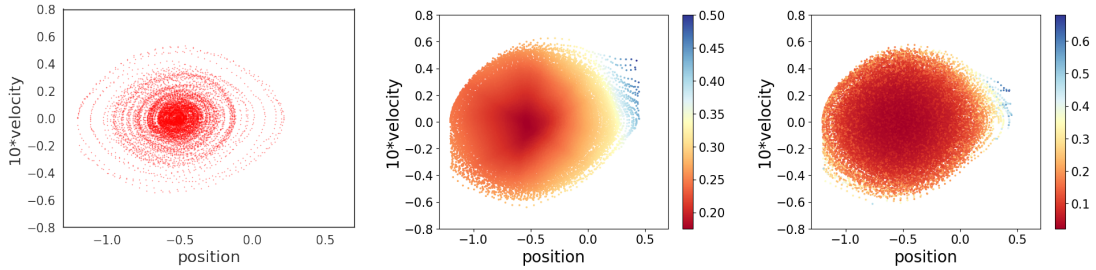

Figure 5.3: Bonus function comparison. [Left]: The trajectories of a chosen policy . [Middle]: the bonus function built by ENIAC upon the policy. [Right]: The bonus built by PC-PG upon the policy. See text for details.

5.7 Conclusion

In this chapter, we present the first set of policy-based techniques for RL with non-linear function approximation. Our methods provide interesting tradeoffs between sample and computational complexities, while also inspire an extremely practical implementation. Empirically, our results demonstrate the benefit of correctly reasoning about the learner's uncertainty under a non-linear function class, while prior heuristics based on linear function approximation

fail to robustly work as we vary the function class. Overall, our results open several interesting avenues of investigation for both theoretical and empirical progress. In theory, it is quite likely that our sample complexity results have scope for a significant improvement. A key challenge here is to enable better sample reuse, typically done with bootstrapping techniques for off-policy learning, while preserving the robustness to model misspecification that our theory exhibits. Empirically, it would be worthwhile to scale these methods to complex state and action spaces such as image-based inputs, and evaluate them on more challenging exploration tasks with a longer effective horizon.

CHAPTER 6

Conclusion and Future Research

In this thesis, we investigate how to achieve efficient RL in various training environments. A series of answers are provided to four settings ranging from the most idealized to the fairly realistic. We focus on improving both statistical and computational efficiency. In the first setting, given an ideal sample oracle, we are interested in how fast we can achieve computationally. We adopt the widely-used asynchronous parallel technique to accelerate training. With careful design, our algorithm enjoys an apparent speedup with a minor increase in sample complexity. This is the first result that theoretically justifies the use of parallel training in RL with an explicit sample complexity result. In the second setting, we answer a fundamental question: how much an approximate (under TV-distance) model can help. With scrutinized analysis, we provide both upper and lower sample complexity bounds to showcase the benefit and the limitation of approximately correct prior knowledge. This is the first systematic answer towards the aforementioned question. For the last two settings, we take efforts to resolve the central challenge in RL: how to achieve efficient exploration with rich observations. We develop two methods: 1. if the observation space has some low-dimensional structure, we use unsupervised learning to abstract it and scale down the problem size for tabular exploration; 2. if no structure is assumed, we use function approximation for its generalization ability to achieve exploration in high-dimensional space. Both of our algorithms are provably efficient and perform pleasantly in preliminary experiments.

These works all together form a systematic answer towards how to achieve efficient reinforcement learning in different environments with both solid theories and practical

algorithms.

For future research, we can take one step further into realistic training by considering offline RL with a batch of data. In this case, no online simulation is available and the information is very limited. If we can successfully conduct RL under this scenario, it can be more widely applied to real-world applications. Some other interesting topics include: safety RL, where hard constraints are imposed to avoid hazardous states and actions; RL with delayed feedback, where observation and reward signals might arrive with time delay and the agent needs to take actions before seeing the true signals; also we can consider applying RL in other mathematical fields such as optimization.

REFERENCES

[ABA18] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar. "Efficient Exploration Through Bayesian Deep Q-Networks." In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9, Feb 2018.

[ABB18] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. "Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments." In *International Conference on Learning Representations*, 2018.

[ABB19] Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. "POLITEX: Regret bounds for policy iteration using expert prediction." In *International Conference on Machine Learning*, pp. 3692–3702. PMLR, 2019.

[AER14] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. "Online multi-task learning for policy gradient methods." In *International Conference on Machine Learning*, pp. 1206–1214, 2014.

[AGK12] Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J. Kappen. "Dynamic Policy Programming." *J. Mach. Learn. Res.*, **13**(1), November 2012.

[AHK20] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. "PC-PG: Policy Cover Directed Exploration for Provable Policy Gradient Learning." In *Advances in Neural Information Processing Systems*, 2020.

[AJ17] Shipra Agrawal and Randy Jia. "Optimistic posterior sampling for reinforcement learning: Worst-case regret bounds." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1184–1194. Curran Associates Inc., 2017.

[AJG18] David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. "Policy and value transfer in lifelong reinforcement learning." In *International Conference on Machine Learning*, pp. 20–29, 2018.

[AJS20] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F. Yang. "Model-Based Reinforcement Learning with Value-Targeted Regression." In *Machine Learning: Proceedings of the Thirty-seventh International Conference (ICML'2020)*, 2020.

[AK01] Sanjeev Arora and Ravi Kannan. "Learning mixtures of arbitrary Gaussians." In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 247–257, 2001.

[AKK20] Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. "Flambe: Structural complexity and representation learning of low rank MDPs." In *Advances in Neural Information Processing Systems*, 2020.

[AKL20a] Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. "On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift.", 2020.

[AKL20b] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. "Optimality and approximation with policy gradient methods in Markov decision processes." In *Conference on Learning Theory*, pp. 64–66. PMLR, 2020.

[AKY19] Alekh Agarwal, Sham Kakade, and Lin F Yang. "On the Optimality of Sparse Model-Based Planning for Markov Decision Processes." *arXiv preprint arXiv:1906.03804*, 2019.

[AM05] Dimitris Achlioptas and Frank McSherry. "On spectral learning of mixtures of distributions." In *International Conference on Computational Learning Theory*, pp. 458–469. Springer, 2005.

[AMG11] Mohammad Gheshlaghi Azar, Remi Munos, Mohammad Ghavamzadeh, and Hilbert Kappen. "Speedy Q-learning." In *Advances in neural information processing systems*, 2011.

[AMK13] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. "Minimax PAC bounds on the sample complexity of reinforcement lear ning with a generative model." *Machine learning*, **91**(3):325–349, 2013.

[AOM17] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. "Minimax Regret Bounds for Reinforcement Learning." In *International Conference on Machine Learning*, pp. 263–272, 2017.

[ASM08] 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*, **71**(1):89–129, 2008.

[AWR17] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. "Hindsight Experience Replay." In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pp. 5048–5058. Curran Associates, Inc., 2017.

[BBC19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris

Hesse, et al. "Dota 2 with large scale deep reinforcement learning." *arXiv preprint arXiv:1912.06680*, 2019.

[BCP16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI gym." *arXiv preprint arXiv:1606.01540*, 2016.

[BES19] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. "Exploration by random network distillation." In *International Conference on Learning Representations*, 2019.

[BF20] Nizar Bouguila and Wentao Fan. *Mixture models and applications*. Springer, 2020.

[BFT16] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. "Reinforcement learning through asynchronous advantage actor-critic on a gpu." *arXiv preprint arXiv:1611.06256*, 2016.

[BKS04] 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*, pp. 831–838, 2004.

[BL13] Emma Brunskill and Lihong Li. "Sample complexity of multi-task reinforcement learning." In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 122–131. AUAI Press, 2013.

[BL14] Emma Brunskill and Lihong Li. "PAC-inspired option discovery in lifelong reinforcement learning." In *International conference on machine learning*, pp. 316–324, 2014.

[BR19] Jalaj Bhandari and Daniel Russo. "Global Optimality Guarantees For Policy Gradient Methods." *CoRR*, **abs/1906.01786**, 2019.

[BSO16] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. "Unifying count-based exploration and intrinsic motivation." In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

[BT89] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[BT91] Dimitri P Bertsekas and John N Tsitsiklis. "Some aspects of parallel and distributed iterative algorithms: a survey." *Automatica*, **27**(1):3–21, 1991.

[BT02] Ronen I Brafman and Moshe Tennenholtz. "R-max-a general polynomial time algorithm for near-optimal reinforcement learning." *Journal of Machine Learning Research*, **3**(Oct):213–231, 2002.

[BT13] Dominique Bontemps and Wilson Toussile. "Clustering and variable selection for categorical multivariate data." *Electronic Journal of Statistics*, **7**:2344–2371, 2013.

[Cha02] Moses S Charikar. "Similarity estimation techniques from rounding algorithms." In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.

[CJ19] Jinglin Chen and Nan Jiang. "Information-Theoretic Considerations in Batch Reinforcement Learning." In *International Conference on Machine Learning*, pp. 1042–1051, 2019.

[CLR14] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. "Sparse multi-task reinforcement learning." In *Advances in Neural Information Processing Systems*, pp. 819–827, 2014.

[CYJ20] Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. "Provably Efficient Exploration in Policy Optimization." In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2020.

[CYS21] Qi Cai, Zhuoran Yang, Csaba Szepesvari, and Zhaoran Wang. "Optimistic Policy Optimization with General Function Approximations.", 2021.

[Dah06] David B Dahl. "Model-based clustering for expression data via a Dirichlet process mixture model." *Bayesian inference for gene expression and proteomics*, **4**:201–218, 2006.

[DBK16] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. "Deep direct reinforcement learning for financial signal representation and trading." *IEEE transactions on neural networks and learning systems*, **28**(3):653–664, 2016.

[DJK18] 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 31*, 2018.

[DKJ19] Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. "Provably efficient RL with Rich Observations via Latent State Decoding." In *International Conference on Machine Learning*, pp. 1665–1674, 2019.

[DKW20] Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. "Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning?" In *International Conference on Learning Representations*, 2020.

[DLB17] Christoph Dann, Tor Lattimore, and Emma Brunskill. "Unifying PAC and Regret: Uniform PAC Bounds for Episodic Reinforcement Learning." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5717–5727, USA, 2017. Curran Associates Inc.

[DLM20] Simon S Du, Jason D Lee, Gaurav Mahajan, and Ruosong Wang. "Agnostic Q-learning with Function Approximation in Deterministic Systems: Near-Optimal Bounds on Approximation Error and Sample Complexity." *Advances in Neural Information Processing Systems*, **33**, 2020.

[DLW19] Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. "Provably efficient Q-learning with function approximation via distribution shift error checking oracle." In *Advances in Neural Information Processing Systems*, pp. 8058–8068, 2019.

[DRC17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator." In *Conference on Robot Learning*, pp. 1–16, 2017.

[DS00] Sanjoy Dasgupta and Leonard J Schulman. "A two-round variant of EM for Gaussian mixtures." In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 152–159, 2000.

[Dur19] Rick Durrett. *Probability: Theory and examples*, volume 49. Cambridge university press, 2019.

[EKM09] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. "Online Markov decision processes." *Mathematics of Operations Research*, **34**(3):726–736, 2009.

[EM03] Eyal Even-Dar and Yishay Mansour. "Learning rates for Q-learning." *Journal of Machine Learning Research*, **5**(Dec):1–25, 2003.

[ERR19] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. "A guide to deep learning in healthcare." *Nature medicine*, **25**(1):24–29, 2019.

[EV13] Ehsan Elhamifar and Rene Vidal. "Sparse subspace clustering: Algorithm, theory, and applications." *IEEE transactions on pattern analysis and machine intelligence*, **35**(11):2765–2781, 2013.

[FAD18] Dylan Foster, Alekh Agarwal, Miroslav Dudik, Haipeng Luo, and Robert Schapire. "Practical contextual bandits with regression oracles." In *International Conference on Machine Learning*, pp. 1539–1548. PMLR, 2018.

[FAP18] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. "Noisy Networks For Exploration." In *International Conference on Learning Representations*, 2018.

[FJ14] Hamid Reza Feyzmahdavian and Mikael Johansson. "On the convergence rates of asynchronous iterations." In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 153–159. IEEE, 2014.

[FWX20] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. "A theoretical analysis of deep Q-learning." In *Learning for Dynamics and Control*, pp. 486–489. PMLR, 2020.

[FWY20] Fei Feng, Ruosong Wang, Wotao Yin, Simon S Du, and Lin Yang. "Provably Efficient Exploration for Reinforcement Learning Using Unsupervised Learning." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 22492–22504. Curran Associates, Inc., 2020.

[FYA21] Fei Feng, Wotao Yin, Alekh Agarwal, and Lin F Yang. "Provably Correct Optimization and Exploration with Non-linear Policies." *arXiv preprint arXiv:2103.11559*, 2021.

[FYY19] Fei Feng, Wotao Yin, and Lin F Yang. "Does knowledge transfer always help to learn a better policy." *arXiv preprint arXiv:1912.02986*, 2019.

[GHL17] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

[GK08] Matthew Grounds and Daniel Kudenko. "Parallel reinforcement learning with linear function approximation." In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, pp. 60–74. Springer, 2008.

[GML18] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. "Meta-reinforcement learning of structured exploration strategies." In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

[GSP19] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. "A Theory of Regularized Markov Decision Processes." In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[Hoe63] Wassily Hoeffding. "Probability inequalities for sums of bounded random variables." *Journal of the American statistical association*, **58**(301):13–30, 1963.

[HY17] Robert Hannah and Wotao Yin. "More Iterations per Second, Same Quality–Why Asynchronous Algorithms may Drastically Outperform Traditional Ones." *arXiv preprint arXiv:1708.05136*, 2017.

[HY18] Robert Hannah and Wotao Yin. "On Unbounded Delays in Asynchronous Parallel Fixed-Point Algorithms." *Journal of Scientific Computing*, **76**(1):299–326, 2018.

[HZA18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

[Its95] Noda Itsuki. "Soccer server: A simulator for RoboCup." In *JSAI AI-Symposium 95: Special Session on RoboCup*. Citeseer, 1995.

[JAB18] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. "Is Q-learning provably efficient?" In *Advances in neural information processing systems*, pp. 4863–4873, 2018.

[Jia18] Nan Jiang. "PAC Reinforcement Learning With an Imperfect Model." In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 3334–3341, 2018.

[JKA17] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. "Contextual decision processes with low Bellman rank are PAC-learnable." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1704–1713. JMLR. org, 2017.

[JOA10] Thomas Jaksch, Ronald Ortner, and Peter Auer. "Near-optimal Regret Bounds for Reinforcement Learning." *Journal of Machine Learning Research*, **11**(4), 2010.

[JV02] Alfons Juan and Enrique Vidal. "On the use of Bernoulli mixture models for text classification." *Pattern Recognition*, **35**(12):2705–2710, 2002.

[JV04] Alfons Juan and Enrique Vidal. "Bernoulli mixture models for binary images." In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pp. 367–370. IEEE, 2004.

[JYW20a] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. "Provably efficient reinforcement learning with linear function approximation." In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

[JYW20b] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. "Provably efficient reinforcement learning with linear function approximation." In *Proceedings of Thirty Third Conference on Learning Theory*, Proceedings of Machine Learning Research, 2020.

[Kak01] Sham M Kakade. "A natural policy gradient." *Advances in neural information processing systems*, **14**, 2001.

[Kak03] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of College London, 2003.

[KAL16] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. "PAC reinforcement learning with rich observations." In *Advances in Neural Information Processing Systems*, pp. 1840–1848, 2016.

[KBJ14] Dileep Kalathil, Vivek S Borkar, and Rahul Jain. "Empirical Q-Value Iteration." *arXiv preprint arXiv:1412.0180*, 2014.

[KBP13] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." *The International Journal of Robotics Research*, p. 0278364913495721, 2013.

[KL02] Sham Kakade and John Langford. "Approximately optimal approximate reinforcement learning." In *ICML*, volume 2, pp. 267–274, 2002.

[KM15] Michael R Kosorok and Erica EM Moodie. *Adaptive TreatmentStrategies in Practice: Planning Trials and Analyzing Data for Personalized Medicine*, volume 21. SIAM, 2015.

[KMN02] Michael Kearns, Yishay Mansour, and Andrew Y Ng. "A sparse sampling algorithm for near-optimal planning in large Markov decision processes." *Machine Learning*, **49**(2-3):193–208, 2002.

[KS02a] Michael Kearns and Satinder Singh. "Near-optimal reinforcement learning in polynomial time." *Machine Learning*, **49**(2-3):209–232, 2002.

[KS02b] Michael Kearns and Satinder Singh. "Near-optimal reinforcement learning in polynomial time." *Machine learning*, **49**(2-3):209–232, 2002.

[KT00] Vijay R Konda and John N Tsitsiklis. "Actor-critic algorithms." In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

[KWY18] Sham Kakade, Mengdi Wang, and Lin F Yang. "Variance reduction methods for sublinear reinforcement learning." *arXiv preprint arXiv:1802.09184*, 2018.

[Laz12] Alessandro Lazaric. "Transfer in reinforcement learning: A framework and a survey." In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

[LG10] Alessandro Lazaric and Mohammad Ghavamzadeh. "Bayesian Multi-Task Reinforcement Learning." In *ICML - 27th International Conference on Machine Learning*, pp. 599–606, Haifa, Israel, June 2010. Omnipress.

[Li17] Yuxi Li. "Deep reinforcement learning: An overview." *arXiv preprint arXiv:1701.07274*, 2017.

[LLG18] Zachary Chase Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. "BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems." In *AAAI*, 2018.

[LLW11] Lihong Li, Michael L Littman, Thomas J Walsh, and Alexander L Strehl. "Knows what it knows: A framework for self-aware learning." *Machine learning*, **82**(3):399–443, 2011.

[LZ06] Jia Li and Hongyuan Zha. "Two-way Poisson mixture models for simultaneous document classification and word clustering." *Computational Statistics & Data Analysis*, **50**(1):163–180, 2006.

[MB88] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York, 1988.

[MBM16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[MC12] Timothy Arthur Mann and Yoonsuck Choe. "Directed Exploration in Reinforcement Learning with Transferred Knowledge." In *Ewrl*, pp. 59–76, 2012.

[MHK20] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. "Kinematic state abstraction and provably efficient rich-observation reinforcement learning." In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

[MJT19] Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. "Sample Complexity of Reinforcement Learning using Linearly Combined Model Ensembles." *arXiv preprint arXiv:1910.10597*, 2019.

[MKS15] V Mnih, K Kavukcuoglu, D Silver, A. A. Rusu, J Veness, M. G. Bellemare, A Graves, M Riedmiller, A. K. Fidjeland, and G Ostrovski. "Human-level control through deep reinforcement learning." *Nature*, **518**(7540):529, 2015.

[MP04] Geoffrey J McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[MT04] Shie Mannor and John N Tsitsiklis. "The sample complexity of exploration in the multi-armed bandit problem." *Journal of Machine Learning Research*, **5**(Jun):623–648, 2004.

[Mun05] Rémi Munos. "Error bounds for approximate value iteration." In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, p. 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[NCD06] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. "Autonomous inverted helicopter flight via reinforcement learning." In *Experimental robotics IX*, pp. 363–372. Springer, 2006.

[NMR20] Amir Najafi, Seyed Abolfazl Motahari, and Hamid R Rabiee. "Reliable clustering of Bernoulli mixture models." *Bernoulli*, **26**(2):1535–1559, 2020.

[NSB15] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. "Massively parallel methods for deep reinforcement learning." *arXiv preprint arXiv:1507.04296*, 2015.

[NYW19] C. Ni, L. F. Yang, and M. Wang. "Learning to Control in Metric Space with Optimal Regret." In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 726–733, 2019.

[OVW16] Ian Osband, Benjamin Van Roy, and Zheng Wen. "Generalization and Exploration via Randomized Value Functions." In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2377–2386. JMLR.org, 2016.

[PAE17] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. "Curiosity-driven Exploration by Self-supervised Prediction." In *ICML*, 2017.

[PCS18] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntak Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. "Agile Autonomous Driving using End-to-End Deep Imitation Learning." In *Robotics: science and systems*, 2018.

[PP13] Jason Pazis and Ronald Parr. "PAC Optimal Exploration in Continuous Space Markov Decision Processes." In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pp. 774–781. AAAI Press, 2013.

[PS08] Jan Peters and Stefan Schaal. "Natural Actor-Critic." *Neurocomput.*, **71**(7-9):1180–1190, 2008.

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[PXY16] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. "ARock: an algorithmic framework for asynchronous parallel coordinate updates." *SIAM Journal on Scientific Computing*, **38**(5):A2851–A2879, 2016.

[RV13] Daniel Russo and Benjamin Van Roy. "Eluder dimension and the sample complexity of optimistic exploration." *Advances in Neural Information Processing Systems*, **26**:2256–2264, 2013.

[RV17] Oded Regev and Aravindan Vijayaraghavan. "On learning mixtures of well-separated Gaussians." In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 85–96. IEEE, 2017.

[RZF19] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables." In *International Conference on Machine Learning*, pp. 5331–5340, 2019.

[SA18] Adam Stooke and Pieter Abbeel. "Accelerated methods for deep reinforcement learning." *arXiv preprint arXiv:1803.02811*, 2018.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SBW92] Richard S Sutton, Andrew G Barto, and Ronald J Williams. "Reinforcement learning is direct adaptive optimal control." *IEEE Control Systems Magazine*, **12**(2):19–22, 1992.

[SD03] Nicolas Schweighofer and Kenji Doya. "Meta-learning in reinforcement learning." *Neural Networks*, **16**(1):5–9, 2003.

[SEC14] Mahdi Soltanolkotabi, Ehsan Elhamifar, Emmanuel J Candes, et al. "Robust subspace clustering." *The Annals of Statistics*, **42**(2):669–699, 2014.

[SER20] Lior Shani, Yonathan Efroni, Aviv Rosenberg, and Shie Mannor. "Optimistic policy optimization with bandit feedback." In *International Conference on Machine Learning*, pp. 8604–8613. PMLR, 2020.

[SG14] Bruno Scherrer and Matthieu Geist. "Local policy search in a convex space and conservative policy iteration as boosted policy search." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 35–50. Springer, 2014.

[Sha18] Zhang Shangtong. "Modularized implementation of deep RL algorithms in pytorch.", 2018.

[SHD18] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. "Meta reinforcement learning with latent variable gaussian processes." *arXiv preprint arXiv:1803.07551*, 2018.

[SHM16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature*, **529**(7587):484–489, 2016.

[SJ19] Max Simchowitz and Kevin G Jamieson. "Non-asymptotic gap-dependent regret bounds for tabular MDPs." In *Advances in Neural Information Processing Systems*, pp. 1151–1160, 2019.

[SJK19] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. "Model-based RL in Contextual Decision Processes: PAC bounds and Exponential Improvements over Model-free Approaches." In *Conference on Learning Theory*, pp. 2898–2933, 2019.

[SLA15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization." In *International conference on machine learning*, pp. 1889–1897, 2015.

[SLW06] 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*, pp. 881–888, 2006.

[SMS99] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation." In *Advances in Neural Information Processing Systems*, volume 99, pp. 1057–1063, 1999.

[SS19] Zhao Song and Wen Sun. "Efficient Model-free Reinforcement Learning in Metric Spaces." *arXiv preprint arXiv:1905.00475*, 2019.

[SW01] Jennie Si and Yu-Tsung Wang. "Online learning control by association and reinforcement." *IEEE Transactions on Neural networks*, **12**(2):264–276, 2001.

[SWD17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347*, 2017.

[SWW18a] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. "Near-Optimal Time and Sample Complexities for Solving Markov Decision Processes with a Generative Model." In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pp. 5186–5196. Curran Associates, Inc., 2018.

[SWW18b] Aaron Sidford, Mengdi Wang, Xian Wu, Lin F Yang, and Yinyu Ye. "Near-Optimal Time and Sample Complexities for Solving Discounted Markov Decision Process with a Generative Model." *arXiv preprint arXiv:1806.01492*, 2018.

[SWW18c] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. "Variance reduced value iteration and faster algorithms for solving markov decision processes." In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 770–787. Society for Industrial and Applied Mathematics, 2018.

[THF17] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. "# Exploration: A study of count-based exploration for deep reinforcement learning." In *Advances in Neural Information Processing Systems*, pp. 2753–2762, 2017.

[TS09] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, **10**(Jul):1633–1685, 2009.

[Tsi94] John N. Tsitsiklis. "Asynchronous stochastic approximation and Q-learning." *Machine Learning*, **16**(3):185–202, Sep 1994.

[TY97] Fumihide Tanaka and Masayuki Yamamura. "An approach to lifelong reinforcement learning through multiple environments." In *6th European Workshop on Learning Robots*, pp. 93–99, 1997.

[Val84] Leslie G Valiant. "A theory of the learnable." *Communications of the ACM*, **27**(11):1134–1142, 1984.

[Van96] R. Vanderbei. "Sailing Strategies, An Application Involving Stochastics, Optimization, and Statistics.", 1996.

[Vid11] René Vidal. "Subspace clustering." *IEEE Signal Processing Magazine*, **28**(2):52–68, 2011.

[VW04] Santosh Vempala and Grant Wang. "A spectral algorithm for learning mixture models." *Journal of Computer and System Sciences*, **68**(4):841–860, 2004.

[Wan17] Mengdi Wang. "Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear (Sometimes Sublinear) Running Time." *arXiv preprint arXiv:1704.01869*, 2017.

[WD92] Christopher JCH Watkins and Peter Dayan. "Q-learning." *Machine learning*, **8**(3-4):279–292, 1992.

[WFR07] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. "Multi-task reinforcement learning: A hierarchical Bayesian approach." In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022, 2007.

[Wie00] MA Wiering. "Multi-agent reinforcement learning for traffic light control." In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151–1158, 2000.

[Wil92] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning*, **8**(3-4):229–256, 1992.

[WSW15] Tim Wallace, Ali Sekmen, and Xiaofei Wang. "Application of subspace clustering in DNA sequence analysis." *Journal of Computational Biology*, **22**(10):940–952, 2015.

[WSY20] Ruosong Wang, Russ R Salakhutdinov, and Lin Yang. "Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension." *Advances in Neural Information Processing Systems*, **33**, 2020.

[WV13] Zheng Wen and Benjamin Van Roy. "Efficient exploration and value function generalization in deterministic systems." In *Advances in Neural Information Processing Systems*, pp. 3021–3029, 2013.

[WXL13] Yu-Xiang Wang, Huan Xu, and Chenlei Leng. "Provable subspace clustering: When LRR meets SSC." In *Advances in Neural Information Processing Systems*, pp. 64–72, 2013.

[YHP18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. "Recent trends in deep learning based natural language processing." *ieee Computational intelligenCe magazine*, **13**(3):55–75, 2018.

[YW19a] Lin F Yang and Mengdi Wang. "Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound." *arXiv preprint arXiv:1905.10389*, 2019.

[YW19b] Lin F Yang and Mengdi Wang. "Sample-optimal parametric Q-learning with linear transition models." *arXiv preprint arXiv:1902.04779*, 2019.

[YZD20] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

[ZAT17] Yusen Zhan, Haitham Bou Ammar, and Matthew E Taylor. "Scalable lifelong reinforcement learning." *Pattern Recognition*, **72**:407–418, 2017.

[ZB19] Andrea Zanette and Emma Brunskill. "Tighter Problem-Dependent Regret Bounds in Reinforcement Learning without Domain Knowledge using Value Function Bounds." In *International Conference on Machine Learning*, pp. 7304–7312, 2019.

[ZCT19] Yunzhi Zhang, Ignasi Clavera, Boren Tsai, and Pieter Abbeel. "Asynchronous Methods for Model-Based Reinforcement Learning." *arXiv preprint arXiv:1910.12453*, 2019.

[ZFY20] Yibo Zeng, Fei Feng, and Wotao Yin. "AsyncQVI: Asynchronous-Parallel Q-Value Iteration for Discounted Markov Decision Processes with Near-Optimal Sample Complexity." In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 713–723. PMLR, 26–28 Aug 2020.