Advances in Simulation-Based Search and Batch Reinforcement Learning

by

Chenjun Xiao

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

Doctor of Philosophy

Department of Computing Science

University of Alberta

© Chenjun Xiao, 2022

Abstract

Reinforcement learning (RL) defines a general computational problem where the learner must learn to make good decisions through interactive experience. To be effective in solving this problem, the learner must be able to explore the environment, make accurate predictions about the future, and compute strategic plans. These joint challenges distinguish RL from other machine learning problems. This dissertation considers two sub-topics of RL: Planning and Batch RL.

For planning, we contribute two novel techniques to improve the efficiency of Monte Carlo Tree Search (MCTS): 1) Memory-augmented MCTS incorporates a memory structure into MCTS in order to generate an approximate value estimate that combines the estimate of similar states; 2) a new MCTS algorithm that applies maximum entropy policy optimization to general sequential decision-making.

For batch RL, we offer three analyses towards a better understanding of the theoretical foundations of batch RL: 1) a minimax and instance-dependent analysis of batch policy optimization algorithms; 2) a characterization of the curse of passive data collection in batch RL; and 3) a theoretical analysis of convergence and generalization properties of value prediction algorithms with overparameterized models.

Acknowledgements

Thanks to my supervisors, Martin Müller and Dale Schuurmans, for their support, advice, and guidance. They have been consistent sources of wisdom and inspiration and gave me complete freedom to explore what I found interesting. I'd also like to thank Csaba Szepesvári for his keen insights and many constructive suggestions. I am so lucky having all of them as my friends and teachers. I'd also like to thank Martha White, Adam White and András György for serving in my supervisory committee and providing insightful feedback for my thesis dissertation.

I greatly appreciate Ruitong Huang for his mentorship at Borealis AI, and sincerely thank Lihong Li and Bo Dai for hosting my internships at Google Brain. I am extremely fortunate to have so much fun and productive internship experience during the PhD journey. Thanks to my colleagues in Reinforcement Learning and Artificial Intelligence (RLAI) and Alberta Machine Intelligence Institute (Amii) groups for interesting research discussions and friendship. In particular, I'd like to thank Jincheng Mei and Yifan Wu for research collaboration.

Finally but most importantly, I'd like to give deepest thanks to my family. Thanks to my parents for their love and constant support; and especially to Mengmeng Fan, for her enduring love and enormous depths of patience. The PhD journey has never been a pain with your company. Thank you Mengmeng.

Contents

1 Introduction **1**
1.1 Contributions . . . 2
1.1.1 Publications . . . 5

I Simulation-Based Search **7**

2 Online Planning **8**
2.1 Online Planning . . . 8
2.2 Simulation-based Planning . . . 10
2.2.1 Monte Carlo Simulation . . . 10
2.2.2 Spare Sampling . . . 11
2.2.3 Monte Carlo Tree Search . . . 12

3 Memory-Augmented Monte Carlo Tree Search **15**
3.1 Introduction . . . 15
3.2 Preliminaries . . . 16
3.3 Value Approximation with Memory . . . 17
3.3.1 Analysis for Independent Sub-gaussian Evaluations . . . 18
3.3.2 Analysis for Tree Search . . . 20
3.4 Memory-Augmented MCTS . . . 21
3.4.1 Approximating $\mathbf{w} = f_\tau(-\mathbf{c})$ . . . 22
3.4.2 Memory Operations . . . 22
3.4.3 Integrating Memory with MCTS . . . 24
3.5 Related Work . . . 25
3.6 Experiments . . . 26
3.6.1 Implementation Details . . . 26
3.6.2 Baseline . . . 27
3.6.3 Results . . . 28
3.7 Conclusion . . . 29

4 Maximum Entropy Monte Carlo Planning **30**
4.1 Introduction . . . 30
4.2 Background . . . 31
4.2.1 Maximum Entropy Policy Optimization . . . 32
4.3 Softmax Value Estimation with Stochastic Bandit . . . 33
4.3.1 The Stochastic Softmax Bandit . . . 33
4.3.2 E2W: an Optimal Sequential Sampling Strategy . . . 34
4.4 Maximum Entropy MCTS . . . 35
4.4.1 Algorithmic Design . . . 35
4.4.2 Theoretical Analysis . . . 36
4.5 Related Work . . . 38

4.6 Experiments . . . 39
4.6.1 Results . . . 40
4.7 Conclusion . . . 42

II Batch Reinforcement Learning 44

5 Reinforcement Learning 45
5.1 Markov Decision Process . . . 45
5.2 Value Functions . . . 46
5.3 Effective Planning Horizon . . . 47
5.4 Discounted Occupancy Measure . . . 48
5.5 Batch Reinforcement Learning . . . 48

6 On the Optimality of Batch Policy Optimization Algorithms 50
6.1 Introduction . . . 50
6.2 Preliminaries . . . 51
6.3 Minimax Analysis . . . 53
6.4 Instance-Dependent Analysis . . . 55
6.4.1 Instance-dependent Upper Bound . . . 55
6.4.2 Instance-dependent Lower Bound . . . 62
6.5 A Characterization of Pessimism . . . 66
6.6 Related work . . . 68
6.7 Conclusion . . . 70

7 The Curse of Passive Data Collection in Batch Reinforcement Learning 71
7.1 Introduction . . . 71
7.2 Notation and Background . . . 75
7.3 Batch Policy Optimization . . . 76
7.4 Lower Bounds . . . 79
7.5 Upper Bounds . . . 83
7.6 Related Work . . . 84
7.7 Conclusion . . . 88

8 Understanding and Leveraging Overparameterization in Recursive Value Estimation 90
8.1 Introduction . . . 90
8.2 Related work . . . 92
8.3 Preliminaries . . . 95
8.3.1 Markov reward processes . . . 95
8.3.2 Linear Function Approximation . . . 96
8.3.3 Batch Value Estimation . . . 96
8.3.4 Over vs Underparameterized Features . . . 97
8.3.5 Residual Minimization (RM) . . . 97
8.3.6 Temporal Difference (TD) Learning . . . 98
8.3.7 Fitted Value Iteration (FVI) . . . 98
8.4 Over-Parameterized Linear Value Function Approximation . . . 99
8.4.1 Overparameterized Residual Minimization . . . 100
8.4.2 Overparameterized TD Learning . . . 100
8.4.3 Overparameterized Fitted Value Iteration . . . 102
8.5 Unified View of Overparameterized Value Estimators . . . 103
8.5.1 Value Prediction Error Bounds . . . 104
8.6 Regularizers for Deep Reinforcement Learning Algorithms . . . 106

8.6.1 Empirical Justification of Regularizers 107
8.7 Conclusion 109

9 Conclusions and Future Directions 111

Bibliography 133

Appendix A Proofs for Chapter 3: Memory-augmented MCTS 133
A.1 Proof 133
A.1.1 Notation 133
A.1.2 Flat UCB 134
A.1.3 Result 137

Appendix B Proofs for Chapter 4: Maximum Entropy Monte Carlo Planning 140
B.1 Experimental Details 140
B.2 Proofs 141
B.2.1 Proofs for softmax stochastic bandit 141
B.2.2 Proof of Theorem 3 142
B.2.3 Concentration of $N_t(a)$ in Bandit (Theorem 5) 144
B.2.4 Proof of Theorem 4 146
B.2.5 Technical Lemmas 149
B.3 Proofs for Tree 151
B.3.1 Proof of Theorem 6 152
B.3.2 Proof of Theorem 7 153

Appendix C Proofs for Chapter 6: On the Optimality of Batch Policy Optimization Algorithms 155
C.1 Proof of Minimax Results 155
C.1.1 Proof of Theorem 8 155
C.1.2 Proof of Theorem 9 158
C.2 Proof of Instance-dependent Results 158
C.2.1 Instance-dependent Upper Bound 158
C.2.2 Instance-dependent Lower Bounds 165
C.3 Proof for Section 6.5 168

Appendix D Proofs for Chapter 7: The Curse of Passive Data Collection in Batch ReinforcementLearning 172
D.1 An absolute bound on the state-action probability ratios under the uniform logging policy and the uniform mix of deterministic policies 172
D.2 Lower Bound Proofs 175
D.3 Upper bound proofs 187
D.3.1 Pessimistic Algorithm 195

Appendix E Proofs for Chapter 8: Understanding and Leveraging Overparameterization in Recursive Value Estimation 198
E.1 Proofs 198
E.1.1 Proof of Theorem 17 198
E.1.2 Proof of Theorem 18 200
E.1.3 Proof of Theorem 19 203
E.1.4 Proof of Theorem 20 and Corollary 9 204
E.1.5 Proof of Theorem 21 204
E.1.6 Proof of Corollary 10 206

E.1.7 Concentration of Eigenvalues and Bounding the Orthogonal Complement . . . 207
E.2 Experiment setup . . . 208
E.2.1 Acrobot . . . 209
E.2.2 Reacher . . . 209
E.2.3 Cartpole . . . 209
E.2.4 Pendulum . . . 210
E.2.5 Mujoco . . . 210

List of Figures

3.1 A brief illustration of M-MCTS. When a leaf state s is searched, the feature representation $\phi(s)$ is generated, which is then used to *query* the memory based value approximation $\hat{v}_{\mathcal{M}}(s)$. $\hat{v}_{\mathcal{M}}(s)$ is used to update s and all its ancestors according to Eq. (3.15), as indicated by the red arrows in the figure. 23

3.2 Experimental results. Figure (a)-(c) shows the results of testing different value of M. Figure (d) shows the results of testing different size of memory. In all figures, x-axis is the number of simulations per move, y-axis means the winning rate against the baseline. 28

4.1 Evaluation of softmax value estimation in the synthetic tree environment. The x-axis shows the number of simulations and y-axis shows the value estimation error. The shaded area shows the standard error. We find that the softmax value can be efficiently estimated by MENTS. 40

4.2 Evaluation of online planning in the synthetic tree environment. The x-axis shows the number of simulations and y-axis shows the planning error. The shaded area shows the standard error. We can observe that MENTS enjoys much smaller error than UCT especially in the large domain. 42

6.1 Comparing UCB, LCB and greedy on synthetic problems (with $k = 100$). (a) and (b): Problem instances where LCB has the best performance. The data set is generated by a behavior policy that pulls an arm i with *high* frequency and the other arms uniformly. In (a) i is the best arm while in (b) i is the 10th-best arm. (c) and (d): Problem instances where UCB has the best performance. The data set is generated by a behavior policy that pulls a set of good arms $\{j : j \leq i\}$ with very *small* frequency and the other arms uniformly. In (c) we use $i = 1$ while in (d) we use $i = 10$. Experiment details are provided in the supplementary material. 61

6.2 Comparing UCB, LCB and greedy on synthetic problems. (a) and (b): A set of two-armed bandit instances where both LCB and UCB dominate half of the instances. (c) and (d): For each k, we first sample 100 vectors $\vec{\mu} = [\mu_1, ..., \mu_k]$ and for each $\vec{\mu}$ we uniformly sample 100 (if exist) subsets $S \subset k, |S| = m$ ($m = k/2$ in (c) and $m = k/4$ in (d)), to generate up to 10k instances. We then count the fraction of instances where each algorithm performs better than the other two algorithms among the randomly sampled set of instances. Experiment details are provided in the supplementary material. 62

8.1 Illustrative example showing the spectrum of $\boldsymbol{W}$ with $k = 2$ and $d = 3$. $\boldsymbol{M} = [\phi_1, \phi_2]^\top$. Without loss of generality, we let $\phi_1 = (\cos\tau, \sin\tau, 0)$ and $\phi_2 = (1, 0, 0)$. $\boldsymbol{N} = [\phi_1', \phi_2']^\top$, where $\phi_1' = \phi_2$, $\phi_2' = (-\cos\tau, \frac{\sqrt{2}}{2}\sin\tau, \frac{\sqrt{2}}{2}\sin\tau)$. Then $\boldsymbol{W} = [[0, 1]^\top, [\frac{\sqrt{2}}{2}, -(1 + \frac{\sqrt{2}}{2}\cos\tau)]^\top]$. Clearly, the spectral norm of $\boldsymbol{W}$ increases as the angle τ between ϕ_1 and ϕ_2 decreases. 102

8.2 We show the results with proposed regularization compared to the baseline algorithms. The algorithms are trained using a fixed offline data set collected by random initialized policies. The x-axis shows the training iterations (in thousands) and y-axis shows the performance. All plots are averaged over 100 runs. The shaded area shows the standard error. 108

8.3 We show the results with proposed regularization compared to the baseline algorithms. The x-axis shows the training iterations (in thousands) and y-axis shows the performance. All plots are averaged over 100 runs. The shaded area shows the standard error. 110

D.1 Illustration of the MDPs used in the proof of Theorem 12. For $\varepsilon > 0$, Let $H = H_{\gamma,2\varepsilon}$. The state space consists of two parts $\mathcal{S} = \{s_0, s_1, \ldots, s_H\} \cup \{z\}$, where s_0 is the initial state, z is a self-absorbing state. For any $s \in \mathcal{S}$, let $a_s = \arg\min_a \pi_{\log}(a|s)$ be the action with minimal chance of being selected by $\pi_{\log}$, and $\mathcal{A}_s = \mathcal{A} \setminus \{a_s\}$. The transitions and rewards are as follows: State z is absorbing under any action. For $i \in \{0, \ldots, H-1\}$, at state s_i under action a_{s_i} the MDP transits to s_{i+1}, while it transits to z under any other actions. From s_H, the next state is also z under any action. The rewards are deterministically zero for any state-action pair except when the state is s_H, and action a_{S_H} is taken, when it is random with either a positive or negative mean. 176

D.2 Illustration of the MDPs used in the proof of Theorem 15. The initial distribution μ concentrates on state $\{s_0\}$. The pair (s', a') is the one where $\mu_{\log}$ takes on the smallest value (which is below $1/(\mathrm{SA})$) and without loss of generality $s' \neq s_0$ and taking any action in s_0 makes the next state s'. We have $p_0 < \bar{p} < p_1$, all in the $[1/2, 1)$ interval. In MDP M_i with $i \in \{0, 1\}$, the probability of transitioning under action a' from s' to z, an absorbing state, is p_i, while with probability $1 - p_i$, the next state is s'. All other actions use probability $\bar{p}$ at this state. All other states under any action lead to z. The rewards are deterministically zero except at state s', when all actions yield a reward of one, regardless of the identity of the next state. 185

Chapter 1

Introduction

Reinforcement learning (RL) is a computational approach to solving sequential decision making problems (Sutton & Barto, 2018). At each time step, a *learner* receives an *observation* from the *environment*, and executes an *action* according to its *policy*. The environment then responds with a feedback signal in the form of a *reward*. The RL problem is to learn an *optimal policy* that maximizes the cumulative reward in this sequential decision making problem.

Unlike many forms of machine learning, the learner is not instructed on which action to take, but instead must explore to find the most promising action. Moreover, the action taken at the current time step may not only affect the immediate reward but also all subsequent rewards. The learner must learn to predict and plan for the future. Therefore, the challenges of value estimation, sequential planning and exploration are jointly raised in reinforcement learning. By eliminating exploration from consideration, this thesis considers two sub-topics of RL: *Planning* and *Batch Reinforcement Learning*.

Planning refers to the problem of computing the optimal sequential decision making strategy when given access to a generative model, a black-box simulator that produces simulated interactions between the learning algorithm and the environment. In many areas of life and research, such as game playing, robot navigation, network routing and logistics optimization, an almost perfect generative model is available for the learner. Such a strong sampling oracle gives the learner the power to collect data cheaply and under its control. This can significantly improve the learning efficiency over problems where one

must collect all data by following real interactions in the environment.

Batch reinforcement learning refers to learning problems where the data available is fixed and has been obtained by interacting with the environment using some *unknown behavior policy*. Interest in this problem has grown recently, as effective solutions hold the promise of extracting powerful decision making strategies from years of logged experience, with important applications to many practical problems. In many settings, such as robotics, healthcare, autonomous driving and hazard management, adaptive online data collection is impractical as it is considered to be either dangerous or expensive. One might prefer to use scalable data-driven learning methods instead, which can utilize previously collected data and become better and better as more training data is provided. Unfortunately, despite the prevalence and importance of batch reinforcement learning, it also poses major algorithmic challenges as many commonly used algorithms cannot be directly applied in the batch setting. One of the fundamental challenges in batch reinforcement learning is insufficient coverage of the dataset. In online reinforcement learning, the learner is allowed to continually explore the environment to collect useful information for the learning task. In contrast, in the batch setting, the learner has to evaluate and optimize over various candidate policies based only on experience that has been collected a priori. The distribution mismatch between the logged experience and agent-environment interaction with a learned policy can cause erroneous value overestimation, which leads to the failure of standard approaches (Fujimoto et al., 2018).

1.1 Contributions

This thesis contributes two novel techniques to improve the efficiency of Monte Carlo Tree Search (MCTS), and three analyses towards a better understanding of the theoretical foundations of batch reinforcement learning. My main contributions are:

Memory-augmented MCTS Chapter 3 proposes and evaluates Memory-augmented MCTS (M-MCTS), which provides a new approach to exploit generalization in online real-time search. The key idea of M-MCTS is to incorporate a memory structure into MCTS, where each entry contains information about a particular state. This memory is used to generate an approximate value estimate by combining the estimates of similar states. It is shown that the memory based value approximation is better than vanilla Monte Carlo estimation with high probability under mild conditions. This work was published as (Xiao et al., 2018) and received the AAAI-18 outstanding paper award.

Maximum entropy MCTS Chapter 4 develops a new algorithm for online planning in large scale sequential decision problems that improves upon the worst case efficiency of UCT. The idea is to augment MCTS with maximum entropy policy optimization, evaluating each search node by softmax values back-propagated from simulation. To establish the effectiveness of this approach, we investigate the single-step decision problem, stochastic softmax bandits, and show that softmax values can be estimated at an optimal convergence rate in terms of mean squared error. We then extend this approach to general sequential decision making by developing a general MCTS algorithm, *Maximum Entropy for Tree Search* (MENTS). It is proven that the probability of MENTS failing to identify the best decision at the root decays exponentially, which fundamentally improves the polynomial convergence rate of UCT. This work was published as (Xiao et al., 2019).

Optimality of batch policy optimization algorithm Chapter 6 studies the problem of *batch policy optimization*, where a learner must infer a behaviour policy given only access to a fixed dataset of previously collected experience, with no further environment interaction available. To advance the understanding of this problem, we provide three results that characterize the limits and possibilities of batch policy optimization in the finite-armed stochastic bandit setting. First, we introduce a class of *confidence-adjusted index* algorithms that unifies optimistic and pessimistic principles in a common

framework, which enables a general analysis. For this family, we show that any confidence-adjusted index algorithm is minimax optimal. This includes optimistic, pessimistic and neutral algorithms as special cases. Our analysis reveals that instance-dependent optimality, commonly used to establish the optimality of on-line stochastic bandit algorithms, cannot be achieved by any algorithm in the batch setting. In particular, for any algorithm that performs optimally in some environment, there exists another environment where the same algorithm suffers arbitrarily larger regret. Therefore, to establish a framework for distinguishing algorithms, we introduce a new weighted-minimax criterion that considers the inherent difficulty of optimal value prediction. We demonstrate how this criterion can be used to justify commonly used pessimistic principles for batch policy optimization. This work was published as (Xiao et al., 2021c).

The curse of passive data collection in batch RL In high stake applications, active experimentation may be considered too risky and thus data are often collected passively. While in simple cases, such as in bandits, passive and active data collection are similarly effective, the price of passive sampling can be much higher when collecting data from a system with controlled states. The main focus of Chapter 7 is the characterization of this price. For example, when learning in episodic finite state-action Markov decision processes (MDPs) with S states and A actions, we show that even with the best (but passively chosen) logging policy, $\Omega(\mathrm{A}^{\min(\mathrm{S}-1,H)}/\varepsilon^2)$ episodes are necessary (and sufficient) to obtain an ϵ-optimal policy, where H is the length of episodes. This shows that the sample complexity blows up exponentially compared to the case of active data collection, a result which is not unexpected, but, as far as we know, has not been published before. The form of the exact expression is perhaps a little surprising. We extend these results in several directions, such as learning in the presence of function approximation, with similar conclusions. A remarkable feature of our result is the sharp characterization of the exponent that appears in the lower bound, which is critical for understanding what makes passive learning hard. This work was published as (Xiao et al., 2021b).

Understanding overparameterization in value estimation The theory of function approximation in RL typically considers low capacity representations that incur a tradeoff between approximation error, stability and generalization. Current deep architectures, however, operate in an overparameterized regime where approximation error is not necessarily a bottleneck. To better understand the utility of deep models in RL, Chapter 8 presents an analysis of recursive value estimation using *overparameterized* linear representations. First, we show that classical updates such as temporal difference (TD) learning or fitted-value-iteration (FVI) converge to *different* fixed points than residual minimization (RM) in the overparameterized linear case. We then develop a unified interpretation of overparameterized linear value estimation as minimizing the Euclidean norm of the weights subject to alternative constraints. A practical consequence is that RM can be modified by a simple alteration of the backup targets to obtain the same fixed points as FVI and TD (when they converge), while universally ensuring stability. Further, we provide an analysis of the generalization error of these methods, demonstrating per iteration bounds on the value prediction error of FVI, and fixed point bounds for TD and RM. Given this new understanding, we also develop algorithmic tools for improving recursive value estimation with deep models. In particular, we develop two regularizers that penalize out-of-span top-layer weights and co-linearity in top-layer features respectively. Empirically we find that these regularizers dramatically improve the stability of TD and FVI, while allowing RM to match and even sometimes surpass their generalization performance with assured stability. This work was published as (Xiao et al., 2021a).

1.1.1 Publications

The papers related to the topics covered in this dissertation are as follows.

- Chenjun Xiao, Jincheng Mei, Martin Mueller. Memory-augmented Monte Carlo Tree Search. AAAI 2018. See (Xiao et al., 2018).
- Chenjun Xiao, Jincheng Mei, Ruitong Huang, Dale Schuurmans, Martin Müller. Maximum Entropy Monte-Carlo Planning. NeurIPS 2019. See

(Xiao et al., 2019).

- Chenjun Xiao*, Yifan Wu*, Tor Lattimore, Bo Dai, Jincheng Mei, Lihong Li, Csaba Szepesvari, Dale Schuurmans[1]. On the Optimality of Batch Policy Optimization Algorithms. ICML 2021. See (Xiao et al., 2021c).

- Chenjun Xiao, Iibin Lee, Bo Dai, Dale Schuurmans, Csaba Szepesvari. The Curse of Passive Data Collection in Batch Reinforcement Learning. AISTATS 2022. See (Xiao et al., 2021b).

- Chenjun Xiao, Bo Dai, Jincheng Mei, Oscar Ramirez, Ramki Gummadi, Chris Harris, Dale Schuurmans. Understanding and Leveraging Over-parameterization in Recursive Value Estimation. ICLR 2022. See (Xiao et al., 2021a).

[1] * indicates equal contribution.

Part I

Simulation-Based Search

Chapter 2

Online Planning

Online planning refers to the problem of computing a near optimal policy for an input state by interacting with a *generative model* of the environment. A generative model is a black-box simulator that takes any state-action pair as input and responds with the immediate reward and a next state randomly sampled from the true transition distribution of the environment. In this section, we introduce the problem setup of online planning, and briefly review three simulation-based online planning algorithms.

2.1 Online Planning

For a set $\mathcal{X}$, $\Delta(\mathcal{X})$ denotes the set of probability distributions over $\mathcal{X}$. We consider an episodic MDP $M = \{\mathcal{S}, \mathcal{A}, P, r, H\}$ (Coquelin & Munos, 2007), where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function that gives the next state distributions for each state-action pair to represent the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that gives the immediate reward incurred by taking a given action on a given state, and H is the maximum episode length. At each episode, the planning algorithm starts at an initial state s_0, and uses a generative model to simulate a H-step trajectory of the form $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_H)$. At the end of the episode, an oracle function φ assigns a stochastic evaluation $r_H = \varphi(s_H)$. We assume that φ is σ^2-subgaussian (Boucheron et al., 2013).

The return

$$G_t = \sum_{k=t}^{H} r_k \tag{2.1}$$

is the total reward accumulated in that episode from time t until reaching the terminal state at time step H. The algorithm's action selection behaviour can be described by a *policy* $\pi(a|s)$ that maps a state s to a probability distribution over actions $a \in \mathcal{A}$. For policy π, the *state value function* is defined to be the expected sum of rewards from s,

$$v^\pi(s) = \mathbb{E}^\pi \left[G_t | s_t = s \right] , \tag{2.2}$$

and the *action value function* is defined similarly,

$$q^\pi(s, a) = \mathbb{E}^\pi \left[G_t | s_t = s, a_t = a \right] . \tag{2.3}$$

The *optimal value functions* are the maximum value achievable by any policy, $v^*(s) = \max_\pi v^\pi(s)$, $q^*(s, a) = \max_\pi q^\pi(s, a)$. An *optimal policy* is defined as a greedy policy with respect to q^*, $\pi^*(s) = \arg\max_a q^*(s, a)$. An online planning algorithm queries a generative model finitely many times and returns a policy π, which is expected to be a good approximation to the optimal policy of the initial state s_0 such that $v^\pi(s_0) \approx v^*(s_0)$. The generative model is defined as follows:

Definition 1 (Generative Model)**.** *A generative model is a black-box oracle that when queried with a state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ returns the reward $r(s, a)$ and a state randomly sampled from the transition distribution $s' \sim P(\cdot|s, a)$. We use $\mathcal{G}$ to denote a generative model, and $s', r \sim \mathcal{G}(s, a)$ to denote the returns of the model.*

The *query complexity or sample complexity* measures how many queries an online planning algorithm needs in the worst case in order to compute a near optimal policy. Since in online planning the algorithm is only asked to solve for the initial state s_0, we expect the sample complexity of an online planning algorithm to be independent of the size of the entire state space.

2.2 Simulation-based Planning

Simulation-based planning is one of the most popular methods for solving the online planning problem. The main idea is to sequentially simulate episodes starting from the input state using the generative model. The statistics collected from such simulations are used to update the values of states or actions.

2.2.1 Monte Carlo Simulation

Monte Carlo (MC) methods refer to a broad class of computational algorithms that rely on repeated random sampling to approximate a numerical solution. The most straightforward application of MC methods in RL is to predict the value of a policy. Recall that the value function $v^\pi(s)$ (Eq. (2.2)) is defined as the expected future discounted total reward, where the expectation is taken over all possible trajectories under the policy π and the transition dynamics of the MDP. Given a generative model $\mathcal{G}$, to approximate such an expectation one can use $\mathcal{G}$ to simulate K independent trajectories

$$\rho^{(j)} = (S_t^{(j)}, A_t^{(j)}, R_t^{(j)}, S_{t+1}^{(j)}, \ldots, S_{H-1}^{(j)}, A_{H-1}^{(j)}, R_{H-1}^{(j)}, S_H^{(j)}),$$

where $S_t^{(j)} = s$ and $A_t^{(j)} \sim \pi(S_t^{(j)}), R_t^{(j)}, S_{t+1}^{(j)} \sim \mathcal{G}(S_t^{(j)})$. Let $v(\rho^{(j)}) = \sum_{h=t}^{H} R_h^{(j)}$ be the sum of the rewards on the sampled trajectory. The value function is approximated by the empirical mean of the simulation results

$$v^\pi(s) \approx \hat{v}^\pi(s) := \frac{1}{K} \sum_{j=1}^{K} v(\rho^{(j)}). \tag{2.4}$$

The state-action value function q^π can also be estimated in a similar way. Let $\hat{q}^\pi$ be the corresponding MC evaluation. The above process requires KH queries of the generative model in total to evaluate $v^\pi(s)$.

Using MC simulations to evaluate actions gives the simplest simulation-based planning algorithm. The algorithm first decides a *simulation policy* $\tilde{\pi}$, and evaluates each candidate action a of the initial state s_0 with MC simulations under policy $\tilde{\pi}$. The algorithm then returns the action with the highest empirical value,

$$\arg\max_a \ \hat{q}^{\tilde{\pi}}(s_0, a).$$

The overall performance of the algorithm is largely determined by the simulation policy. A simulation policy with appropriate domain knowledge can dramatically outperform a uniform random simulation policy (Gelly et al., 2006; Gelly & Silver, 2007; Coulom, 2007). Arguably, automatically learning an efficient simulation policy is also a very challenging research problem (Silver & Tesauro, 2009). Another major limitation of MC simulation is that the algorithm only uses the simulation results to update the policy of the input state at the end of the search. For all subsequent states, the simulation policy is not updated during the whole procedure. Due to this fact, the simple MC simulation algorithm does not enjoy any theoretical guarantee on the returned policy. Despite all these limitations, MC simulation serves as the basis of many advanced variants that we discuss next.

2.2.2 Spare Sampling

Sparse sampling (Kearns et al., 2002) is a depth-first search algorithm to solve the online planning problem. The basic idea is to extensively explore all subsequent states of the initial state s_0 by running simulations with the generative model. This allows the planner to construct a "sub-MDP" M' of the original MDP M such that the optimal policy at s_0 in M' is also near optimal at s_0 in M. Importantly, the size of the sub-MDP M' does not depend on the number of states in M. The complexity of finding an optimal policy of M' has no dependence on the size of the state space of M.

In sparse sampling, the sub-MDP M' is implemented as a *lookahead search tree* $\mathcal{T}$. Each interior node in $\mathcal{T}$ is labeled by a state $s \in \mathcal{S}$, and the root node is labeled by the initial state s_0. The search tree is grown in the following way. Starting from the root, the algorithm expands the node s by executing each candidate action $a \in \mathcal{A}$ and sampling n successor states $\mathcal{C}(s, a)$ given by querying the generative model $\mathcal{G}(s, a)$, which generates a total of $n|\mathcal{A}|$ children. Each child $s' \in \mathcal{C}(s, a)$ is then expanded recursively in a depth-first order until the horizon H is reached. After the search tree is built, the value of an in-tree

node labeled by state s is evaluated by

$$\hat{v}(s) = \max_{a \in \mathcal{A}} \hat{q}(s,a) := r(s,a) + \gamma \frac{1}{n} \sum_{s' \in \mathcal{C}(s,a)} \hat{v}(s')\,, \tag{2.5}$$

where at depth H the algorithm assigns the MC estimation $\hat{v}(s')$ for all leaves s' of $\mathcal{T}$. The size of the search tree is $(n|\mathcal{A}|)^H$, which is independent of the state space size of the original MDP.

The main advantage of sparse sampling over pure MC simulation is that all simulated trajectories are not wasted and they are instead organized in a structured way. One can think of the search tree $\mathcal{T}$ as a data structure that maintains all of the simulated trajectories starting from the state of interest under a "uniform" simulation policy that goes through all actions. At each node of the search tree, all candidate actions are still estimated using MC methods by taking the empirical average of the children's estimations, which are computed through an empirical max backup operator defined in a recursive manner. The main limitation of sparse sampling is that it builds the search tree uniformly. As more simulations have been collected, we expect a good algorithm to take advantage of the simulation results and gradually adapt and improve the simulation policy, such that it continually refocuses its attention on the most promising parts of the state space and only grows the search tree there. This simple idea motives the development of the Monte Carlo tree search algorithm.

2.2.3 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) relies on a *bandit algorithm* to guide the growth of the search tree online. Bandit algorithms consider the problem of efficiently allocating computation resources in sequential decision making. In a k-armed stochastic bandit, the learner has to choose among k arms (actions) $\{1, \ldots, k\}$ with unknown reward distribution $\{\nu_i\}_{1 \le i \le k}$. In each round $t = 1, \ldots, n$, the learner pulls one arm a and observes a random reward $X_t \sim \nu_a$. Let $\{\mu_i\}_{1 \le i \le k}$ be the mean reward of the arms such that $\mu_i = \mathbb{E}_{X \sim \nu_i}[X]$, and

$\mu^* = \max_i \mu_i$. The goal of the learner is to minimize the *cumulative regret*,

$$\mathcal{R}_n = \sum_{t=1}^{n} \mu^* - X_t \,. \tag{2.6}$$

Since the reward distribution of all arms is unknown at the beginning of this procedure, the learner needs to pull each arm several times in order to collect more information (i.e. *exploration*). Once its knowledge improves, the learner should focus more often on the apparently best choice (i.e. *exploitation*). The key to solve a bandit problem efficiently is to develop a good strategy to deal with the trade-off between exploration and exploitation.

The Upper Confidence Bounds (UCB) algorithm (Auer et al., 2002) is based on the principle of optimism in the face of uncertainty. It selects the action that maximizes the UCB score at each round t

$$a_t = \arg\max_a \hat{\mu}_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}\,, \tag{2.7}$$

where $\hat{\mu}_t(a)$ is the empirical reward estimate of arm a, $N_t(a)$ is the number of pulls of arm a at round t, and c is a constant. The UCB algorithm is known as an optimal algorithm for the stochastic bandit problem. It is both instance-dependent optimal and minimax optimal (Lattimore & Szepesvári, 2020).

We now are ready to introduce the MCTS algorithm. Like sparse sampling, MCTS builds a search tree $\mathcal{T}$ and evaluates states with MC simulations (Coulom, 2006). Instead of fully expanding the entire search space rooted at the initial state s_0, MCTS builds the search tree in an incremental fashion. For each interior node of $\mathcal{T}$ that is labeled by a state s, the algorithm stores a value estimate $\hat{q}(s, a)$ and a visit count $N(s, a)$ for all actions a. The estimate $\hat{q}(s, a)$ is the mean return of all simulations starting from s and a. At each iteration of the algorithm, one simulation starts from the root of the search tree, and proceeds by using a *tree policy* to select actions within the tree until a leaf of $\mathcal{T}$ is reached. An evaluation function is used at the leaf to obtain a simulation return. Typical choices of the evaluation function include function approximation with a neural network, and Monte Carlo simulations using a

roll-out policy. The return is propagated upwards to all nodes along the path to the root. Finally, the search tree $\mathcal{T}$ is grown by expanding the leaf.

The advance of MCTS over sparse sampling is that it uses bandit algorithms to balance between exploring the most uncertain branches and exploiting the most promising ones. The UCT algorithm applies the UCB algorithm (Eq. (2.7)) as its tree policy to balance the growth of the search tree (Kocsis & Szepesvári, 2006). At a node at depth h of $\mathcal{T}$ labed by s', the tree policy selects the action

$$\arg\max_a \hat{q}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}},$$

where $N(s) = \sum_a N(s,a)$, and c is a parameter controlling exploration. The UCT algorithm and its many variants have proven to be effective in many practical problems. The most famous example is the usage of its variant PUCT in AlphaGo (Silver et al., 2016, 2017; Schrittwieser et al., 2020). UCT is asymptotically optimal: the value estimated by UCT converges in probability to the optimal value, $q(s,a) \xrightarrow{p} q^*(s,a)$. The probability of finding a suboptimal action at the root converges to zero at a polynomial rate (Kocsis & Szepesvári, 2006, Theorem 6).

Chapter 3

Memory-Augmented Monte Carlo Tree Search

3.1 Introduction

The key idea of Monte Carlo Tree Search (MCTS) is to construct a search tree of states evaluated by fast Monte Carlo simulations (Coulom, 2006). Starting from a given game state, many thousands of games are simulated by randomized self-play until an outcome is observed. The state value is then estimated as the mean outcome of the simulations. Meanwhile, a search tree is maintained to guide the direction of simulation, for which bandit algorithms can be employed to balance exploration and exploitation (Kocsis & Szepesvári, 2006). However, with large state spaces, the accuracy of value estimation cannot be effectively guaranteed, since the mean value estimation is likely to have high variance under relatively limited search time. Inaccurate estimation can mislead building the search tree and severely degrade the performance of the program.

Recently, several machine learning approaches have been proposed to deal with this drawback of MCTS. For example, deep neural networks are employed to learn domain knowledge and approximate a state value function. They are integrated with MCTS to provide heuristics which can improve the search sample efficiency in practice (Silver et al., 2016; Tian & Zhu, 2015).

The successes of the machine learning methods can be mostly contributed to the power of *generalization*, *i.e.*, similar states share information. General-

ized domain knowledge is usually represented by function approximation, such as a deep neural network, which is trained *offline* from an expert move dataset or self-generated simulations (Silver et al., 2016).

Compared with the amount of research done on discovering generalization from an offline learning procedure, not too much attention has focused on exploiting the benefits of generalization during the online real-time search. The current chapter proposes and evaluates a Memory-Augmented MCTS algorithm to provide an alternative approach that takes advantage of *online* generalization. We design a *memory*, where each entry contains information about a particular state, as the basis to construct an online value approximation. We demonstrate that this memory-based framework is useful for improving the performance of MCTS in both theory and practice, using an experiment in the game of Go.

The remainder of the chapter is organized as follows: After preliminaries introduced in Section 3.2, we theoretically analyze the memory framework in Section 3. The proposed Memory-Augmented MCTS algorithm is presented in Section 4. Related work and experimental results are shown in Section 5 and 6, respectively. In Section 7, we come to our conclusion and future work.

3.2 Preliminaries

We consider the MCTS algorithm described in Section 2.2.3. Let $\mathcal{S}$ be the set of all possible states of a search problem. For $s \in \mathcal{S}$, let $\hat{v}(s) = \frac{1}{N(s)} \sum_{i=1}^{N(s)} R_{s,i}$ denote the value estimation of state s from simulations, where $R_{s,i}$ is the outcome of a simulation, $N(s)$ is the number of simulations starting from state s. The true value of a state s is denoted by $v^*(s)$. Let the value estimation error of state s be $\delta(s) = |\hat{v}(s) - v^*(s)|$, and the true value difference between states s and x be $\varepsilon(s, x) = |v^*(s) - v^*(x)|$.

Our analysis is based on the *entropy regularized policy optimization* framework. We denote the probability simplex by $\Delta = \{\mathbf{w} : \mathbf{w} \geq 0, \mathbf{1} \cdot \mathbf{w} = 1\}$, and denote the entropy function by $H(\mathbf{w}) = -\mathbf{w} \cdot \log \mathbf{w}$. For any k-dimensional vector $\mathbf{q} \in \mathbb{R}^k$, the entropy-regularized optimization problem is to find the

solution of

$$\max_{\mathbf{w}\in\Delta}\{\mathbf{w}\cdot\mathbf{q}+\tau H(\mathbf{w})\} \tag{3.1}$$

where $\tau > 0$ is the *temperature* parameter. This problem has recently drawn much attention in the reinforcement learning community (Nachum et al., 2017; Haarnoja et al., 2017; Ziebart et al., 2008). One nice property of this problem is that given the vector $\mathbf{q}$, it has a closed form solution. We define the scalar value function F_τ (the "softmax"),

$$F_\tau(\mathbf{q}) = \tau \log\left(\sum_{i=1}^{M} e^{q_i/\tau}\right), \tag{3.2}$$

and the vector-valued function $f_\tau(\mathbf{q})$ (the "soft indmax"),

$$f_\tau(\mathbf{q}) = \frac{e^{\mathbf{q}/\tau}}{\sum_{i=1}^{M} e^{q_i/\tau}} = e^{(\mathbf{q}-F_\tau(\mathbf{q}))/\tau}, \tag{3.3}$$

where the exponentiation is component-wise. Note that f_τ maps any real valued vector into a probability distribution. The next lemma states the connection between F_τ, f_τ and the entropy regularized optimization problem.

Lemma 1. *(Nachum et al., 2017; Haarnoja et al., 2017; Ziebart et al., 2008)*

$$F_\tau(\mathbf{q}) = \max_{\mathbf{w}\in\Delta}\{\mathbf{w}\cdot\mathbf{q}+\tau H(\mathbf{w})\} = f_\tau(\mathbf{q})\cdot\mathbf{q}+\tau H(f_\tau(\mathbf{q}))\,.$$

3.3 Value Approximation with Memory

The main idea of our Memory-Augmented MCTS algorithm is to approximate value estimates with the help of a memory, each entry of which contains the feature representation and simulation statistics of a particular state. Approximate value estimation is performed as follows: given a memory $\mathcal{M}$ and a state s, we find the M most similar states $\mathcal{M}_s \subset \mathcal{M}$ according to a distance metric $d(\cdot, s)$, such that $\hat{v}(s')$ is independent with $\hat{v}(s)$ for $s' \in \mathcal{M}_s$[1], and compute a memory-based value estimate

$$\hat{v}_{\mathcal{M}}(s) = \sum_{i=1}^{M} w_i(s)\hat{v}(s_i)\,, \qquad s.t. \sum_{i=1}^{M} w_i(s) = 1 \tag{3.4}$$

[1]This can be guaranteed in implementation as discussed in Section 3.4.2.

One question naturally arises, is this memory-based value approximation better than the vanilla mean outcome estimation?

3.3.1 Analysis for Independent Sub-gaussian Evaluations

We first give an analysis by considering the following assumption.

Assumption 1. *We assume that (1) $\hat{v}(s)$ are independent with each other for all states in the memory; (2) the error of MC simulation, $R_{s,t} - v^*(s)$, is σ^2-subgaussian for any state $s \in \mathcal{S}$ and $t \geq 1$.*

Based on this assumption, we attempt to show $|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \delta(s)$ for state s with high probability under some mild condition. We first show a trivial bound for $\mathbb{P}(|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \delta(s))$, then provide an improved bound with entropy regularized policy. Let $\delta_M = \max_{x \in \mathcal{M}_s} \delta(x)$ and $\varepsilon_M = \max_{x \in \mathcal{M}_s} \varepsilon(x, s)$. We assume that our memory addressing scheme is able to control ε_M within the range $[0, \varepsilon]$.

A Trivial Probability Bound

The first step is to upper bound $|\hat{v}_{\mathcal{M}}(s) - v^*(s)|$ using the triangle inequality:

$$\begin{aligned}
&\left| \sum_{i=1}^{M} w_i(s) \hat{v}(s_i) - v^*(s) \right| \\
\leq &\sum_{i=1}^{M} w_i(s) |\hat{v}(s_i) - v^*(s)| \\
\leq &\sum_{i=1}^{M} w_i(s) (|\hat{v}(s_i) - v^*(s)| + |v^*(s_i) - v^*(s)|) \\
= &\sum_{i=1}^{M} w_i(s) (\delta(s_i) + \varepsilon(s_i, s)) \qquad (3.5)
\end{aligned}$$

Using the fact that $\sum_{i=1}^{M} w_i(s) = 1$, we can take an upper bound of (Eq. (3.5)) by $\sum_{i=1}^{M} w_i(s)(\delta(s_i) + \varepsilon(s_i, s)) \leq \delta_M + \varepsilon_M$. This upper bound is very loose, since we do not specify any particular choice of the weights $\mathbf{w}$. With a standard probability argument we can immediately get the following:

Proposition 1. *For state s satisfying $\alpha(s) = \delta(s) - \varepsilon > 0$, let $n_{min} = min_{x \in \mathcal{M}_s} n(x)$. Under Assumption 1, with probability at least $1-\beta$, our memory-based value function approximation has less error than δ_x provided that:*

$$n_{min} \geq \frac{2\sigma^2}{\alpha(s)^2} \log(M/\beta)\,. \tag{3.6}$$

Improved Probability Bound with Maximum Entropy Regularization

We now provide an improvement of the previous bound by specifying the choice of the weights $\mathbf{w}$ using entropy regularized optimization. Let $\mathbf{c}$ be a vector where $c_i = \delta(s_i) + \varepsilon(s_i, s), 1 \leq i \leq M$. Our choice of $\mathbf{w}$ should minimize the upper bound (Eq. (3.5)), which is equivalent to:

$$\max_{\mathbf{w} \in \Delta} \{\mathbf{w} \cdot (-\mathbf{c})\} \tag{3.7}$$

This linear optimization problem has the solution $w_j = 1$ for $j = \text{argmin}_i c_i$ and $w_k = 0$ for $k \neq j$. However, in practice we do not know the accurate value of $\delta(s_i)$ and $\varepsilon(s_i, x)$ and applying this deterministic policy may cause the problem of addressing the wrong entries. We provide an approximation by solving the entropy regularized version of this optimization problem:

$$\max_{\mathbf{w} \in \Delta} \{\mathbf{w} \cdot (-\mathbf{c}) + \tau H(\mathbf{w})\} \tag{3.8}$$

As τ approaches zero, we recover the original problem (3.7). According to Lemma 1, the closed form solution of problem (3.8) is

$$F_\tau(-\mathbf{c}) = \tau \log \left(\sum_{i=1}^{M} e^{-c_i/\tau} \right) \tag{3.9}$$

by setting $\mathbf{w} = f_\tau(-\mathbf{c})$. Note that

$$-f_\tau(-\mathbf{c}) \cdot (-\mathbf{c}) = -F_\tau(-\mathbf{c}) + \tau H(f_\tau(-\mathbf{c})) \leq -F_\tau(-\mathbf{c}) + \tau \log M \tag{3.10}$$

Thus, to show $\mathbb{P}\{(Eq.\ (3.5)) \leq \delta\} \geq 1-\beta$ for some small constant β, it suffices to show that $\mathbb{P}\{-F_\tau(-\mathbf{c}) + \tau \log M \leq \delta\} \geq 1 - \beta$.

Theorem 1. *For states s satisfying $\alpha(s) = \delta(s) - \varepsilon > 0$, let $n = \sum_{i=1}^{M} n(s_i)$. Under Assumption 1, by choosing the weight $\mathbf{w} = f_\tau(-\mathbf{c}) = e^{-\mathbf{c}/\tau} / \sum_{i=1}^{M} e^{-c_i/\tau}$,*

with probability at least $1-\beta$ *our memory-based value function approximation has less error than* $\delta(s)$ *provided that:*

$$n \geq \frac{2\sigma^2}{(\alpha(s) - \tau \log M)^2} \log(1/\beta)\,. \tag{3.11}$$

Proof. We show that under condition (Eq. (3.11)), it can be guaranteed that $\mathbb{P}\left(-F_\tau(-\mathbf{c}) + \tau \log M \leq \delta(s)\right) \geq 1-\beta$.

$$\begin{aligned}
&\mathbb{P}\left(-\tau \log\left(\sum_{i=1}^{M} \exp(-c_i/\tau)\right) \leq \delta(s) - \tau \log M\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{M} \exp(-c_i/\tau) \geq \exp(-(\delta(s) - \tau \log M)/\tau)\right) \\
&\geq \mathbb{P}\left(\sum_{i=1}^{M} \exp(-\delta_i/\tau) \geq \exp(-(\delta(s) - \varepsilon - \tau \log M)/\tau)\right) \\
&\geq \mathbb{P}(\exists\ i, \exp(\delta(s_i)/\tau) \leq \exp((\delta(s) - \varepsilon - \tau \log M)/\tau) \\
&= 1 - \prod_{i=1}^{M} \mathbb{P}\left(\delta(s_i) \geq \alpha(s) - \tau \log M\right) \\
&\geq 1 - \prod_{i=1}^{M} \exp\left(-\frac{(\alpha(s) - \tau \log M)^2 n(s_i)}{2\sigma^2}\right) \\
&= 1 - \exp\left(-\frac{(\alpha_x - \tau \log M)^2 n}{2\sigma^2}\right)
\end{aligned} \tag{3.12}$$

The first inequality comes from our assumption that all $\varepsilon(s_i, s) \leq \varepsilon$, and the last inequality comes from the assumption that $\hat{v}(s) - v^*(s)$ is subgaussian and Hoeffding's inequality. All other inequalities can be obtained using standard probability arguments. Eq. (3.11) can be derived directly with standard algebra. □

The probability bound provided by Theorem 1 is much better than the one in Proposition 1, since n is the sum of simulation counts of all addressed memory entries, which has to be greater than n_{min}.

3.3.2 Analysis for Tree Search

Theorem 1 only holds under Assumption 1. Both independence and subgaussian assumptions do not hold in general when applying to MCTS, since

the payoff sequences experienced for any non-leaf state will drift in time as the policy at nodes in the search tree is changing, and nodes in a sub-tree are not independent. We now prove the convergence of augmenting memory-based evaluation with MCTS.

Our analysis considers the *Flat UCB* algorithm (Coquelin & Munos, 2007), which is proposed as an improvement of UCT in terms of worst-case performance. Our technique can also be directly applied for the *Bandit Algorithm for Smooth Trees* algorithm (Coquelin & Munos, 2007), which takes into account the smoothness of the rewards for performing efficient pruning sub-optimal branches during the search. The next result gives the convergence of memory-based value estimation $\hat{v}_{\mathcal{M}}$ when $\hat{v}$ is obtained from MCTS. More details of Flat UCB and the proof of this result can be found in Section A.1.

Theorem 2. *Consider a max search in a tree where each leaf is assigned a sub-gaussian reward distribution and the goal is to identify the optimal leaf. Consider running Flat UCB in a tree with branching factor K and depth D. Let N be the total number of simulations of Flat UCB. There exists constants C_1 and C_2 that only depend on K and D, such that for any $\alpha > \varepsilon + \tau(D+1)\log K$,*

$$\begin{aligned}&\mathbb{P}(\forall s, |\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \alpha) \\ &\geq 1 - \frac{C_1}{(\alpha - \varepsilon - \tau(D+1)\log K)^2} e^{-C_2(\alpha-\varepsilon-\tau(D+1)\log K)^2 \frac{N}{K^{D+1}-1}}.\end{aligned}$$

We note that the above result shows that the memory-based evaluation is consistent with MCTS: as N goes to infinity, we have for any s in the tree $\hat{v}_{\mathcal{M}}(s)$ converges to $v^*(s)$ almost surely.

3.4 Memory-Augmented MCTS

In the previous section, we prove that our memory-based value function approximation is better than the mean outcome evaluation used in MCTS with high probability under mild conditions. The remaining question is to design a practical algorithm and incorporate it with MCTS. In particular, this first requires choosing an approximation of the weight function $\mathbf{w} = f_\tau(-\mathbf{c})$.

3.4.1 Approximating $\mathbf{w} = f_\tau(-\mathbf{c})$

Let $\phi : \mathcal{S} \to \mathbb{R}^D$ be a function to generate the feature representation of a state. For two states $s, x \in \mathcal{S}$, we approximate the difference between $v^*(s)$ and $v^*(x)$ by a distance function $d(s, x)$ which is set to be the negative cosine of the two states' feature representations:

$$\varepsilon(s, x) \approx d(s, x) = -\cos(\phi(s), \phi(x)) \tag{3.13}$$

We apply two steps to create ϕ. First, take the output of an inner layer of a deep convolutional neural network and normalize it. We denote this process as $\zeta : \mathcal{S} \to \mathbb{R}^L$. In practice L will be very large which is time-consuming when computing (Eq. (3.13)). We overcome this problem by applying a feature hashing function $h : \mathbb{R}^L \to \mathbb{R}^D$ (Weinberger et al., 2009), and the feature representation is computed by $\phi(s) = h(\zeta(s))$. One nice property of feature hashing is that it can keep the inner product unbiased. Since $\zeta(s)$ is normalized, we have for $s, x \in \mathcal{S}$:

$$\mathbb{E}[\cos(\phi(s), \phi(x))] = \cos(\zeta(s), \zeta(x))$$

Let $\delta(s)$ be the term corresponding to the sampling error, which is inversely proportional to the simulation numbers: $\delta(s) \propto 1/N(s)$. Combining with (Eq. (3.13)) and the fact that e^y is very close to $y + 1$ for small y we can get our approximation of $f_\tau(-\mathbf{c})$:

$$w_i(s) = \frac{N(s_i) \exp(-d(s_i, s)/\tau)}{\sum_{j=1}^{M} N(s_j) \exp(-d(s_j, s)/\tau)} \tag{3.14}$$

By applying these approximations our model becomes a special case of kernel based methods, such as Locally Weighted Regression and Kernel Regression (Hastie et al., 2001), where τ acts like the smoothing factor in those kernel based methods. Our model is also similar to the "attention" scheme used in memory based neural networks (Graves et al., 2016; Weston et al., 2014; Vinyals et al., 2016; Pritzel et al., 2017).

3.4.2 Memory Operations

One memory $\mathcal{M}$ is maintained in our approach. Each entry of $\mathcal{M}$ corresponds to one particular state $s \in \mathcal{S}$. It contains the state's feature representation $\phi(s)$

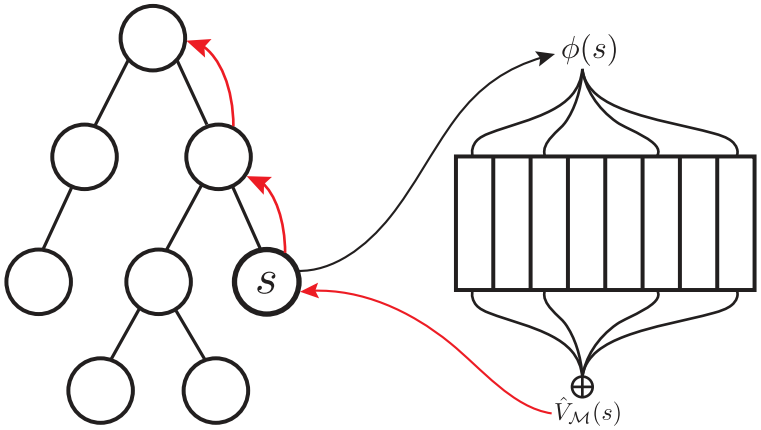


Figure 3.1: A brief illustration of M-MCTS. When a leaf state s is searched, the feature representation $\phi(s)$ is generated, which is then used to *query* the memory based value approximation $\hat{v}_{\mathcal{M}}(s)$. $\hat{v}_{\mathcal{M}}(s)$ is used to update s and all its ancestors according to Eq. (3.15), as indicated by the red arrows in the figure.

as well as its simulation statistics $\hat{v}(s)$ and $N(s)$. There are three operations to access $\mathcal{M}$: *update*, *add* and *query*.

Update

If the simulation statistics of a state s have been updated during MCTS, we also update its corresponding values $\hat{v}(s)$ and $N(s)$ in the memory.

Add

To include state s, we add a new memory entry $\{\phi(s), \hat{v}(s), N(s)\}$. If s has already been stored in the memory, we only update $\hat{v}(s)$ and $N(s)$ at the corresponding entry. If the maximum size of the memory is reached, we replace the least recently *queried* or *updated* memory entry with the new one.

Query

The *query* operation computes a memory based approximate value given a state $s \in \mathcal{S}$. We first find the top M similar states in $\mathcal{M}$ based on the distance function $d(\cdot, s)$. Note that the memory value computation requires that the evaluations of queried states must be independent with $\hat{v}(s)$. This can be guaranteed by checking that if a queried state and s are in the same path of

the current search tree. The approximated memory value is then computed by $\hat{v}_{\mathcal{M}}(x) = \sum_{i=1}^{M} w_i(s)\hat{v}(s_i)$ where the weights are computed according to Eq. (3.14). The two advantages of addressing the top M similar states are: first, to restrict the maximum value difference of addressed states with $v^*(x)$ within a range, which is shown to be useful in our analysis; second, to make queries in a very large memory scalable. We use an approximate nearest neighbours algorithm to perform the queries based on SimHash (Charikar, 2002).

3.4.3 Integrating Memory with MCTS

We are now ready to introduce our Memory-Augmented MCTS (M-MCTS) algorithm. Fig. 3.1 provides a brief illustration. The main difference between the proposed M-MCTS and regular MCTS is that, in each node of a M-MCTS search tree, we store an extended set of statistics:

$$\{N(s), \hat{v}(s), N_{\mathcal{M}}(s), \hat{v}_{\mathcal{M}}(s)\}$$

Here, $N_{\mathcal{M}}$ is the number of evaluations of the approximated memory value $\hat{v}_{\mathcal{M}}(s)$. During in-tree search of MCTS, instead of $\hat{v}(s)$, we use $(1-\lambda_s)\hat{v}(s) + \lambda_s\hat{v}_{\mathcal{M}}(s)$ as the value of state s, which is used for in-tree selection, for example in the UCB formula. λ_s is a decay parameter to guarantee no bias asymptotically.

In the original MCTS, a trajectory of visited states $\mathcal{T} = \{s_0, s_1, \ldots, s_T\}$ is obtained at the end of each simulation. The statistics of all states $s \in \mathcal{T}$ in the tree are updated. In M-MCTS, we also update the in-memory statistics by performing the *update*(s) operation of $\mathcal{M}$. Furthermore, when a new state s is searched by MCTS, we compute $\phi(s)$ and use the *add*(s) operation to include s in the memory $\mathcal{M}$.

The most natural way to obtain $\hat{v}_{\mathcal{M}}(s)$ and $n_{\mathcal{M}}(s)$ is to compute and update their value every time s is visited during the in-tree search stage. However, this direct method is time-consuming, especially when the memory size is large. Instead, we only compute the memory value at the leaf node and backpropagate the value to its ancestors. Specifically, let $s_h \in \mathcal{T}$ be the

state just added to the tree whose feature representation $\phi(s_h)$ has already been computed, and its memory approximated value $\hat{v}_{\mathcal{M}}(s_h)$ is computed by $query(s_h)$. Let $R = \hat{v}_{\mathcal{M}}(s_h) * N_{\mathcal{M}}(s_h)$ and $N_{\mathcal{M}}(s_h) = \sum_{i=1}^{M} k_i(s_h) N(s_i)$, where $k_i(s) = \exp(-d(s_i, s)/\tau) / \sum_{j=1}^{M} \exp(-d(s_j, s)/\tau)$. For state $s_i \in \{s_0, \ldots, s_h\}$, we perform the following updates, where $\eta \geq 1$ is a decay parameter.

$$\begin{aligned} X &\leftarrow \max(N_{\mathcal{M}}(s_h)/\eta^{|i-h|}, 1) \\ N_{\mathcal{M}}(s_i) &\leftarrow N_{\mathcal{M}}(s_i) + X \\ \hat{v}_{\mathcal{M}}(s_i) &\leftarrow \hat{v}_{\mathcal{M}}(s_i) + \frac{R - \hat{v}_{\mathcal{M}}(s_i) * X}{N_{\mathcal{M}}(s_i)} \end{aligned} \tag{3.15}$$

The reason for the decay parameter η is because the memory-approximated value of a state is more similar to its closer ancestors.

3.5 Related Work

The idea of utilizing information from similar states has been previously studied in game solving. Kawano (1996) provided a technique where proofs of similar positions are reused for proving other nodes in a game tree. Kishimoto & Müller (2004) applied this to provide an efficient Graph History Interaction solution, for solving the games of Checkers and Go.

Memory architectures for neural networks and reinforcement learning have been recently described in Memory Networks (Weston et al., 2014), Differentiable Neural Computers (Graves et al., 2016), Matching Network (Vinyals et al., 2016) and Neural Episodic Control (NEC) (Pritzel et al., 2017). The most similar work with our M-MCTS algorithm is NEC, which applies a memory framework to provide action value function approximation in reinforcement learning. The memory architecture and addressing method are similar to ours. In contrast to their work, we provide theoretical analysis about how the memory can affect value estimation. Furthermore, to our best knowledge, this work is the first one to apply a memory architecture in MCTS.

The role of generalization has been previously exploited in transposition tables (Childs et al., 2008), Temporal-Difference search (TD search) (Silver et al., 2012), Rapid Action Value Estimation (RAVE) (Gelly & Silver, 2011),

and mNN-UCT (Srinivasan et al., 2015). A transposition table provides a simple form of generalization. All nodes in the tree corresponding to the same state share the same simulation statistics. Our addressing scheme can closely resemble a transposition table by setting τ close to zero. In M-MCTS with $\tau > 0$ the memory can provide more generalization, which we show to be beneficial both theoretically and practically.

TD search uses linear function approximation to generalize between related states. This linear function approximation is updated during the online real-time search. However, with complex non-linear function approximation such as neural networks, such updates are impossible to perform online. Since our memory based method is non-parametric, it provides an alternative approach for generalization during real time search.

RAVE uses the all-moves-as-first heuristic based on the intuition that the value of an action is independent of when it is taken. Simulation results are not only updated to one, but to all actions along the simulation path. mNN-UCT applies kernel regression to approximate a state value function, which has been shown equivalent to our addressing scheme using our choice of approximations in Section 4. However, we use the difference between feature representations as the distance metric, while mNN-UCT applies the distance between nodes in the tree. Also, both RAVE and mNN-UCT do not provide any theoretical justifications.

3.6 Experiments

We evaluate M-MCTS in the game of Go (Müller, 2002).

3.6.1 Implementation Details

Our implementation applies a deep convolutional neural network (DCNN) from (Clark & Storkey, 2015), which is trained for move prediction by professional game records. It has 8 layers in total, including one convolutional layer with 64 7×7 filters, two convolutional layers with 64 5×5 filters, two layers with 48 5×5 filters, two layers with 32 5×5 filters, and one fully connected layer.

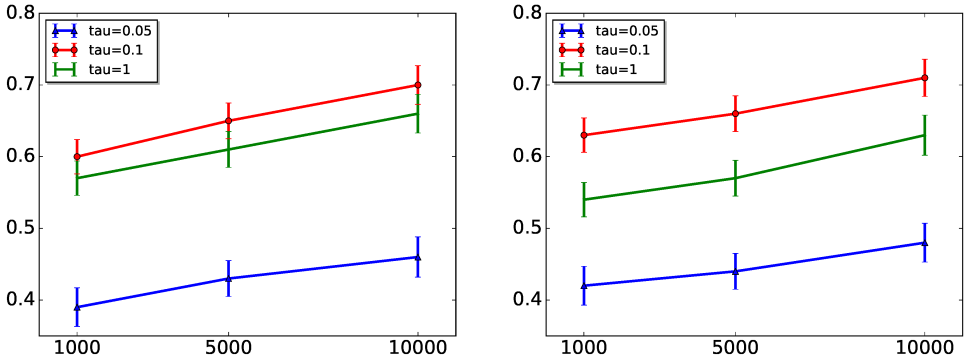
The network has about 44% prediction accuracy on professional game records. The feature vector $\phi(s)$ is first extracted from the output of Conv7 which is the last layer before the final fully connected layer of the neural network. The dimension of this output is 23104. A dimension reduction step using feature hashing as described in Section 4 is then applied. The feature hashing dimension is set to 4096, which gives $\phi(s) \in \mathbb{R}^{4096}$.

The hash code in our SimHash implementation has 16 bits. We use 8 hash tables, each of which corresponds to a unique hash function. We also apply a multiple probing strategy. Suppose that a feature vector $\phi(s)$ is mapped to the hash bin b_i at the ith hash table. Let the hash code of b_i be h_i. To search the neighbours of $\phi(s)$ in the ith table, we search those bins whose hash codes' hamming distance to h_i is less than a threshold, set to 1 in our implementation. The discount parameter η in Eq. (3.15) to update memory approximated values is set to 2.

3.6.2 Baseline

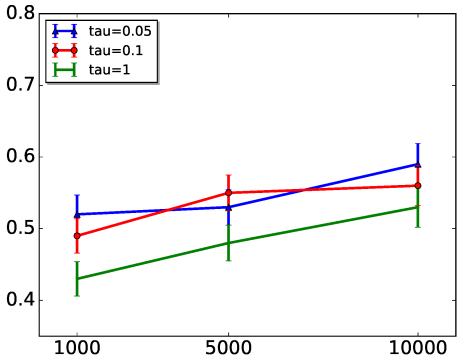
Our baseline is based on the open source Go program Fuego (Enzenberger et al., 2010), but adds DCNN to improve performance. We adopt the method from (Gelly & Silver, 2007) and use DCNN to initialize simulation statistics. Suppose that DCNN is called on the state s that has just been added to the tree. For a move m, let p_m be the move probability from the network, and s' the state transformed by taking m on s. Let $p_{\max}$ be the maximum of the network's output move probabilities. We compute two statistics $\hat{V}_{\text{DCNN}}(s') = 0.5 * (1.0 + p_m/p_{\max})$ and $\hat{N}_{\text{DCNN}}(s) = \text{CNN\_STRENGTH} * p_m/p_{\max}$. These two values are used as the initial statistics when creating s'. We set CNN_STRENGTH to 200 in our experiment.

We implement DCNN in MCTS in a synchronized way, where the search continues after the DCNN evaluation is returned. To increase speed, we restrict DCNN calls to the first 100 nodes visited during the search. This baseline achieves a win rate of 97% against original Fuego with 10,000 simulations per move. We implement M-MCTS based on this baseline. The same DCNN is used to extract features for the memory.

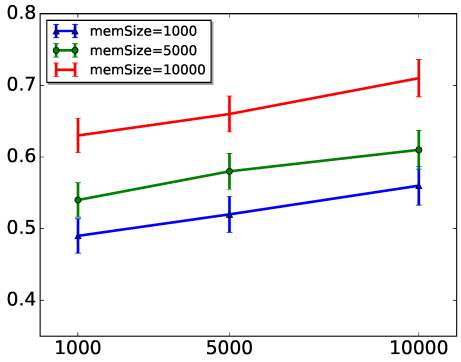


(a) M=20

(b) M=50



(c) M=100



(d) Testing Memory Size

Figure 3.2: Experimental results. Figure (a)-(c) shows the results of testing different value of M. Figure (d) shows the results of testing different size of memory. In all figures, x-axis is the number of simulations per move, y-axis means the winning rate against the baseline.

3.6.3 Results

We first study how the parameters M and τ can affect the performance of M-MCTS, since these two parameters together control the degree of generalization. The parameter M is chosen from $\{20, 50, 100\}$, and τ from $\{0.05, 0.1, 1\}$. The size of $\mathcal{M}$ is set to 10000. We vary the number of simulations per move from $\{1000, 5000, 10000\}$. Results are summarized in Figure 3.2(a)-(c). The best result we have is from the setting $\{M = 50, \tau = 0.1\}$, which achieves a 71% win rate against the baseline with 10,000 simulations per move. The standard error of each result is around 2.5%. For $M = 20$ and $M = 50$, the performance of M-MCTS scales well with the number of simulations per move

with $\tau = 1$ and $\tau = 0.1$. A small temperature $\tau = 0.05$ cannot beat the baseline at all. We believe the reason is that in this setting M-MCTS only focuses on the closest neighbours for generalization, but does not do enough exploration. For $M = 100$, M-MCTS does not perform well in any setting of τ, since larger M increases the chance of including less similar states.

We then investigate the impact of the size of $\mathcal{M}$ by varying it from $\{1000, 5000, 10000\}$. M and τ are set to 50 and 0.1 respectively. Results with a different number of simulations per move are summarized in Figure 3.2(d). Intuitively, a large memory can provide better performance, since more candidate states are included for querying. The results shown in Figure 3.2(d) confirm this intuition: M-MCTS achieves the best performance with $|\mathcal{M}| = 10000$, while a small memory size $|\mathcal{M}| = 1000$ can even lead to negative effects for value estimation in MCTS. We also compare M-MCTS with the baseline with equal computational time per move. By setting $M = 50$, $\tau = 0.1$ and with 5 seconds per move, M-MCTS achieves a 61% win rate against the baseline.

3.7 Conclusion

In this chapter, we present an efficient approach to exploit online generalization during real-time search. Our method, Memory-Augmented Monte Carlo Tree Search (M-MCTS), combines the original MCTS algorithm with a memory framework, to provide a memory-based online value approximation. We demonstrate that this can improve the performance of MCTS in both theory and practice.

Chapter 4

Maximum Entropy Monte Carlo Planning

4.1 Introduction

Monte Carlo planning algorithms have been widely applied in many challenging problems (Silver et al., 2016, 2017). One particularly powerful and general algorithm is Monte Carlo Tree Search (MCTS) (Coulom, 2006). The key idea of MCTS is to construct a search tree of states that are evaluated by averaging over outcomes from simulations. MCTS provides several major advantages over traditional online planning methods. It breaks the curse of dimensionality by simulating state-action trajectories using a domain generative model, and building a search tree online by collecting information gathered during the simulations in an incremental manner. It can be combined with domain knowledge such as function approximations learned either online (Xiao et al., 2018) or offline (Silver et al., 2016, 2017). It is highly selective, where bandit algorithm are applied to balance between exploring the most uncertain branches and exploiting the most promising ones (Kocsis & Szepesvári, 2006). MCTS has demonstrated outstanding empirical performance in many game playing problems, but most importantly, it is provable to converge to the optimal policy if the exploitation and exploration balanced appropriately (Kocsis & Szepesvári, 2006; Kearns et al., 2002).

The convergence property of MCTS highly replies on the state value estimates. At each node of the search tree, the value estimate is also used to

calculate the value of the action leading to that node. Hence, the convergence rate of the state value estimate influences the rate of convergence for states further up in the tree. However, the Monte Carlo value estimate (average over simulations outcomes) used in MCTS does not enjoy an effective convergence guarantee when this value is back-propagated in the search tree, since for any given node, the sampling policy in the subtree is changing and the payoff sequences experienced will drift in time. In summary, the compounding error, caused by the structure of the search tree as well as the uncertainty of the Monte Carlo estimation, makes that UCT can only guarantee a polynomial convergence rate of finding the best action at the root.

Ideally, one would like to adopt a state value that can be efficiently estimated and back-propagated in a search tree. In this chapter, we exploit the usage of *softmax value estimate* in MCTS based on the maximum entropy policy optimization framework. To establish the effectiveness of this approach, we first propose a new *stochastic softmax bandit* framework for the single-step decision problem, and show that softmax values can be estimated in a sequential manner at an optimal convergence rate in terms of mean squared error. Our next contribution is to extend this approach to general sequential decision making by developing a general MCTS algorithm, *Maximum Entropy for Tree Search* (MENTS). We contribute new observations that the softmax state value can be efficiently back-propagated in the search tree, which enables the search algorithm to achieve faster convergence rate towards finding the optimal action at the root. Our theoretical analysis shows that MENTS enjoys an exponential convergence rate to the optimal solution, improving the polynomial convergence rate of UCT fundamentally. Our experiments also demonstrate that MENTS is much more sample efficient compared with UCT in practice.

4.2 Background

We consider the online planning problem described in Section 2.1. We assume the transition and reward functions are deterministic for simplicity, while all of

our techniques can easily generalize to the case with stochastic transitions and rewards, with an appropriate dependence on the variances of the transition and reward distributions.

4.2.1 Maximum Entropy Policy Optimization

The maximum entropy policy optimization problem, which augments the standard expected reward objective with a entropy regularizer, has recently drawn much attention in the reinforcement learning community (Haarnoja et al., 2017, 2018; Nachum et al., 2017). Given K actions and the corresponding K-dimensional reward vector $\mathbf{r} \in \mathbb{R}^K$, the entropy regularized policy optimization problem finds a policy by solving

$$\max_{\pi} \Big\{ \pi \cdot \mathbf{r} + \tau \mathcal{H}(\pi) \Big\}. \tag{4.1}$$

where $\tau \geq 0$ is a user-specified temperature parameter which controls the degree of exploration. The most intriguing fact about this problem is that it has a closed form solution. Define the *softmax* $\mathcal{F}_\tau$ and the *soft indmax* $\mathbf{f}_\tau$ functions,

$$\mathbf{f}_\tau(\mathbf{r}) = \exp\{(\mathbf{r} - \mathcal{F}_\tau(\mathbf{r}))/\tau\} \qquad \mathcal{F}_\tau(\mathbf{r}) = \tau \log \sum\nolimits_a \exp(r(a)/\tau).$$

Note that the softmax $\mathcal{F}_\tau$ outputs a scalar while the soft indmax $\mathbf{f}_\tau$ maps any reward vector $\mathbf{r}$ to a Boltzmann policy. $\mathcal{F}_\tau(\mathbf{r})$, $\mathbf{f}_\tau(\mathbf{r})$ and (4.1) are connected by as shown in Haarnoja et al. (2017); Nachum et al. (2017),

$$\mathcal{F}_\tau(\mathbf{r}) = \max_{\pi} \Big\{ \pi \cdot \mathbf{r} + \tau \mathcal{H}(\pi) \Big\} = \mathbf{f}_\tau(\mathbf{r}) \cdot \mathbf{r} + \tau \mathcal{H}(\mathbf{f}_\tau(\mathbf{r})). \tag{4.2}$$

This relation suggests the softmax value is an upper bound on the maximum value, and the gap can be upper bounded by the product of τ and the maximum entropy. Note that as $\tau \to 0$, (4.1) approaches the standard expected reward objective, where the optimal solution is the hard-max policy. Therefore, it is straightforward to generalize the entropy regularized optimization to define the *softmax value functions*, by replacing the hard-max operator with the softmax operators (Haarnoja et al., 2017; Nachum et al., 2017),

$$q^*_{\text{sft}}(s,a) = r(s,a) + \mathbb{E}_{s'|s,a}\left[v^*_{\text{sft}}(s')\right], \qquad v^*_{\text{sft}}(s) = \tau \log \sum\nolimits_a \exp\Big\{ q^*_{\text{sft}}(s,a)/\tau \Big\}. \tag{4.3}$$

Finally, according to (4.2), we can characterize the optimal *softmax policy* by,

$$\pi^*_{\text{sft}}(a|s) = \exp\left\{ \left(q^*_{\text{sft}}(s,a) - v^*_{\text{sft}}(s)\right)/\tau \right\}. \tag{4.4}$$

We combine maximum entropy policy optimization with MCTS, by estimating the softmax values backpropagated from simulations. We show that the softmax values can be efficiently backpropagated in the search tree, which leads to a faster convergence rate to the optimal policy at the root.

4.3 Softmax Value Estimation with Stochastic Bandit

We begin by introducing the *stochastic softmax bandit* problem. We provide an asymptotic lower bound of this problem, propose a new bandit algorithm for it and show a tight upper bound on its convergence rate. Our upper bound matches the lower bound not only in order, but also in the coefficient of the dominating term. All proofs are provided in Section B.2.

4.3.1 The Stochastic Softmax Bandit

Consider a stochastic bandit setting with arms set $\mathcal{A}$. At each round t, a learner chooses an action $A_t \in \mathcal{A}$. Next, the environment samples a random reward R_t and reveals it to the learner. Let $r(a)$ be the expected value of the reward distribution of action $a \in \mathcal{A}$. We assume $r(a) \in [0,1]$, and that all reward distributions are σ^2-subgaussian [1]. For round t, we define $N_t(a)$ as the number of times a is chosen so far, and $\hat{r}_t(a)$ as the empirical estimate of $r(a)$,

$$N_t(a) = \sum_{i=1}^{t} \mathbb{I}\{A_t = a\} \qquad \hat{r}_t(a) = \sum_{i=1}^{t} \mathbb{I}\{A_i = a\} R_i / N_t(a),$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. Let $\mathbf{r} \in [0,1]^K$ be the vector of expected rewards, and $\hat{\mathbf{r}}_t$ be the empirical estimates of $\mathbf{r}$ at round t. We denote $\pi^*_{\text{sft}} =$

[1] For prudent readers, we follow the finite horizon bandits setting in (Lattimore & Szepesvári, 2020), where the probability space carries the tuple of random variables $S_T = \{A_0, R_0, \ldots, A_T, R_T\}$. For every time step $t-1$ the historical observation defines a σ-algebra $\mathcal{F}_{t-1}$ and A_t is $\mathcal{F}_{t-1}$-measurable, the conditional distribution of A_t is our policy at time π_t, and the conditoinal distribution of the reward $R_{A_t} - r(A_t)$ is a martingale difference sequence.

$\mathbf{f}_\tau(\mathbf{r})$ the optimal soft indmax policy defined by the mean reward vector $\mathbf{r}$. The stochastic bandit setting can be considered as a special case of an episodic MDP with $H = 1$.

In a stochastic softmax bandit problem, instead of finding the policy with maximum expected reward as in the original stochastic bandits (Lattimore & Szepesvári, 2020), our objective is to estimate the softmax value $v^*_{\text{sft}} = \mathcal{F}_\tau(\mathbf{r})$ for some $\tau > 0$. We define $U^* = \sum_a \exp\{r(a)/\tau\}$ and $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\}$, and propose to use the estimator $v_t = \mathcal{F}_\tau(\hat{\mathbf{r}}_t) = \tau \log U_t$. Our goal is to find a sequential sampling algorithm that can minimize the mean squared error, $\mathcal{E}_t = \mathbb{E}[(U^* - U_t)^2]$. The randomness in $\mathcal{E}_t$ comes from both the sampling algorithm and the observed rewards. We first give a lower bound on $\mathcal{E}_t$.

Theorem 3. *In the stochastic softmax bandit problem, for any algorithm that achieves $\mathcal{E}_t = O(\frac{1}{t})$, there exists a problem setting such that*

$$\lim_{t\to\infty} t\mathcal{E}_t \geq \frac{\sigma^2}{\tau^2}\left(\sum_a \exp(r(a)/\tau)\right)^2.$$

Also, to achieve this lower bound, there must be for any $a \in \mathcal{A}$, $\lim_{t\to\infty} N_t(a)/t = \pi^*_{sft}(a)$.

Note that in Theorem 3, we only assume $\mathcal{E}_t = O(1/t)$, but not that the algorithm achieves (asymptotically) unbiased estimates for each arm. Furthermore, this lower bound also reflects the consistency between the softmax value and the soft indmax policy equation 4.2: in order to achieve the lower bound on the mean squared error, the sampling policy must converge to π^*_{sft} asymptotically.

4.3.2 E2W: an Optimal Sequential Sampling Strategy

Inspired by the lower bound, we propose an optimal algorithm, Empirical Exponential Weight (E2W), for the stochastic softmax bandit problem. The main idea is very intuitive: enforce enough exploration to guarantee good estimation of $\hat{\mathbf{r}}$, and make the policy converge to π^* asymptotically, as suggested by the lower bound. Specifically, at round t, the algorithm selects an action

by sampling from the distribution

$$\pi_t(a) = (1 - \lambda_t)\mathbf{f}_\tau(\hat{\mathbf{r}})(a) + \lambda_t \frac{1}{|\mathcal{A}|} \ . \tag{4.5}$$

In (4.5), $\lambda_t = \varepsilon|\mathcal{A}|/\log(t+1)$ is a decay rate for exploration, with exploration parameter $\varepsilon > 0$. Our next theorem provides an exact convergence rate for E2W.

Theorem 4. *For the softmax stochastic bandit problem, E2W can guarantee,*

$$\lim_{t\to\infty} t\mathcal{E}_t = \frac{\sigma^2}{\tau^2}\left(\sum_a \exp(r(a)/\tau)\right)^2 .$$

Theorem 4 shows that E2W is an asymptotically optimal sequential sampling strategy for estimating the softmax value in stochastic multi-armed bandits. The main contribution of the present chapter is the introduction of the softmax bandit algorithm for the implementation of tree policy in MCTS. In our proposed new algorithm, softmax bandit is used as the fundamental tool both for estimating each state's softmax value, and balancing the growth of the search tree.

4.4 Maximum Entropy MCTS

We now describe the main technical contributions of this chapter, which combine maximum entropy policy optimization with MCTS. Our proposed method, MENTS (Maximum Entropy for Tree Search), applies a similar algorithmic design as UCT with two innovations: using E2W as the tree policy, and evaluating each search node by softmax values back-propagated from simulations.

4.4.1 Algorithmic Design

Let $\mathcal{T}$ be a look-ahead search tree built online by the algorithm. Each node $n(s) \in \mathcal{T}$ is labeled by a state s, contains a softmax value estimate $q_{\text{sft}}(s,a)$, and a visit count $N(s,a)$ for each action a. We use $\mathbf{q}_{\text{sft}}(s)$ to denote a $|\mathcal{A}|$-dimensional vector with components $q_{\text{sft}}(s,a)$. Let $N(s) = \sum_a N(s,a)$ and

$v_{\text{sft}}(s) = \mathcal{F}_\tau(\mathbf{q}_{\text{sft}}(s))$. During the in-tree phase of the simulation, the tree policy selects an action according to

$$\pi_t(a|s) = (1-\lambda_s)\mathbf{f}_\tau(\mathbf{q}_{\text{sft}}(s))(a) + \lambda_s \frac{1}{|\mathcal{A}|} \tag{4.6}$$

where $\lambda_s = \varepsilon|\mathcal{A}|/\log(\sum_a N(s,a)+1)$. Let $\{s_0, a_0, s_1, a_1, \ldots, s_T\}$ be the state action trajectory in the simulation, where $n(s_T)$ is a leaf node of $\mathcal{T}$. An evaluation function is called on s_T and returns an estimate R [2]. $\mathcal{T}$ is then grown by expanding $n(s_T)$. Its statistics are initialized by $q_{\text{sft}}(s_T, a) = 0$ and $N(s_T, a) = 0$ for all actions a. For all nodes in the trajectory, we update the visiting counts by $N(s_t, a_t) = N(s_t, a_t) + 1$, and the Q-values using a *softmax backup*,

$$q_{\text{sft}}(s_t, a_t) = \begin{cases} r(s_t, a_t) + R & t = T-1 \\ r(s_t, a_t) + \mathcal{F}_\tau(\mathbf{q}_{\text{sft}}(s_{t+1})) & t < T-1 \end{cases} \tag{4.7}$$

The algorithm MENTS can also be extended to use domain knowledge, such as function approximations learned offline. For instance, suppose that a policy network $\tilde{\pi}(\cdot|s)$ is available. Then the statistics can be initialized by $q_{\text{sft}}(s_T, a) = \log \tilde{\pi}(a|s_T)$ and $N(s_T, a) = 0$ for all actions a during the expansion. Finally, at each time step t, MENTS proposes the action with the maximum estimated softmax value at the root s_0; i.e. $a_t = \text{argmax}_a q_{\text{sft}}(s_0, a)$.

4.4.2 Theoretical Analysis

This section provides the key steps in developing a theoretical analysis of the convergence property for MENTS. We first show that for any node in the search tree, after its subtree has been fully explored, the estimated softmax value will converge to the optimal value at an exponential rate. Recall that in Theorem 3, an optimal sampling algorithm for the softmax stochastic bandit problem must guarantee $\lim_{t\to\infty} N_t(a)/t = \pi^*_{\text{sft}}(a)$ for any action a. Our first result shows that this holds for true in E2W with high probability. This directly comes from the proof of Theorem 4.

[2] We adapt a similar setting to Section 4.3, where R_t is replaced by the sample from the evaluation function, and the martingale assumption is extended to the the selection policy and the evaluation function on the leaves.

Theorem 5. *Consider E2W applied to the stochastic softmax bandit problem. Let* $N_t^*(a) = \pi_{sft}^*(a) \cdot t$. *Then there exists some constants* C *and* $\tilde{C}$ *such that,*

$$\mathbb{P}\left(|N_t(a) - N_t^*(a)| > \frac{Ct}{\log t}\right) \leq \tilde{C}|\mathcal{A}|t \exp\left\{-\frac{t}{(\log t)^3}\right\}.$$

In the bandit case, the reward distribution of each arm is assumed to be subgaussian. However, when applying bandit algorithms at the internal nodes of a search tree, the payoff sequence experienced from each action will drift over time, since the sampling probability of the actions in the subtree is changing. The next result shows that even under this drift condition, the softmax value can still be efficiently estimated according to the backup scheme (4.7).

Theorem 6. *For any node* $n(s) \in \mathcal{T}$, *define the event,*

$$E_s = \left\{\forall a \in \mathcal{A}, |N(s,a) - N^*(s,a)| < \frac{N^*(s,a)}{2}\right\}$$

where $N^*(s,a) = \pi_{sft}^*(a|s) \cdot N(s)$. *For* $\epsilon \in [0,1)$, *there exist some constant* C *and* $\tilde{C}$ *such that for sufficiently large* t,

$$\mathbb{P}\left(|v_{sft}(s) - v_{sft}^*(s)| \geq \epsilon | E_s\right) \leq \tilde{C} \exp\left\{-\frac{N(s)\tau^2\epsilon^2}{C\sigma^2}\right\}.$$

Without loss of generality, we assume $Q^*(s,1) \geq Q^*(s,2) \geq \cdots \geq Q^*(s,|\mathcal{A}|)$ for any $n(s) \in \mathcal{T}$, and define $\Delta = Q^*(s,1) - Q^*(s,2)$. Recall that by (4.2), the gap between the softmax and maximum value is upper bounded by τ times the maximum of entropy. Therefore as long as τ is chosen small enough such that this gap is smaller than Δ, the best action also has the largest softmax value. Finally, as we are interested in the probability that the algorithm fails to find the optimal arm at the root, we prove the following result.

Theorem 7. *Let* a_t *be the action returned by MENTS at iteration* t. *Then for large enough* t *with some constant* C,

$$\mathbb{P}\left(a_t \neq a^*\right) \leq Ct \exp\left\{-\frac{t}{(\log t)^3}\right\}.$$

Remark. MENTS enjoys a fundamentally faster convergence rate than UCT. We highlight two main reasons behind this success result from the innovated algorithmic design. First, MENTS applies E2W as the tree policy

during simulations. This assures that the softmax value functions used in MENTS could be effectively estimated in an optimal rate, and the tree policy would converge to the optimal softmax policy π^*_{sft} asymptotically, as suggested by Theorem 3 and Theorem 4. Secondly, Theorem 6 shows that the softmax value can also be efficiently back-propagated in the search tree. Due to these facts, the probability of MENTS failing to identify the best decision at the root decays exponentially, significantly improving the polynomial rate of UCT.

4.5 Related Work

Maximum entropy policy optimization is a well studied topic in reinforcement learning (Haarnoja et al., 2017, 2018; Nachum et al., 2017). The maximum entropy formulation provides a substantial improvement in exploration and robustness, by adopting a smoothed optimization objective and acquiring diverse policy behaviors. The proposed algorithm MENTS is built on the softmax Bellman operator Eq. (4.3), which is used as the value propagation formula in MCTS. To our best knowledge, MENTS is the first algorithm that applies the maximum entropy policy optimization framework for simulation-based planning algorithms.

Several works have been proposed for improving UCT, since it is arguably "over-optimistic" (Coquelin & Munos, 2007) and does not explore sufficiently: UCT can take a long time to discover an optimal branch that initially looked inferior. Previous work has proposed to use flat-UCB, which enforces more exploration, as the tree policy for action selection at internal nodes (Coquelin & Munos, 2007). Minimizing simple regret in MCTS is discussed in (Pepels et al., 2015; Tolpin & Shimony, 2012). Instead of using UCB1 as the tree policy at each node, these works employ a hybrid architecture, where a best-arm identification algorithm such as Sequential Halving (Karnin et al., 2013) is applied at the upper levels, while the original UCT is used for the deeper levels of the tree.

Various value back-propagation strategies, particularly back-propagate the maximum estimated value over the children, were originally studied in (Coulom,

2006). It has been shown that the maximum backup is a poor option, since Monte-Carlo estimation is too noisy when the number of simulations is low, which misguides the algorithm, particularly at the beginning of search. Complex back-propagation strategies in MCTS have been investigated in (Khandelwal et al., 2016), where a mixture of maximum backup with the well known TD-λ operator (Sutton & Barto, 2018) is proposed. In contrast to these approaches, MENTS exploits the softmax backup to achieve a faster convergence rate of value estimation.

4.6 Experiments

We evaluate the proposed algorithm, MENTS, across several different benchmark problems against strong baseline methods. Our first test domain is a *Synthetic Tree* environment. The tree has branching factor (number of actions) k of depth d. At each leaf of the tree, a standard Gaussian distribution is assigned as an evaluation function, that is each time a leaf is visited, the distribution is used to sample a stochastic return. The mean of each Gaussian distribution is determined in the following way: when initializing the environment each edge is assigned a random value, then the mean of the Gaussian distribution at a leaf is the sum of values along the path from the root to the leaf. This environment is similar to the P-game tree environment (Kocsis & Szepesvári, 2006; Smith & Nau, 1994) used to model two player minimax games, while here we consider the single (max) player version. Finally, we normalize all the means in $[0, 1]$.

We then test MENTS on five Atari games: *BeamRider*, *Breakout*, *Q*bert*, *Seaquest* and *SpaceInvaders*. For each game, we train a vanilla DQN and use it as an evaluation function for the tree search as discussed in the AlphaGo (Silver et al., 2016, 2017). In particular, the softmax of Q-values is used as the state value estimate, and the Boltzmann distribution over the Q-values is used as the policy network to assign a probability prior for each action when expanding a node. The temperature is set to 0.1. The UCT algorithm adopts

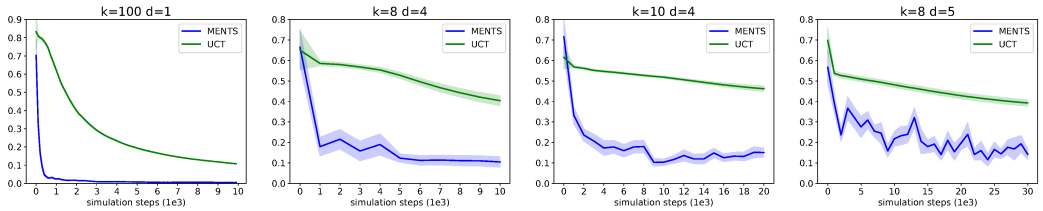


Figure 4.1: Evaluation of softmax value estimation in the synthetic tree environment. The x-axis shows the number of simulations and y-axis shows the value estimation error. The shaded area shows the standard error. We find that the softmax value can be efficiently estimated by MENTS.

the following tree-policy introduced in AlphaGo (Silver et al., 2017),

$$\mathrm{PUCT}(s,a) = Q(s,a) + cP(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$

where $P(s,a)$ is the prior probability. MENTS also applies the same evaluation function. The prior probability is used to initialize the q_{sft} as discussed in Section 4.4.1. We note that the DQN is trained using a hard-max target. Training a neural network using softmax targets such as soft Q-learning or PCL might be more suitable for MENTS (Haarnoja et al., 2017; Nachum et al., 2017). However, in the experiments we still use DQN in MENTS to present a fair comparison with UCT, since both algorithms apply the exactly same evaluation function. The details of the experimental setup are provided in Section B.1.

4.6.1 Results

Value estimation in synthetic tree. As shown in Section 4.4.2, the main advantage of the softmax value is that it can be efficiently estimated and back-propagated in the search tree. To verify this observation, we compare the value estimation error of MENTS and UCT in both the bandit and tree search setting. For MENTS, the error is measured by the absolute difference between the estimated softmax value $v_{\mathrm{sft}}(s_0)$ and the true softmax state value $v^*_{\mathrm{sft}}(s_0)$ of the root s_0. For UCT, the error is measured by the absolute difference between the Monte Carlo value estimation $V(s_0)$ and the optimal state value $V^*(s_0)$ at the root. We report the results in Figure 4.1. Each data point is averaged over

5×5 independent experiment (5 runs on 5 randomly initialized environment). In all of the test environments, we observe that MENTS estimates the softmax values efficiently. By comparison, we find that the Monte Carlo estimation used in UCT converges far more slowly to the optimal state value, even in the bandit setting ($d = 1$).

Online planning in a synthetic tree. We next compare MENTS with UCT for online planning in the synthetic tree environment. Both algorithms use Monte Carlo simulation with a uniform rollout policy as the evaluation function. The error is evaluated by $V^*(s_0) - Q^*(s_0, a_t)$, where a_t is the action proposed by the algorithm at simulation step t, and s_0 is the root of the synthetic tree. The optimal values Q^* and V^* are computed by back-propagating the true values from the leaves when the environment is initialized. Results are reported in Figure 4.2. As in the previous experiments, each data point is averaged over 5×5 independent experiment (5 runs on 5 randomly initialized environments). UCT converges faster than our method in the bandit environment ($d = 1$). This is because that the main advantage of MENTS is the use of softmax state values, which can be efficiently estimated and back-propagated in the search tree. In the bandit case such an advantage does not exist. In the tree case ($d > 0$), we find that MENTS significantly outperforms UCT, especially in the large domain. For example, in synthetic tree with $k = 8$ $d = 5$, UCT fails to identify the optimal action at the root in some of the random environments, result in the large regret given the simulation budgets. However, MENTS can continuously make progress towards the optimal solution in all random environments, confirming MENTS scales with larger tree depth.

Online planning in Atari 2600 games. Finally, we compare MENTS and UCT using Atari games. At each time step we use 500 simulations to generate a move. Results are provided in Table 4.1, where we highlight scores where MENTS significantly outperforms the baselines. Scores obtained by DQN are also provided. In *Breakout*, *Q*bert* and *SpaceInvaders*, MENTS significantly outperforms UCT as well as the DQN agent. In *BeamRider* and *Seaquest* all algorithms performs similarly, since the search algorithms only use the DQN as the evaluation function and only 500 simulations are applied to

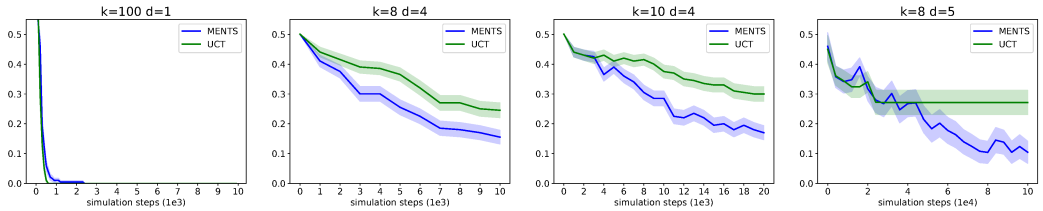


Figure 4.2: Evaluation of online planning in the synthetic tree environment. The x-axis shows the number of simulations and y-axis shows the planning error. The shaded area shows the standard error. We can observe that MENTS enjoys much smaller error than UCT especially in the large domain.

Table 4.1: Performance comparison of Atari games playing.

Agent	*BeamRider*	*Breakout*	*Q*bert*	*Seaquest*	*SpaceInvaders*
DQN	19280	345	14558	1142	625
UCT	21952	367	16010	1129	656
MENTS	18576	**386**	**18336**	1161	**1503**

generate a move. We can expect better performance when a larger simulation budget is used.

4.7 Conclusion

In this chapter, we propose a new online planning algorithm, Maximum Entropy for Tree Search (MENTS), for large scale sequential decision making. The main idea of MENTS is to augment MCTS with maximum entropy policy optimization, evaluating each node in the search tree using softmax values back-proagated from simulations. We contribute two new observations that are essential to establishing the effectiveness of MENTS: first, we study *stochastic softmax bandits* for single-step decision making and show that softmax values can be estimated at an optimal convergence rate in terms of mean squared error; second, the softmax values can be efficiently back-propagated from simulations in the search. We prove that the probability of MENTS failing to identify the best decision at the root decays exponentially, which fundamen-

tally improves the worst case efficiency of UCT. Empirically, MENTS exhibits a significant improvement over UCT in both synthetic tree environments and Atari game playing.

Part II

Batch Reinforcement Learning

Chapter 5

Reinforcement Learning

5.1 Markov Decision Process

Recall that *reinforcement learning* is a computational approach to solving sequential decision making problems. At each time-step, a *learner* receives an observation from the *environment*, and executes an action according to its *policy*. The environment then responds with a feedback signal in the form of a *reward*. The goal of reinforcement learning is to improve the learning agent's performance by computing an *optimal policy* that maximizes the total reward.

A reinforcement learning (RL) problem is typically formulated as a Markov Decision Process (MDP), which is a mathematical model for modeling sequential decision making (Puterman, 2014). For set $\mathcal{X}$, let $\Delta(\mathcal{X})$ be the set of probability distributions over $\mathcal{X}$. An MDP is defined as $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function that gives the next state distributions for each state-action pair to represent the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that gives the immediate reward incurred by taking a given action on a given state, $\gamma \in [0, 1)$ is the discount factor. The learner's behaviour is described by a (memoryless)[1] policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, that computes a probability distribution over actions. Let μ be the initial state distribution. Given a state s_0 randomly sampled from μ, the interconnection of a policy π and the MDP M results in a distribution $\mathbb{P}^\pi$ over the random trajectory $s_0, a_0, r_0, s_1, a_1, r_1, \ldots,$

[1] A memoryless policy does not depend on the history and takes only the current state into account.

where $a_t \sim \pi(s_t)$, $r_t = r(s_t, a_t), s_{t+1} \sim P(\cdot|s_t, a_t)$. We use $\mathbb{E}^\pi$ to denote the expectation operator under the distribution $\mathbb{P}^\pi$.

5.2 Value Functions

Most RL solution methods use a *value function* to predict the long-term reward consequences of a particular policy. For the *discounted total reward criterion* with discount factor $0 \leq \gamma < 1$, the *state value function* $v^\pi : \mathcal{S} \to \mathbb{R}$ under π is defined as

$$v^\pi(s) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s \right] . \tag{5.1}$$

Let $v^\pi(\mu) = \mathbb{E}_{s\sim\mu}[v^\pi(s)]$, where $\mu \in \Delta(\mathcal{S})$ is the initial state distribution. The *state-action value function* of π, $q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, is defined as

$$q^\pi(s, a) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s, A_0 = a \right] . \tag{5.2}$$

There exists a unique value function $v^* : \mathcal{S} \to \mathbb{R}$, called as the *optimal value function*, that maximizes the value over all states $s \in \mathcal{S}$, $v^*(s) = \sup_\pi v^\pi$. A policy is *optimal* in state s if $v^\pi(s) = v^*(s)$. Optimal policies also share the same state-action value $q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $q^*(s, a) = \sup_\pi q^\pi(s, a)$

One of the key properties of value functions is that they satisfy the *Bellman equations*, which relate the value of a state (state-action pair) to its successor states (state-action pairs) (Bellman, 1957). In particular, for any policy π

$$v^\pi(s) = T^\pi v^\pi(s) := \sum_a \pi(a|s) \left[r(s, a) + \gamma \sum_{s'} P(s'|s, a) v^\pi(s') \right] , \tag{5.3}$$

and

$$q^\pi(s, a) = T^\pi q^\pi(s, a) := r(s, a) + \gamma \sum_{s'} P(s'|s, a) \sum_{a'} \pi(a'|s') q^\pi(s', a') , \tag{5.4}$$

or in short, $v^\pi = T^\pi v^\pi$ and $q^\pi = T^\pi q^\pi$, where T^π is the *Bellman operator*. The Bellman equations suggest an *iterative policy evaluation* algorithm to estimate the value functions: 1) initialize a value function estimate v_0; 2) at iteration $k > 0$, update the value function by using the reward and transition functions

to perform a full-width lookahead over all possible actions and state transitions $v_k = T^\pi v_{k-1}$; 3) stop when the algorithm converges.

The optimal value function also satisfies a recursive definition, which is known as the *Bellman optimality equations*:

$$v^*(s) = Tv^*(s) := \max_a \; r(s,a) + \gamma \sum_{s'} P(s'|s,a) v^*(s') \,, \tag{5.5}$$

and

$$q^*(s,a) = Tq^*(s,a) := r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} q^*(s',a') \,, \tag{5.6}$$

or in short, $v^* = Tv^*$ and $q^* = Tq^*$, where T is known as the *Bellman optimality operator*. The Bellman optimality equations provide the basis for the *value iteration* algorithm that computes the optimal value functions. The algorithm works similarly to the iterative policy evaluation described above, except that the Bellman operator T^π is replaced with the Bellman optimality operator T. Alternatively, in order to compute the optimal value function one can also use the *policy iteration* algorithm: 1) initialize an arbitrary policy π_0; 2) at iteration $k > 0$, compute $q^{\pi_{k-1}}$ using the iterative policy evaluation algorithm, and define π_k as the greedy policy with respect to $q^{\pi_{k-1}}$; 3) stop when the algorithm converges. The second step is also known as *policy improvement* (Sutton & Barto, 2018). Convergence analyses of all these algorithms can be found in (Szepesvári, 2010).

5.3 Effective Planning Horizon

In the discounted infinite horizon setting, the discount factor γ makes the future rewards less important than the present reward. Suppose that all rewards belong to the interval $[0, r_{\max}]$. If we truncate the discounted total rewards after $h \geq 0$ steps, the difference between the truncated return and the true return is at most

$$\gamma^h (r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} + \dots) \leq \gamma^h \sum_{t=0}^{\infty} r^t = \frac{\gamma^h r_{\max}}{1-\gamma} \,,$$

where the last equality follows by the summation rule for the geometric series. Given an error tolerance parameter ε, this difference is bounded by ε as long as the horizon h satisfies

$$h \geq H_{\gamma,\varepsilon} := \frac{\ln\left(\frac{r_{\max}}{\varepsilon(1-\gamma)}\right)}{1-\gamma}\,. \tag{5.7}$$

The quantity $H_{\gamma,\varepsilon}$ is referred to as the *effective planning horizon*. Ignoring an error of at most ε, it suffices to find a policy that maximizes the discounted total rewards of the first $H_{\gamma,\varepsilon}$ steps.

5.4 Discounted Occupancy Measure

Given an initial state distribution $\mu \in \Delta(\mathcal{S})$ and a policy π, the (unnormalized) discounted occupancy measure ν_μ^π induced by μ, π, and the underlying MDP M is defined as

$$\nu_\mu^\pi(s,a) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(S_t = s, A_t = a | S_0 \sim \mu)\,. \tag{5.8}$$

One interesting property of this occupancy measure is that the value function can be represented as an inner product between the immediate reward function r and the occupancy measure ν_μ^π

$$v^\pi(\mu) = \mathbb{E}^\pi_{s\sim\mu}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s\right] \tag{5.9}$$

$$= \sum_{s,a}\sum_{t=0}^{\infty} \gamma^t \mathbb{E}^\pi_\mu\left[r(S_t, A_t)\mathbb{I}(S_t = s, A_t = a)\right] \tag{5.10}$$

$$= \sum_{s,a} r(s,a) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi_\mu\left[\mathbb{I}(S_t = s, A_t = a)\right] \tag{5.11}$$

$$= \sum_{s,a} r(s,a)\nu_\mu^\pi(s,a) \tag{5.12}$$

$$= \langle \nu_\mu^\pi, r\rangle\,. \tag{5.13}$$

5.5 Batch Reinforcement Learning

Batch RL[2] is concerned with problems where one must solve a learning task given only access to a fixed dataset of previously collected experience, with-

[2]This problem has also been referred to as *offline* RL.

out further environment interaction. In batch RL, the learner is given a fixed dataset $\mathcal{D} = \{s_i, a_i, r_i, s'_i\}_{i=1}^n$ which is collected by interacting with the underlying MDP M using some unknown behavior policy $\pi_{\log}$. The batch nature requires that the learning algorithm has no control and knowledge of $\pi_{\log}$. Similar to online RL, there are two basic tasks in the batch setting: *policy evaluation*, where the goal is to predict the value of a given target policy π_{target} in M, and *policy optimization*, where the goal is to compute a near optimal policy in the MDP M. We are particularly interested in the sample complexity of a batch RL algorithm to solve a task. For example, when considering the batch policy optimization problem, our goal is to design an algorithm which can compute a policy π such that $v^* - v^\pi < \varepsilon$ with a minimum number of samples n, where ε is a given target accuracy and v^* and v are value functions in M.

Chapter 6

On the Optimality of Batch Policy Optimization Algorithms

6.1 Introduction

A fundamental challenge in batch policy optimization is insufficient coverage of the dataset. In online reinforcement learning (RL), the learner is allowed to continually explore the environment to collect useful information for the learning task. By contrast, in the batch setting, the learner has to evaluate and optimize over various candidate policies based only on experience that has been collected a priori. The distribution mismatch between the logged experience and agent-environment interaction with a learned policy can cause erroneous value overestimation, which leads to the failure of standard policy optimization methods (Fujimoto et al., 2019). To overcome this problem, recent studies propose to use the *pessimistic principle*, by either learning a pessimistic value function (Wu et al., 2019; Jaques et al., 2019; Kumar et al., 2019, 2020) or pessimistic surrogate (Buckman et al., 2020), or planning with a pessimistic model (Kidambi et al., 2020; Yu et al., 2020). However, it still remains unclear how to maximally exploit the logged experience without further exploration.

In this chapter, we investigate batch policy optimization with finite-armed stochastic bandits, and make three contributions toward better understanding the statistical limits of this problem. *First*, we prove a minimax lower bound of $\Omega(1/\sqrt{\min_i n_i})$ for batch policy optimization with stochastic bandits, where n_i is the count of arm i in the dataset. We then introduce a notion

of confidence-adjusted index algorithm that unifies both the optimistic and pessimistic principles in a single algorithmic framework. Our analysis suggests that any index algorithm with an appropriate adjustment, whether pessimistic or optimistic, is minimax optimal.

Second, we analyze the instance-dependent regret bound of batch policy optimization algorithms. Perhaps surprisingly, our main result shows that instance-dependent optimality, which is commonly used in the literature of minimizing cumulative regret of stochastic bandits, does not exist in the batch setting. Together with our first contribution, this finding challenges recent theoretical findings in batch RL that claim pessimistic algorithms are an optimal choice (Buckman et al., 2020; Jin et al., 2020c). In fact, our analysis suggests that for any algorithm that performs optimally in some environment, there must always exist another environment where the algorithm suffers arbitrarily larger regret than an optimal strategy there. Therefore, any reasonable algorithm is equally optimal, or not optimal, depending on the exact problem instance the algorithm is facing. In this sense, for batch policy optimization, there remains a lack of a well-defined optimality criterion that can be used to choose between algorithms.

Third, we provide a characterization of the pessimistic algorithm by introducing a weighted-minimax objective. In particular, the pessimistic algorithm can be considered to be optimal in the sense that it achieves a regret that is comparable to the inherent difficulty of optimal value prediction on an instance-by-instance basis. Overall, the theoretical study we provide consolidates recent research findings on the impact of being pessimistic in batch policy optimization (Buckman et al., 2020; Jin et al., 2020c; Kumar et al., 2020; Kidambi et al., 2020; Yu et al., 2020; Liu et al., 2020).

6.2 Preliminaries

To simplify the exposition, we express our results for batch policy optimization in the setting of stochastic finite-armed bandits. In particular, we assume the action space consists of $k > 0$ arms, where the available data takes the

form of $n_i > 0$ real-valued observations $X_{i,1}, \ldots, X_{i,n_i}$ for each arm $i \in [k] := \{1, \ldots, k\}$. This data represents the outcomes of n_i pulls of each arm i. We assume further that the data for each arm i is *i.i.d.* with $X_{i,j} \sim P_i$ such that P_i is the reward distribution for arm i. Let $\mu_i = \int x P_i(dx)$ denote the mean reward that results from pulling arm i. All observations in the data set $X = (X_{ij})_{i \in [k], j \in [n_i]}$ are assumed to be independent.

We consider the problem of designing an algorithm that takes the counts $(n_i)_{i \in [k]}$ and observations $X \in \times_{i \in [k]} \mathbb{R}^{n_i}$ as inputs and returns the index of a single arm in $[k]$, where the goal is to select an arm that achieves the highest mean reward. Let $\mathcal{A}(X) \in [k]$ be the output of algorithm $\mathcal{A}$, Then the (simple) regret of $\mathcal{A}$ can be defined as

$$\mathcal{R}(\mathcal{A}, \theta) = \mu^* - \mathbb{E}_{X \sim \theta}[\mu_{\mathcal{A}(X)}],$$

where $\mu^* = \max_i \mu_i$ is the maximum reward. Here, the expectation $\mathbb{E}_{X \sim \theta}$ considers the randomness of the data X generated from problem instance θ, and also any randomness in the algorithm $\mathcal{A}$, which together induce the distribution of the random choice $\mathcal{A}(X)$. Note that this definition of regret depends both on the algorithm $\mathcal{A}$ and the problem instance $\theta = ((n_i)_{i \in [k]}, (P_i)_{i \in [k]})$. When θ is fixed, we will use $\mathcal{R}(\mathcal{A})$ to reduce clutter.

For convenience, we also let $n = \sum_i n_i$ and $n_{\min}$ denote the total number of observations and the minimum number of observations in the data, let a^* denote the optimal arm, let $\Delta_i = \mu^* - \mu_i$ denote the reward gap of arm i, and let $\Delta_{\max} = \max_i \Delta_i$ and $\Delta_{\min} = \min_{i: \Delta_i > 0} \Delta_i$ be the maximum and minimum reward gap. In what follows, we assume that the distributions P_i are 1-subgaussian with means in the unit interval $[0, 1]$. We denote the set of these distributions by $\mathcal{P}$. The set of all instances where the distributions satisfy these properties is denoted by Θ. The set of instances with $\mathbf{n} = (n_i)_{i \in [k]}$ fixed is denoted by $\Theta_{\mathbf{n}}$. Thus, $\Theta = \cup_{\mathbf{n}} \Theta_{\mathbf{n}}$. Finally, we define $|\mathbf{n}| = \sum_i n_i$ for $\mathbf{n} = (n_i)_{i \in [k]}$.

6.3 Minimax Analysis

In this section, we introduce the notion of a *confidence-adjusted index algorithm*, and prove that a broad range of such algorithms are minimax optimal up to a constant. A confidence-adjusted index algorithm is one that calculates an index for each arm based on the data for that arm only, then chooses an arm that maximizes the index. We consider index algorithms where the index of arm $i \in [k]$ is defined as the sum of the sample mean of this arm, $\hat{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} X_{i,j}$ plus a bias term of the form $\alpha/\sqrt{n_i}$ with $\alpha \in \mathbb{R}$. That is, given the input data X, the algorithm selects an arm according to

$$\arg\max_{i \in [k]} \ \hat{\mu}_i + \frac{\alpha}{\sqrt{n_i}} \,. \tag{6.1}$$

The reason we call these confidence-adjusted is because for a given confidence level $\delta > 0$, by Hoeffding's inequality, it follows that

$$\mu_i \in \left[\hat{\mu}_i - \frac{\beta_\delta}{\sqrt{n_i}}, \ \hat{\mu}_i + \frac{\beta_\delta}{\sqrt{n_i}} \right] \tag{6.2}$$

with probability at least $1 - \delta$ for all arms with

$$\beta_\delta = \sqrt{2 \log\left(\frac{k}{\delta}\right)} \,.$$

Thus, the family of confidence-adjusted index algorithms consists of all algorithms that follow this strategy, where each particular algorithm is defined by a (data independent) choice of α. For example, an algorithm specified by $\alpha = -\beta_\delta$ chooses the arm with highest lower-confidence bound (highest LCB value), while an algorithm specified by $\alpha = \beta_\delta$ chooses the arm with the highest upper-confidence bound (highest UCB value). Note that $\alpha = 0$ corresponds to what is known as the *greedy* (sample mean maximizing) choice.

Readers familiar with the literature on batch policy optimization will recognize that $\alpha = -\beta_\delta$ implements what is known as the pessimistic algorithm (Jin et al., 2020c; Buckman et al., 2020; Kidambi et al., 2020), or distributionally robust choice, or risk-adverse strategy. It is therefore natural to question the utility of considering batch policy optimization algorithms that *maximize*

UCB values (i.e., implement optimism in the presence of uncertainty, or risk-seeking behavior, even when there is no opportunity for exploration). However, our first main result is that for batch policy optimization a risk-seeking (or greedy) algorithm cannot be distinguished from the more commonly proposed pessimistic approach in terms of minimax regret. To establish this finding, we first provide a lower bound on the minimax regret:

Theorem 8. *Fix* $\mathbf{n} = (n_i)_{i\in[k]}$ *with* $n_1 \leq \cdots \leq n_k$. *Then, there exists a universal constant* $c > 0$ *such that*

$$\inf_{\mathcal{A}} \sup_{\theta\in\Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A},\theta) \geq c \max_{m\in[k]} \sqrt{\frac{\max(1,\log(m))}{n_m}}\,.$$

The assumption of increasing counts, $n_1 \leq \cdots \leq n_k$, is only needed to simplify the statement; the arm indices can always be re-ordered without loss of generality. The proof follows by arguing that the minimax regret is lower bounded by the Bayesian regret of the Bayesian optimal policy for any prior. Then, with a judicious choice of prior, the Bayesian optimal policy has a simple form. Intuitively, the available data permits estimation of the mean of action a with accuracy $O(\sqrt{1/n_a})$. The additional logarithmic factor appears when $n_1, \ldots, n_m$ are relatively close, in which case the lower bound is demonstrating the necessity of a union bound that appears in the upper bound that follows. The full proof appears in the supplementary material.

Next we show that a wide range of confidence-adjusted index algorithms are nearly minimax optimal when their confidence parameter is properly chosen:

Theorem 9. *Fix* $\mathbf{n} = (n_i)_{i\in[k]}$. *Let* δ *be the solution of* $\delta = \sqrt{\frac{32\log(k/\delta)}{\min_i n_i}}$, *and* $\mathcal{I}$ *be the confidence-adjusted index algorithm with parameter* α. *Then, for any* $\alpha \in [-\beta_\delta, \beta_\delta]$, *we have*

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{I}(\alpha),\theta) \leq 12\sqrt{\frac{\log(k/\delta)}{\min_i n_i}}\,.$$

Remark 1. *Theorem 9 also holds for algorithms that use different* $\alpha_i \in [-\beta_\delta, \beta_\delta]$ *for different arms.*

Perhaps a little unexpectedly, we see that *regardless* of optimism vs. pessimism, index algorithms with the right amount of adjustment, or *even no adjustment*, are minimax optimal, up to an order $\sqrt{\log(k)}$ factor. We note that although these algorithms have the same worst case performance, they can behave very differently indeed on individual instances, as we show in the next section.

In effect, what these two results tell us is that minimax optimality is too weak as a criterion to distinguish between pessimistic versus optimistic (or greedy) algorithms when considering the "fixed count" setting of batch policy optimization. This leads us to ask whether more refined optimality criteria are able to provide nontrivial guidance in the selection of batch policy optimization methods. One such criterion, considered next, is known as instance-optimality in the literature of cumulative regret minimization for stochastic bandits.

6.4 Instance-Dependent Analysis

To better distinguish between algorithms we require a much more refined notion of performance that goes beyond merely considering worst-case behavior over all problem instances. Even if two algorithms have the same worst case performance, they can behave very differently on individual instances. Therefore, we consider the instance dependent performance of confidence-adjusted index algorithms.

6.4.1 Instance-dependent Upper Bound

Our next result provides a regret upper bound for a general form of index algorithm. All upper bounds in this section hold for any $\theta \in \Theta_{\mathbf{n}}$ unless otherwise specified, and we use $\mathcal{R}(\mathcal{A})$ instead of $\mathcal{R}(\mathcal{A}, \theta)$ to simplify the notation.

Theorem 10. *Consider a general form of index algorithm,* $\mathcal{A}(X) = \arg\max_i \hat{\mu}_i + b_i$*, where* b_i *denotes the bias for arm* $i \in [k]$ *specified by the algorithm. For* $2 \leq i \leq k$ *and* $\eta \in \mathbb{R}$*, define*

$$g_i(\eta) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2}$$

and $g_i^* = \min_\eta g_i(\eta)$. *Assuming* $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$, *for the index algorithms (6.1) we have*

$$\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\} \tag{6.3}$$

and

$$\mathcal{R}(\mathcal{A}) \leq \sum_{2 \leq i \leq k} \Delta_i \left(\min\{1, g_i^*\} - \min\{1, g_{i+1}^*\}\right) \tag{6.4}$$

where we define $g_{k+1}^* = 0$.

The assumption $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$ is only required to express the statement simply; the indices can be reordered without loss of generality. The expression in equation 6.3 is a bit difficult to work with, so to make the subsequent analysis simpler we provide a looser but more interpretable bound for general index algorithms as follows.

Corollary 1. *Following the setting of Theorem 10, consider any index algorithm and any* $\delta \in (0, 1)$. *Define* $U_i = \mu_i + b_i + \beta_\delta/\sqrt{n_i}$ *and* $L_i = \mu_i + b_i - \beta_\delta/\sqrt{n_i}$. *Let* $h = \max\{i \in [k] : \max_{j<i} L_j < \max_{j' \geq i} U_{j'}\}$. *Then we have*

$$\mathcal{R}(\mathcal{A}) \leq \Delta_h + \frac{\delta}{k}\Delta_{\max} + \frac{\delta}{k}\sum_{i>h}(\Delta_i - \Delta_{i-1}) \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2}.$$

Remark 2. *The upper bound in Corollary 1 can be further relaxed as* $\mathcal{R}(\mathcal{A}) \leq \Delta_h + \delta\Delta_{\max}$.

Remark 3. *The minimax regret upper bound (Theorem 9) can be recovered a result of Corollary 1.*

Corollary 1 highlights an inherent optimization property of index algorithms: they work by designing an additive adjustment for each arm, such that all of the bad arms ($i > h$) can be eliminated efficiently, i.e., it is desirable to make h as small as possible. We note that although one can directly plug in the specific choices of $\{b_i\}_{i \in [k]}$ to get instance-dependent upper bounds for different algorithms, it is not clear how their performance compares to one

another. Therefore, we provide simpler relaxed upper bounds for the three specific cases, greedy, LCB and UCB, to allow us to better differentiate their performance across different problem instances (see supplement for details).

Corollary 2 (Regret Upper bound for Greedy)**.** *Following the setting of Theorem 10, for any $0 < \delta < 1$, the regret of greedy ($\alpha = 0$) on any problem instance is upper bounded by*

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \left(\Delta_i + \sqrt{\frac{2}{n_i} \log \frac{k}{\delta}} + \max_{j > i} \sqrt{\frac{2}{n_j} \log \frac{k}{\delta}} \right) + \delta \,.$$

Corollary 3 (Regret Upper bound for LCB)**.** *Following the setting of Theorem 10, for any $0 < \delta < 1$, the regret of LCB ($\alpha = -\beta_\delta$) on any problem instance is upper bounded by*

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \Delta_i + \sqrt{\frac{8}{n_i} \log \frac{k}{\delta}} + \delta \,.$$

Corollary 4 (Regret Upper bound for UCB)**.** *Following the setting of Theorem 10, for any $0 < \delta < 1$, the regret of UCB ($\alpha = \beta_\delta$) on any problem instance is upper bounded by*

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \left(\Delta_i + \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \right) + \delta \,.$$

Remark 4. *The results in these corollaries sacrifice the tightness of instance-dependence to obtain cleaner bounds for the different algorithms. The tightest instance dependent bounds can be derived from Theorem 10 by optimizing η.*

Discussion. The regret upper bounds presented above suggest that although they are all nearly minimax optimal, UCB, LCB and greedy exhibit distinct behavior on individual instances. Each will eventually select the best arm with high probability when n_i gets large for *all* $i \in [k]$, but their performance can

be very different when n_i gets large for only a *subset* of arms $S \subset [k]$. For example, LCB performs well whenever S contains a good arm (i.e., with small Δ_i and large n_i). UCB performs well when there is a good arm i such that all worse arms are in S (n_j large for all $j > i$). For the greedy algorithm, the regret upper bound is small only when there is a good arm i where n_j is large for all $j \geq i$, in which situation both LCB and UCB perform well.

Clearly there are instances where LCB performs much better than UCB and vice versa. Consider an environment where there are two groups of arms: one with higher rewards and another with lower rewards. The behavior policy plays a subset of the arms $S \subset [k]$ a large number of times and ignores the rest. If S contains at least one good arm but no bad arm, LCB will select a good played arm (with high probability) while UCB will select a bad unplayed arm. If S consists of all bad arms, then LCB will select a bad arm by being pessimistic about the unobserved good arms while UCB is guaranteed to select a good arm by being optimistic.

This example actually raises a potential reason to favor LCB, since the condition for UCB to outperform LCB is stricter: requiring the behavior policy to play all bad arms while ignoring all good arms. To formalize this, we compare the upper bounds for the two algorithms by taking the n_i for a subset of arms $i \in S \subset [k]$ to infinity. For $\mathcal{A} \in \{\text{greedy}, \text{LCB}, \text{UCB}\}$, let $\hat{\mathcal{R}}_S(\mathcal{A})$ be the regret upper bounds with $\{n_i\}_{i \in S} \to \infty$ and $\{n_i\}_{i \notin S} = 1$ while fixing $\mu_1, ..., \mu_k$ in Corollary 2, 3, and 4 respectively. Then LCB dominates the three algorithms with high probability under a uniform prior for S:

Proposition 2. *Suppose $\mu_1 > \mu_2 > ... > \mu_k$ and $S \subset [k]$ is uniformly sampled from all subsets with size $m < k$, then*

$$\mathbb{P}\left(\hat{\mathcal{R}}_S(\text{LCB}) < \hat{\mathcal{R}}_S(\text{UCB})\right) \geq 1 - \frac{(k-m)!m!}{k!}.$$

This lower bound is $1/2$ when $k = 2$ and approaches 1 when k increases for any $0 < m < k$ since it is always lower bounded by $1 - 1/k$. The same argument applies when comparing LCB to greedy.

To summarize, when comparing different algorithms by their upper bounds, we have the following observations: (i) These algorithms behave differently on different instances, and none of them outperforms the others on all instances. (ii) Both scenarios where LCB is better and scenarios where UCB is better exist. (iii) LCB is more favorable when k is not too small because it is the best option among these algorithms on most of the instances.

Simulation results. Since our discussion is based on comparing only the upper bounds (instead of the exact regret) for different algorithms, it is a question that whether these statements still hold in terms of their actual performance. To answer this question, we verify these statements through experiments on synthetic problems. The details of these synthetic experiments can be found in the supplementary material.

We first verify that there exist instances where LCB is the best among the three algorithms as well as instances where UCB is the best. For LCB to perform well, we construct two ϵ-greedy behavior policies on a 100-arm bandit where the best arm or a near-optimal arm is selected to be played with a high frequency while the other arms are uniformly played with a low frequency. In more details, the reward distribution for each arm $i \in [100]$ is a Gaussian with unit variance. The mean rewards μ_i are uniformly spread over $[0, 1]$. In particular, we have $\mu_1 \geq \ldots \mu_{100}$, $\mu_i - \mu_{i+1} = 0.01$ for $1 \leq i < 99$, and $\mu_1 = 1$. When generating the data set, we split the arms into two sets S_1 and $S_2 = [k] \setminus S_2$. For each arm $i \in \mathcal{S}_1$, we collect πn data; for each arm $i \in S_2$, we collect $n(1 - \pi|S_1|)/|S_2|$ data, where n is the total sample size, and $0 \leq \pi \leq 1/|\mathcal{S}_1|$ is a parameter to be chosen to generate different data sets. We consider four data sets: *LCB*-1 ($S_1 = \{1\}$, $\pi = 0.3$); *LCB*-2 ($S_1 = \{10\}$, $\pi = 0.3$); *UCB*-1 ($S_1 = \{1\}$, $\pi = 1e^{-4}$); *UCB*-2 ($S_1 = \{1, \ldots, 10\}$, $\pi = 1e^{-4}$). For each instance, we run each algorithm 500 times and use the average performance to approximate the expected simple regret. Error bars are the standard deviation of the simple regret over the 500 runs.

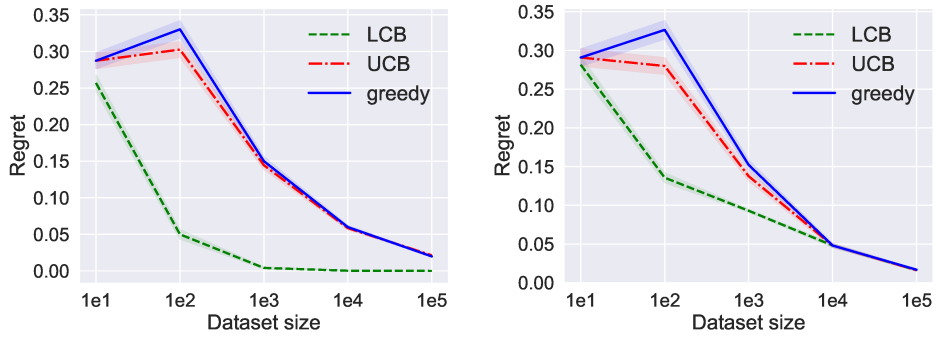
Figure 6.1(a) and 6.1(b) show that LCB outperforms UCB and greedy on these two instances, verifying our observation from the upper bound (Corol-

lary 3) that LCB only requires a good behavior policy while UCB and greedy require bad arms to be eliminated (which is not the case for ϵ-greedy policies). For UCB to outperform LCB, we set the behavior policy to play a set of near-optimal arms with only a small number of times and play the rest of the arms uniformly. Figure 6.1(c) and 6.1(d) show that UCB outperforms LCB and greedy on these two instances, verifying our observation from the upper bound (Corollary 4) that UCB only requires all worse arms to be identified.

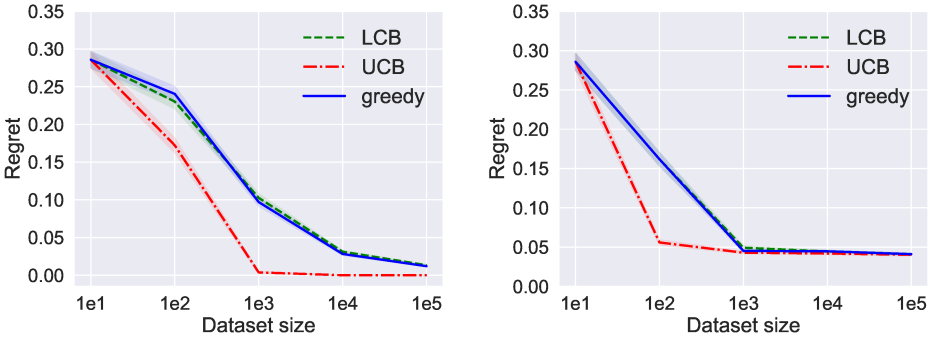
We then verify the statement that LCB is the best option on most of the instances when k is not too small. We verify this statement in two aspects: First, we show that when $k = 2$, LCB and UCB have an equal chance to be the better algorithm. More specifically, we fix $n_1 > n_2$ (note that if $n_1 = n_2$ all index algorithms are the same as greedy) and vary $\mu_1 - \mu_2$ from -1 to 1. The reward distribution for each arm $i \in [2]$ is a Gaussian with unit variance. We fix $\mu_1 = 0$ and vary μ_2 accordingly. Intuitively, when $|\mu_1 - \mu_2|$ is large, the problem is relatively easy for all algorithms. For $\mu_1 - \mu_2$ in the medium range, as it becomes larger, the good arm is tried more often, thus the problem becomes easier for LCB and harder for UCB. Figure 6.2(a) and 6.2(b) confirm this and show that both LCB and UCB are the best option on half of the instances. In Figure 6.2(a), $n_1 = 10, n_2 = 5$. In Figure 6.2(b), $n_1 = 100, n_2 = 10$. For each instance, we run each algorithm 100 times and use the average performance to approximate the expected simple regret. Error bars are the standard deviation of the simple regret over the 100 runs.

Second, we show that as k grows, LCB quickly becomes the more favorable algorithm, outperforming UCB and greedy on an increasing fraction of instances. More specifically, we vary k and sample a set of instances from the prior distribution introduced in Proposition 2. For each k, we first sample 100 vectors $\vec{\mu} = [\mu_1, ..., \mu_k]$ in the following way: We generate $\vec{\mu}_0$ with $\vec{\mu}_{0,i} = \frac{i-1}{2(k-1)} + \frac{1}{4}$ such that all reward means are evenly distributed with in $[\frac{1}{4}, \frac{3}{4}]$. We then add independent Gaussian noise with standard deviation 0.05 to each $\vec{\mu}_{0,i}$ to get a sampled $\vec{\mu}$. Generating 100 noise vectors with size k gives 100 samples of $\vec{\mu}$. For each $\vec{\mu}$ we uniformly sample 100 (if exist) subsets $S \subset k, |S| = m$ ($m = k/2$ in (c) and $m = k/4$ in (d)), to generate up to 10k in-

stances. We set $n_i = 100$ for $i \in S$ and $n_i = 1$ for $i \notin S$. For each instance, we run each algorithm 100 times and use the average performance to approximate the expected simple regret. We then select the algorithm with the best average performance for each instance and count the fraction of instances where each algorithm performs the best. Experiment details are provided in the supplementary material. Error bars are representing the standard deviation of the reported fraction over 5 different runs of the whole procedure. Figure 6.2(c) and 6.2(d) shows that the fraction of instances where LCB is the best quickly approaches 1 as k increases.

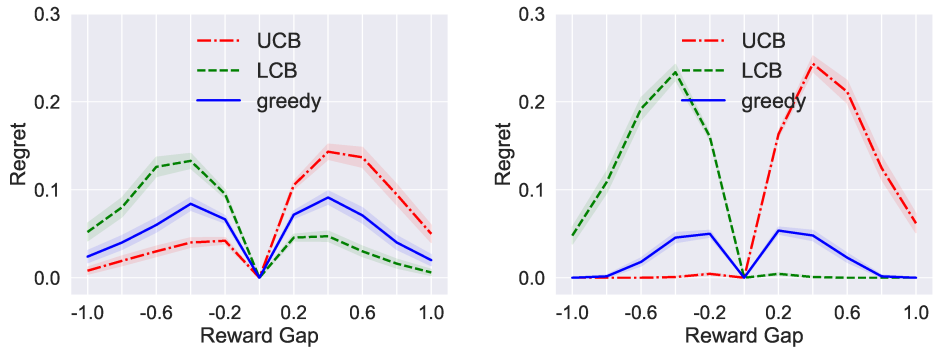


(a) LCB-1 (b) LCB-2



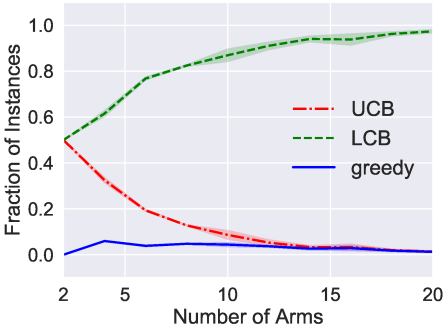
(c) UCB-1 (d) UCB-2

Figure 6.1: Comparing UCB, LCB and greedy on synthetic problems (with $k = 100$). (a) and (b): Problem instances where LCB has the best performance. The data set is generated by a behavior policy that pulls an arm i with *high* frequency and the other arms uniformly. In (a) i is the best arm while in (b) i is the 10th-best arm. (c) and (d): Problem instances where UCB has the best performance. The data set is generated by a behavior policy that pulls a set of good arms $\{j : j \leq i\}$ with very *small* frequency and the other arms uniformly. In (c) we use $i = 1$ while in (d) we use $i = 10$. Experiment details are provided in the supplementary material.

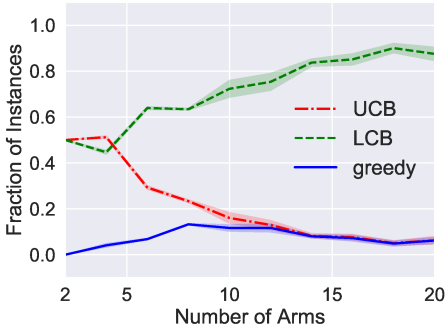


(a) $k = 2, n_1/n_2 = 2$

(b) $k = 2, n_1/n_2 = 10$



(c) $|S| = k/2$



(d) $|S| = k/4$

Figure 6.2: Comparing UCB, LCB and greedy on synthetic problems. (a) and (b): A set of two-armed bandit instances where both LCB and UCB dominate half of the instances. (c) and (d): For each k, we first sample 100 vectors $\vec{\mu} = [\mu_1, ..., \mu_k]$ and for each $\vec{\mu}$ we uniformly sample 100 (if exist) subsets $S \subset k, |S| = m$ ($m = k/2$ in (c) and $m = k/4$ in (d)), to generate up to 10k instances. We then count the fraction of instances where each algorithm performs better than the other two algorithms among the randomly sampled set of instances. Experiment details are provided in the supplementary material.

6.4.2 Instance-dependent Lower Bound

We have established that, despite all being minimax optimal, index algorithms with different adjustment can exhibit very different performance on specific problem instances. One might therefore wonder if instance optimal algorithms exist for batch policy optimization with finite-armed stochastic bandits. To answer this question, we next show that there is no instance optimal algorithm in the batch optimization setting for stochastic bandits, which is a very different outcome from the setting of cumulative regret minimization for online

stochastic bandits.

For cumulative regret minimization, (Lai & Robbins, 1985) introduced an asymptotic notion of instance optimality (Lattimore & Szepesvári, 2020). The idea is to first remove algorithms that are insufficiently adaptive, then define a yardstick (or benchmark) for each instance as the best (normalized) asymptotic performance that can be achieved with the remaining adaptive algorithms. An algorithm that meets this benchmark over all instances is then considered to be an instance optimal algorithm.

When adapting this notion of instance optimality to the batch setting there are two decisions that need to be made: what is an appropriate notion of "sufficient adaptivity" and whether, of course, a similar asymptotic notion is sought or optimality can be adapted to the finite sample setting. Here, we consider the asymptotic case, as one usually expects this to be easier.

We consider the 2-armed bandit case ($k = 2$) with Gaussian reward distributions $\mathcal{N}(\mu_1, 1)$ and $\mathcal{N}(\mu_2, 1)$ for each arm respectively. Recall that, in this setting, fixing $\mathbf{n} = (n_1, n_2)$ each instance $\theta \in \Theta_{\mathbf{n}}$ is defined by (μ_1, μ_2). We assume that algorithms only make decisions based on the sufficient statistic — empirical means for each arm, which in this case reduces to $X = (X_1, X_2, \mathbf{n})$ with $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$.

To introduce an asymptotic notion, we further denote $n = n_1 + n_2$, $\pi_1 = n_1/n$, and $\pi_2 = n_2/n = 1 - \pi_1$. Assume $\pi_1, \pi_2 > 0$; then each $\mathbf{n}$ can be uniquely defined by (n, π_1) for $\pi_1 \in (0, 1)$. We also ignore the fact that n_1 and n_2 should be integers since we assume the algorithms can only make decisions based on the sufficient statistic $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$, which is well defined even when n_i is not an integer.

Definition 2 (c-Minimax Optimal)**.** *Given a constant $c \geq 1$, an algorithm is said to be minimax optimal if its worst case regret is bounded by the minimax value of the problem up to a multiplicative factor c. We define the set of c-minimax optimal algorithms as*

$$\mathcal{M}_{\mathbf{n},c} = \left\{ \mathcal{A} : \sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}, \theta) \leq c \cdot \inf_{\mathcal{A}'} \sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}', \theta) \right\}.$$

Definition 3 (Instance-dependent Lower Bound)**.** *Given a set of algorithms $\mathcal{M}$, for each $\theta \in \Theta_{\mathbf{n}}$, we define the instance-dependent lower bound as $\mathcal{R}^*_{\mathcal{M}}(\theta) = \inf_{\mathcal{A}\in\mathcal{M}} \mathcal{R}(\mathcal{A}, \theta)$.*

The following theorem states the non-existence of instance optimal algorithms up to a constant multiplicative factor.

Theorem 11. *Let c_0 be the constant in minimax lower bound such that*

$$\inf_{\mathcal{A}} \sup_{\theta\in\Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}, \theta) \geq c_0/\sqrt{n_{\min}}\,.$$

Then for any $c > 2/c_0$ and any algorithm $\mathcal{A}$ we have

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} \geq \frac{n_{\min}}{n_{\min}+4} e^{\frac{\beta^2}{4}+\frac{\beta}{4}\sqrt{n_{\min}}}$$

where $\beta = cc_0 - 2$.

Corollary 5. *There is no algorithm that is instance optimal up to a constant multiplicative factor. That is, fixing $\pi_1 \in (0, 1)$, given any $c > 2/c_0$ and for any algorithm $\mathcal{A}$, we have*

$$\limsup_{n\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} = +\infty\,.$$

The proof of Theorem 11 follows by constructing two competing instances where the performance of any single algorithm cannot simultaneously match the performance of the adapted algorithm on each specific instance. Here we briefly discuss the proof idea – the detailed analysis is provided in the supplementary material.

Step 1, define the algorithm $\mathcal{A}_\beta$ as

$$\mathcal{A}_\beta(X) = \begin{cases} 1 & \text{if } X_1 - X_2 \geq \frac{\beta}{\sqrt{n_{\min}}} \\ 2 & \text{otherwise} \end{cases}.$$

For any β within a certain range, it can be shown that $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$, hence $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_\beta, \theta)$.

Step 2, construct two problem instances as follows. Fix a $\lambda \in \mathbb{R}$ and $\eta > 0$, and define

$$\theta_1 = (\mu_1, \mu_2) = (\lambda + \frac{\eta}{n_1}, \lambda - \frac{\eta}{n_2})\,,$$

$$\theta_2 = (\mu_1', \mu_2') = (\lambda - \frac{\eta}{n_1}, \lambda + \frac{\eta}{n_2}) .$$

Since we have $X_1 - X_2 \sim \mathcal{N}(\Delta, \sigma^2)$ on instance θ_1 and $X_1 - X_2 \sim \mathcal{N}(-\Delta, \sigma^2)$ on instance θ_2, where $\Delta = (\frac{1}{n_1} + \frac{1}{n_2})\eta$ and $\sigma^2 = \frac{1}{n_1} + \frac{1}{n_2}$, the regret of $\mathcal{A}_\beta$ on both instances can be computed using the CDF of Gaussian distributions. Note that $\mathcal{R}(\mathcal{A}_{-\beta}, \theta_1) = \mathcal{R}(\mathcal{A}_\beta, \theta_2)$. We now chose a $\beta_1 < 0$ for θ_1 to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)$ by $\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)$ and use $\beta_2 = -\beta_1 > 0$ to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)$ by $\mathcal{R}(\mathcal{A}_{\beta_2}, \theta_1)$.

Then applying the Neyman-Pearson Lemma (Neyman & Pearson, 1933) to this scenario gives that $\mathcal{A}_0$ is the optimal algorithm in terms of balancing the regret on θ_1 and θ_2:

$$\mathcal{R}(\mathcal{A}_0, \theta_1) = \mathcal{R}(\mathcal{A}_0, \theta_2) = \min_{\mathcal{A}} \max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\} .$$

Step 3, combining the above results gives

$$\begin{aligned}
\sup_{\theta \in \Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} &\geq \max\left\{ \frac{\mathcal{R}(\mathcal{A}, \theta_1)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)}, \frac{\mathcal{R}(\mathcal{A}, \theta_2)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)} \right\} \\
&\geq \max\left\{ \frac{\mathcal{R}(\mathcal{A}, \theta_1)}{\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)}, \frac{\mathcal{R}(\mathcal{A}, \theta_2)}{\mathcal{R}(\mathcal{A}_{\beta_2}, \theta_2)} \right\} \\
&= \frac{\max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\}}{\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)} \\
&\geq \frac{\mathcal{R}(\mathcal{A}_0, \theta_1)}{\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)} .
\end{aligned}$$

Note that both the regret $\mathcal{R}(\mathcal{A}_0, \theta_1)$ and $\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)$ can be exact expressed as CDFs of Gaussian distributions: $\mathcal{R}(\mathcal{A}_0, \theta_1) = \Phi(-\Delta/\sigma)$ and $\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1) = \Phi\left(-\beta/(\sigma\sqrt{n_{\min}}) - \Delta/\sigma\right)$ where Φ is the CDF of the standard normal distribution. Now we can conclude the proof by picking $\lambda = 1/2$ and $\eta = n_{\min}/2$ such that $\theta_1, \theta_2 \in [0, 1]^2$. Then the result in Theorem 11 can be proved by applying an approximation of Φ and setting $\beta_1 = -\beta_2 = 2 - cc_0$ such that both β_1 and β_2 are within the range that makes $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$.

To summarize, for any algorithm that performs well on some problem instance, there exists another instance where the same algorithm suffers arbitrarily larger regret. Therefore, any reasonable algorithm is equally optimal, or not optimal, depending on whether the minimax or instance optimality

is considered. In this sense, there remains a lack of a well-defined optimality criterion that can be used to choose between algorithms for batch policy optimization.

6.5 A Characterization of Pessimism

It is known that the pessimistic algorithm, maximizing a lower confidence bound on the value, satisfies many desirable properties: it is consistent with rational decision making using preferences that satisfy uncertainty aversion and certainty-independence (Gilboa & Schmeidler, 1989), it avoids the optimizer's curse (Smith & Winkler, 2006a), it allows for optimal inference in an asymptotic sense Lam (2019), and in a certain sense it is the unique strategy that achieves these properties (Van Parys et al., 2017; Sutter et al., 2020). However, a pure statistical decision theoretic justification (in the sense of (Berger, 1985)) is still lacking.

The instance-dependent lower bound presented above attempts to characterize the optimal performance of an algorithm on an instance-by-instance basis. In particular, one can interpret the objective $\mathcal{R}(\mathcal{A}, \theta)/\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$ defined in Theorem 11 as weighting each instance θ by $1/\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$, where this can be interpreted as a measure of instance difficulty. It is natural to consider an algorithm to be optimal if it can perform well relative to this weighted criteria. However, given that the performance of an algorithm can be arbitrarily different across instances, no such optimal algorithm can exist under this criterion. The question we address here is whether other measures of instance difficulty might be used to distinguish some algorithms as naturally advantageous over others.

In a recent study, Jin et al. (2020c) show that the pessimistic algorithm is minimax optimal when weighting each instance by the variance induced by the optimal policy. In another recent paper, Buckman et al. (2020) point out that the pessimistic choice has the property that its regret improves whenever the optimal choice's value is easier to predict. In particular, with our notation, their most relevant result (Theorem 3) implies the following: if b_i defines an

interval such that $\mu_i \in [\hat{\mu}_i - b_i, \hat{\mu}_i + b_i]$ for all $i \in [k]$, then for $i' = \arg\max_i \hat{\mu}_i - b_i$ one obtains [1]

$$\mu^* - \mu_{i'} \leq 2b_{a^*} \,. \tag{6.5}$$

If we (liberally) interpret b_{a^*} as a measure of how hard it is to predict the value of the optimal choice, this inequality suggests that the pessimistic choice could be justified as the choice that makes the regret comparable to the error of predicting the optimal value.

To make this intuition precise, consider the same problem setup as discussed in Section 6.2. Suppose that the reward distribution for each arm $i \in [k]$ is a Gaussian with unit variance. Consider the problem of estimating the optimal value μ^* where the optimal arm a^* is also provided to the estimator. We define the set of minimax optimal estimators.

Definition 4 (Minimax Estimator)**.** *For fixed* $\mathbf{n} = (n_i)_{i\in[k]}$*, an estimator is said to be minimax optimal if its worst case error is bounded by the minimax estimate error of the problem up to some constant. We define the set of minimax optimal estimators as*

$$\mathcal{V}^*_{\mathbf{n}} = \left\{ \nu : \sup_{\theta \in \Theta_{\mathbf{n}}} \mathbb{E}_\theta[|\mu^* - \nu|] \leq c \inf_{\nu' \in \mathcal{V}} \sup_{\theta \in \Theta_{\mathbf{n}}} \mathbb{E}_\theta[|\mu^* - \nu'|] \right\}$$

where c is a universal constant, and $\mathcal{V}$ is the set of all possible estimators.

Now consider using this optimal value estimation problem as a measure of how difficult a problem instance is, and then use this to weight each problem instance as in the definition of instance-dependent lower bound. In particular, let

$$\mathcal{E}^*(\theta) = \inf_{\nu \in \mathcal{V}^*_{\mathbf{n}}} \mathbb{E}_\theta[|\mu^* - \nu|]$$

[1] This inequality follows directly from the definitions: $\mu^* - \mu_{i'} \leq \mu^* - (\hat{\mu}_{i'} - b_{i'}) \leq \mu^* - (\hat{\mu}_{a^*} - b_{a^*}) \leq 2b_{a^*}$ and we believe this was known as a folklore result, although we are not able to point to a previous paper that includes this inequality. The logic of this inequality is the same as that used in proving regret bounds for UCB policies (Buckman et al., 2020; Lattimore & Szepesvári, 2020). It is also clear that the result holds for any data-driven stochastic optimization problem regardless of the structure of the problem. Theorem 3 of (Buckman et al., 2020) with this notation states that $\mu^* - \mu_{i'} \leq \min_i \mu^* - \mu_i + 2b_i$.

be the inherent difficulty of estimating the optimal value μ^* on problem instance θ. The result (6.5) suggests (but does not prove) that $\sup_\theta \frac{\mathcal{R}(\mathrm{LCB},\theta)}{\mathcal{E}^*(\theta)} < +\infty$. We now show that not only does this hold, but up to a constant factor, the LCB algorithm is nearly weighted minimax optimal with the weighting given by $\mathcal{E}^*(\theta)$.

Proposition 3. *For any* $\mathbf{n} = (n_i)_{i\in[k]}$,

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathrm{LCB},\theta)}{\mathcal{E}^*(\theta)} < c\sqrt{\log|\mathbf{n}|}\,,$$

where c is some universal constant.

Proposition 4. *There exists a sequence* $\{\mathbf{n}_j\}$ *such that*

$$\limsup_{j\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\mathrm{UCB},\theta)}{\sqrt{\log|\mathbf{n}_j|}\cdot\mathcal{E}^*(\theta)} = +\infty$$
$$\limsup_{j\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\mathrm{greedy},\theta)}{\sqrt{\log|\mathbf{n}_j|}\cdot\mathcal{E}^*(\theta)} = +\infty$$

That is, the pessimistic algorithm can be justified by weighting each instance using the difficulty of predicting the optimal value. We note that this result does not contradict the no-instance-optimality property of batch policy optimization with stochastic bandits (Corollary 5). In fact, it only provides a characterization of pessimism: the pessimistic choice is beneficial when the batch dataset contains enough information that is good for predicting the optimal value.

6.6 Related work

In the context of offline bandit and RL, a number of approaches based on the pessimistic principle have been proposed and demonstrated great success in practical problems (Swaminathan & Joachims, 2015; Wu et al., 2019; Jaques et al., 2019; Kumar et al., 2019, 2020; Buckman et al., 2020; Kidambi et al., 2020; Yu et al., 2020; Siegel et al., 2020). We refer interested readers to the survey by (Levine et al., 2020) for recent developments on this topic. To

implement the pessimistic principle, distributional robust optimization (DRO) has become a powerful tool in bandits (Faury et al., 2019; Karampatziakis et al., 2019) and RL (Xu & Mannor, 2010; Yu & Xu, 2015; Yang, 2017; Chen et al., 2019; Dai et al., 2020; Derman & Mannor, 2020).

From a theoretical perspective, the statistical properties of general DRO, *e.g.*, the consistency and asymptotic expansion of DRO, is analyzed in (Duchi et al., 2016). Liu et al. (2020) provides regret analysis for a pessimistic algorithm based on stationary distribution estimation in offline RL with insufficient data coverage. Jin et al. (2020c) and Kidambi et al. (2020) recently prove that the pessimistic algorithm is nearly minimax optimal for batch policy optimization. However, the theoretical justification of the benefits of pessimitic principle vs. alternatives are missing in offline RL.

Decision theory motivates DRO with an axiomatic characterization of minmax (or distributionally robust) utility: Preferences of decision makers who face an uncertain decision problem and whose preference relationships over their choices satisfy certain axioms follow an ordering given by assigning maxmin utility to these preferences (Gilboa & Schmeidler, 1989). Thus, if we believe that the preferences of the user follow the axioms stated in the above work, one must use a distributionally optimal (pessimistic) choice. On the other hand, (Smith & Winkler, 2006b) raise the "optimizer's curse" due to statistical effect, which describes the phenomena that the resulting decision policy may disappoint on unseen out-of-sample data, *i.e.*, the actual value of the candidate decision is below the predicted value. Van Parys et al. (2017); Sutter et al. (2020) justify the optimality of DRO in combating with such an overfitting issue to avoid the optimizer's curse. Moreover, Delage et al. (2019) demonstrate the benefits of randomized policy from DRO in the face of uncertainty comparing with deterministic policy. While reassuring, these still leave open the question whether there is a justification for the pessimistic choice dictated by some alternate logic, or perhaps a more direct logic reasoning in terms of regret in decision problem itself (Lattimore & Szepesvári, 2020).

Our theoretical analysis answer this question, and provides a complete and direct justification for all confidence-based index algorithms. Specifically,

we show all confidence-based index algorithms are nearly minimax optimal in terms of regret. More importantly, our instance-dependent analysis shows that for any algorithm one can always find a problem instance where the algorithm will suffer arbitrarily large regret. Therefore, one cannot directly compare the performance of two algorithms without specifying the problem instance. Buckman et al. (2020) state that for the pessimistic choice to be a good one, it suffices to have data that makes predicting the value of the optimal policy feasible. We provide a formal analysis to support this intuition: the pessimistic algorithm is nearly minimax optimal when weighting individual instance by its inherent difficulty of estimating the optimal value. This weighted criterion can be used to distinguish pessimistic algorithm from other confidence-adjusted index algorithms.

6.7 Conclusion

In this chapter we study the statistical limits of batch policy optimization with finite-armed bandits. We introduce a family of confidence-adjusted index algorithms that provides a general analysis framework to unify the commonly used optimistic and pessimistic principles. For this family, we show that any index algorithm with an appropriate adjustment is nearly minimax optimal. Our analysis also reveals another important finding, that for any algorithm that performs optimally in some environment, there exists another environment where the same algorithm can suffer arbitrarily large regret. Therefore, the instance-dependent optimality cannot be achieved by any algorithm. To distinguish the algorithms in offline setting, we introduce a weighted minimax objective and justify the pessimistic algorithm is nearly optimal under this criterion.

Chapter 7

The Curse of Passive Data Collection in Batch Reinforcement Learning

7.1 Introduction

Batch reinforcement learning (RL) broadly refers to the problem of finding a policy with high expected return in a stochastic control problem when only a batch of data collected from the controlled system is available. Here, we consider this problem for finite state-action Markov decision processes (MDPs), with or without function approximation, when the data is in the form of trajectories obtained by following some *logging policy*. In more detail, the trajectories are composed of sequences of states, actions, and rewards, where the action is chosen by the logging policy, and the next states and rewards follow the distributions specified by the MDP's transition parameters.

There are two subproblems underlying batch RL: the *design problem*, where the learner needs to specify a data collection mechanism that will be used to collect the batch data; and the *policy optimization problem*, where the learner needs to specify the algorithm that produces a policy given the batch data. For the design problem, often times one can use an adaptive data collection process where the next action to be taken is determined by the past data. Another way to say this is that the data collection is done in an *active* way. Recent theoretical advances in reward-free exploration (e.g., Jin et al., 2020a; Kaufmann et al., 2021; Zhang et al., 2021b) show that one can design algo-

rithms to collect a batch data set with only polynomial samples to have good coverage over all possible scenarios in the environment. Near-optimal policies can be obtained for any given reward functions using standard policy optimization algorithms with the collected data. A complication arises in applications, such as healthcare, education, autonomous driving, or hazard management, where active data collection is either impractical or dangerous (Levine et al., 2020). In these applications, the best one could do is to collect data using a fixed, logging policy, which needs to be chosen *a priori*, that is before the data collection process begins, so that the stakeholders can approve it. Arguably, this is the most natural problem setting to consider in batch learning. The fundamental questions are: how should one choose the logging policy so as to maximize the chance of obtaining a good policy with as little data as possible and how many samples are sufficient and necessary to obtain a near optimal policy given a logging policy, and which algorithm to use to obtain such a policy?

Perhaps surprisingly, before this work, these questions remained unanswered. In particular, while much work have studied the sample complexity of batch RL, the results in these works are focusing only on the policy optimization subproblem and as such fall short in providing an answer to our questions. In particular, some authors give sample complexity upper and lower bounds as a function of a parameter, d_m, which could be the smallest visit probability of state-action pairs under the logging policy (Chen & Jiang, 2019; Yin & Wang, 2020; Yin et al., 2021a; Ren et al., 2021; Uehara et al., 2021; Yin & Wang, 2021; Xie & Jiang, 2021; Xie et al., 2021a), or the smallest ratio of visit probabilities of the logging versus the optimal policies, again over all state-action pairs (Liu et al., 2019, 2020; Yin et al., 2021b; Jin et al., 2021; Rashidinejad et al., 2021; Xie et al., 2021b). The sample complexity results depend polynomially on $1/d_m$, S, A and H, where H is the episode length or the effective horizon. Although these results are valuable in informing us the policy optimization step of batch RL, they provide no clue as to how to choose the logging policy to get a high value for d_m and whether d_m will be uniformly bounded from below when adopting such a logging policy. In particular, if

we take the first definition for d_m, in some MDPs d_m will be zero regardless of the logging policy if some state is not accessible from the initial distribution. While this predicts an infinite sample complexity for our problem, this is clearly too conservative, since if a state is not accessible under any policy, it is unimportant to learn about it. This is corrected by the second definition. However, even with this definition it remains unclear whether d_m will be uniformly bounded away from zero for an MDP with a fixed number of states and actions and the best instance-agnostic choice of the logging policy. The lower bounds in these work also fail to provide a lower bound for our setting. This is because in these lower bounds the instance will include an adversarially chosen logging policy, again falling short of helping us. Essentially, these are the gaps that we fill with this chapter.

In particular, we first show that in tabular MDPs the number of transitions necessary and sufficient to obtain a good policy, *the sample complexity of learning*, is an exponential function of the minimum of the number of states and the planning horizon. In more details, we prove that the sample complexity of obtaining ϵ-optimal policies is at least $\Omega(\mathrm{A}^{\min(\mathrm{S}-1,H+1)})$ for γ-discounted problems, where S is the number of states, A is the number of actions (per state), and H is the effective horizon defined as $H = \lfloor \frac{\ln(1/\epsilon)}{\ln(1/\gamma)} \rfloor$. For finite horizon problems with horizon H, we prove the analogue $\Omega(\mathrm{A}^{\min(\mathrm{S}-1,H)}/\varepsilon^2)$ lower bound. These results for tabular MDPs immediately imply exponential lower bounds when linear value function approximation is applied with S replaced by d, the number of features. We also show that warm starts (when one starts with a policy which is achieving almost as much as the optimal policy) do not help either, crushing the hope that one the "curse" of passivity can be broken by adopting a straightforward two-phase data collection process (Bai et al., 2019; Zhang et al., 2020; Gao et al., 2021). We then establish nearly matching upper bounds for both the plug-in algorithm and pessimistic algorithm, showing that the sample complexity behaves essentially as shown in the lower bounds. While the upper bounds for these two algorithms may be off by a polynomial factor, we do not expect the pessimistic algorithm to have a major advantage over the plug-in method in the worst-case setting. In fact,

the recent work of Xiao et al. (2021c) established this in a rigorous fashion for the bandit setting by showing an algorithm independent lower bound that matched the upper bound for both the plug-in method and the pessimistic algorithm. In the average reward case we show that the sample complexity is infinite.

How do our results relate to the expectations of RL practitioners (and theoreticians)? We believe that most members of our community recognize that batch RL is hard at the fundamental level. Yet, it appears that many in the community are still highly optimistic about batch RL, as demonstrated by the numerous empirical papers that document successes of various levels and kinds (e.g., Laroche et al., 2019; Kumar et al., 2019; Wu et al., 2019; Jaques et al., 2019; Agarwal et al., 2020c; Kidambi et al., 2020; Yu et al., 2020; Gulcehre et al., 2020; Fu et al., 2020), or by the optimistic tone of the above-mentioned theoretical results. The enthusiasm of the community is of course commendable and nothing is farthest from our intentions than to break it. In connection to this, we would like to point out that our results show that if *either* H, *or* S (or d when we have d features) is fixed, batch RL is in fact tractable. Yet, the lower bound assures us that we cannot afford batch RL if *both* of these parameters are large, a result which one should not hide from. Perhaps the most important finding here is the curious interplay between the horizon and the number of states (more generally, we expect a complexity parameter of the MDP to stand here), which is reasonable yet we find it non-obvious. Certainly, the proof that shows that the interplay is "real" required some new ideas. Returning to the empirical works, recent studies identify the tandem effect from the issue of function approximation in the batch RL with passive data collection (Ostrovski et al., 2021). Our results suggest that there is a need to rethink how batch RL methods are benchmarked. In particular, a tedious examination of the benchmarks shows that the promising results are almost always produced on data sets that are collected by a noisy version of a near-optimal policy. The problem is that this choice biases the development of algorithms towards those that exploit this condition (e.g., the pessimistic

algorithm), yet, this mode of data collection is unrealistic.[1]

Another highlight of our result is that it implies an exponential gap between the sample complexity of passive and active learning. Recently, a similar conclusion is drawn in the paper of Zanette (2021), which served as the main motivation for our work. Zanette (2021) demonstrated an exponential separation for the case when batch learning is used in the presence of linear function approximation. A careful reading of the paper shows that the lower bound shown there does not apply to the tabular setting as the data collection method of this chapter allows sampling from any distribution over the state-action space, which, in the tabular setting is sufficient for polynomial sample complexity.

7.2 Notation and Background

Notation We let $\mathbb{R}$ denote the set of real numbers, and for a positive integer i, let $[i] = \{0, \dots, i-1\}$ be the set of integers from 0 to $i-1$. We also let $\mathbb{N} = \{0, 1, \dots\}$ be the set of nonnegative integers and $\mathbb{N}_+ = \{1, 2, \dots\}$ be the set of positive integers. For a finite set $\mathcal{X}$, we use $\Delta(\mathcal{X})$ to denote the set of probability distributions over $\mathcal{X}$. We also use the same notation for infinite sets when the set has a clearly identifiable measurability structure such as $\mathbb{R}$, which in this context is equipped by the σ-algebra of Borel measurable sets. We use $\mathbb{I}$ to denote the indicator function. We also use $\mathbf{1}$ to be the identically one function/vector; the domain/dimension is so that the expression that involves $\mathbf{1}$ is well-defined.

We consider finite Markov decision processes (MDPs) given by $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are finite set of states and actions, P is the transition function, r is the reward function, and γ is the discounted factor. See Section 5.1 for the definition of MDP. Since $\mathcal{S}$ and $\mathcal{A}$ are finite, without loss of generality, we

[1] Qin et al. (2021) points to another problem with the benchmarks; namely that they fail to compare to the noise-free version of the near-optimal policy used in the data collection. Brandfonbrener et al. (2021) observe that simply doing one step of constrained/regularized policy improvement using an value estimate of the behavior policy performs surprisingly well in offline RL benchmarks. They hypothesize that the strong performance is due to a combination of favorable structure in the environment and behavior policy.

assume that the immediate mean rewards lie in the $[-1, 1]$ interval. We also assume that the reward distribution is ρ-subgaussian with a constant $\rho > 0$. We denote $\mathcal{M}(\mathcal{S}, \mathcal{A})$ the set of MDPs sharing the same $\mathcal{S}$ and $\mathcal{A}$. Since the identity of the states and actions plays no role, without loss of generality, in what follows we assume that $\mathcal{S} = [\mathrm{S}]$ and $\mathcal{A} = [\mathrm{A}]$ for some S, A positive integers. We also use $\mathcal{M}(\mathrm{S}, \mathrm{A})$ to denote the set of MDPs with S states and A actions (say, over the canonical sets $\mathcal{S} = [\mathrm{S}]$ and $\mathcal{A} = [\mathrm{A}]$). Finally, for $\varepsilon > 0$, we define the effective horizon $H_{\gamma,\varepsilon} := \lfloor \frac{\ln(1/\epsilon)}{\ln(1/\gamma)} \rfloor$, which is different with the normally used effective horizon Eq. (5.7) with only a $\ln(1/(1-\gamma))$ factor.

The standard goal in a finite MDP under the discounted criterion is to identify the optimal policy π^* that maximizes the value function in every state $s \in \mathcal{S}$ such that $v^*(s) = \sup_\pi v^\pi(s)$. In this chapter though, we consider the less demanding problem of finding a policy π that maximizes $v^\pi(\mu)$ for a fixed initial state distribution μ, i.e., finding a policy π which achieves, or nearly achieves $v^*(\mu)$. For an initial state distribution $\mu \in \Delta(\mathcal{S})$ and a policy π, recall that the (unnormalized) discounted occupancy measure ν_μ^π Eq. (5.8) induced by μ, π, and the MDP, is defined by

$$\nu_\mu^\pi(s, a) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(S_t = s, A_t = a | S_0 \sim \mu) . \tag{7.1}$$

As shown in Eq. (5.13), $v^\pi(\mu)$ can be represented as an inner product between the immediate reward function r and the occupancy measure ν_μ^π

$$v^\pi(\mu) = \sum\nolimits_{s,a} r(s, a) \nu_\mu^\pi(s, a) = \langle \nu_\mu^\pi, r \rangle . \tag{7.2}$$

7.3 Batch Policy Optimization

We consider *policy optimization* in a *batch mode*, or, in short, *batch policy optimization* (BPO). A *BPO problem* for a fixed sample size n is given by the tuple $\mathcal{B} = (\mathcal{S}, \mathcal{A}, \mu, n, \mathcal{P})$ where $\mathcal{S}$ and $\mathcal{A}$ are finite sets, μ is a probability distribution over $\mathcal{S}$, n is a positive integer, and $\mathcal{P}$ is a set of MDP-distribution pairs of the form (M, G), where $M \in \mathcal{M}(\mathcal{S}, \mathcal{A})$ is an MDP over $(\mathcal{S}, \mathcal{A})$ and G is a probability distribution over $(\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S})^n$. In what follows a pair (M, G) of the above form will be called a BPO instance.

A *BPO algorithm* for a given sample size n and sets $\mathcal{S}, \mathcal{A}$ takes data $\mathcal{D} \in (\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S})^n$ and returns a policy π (possibly history-dependent). Ignoring computational aspects, we will identify BPO algorithms with (possibly randomized) maps $\mathcal{L} : (\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S})^n \to \Pi$, where Π is the set of all policies. The aim is to find BPO algorithms that find near-optimal policies with high probability on every instance within a BPO problem:

Definition 5 ((ε, δ)-sound algorithm)**.** *Fix $\epsilon > 0$ and $\delta \in (0, 1)$. A BPO algorithm $\mathcal{L}$ is (ε, δ)-sound on instance (M, G) given initial state distribution μ if*

$$\mathbb{P}_{\mathcal{D} \sim G} \left(v^{\mathcal{L}(\mathcal{D})}(\mu) > v^*(\mu) - \varepsilon \right) > 1 - \delta \,,$$

where the value functions are for the MDP M. Further, we say that a BPO algorithm is (ϵ, δ)-sound on a BPO problem $\mathcal{B} = (\mathcal{S}, \mathcal{A}, \mu, n, \mathcal{P})$ if it is sound on any $(M, G) \in \mathcal{P}$ given the initial state distribution μ.

Data collection mechanisms A *data collection mechanism* is a way of arriving at a distribution G over the data given an MDP and some other inputs, such as the sample size. We consider two types of data collection mechanisms. One of them is governed by a distribution over the state-action pairs, the other is governed by a policy and a way of deciding how a fixed sample size n should be split up into episodes in which the policy is followed. We call the first SA-sampling, the second policy-induced data collection.

- *SA-sampling:* An SA-sampling scheme is specified by a probability distribution $\mu_{\log} \in \Delta(\mathcal{S} \times \mathcal{A})$ over the state-action pairs. For a given sample size n, $\mu_{\log}$ together with an MDP M induces a distribution $G_n(M, \mu_{\log})$ over n tuples $\mathcal{D} = (S_i, A_i, R_i, S'_i)_{i=0}^{n-1}$ so that the elements of this sequence form an i.i.d. sequence such that for any $i \in [n]$, $(S_i, A_i) \sim \mu_{\log}$, $(R_i, S'_i) \sim Q(\cdot|S_i, A_i)$.

- *Policy-induced data collection*: A policy induced data collection scheme is specified by $(\pi_{\log}, \mathbf{h})$, where $\pi_{\log} : \mathcal{S} \to \Delta(\mathcal{A})$ is a policy, which we shall call the *logging policy*, and $\mathbf{h} = (\mathbf{h}_n)_{n \geq 1}$: For each $n \geq 1$, $\mathbf{h}_n$ is an m-tuple

$(h_j)_{j\in[m]}$ of positive integers for some m, specifying the length of the m episodes in the data whose total length is n. Then, for any n, the pair $(\pi_{\log}, \mathbf{h}_n)$ together with an MDP M and an initial distribution μ induces a distribution $G(M, \pi_{\log}, \mathbf{h}_n, \mu)$ over the n tuples $\mathcal{D} = (S_i, A_i, R_i, S'_i)_{i=0}^{n-1}$ as follows: The data consists of m episodes, with episode $j \in [m]$ having length h_j and taking the form $\tau_j = (S_0^{(j)}, A_0^{(j)}, R_0^{(j)}, \ldots, S_{h_j-1}^{(j)}, A_{h_j-1}^{(j)}, R_{h_j-1}^{(j)}, S_{h_j}^{(j)})$, where $S_0^{(j)} \sim \mu$, $A_t^{(j)} \sim \pi_{\log}(\cdot|S_t^{(j)})$, $(R_t^{(j)}, S_{t+1}^{(j)}) \sim Q(\cdot|S_t^{(j)}, A_t^{(j)})$. Then, for $i \in [n]$, $(S_i, A_i, R_i, S'_i) = (S_t^{(j)}, A_t^{(j)}, R_t^{(j)}, S_{t+1}^{(j)})$ where $j \in [m]$, $t \in [h_j]$ are unique integers such that $i = \sum_{j'<j} h_j + t$.

Now, under SA-sampling, the sets $\mathcal{S}$, $\mathcal{A}$, a logging distribution $\mu_{\log}$ and state-distribution μ over the respective sets give rise to the BPO problem $\mathcal{B}(\mu_{\log}, \mu, n) = (\mathcal{S}, \mathcal{A}, \mu, n, \mathcal{P}(\mu_{\log}, n))$, where $\mathcal{P}(\mu_{\log}, n)$ is the set of all pairs of the form $(M, G_n(M, \mu_{\log}))$, where $M \in \mathcal{M}(\mathcal{S}, \mathcal{A})$ is an MDP with the specified state-action spaces and $G_n(M, \mu_{\log})$ is defined as above. Similarly, a fixed policy $\pi_{\log}$, fixed episode lengths $\mathbf{h} \in \mathbb{N}_+^m$ for some m integer and a fixed state-distribution μ give rise to a BPO problem $\mathcal{B}(\pi_{\log}, \mu, \mathbf{h}) = (\mathcal{S}, \mathcal{A}, \mu, |\mathbf{h}|, \mathcal{P}(\pi_{\log}, \mathbf{h}))$, where $\mathcal{P}(\pi_{\log}, \mathbf{h})$ is the set of pairs of the form $(M, G(M, \pi_{\log}, \mathbf{h}, \mu))$ where $M \in \mathcal{M}(\mathcal{S}, \mathcal{A})$ and $G(M, \pi_{\log}, \mathbf{h}, \mu)$ is a distribution as defined above. Here, we use $|\mathbf{h}|$ to denote $\sum_{s=0}^{m-1} h_s$ which is the sample size specified by $\mathbf{h}$.

The *sample-complexity of BPO with SA-sampling* for a given pair (ε, δ) and a criterion (discounted, finite horizon, or average reward) is the smallest integer n such that for each μ there exists a logging distribution $\mu_{\log}$ and a BPO algorithm $\mathcal{L}$ for this sample size such that $\mathcal{L}$ is (ε, δ)-sound on the BPO problem $\mathcal{B}(\mu_{\log}, \mu, n)$. Similarly, *the sample-complexity of BPO with policy-induced data collection* for a given pair (ε, δ) and a criterion is the smallest integer n such that for each μ there exists a logging policy $\pi_{\log}$ and episode lengths $\mathbf{h} \in \mathbb{N}_+^m$ with $|\mathbf{h}| = n$ and a BPO algorithm that is (ε, δ)-sound on $\mathcal{B}(\pi_{\log}, \mu, \mathbf{h})$.

The SA-sampling based data collection is realistic when there is a simulator that allows this type of data collection (Agarwal et al., 2020b; Azar et al., 2013; Cui & Yang, 2020; Li et al., 2020). Besides this scenario, it is hard to

imagine a case when SA-sampling can be realistically applied. Indeed, in most practical settings, data collection happens by following some policy, usually from the same initial state distribution that is used in the objective of policy optimization.

For policy-induced data collection, a key restriction on the logging policy is that it is chosen without any knowledge of the MDP. Moreover, that the logging policy is memoryless rules out any adaptation to the MDP. The intention here is to model a "tabula rasa" setting, which is relevant when one must find a good policy in a completely new environment but only passive data collection is available. However, our lower bound Corollary 7 shows that there is not much to be gained even if the logging policy is known to be a good policy: If the goal is to improve the suboptimality level of the logging policy, by saying, a factor of two, the exponential sample complexity lower bound still applies.

From a statistical perspective, the main difference between these two data collection mechanisms is that for policy-induced data-collection the distribution of (S_i, A_i) will depend on the specific MDP instance, while this is not the case for SA-sampling. As we shall see in the next section, this makes BPO under SA-sampling provably exponentially more efficient.

7.4 Lower Bounds

We first give a lower bound on the sample complexity for BPO when the data available for learning is obtained by following some logging policy:

Theorem 12 (Exponential sample complexity with policy-induced data collection in discounted problems)**.** *For any positive integers* S *and* A*, discount factor* $\gamma \in [0, 1)$ *and a pair* (ϵ, δ) *such that* $0 < \epsilon < 1/2$ *and* $\delta \in (0, 1)$*, any* (ϵ, δ)*-sound algorithm needs at least* $\Omega(\mathrm{A}^{\min(\mathrm{S}-1, H_{\gamma,2\varepsilon}+1)} \ln(1/\delta))$ episodes *of any length with policy-induced data collection for MDPs with* S *states and* A *actions under the* γ*-discounted total expected reward criterion. The result remains true if the MDPs are restricted to have deterministic transitions.*

Remark 5. *Random rewards are not essential in proving Theorem 12 as long as stochastic transitions are allowed: First, the proof can be modified to use*

Bernoulli rewards and stochastic transitions can be used to emulate Bernoulli rewards. Note also that for ρ-subgaussian random reward, the sample complexity in Theorem 12 becomes $\Omega(\max\{1,\rho^2\}\mathrm{A}^{\min(\mathrm{S}-1,H+1)}\ln(1/\delta))$. The maximum appears exactly because stochastic transitions can emulate Bernoulli rewards.

Simplifying things a bit, the theorem states that the sample complexity is exponential as the number of states and the planning horizon grow together and without a limit. Note that this is in striking contrast to sample complexity of learning actively, or even with SA-sampling, as we shall soon discuss it. The hard MDP instance used to construct the lower bound is adopted from the combination lock problem (Whitehead, 1991). The detailed proof is provided in the Appendix, as are the proofs of the other statements. By realizing that tabular MDPs can be considered as using one-hot features, the exponential lower bound still holds for linear function approximation.

Corollary 6 (Exponential sample complexity with linear function approximation in the discounted problem)**.** *Let d, A be positive integers. Then the same result as Theorem 12 with S replaced by d holds when the data collection happens for some MDP $M \in \mathcal{M}(\mathcal{S}, [\mathrm{A}])$ and in addition to the dataset the learner is also given access to a featuremap $\phi : \mathcal{S} \to \mathbb{R}^d$ such that for every policy π of this MDP, there exists $\theta \in \mathbb{R}^d$ such that $v^\pi(s) = \phi(s)^\top\theta$, $\forall s \in \mathcal{S}$, where v^π is the value function of π in M. The result also remains true if the MDPs are restricted to have deterministic transitions.*

One may wonder about whether this exponential complexity can be avoided if more is assumed about the logging policy. In particular, one may hope that improving on an already good logging policy (i.e., one that is close to optimal) should be easier. Our next result shows that this is not the case.

Corollary 7 (Warm starts do not help)**.** *Fix $\pi_{\log}$, $0 < \varepsilon < 1/2$, $\delta \in (0,1)$, S and A as before. Let $\mathcal{M}_{2\varepsilon}^{\pi_{\log}}$ denote a set of MDPs with deterministic transitions, state space $\mathcal{S} = [\mathrm{S}]$ and action space $\mathcal{A} = [\mathrm{A}]$ such that $\pi_{\log}$ is 2ε-optimal for all $M \in \mathcal{M}_{2\varepsilon}^{\pi_{\log}}$. Then for any length of sampled episodes, any (ε,δ)-sound algorithm needs at least $\Omega(\mathrm{A}^{\min(\mathrm{S}-1,H+1)}\ln(1/\delta))$ episodes, where $H = H_{\gamma,2\varepsilon}$.*

The third corollary shows that when the logging policy is not uniform, the lower bound gets worse.

Corollary 8 (The uniform policy is the best logging policy)**.** *If $\pi_{\log}$ is not uniform at every state, then the sample complexity in Theorem 12 increases, and in particular, A^u can be replaced by $\max_{\mathcal{S}' \subset \mathcal{S}, |\mathcal{S}'|=u} \prod_{s \in \mathcal{S}'} \max_a \frac{1}{\pi_{\log}(a|s)}$ where $u = \min(\mathrm{S}-1, H+1)$.*

For fixed-horizon policy optimization, we have the following result similar to Theorem 12.

Theorem 13 (Exponential sample complexity with policy-induced data collection in finite-horizon problems)**.** *For any positive integers* S *and* A*, planning horizon $H > 0$ and a pair (ϵ, δ) such that $0 < \epsilon < 1/2$ and $\delta \in (0,1)$, any (ϵ, δ)-sound algorithm needs at least $\Omega(\mathrm{A}^{\min(\mathrm{S}-1,H)} \ln(1/\delta)/\varepsilon^2)$ episodes with policy-induced data collection for MDPs with* S *states and* A *actions under the H-horizon total expected reward criterion. The result also remains true if the MDPs are restricted to have deterministic transitions.*

Remark 5 and Corollaries 6 to 8 also remain essentially true; we omit these to save space. As shown in the next result, the sample complexity could be even worse in average reward MDPs. The different sample complexities of the average reward problems and the two previous settings can be explained as follows: In discounted and finite-horizon problems, rewards beyond the planning horizon do not have to be observed in data to find a near optimal policy. In contrast, this is not the case for the *average reward* setting, where the value function is redefined to be

$$v^\pi(\mu) = \liminf_{T \to \infty} \mathbb{E}^\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} r(S_t, A_t) \,\middle|\, S_0 \sim \mu \right] .$$

Rewards at states that are "hard" to reach may have to be observed enough in data. Thus, the fact that the planning horizon is finite is crucial for a finite sample complexity.

Theorem 14 (Infinite sample complexity with policy-induced data collection in average reward problems)**.** *For any positive integers* S *and* A*, any pair*

(ϵ, δ) *such that* $0 < \epsilon < 1/2$ *and* $\delta \in (0,1)$*, the sample complexity of BPO with policy-induced data collection for MDPs with* S *states and* A *actions under the average reward criterion is infinite.*

For SA-sampling, the sample complexity becomes polynomial in the relevant quantities: Staying with discounted problems, this is implied by the results of Agarwal et al. (2020b) who study plug-in methods when a generative model is used to generate the same number of observations for each state-action pair. In particular, they show that in this setting, if N samples are available in each state-action pair then the plug-in algorithm will find a policy with $v^\pi \geq v^* - \varepsilon \mathbf{1}$ provided that $N = \Omega(\ln \frac{\mathrm{SA}}{(1-\gamma)\delta}/(\varepsilon^2(1-\gamma)^3))$. This implies a sample complexity upper bound of size $\tilde{O}(\mathrm{SA}H^3 \ln(1/\delta)/\varepsilon^2)$ where $H = 1/(1-\gamma)$, though for the stronger requirement that π is optimal not only from μ, but from each state. The upper bound is essentially matched by a lower bound by Sidford et al. (2018) who prove their result in Section D of their paper using a reduction to a result of Azar et al. (2013) that stated a similar sample complexity lower bound for estimating the optimal value. Our result is stronger than these results, which require the algorithm to produce a "globally good" policy, i.e., a policy that is near-optimal no matter the initial state, while in our result, the policy needs to be good only at a *fixed* initial state distribution.

Theorem 15. *Fix any* $\gamma_0 > 0$*. Then, there exist some constants* $c_0, c_1 > 0$ *such that for any* $\gamma \in [\gamma_0, 1)$*, any positive integers* S *and* A*,* $\delta \in (0,1)$*, and* $0 < \varepsilon \leq c_0/(1-\gamma)$*, the sample size* n *needed by any* (ε, δ)*-sound algorithm that produces as output a memoryless policy and works with* SA*-sampling for MDPs with* S *states and* A *actions under the* γ*-discounted expected reward criterion must be at least* $c_1 \frac{\mathrm{SA}\ln(1/(4\delta))}{\varepsilon^2(1-\gamma)^3}$*.*

Our proof for the lower bound essentially follows the ideas of Azar et al. (2013), but an effort was made to make the proof more streamlined. In particular, the new proof uses Le Cam's method (LeCam, 1973). We leave it for future work to extend the result to algorithms whose output is not restricted to memoryless policies.

7.5 Upper Bounds

In this section, we consider the "plug-in" algorithm for BPO and the discounted total expected reward criterion and will present a result for it that shows that this simple approach essentially matches the sample complexity lower bound of Theorem 12. For simplicity, we assume that the reward function is known.[2] Given a batch of data, the plug-in algorithm uses sample means to construct estimates for the transition probabilities. These can then be fed into any MDP solver to get a policy. The plug-in method is an obvious first choice that has proved its value in a number of different settings (Agarwal et al., 2020b; Azar et al., 2013; Cui & Yang, 2020; Li et al., 2020; Ren et al., 2021; Xiao et al., 2021c).

Let $\mathcal{D} = (S_i, A_i, R_i, S'_i)_{i=0}^{n-1}$ be the data available to the algorithm, and

$$N(s,a,s') = \sum_{i=0}^{n-1} \mathbb{I}\left(S_i = s, A_i = a, S'_i = s'\right)$$

denote the number of transitions observed in the data from s to s' while action a is used. Let $N(s,a) = \sum_{s'} N(s,a,s')$ be the number of times the pair (s,a) is seen in the data. Provided that the visit count $N(s,a)$ is positive, let

$$\hat{P}(s'|s,a) = \frac{N(s,a,s')}{N(s,a)}$$

be the estimated probability of transitioning to s' given that a is taken in state s. Let $\hat{P}(s'|s,a) = 0$ for all $s' \in \mathcal{S}$ when $N(s,a)$ is zero.[3] The plug-in algorithm returns a policy by solving the planning problem defined with $(\hat{P}, r)$, exploiting that planning algorithms need only the mean rewards and the transition probabilities (Puterman, 2014). By slightly abusing the definitions, we will hence treat $(\hat{P}, r)$ as an MDP and denote it by $\hat{M}$. In the result stated below we also allow a little slack for the planner; i.e., the planner is allowed to return a policy which is ϵ_{opt}-optimal.

[2] As noted also, e.g., by (Agarwal et al., 2020b), the sample size requirements stemming from the need to obtain a sufficiently accurate estimate of the reward function is a lower order term compared to that needed to accurately estimate the transition probabilities.

[3] The particular values chosen here do not have an essential effect on the results. For example, when $\hat{P}(\cdot|s,a)$ is the uniform distribution over $\mathcal{S}$, it will only effect the constant factor in Theorem 16 (see Eq. (D.15) in Section D.3).

The main result for this section is as follows:

Theorem 16. *Fix* S, A, *an MDP* $M \in \mathcal{M}(\mathrm{S}, \mathrm{A})$ *and a distribution* μ *on the state space of* M. *Suppose* $\delta > 0$, $\varepsilon > 0$, *and* $\varepsilon_{opt} > 0$. *Assume that the data is collected by following the uniform policy and it consists of* m *episodes, each of length* $H = H_{\gamma,(1-\gamma)\varepsilon/(2\gamma)}$. *Let* $\hat{\pi}$ *be any deterministic,* ε_{opt}*-optimal policy for* $\hat{M} = (\hat{P}, r)$ *where* $\hat{P}$ *is the sample-mean based empirical estimate of the transition probabilities based on the data collected. Then if*

$$m = \tilde{\Omega}\left(\frac{\mathrm{S}^3\mathrm{A}^{\min(H,\mathrm{S})+2}\ln\frac{1}{\delta}}{(1-\gamma)^4\epsilon^2}\right),$$

where $\tilde{\Omega}$ *hides log factors of* S, A *and* H, *we have* $v^{\hat{\pi}}(\mu) \geq v^*(\mu) - 4\varepsilon - \varepsilon_{opt}$ *with probability at least* $1 - \delta$.

Remark 6. *Our proof technique for the upper bound can be directly applied to the fixed* H*-horizon setting and gives an identical result.*

In summary, the theorem states that when the logging policy is uniform, then the plug-in algorithm will find an $O(\varepsilon)$ optimal policy with

$$\tilde{O}(\mathrm{S}^3\mathrm{A}^{\min(H,\mathrm{S})+2}\ln(1/\delta)/(\varepsilon^2(1-\gamma)^4))$$

episodes. For BPO with policy-induced data collection, it is not possible to directly apply a reductionist approach based on analysis for SA-sampling, which requires a uniform lower bound on the number of transitions observed at all the state-action pairs. As a result, the upper bound proven with this reductionist approach would depend on $1/\min_{s,a}\nu_{\mu}^{\pi_{\log}}(s,a)$. Our direct analysis avoids this issue, essentially replacing this minimum probability with a horizon-dependent constant. We provide the proof of Theorem 16, as well as an analogous result for the pessimistic policy (Jin et al., 2021; Buckman et al., 2020; Kidambi et al., 2020; Yu et al., 2020; Kumar et al., 2020; Liu et al., 2020; Xiao et al., 2021c) in Section D.3

7.6 Related Work

Our work is motivated by that of Zanette (2021) who considers the sample complexity of BPO in MDPs and linear function approximation. One of the

main results in Zanette's paper (Theorem 2) is that the $(1/2, 1/2)$ sample complexity with a "reinforced" policy-induced data collection in MDPs whose optimal action-value function is realizable with a d-dimensional feature map given to the learner is at least $\Omega((1/(1-\gamma))^{\frac{d-1}{2}})$. The "reinforced" data collection gives the learner full access to the transitional kernel and rewards at states that are reachable from the start states with the policy (or policies) chosen. Thus, the learner here has more information than in our setting, but the problem is made hard by the presence of linear function approximators. As noted by Zanette, the same setting is trivially easy in the finite horizon setting, thus the result shows a separation between the infinite and finite horizon settings. The weakness of this separation is that the "reinforced data collection" mechanism is unrealistic. A second result in Zanette's paper (Theorem 3) shows that in the presence of function approximation, even under SA-sampling, the sample complexity is still exponential in d (as in Theorem 2 mentioned above) even when the features are so that the action-value functions of *any* policy can be realized. This exponential sample complexity is to be contrasted with the fully polynomial result available for the same setting when a generative model is available (Lattimore et al., 2020). Thus, this second result shows a real exponential separation between "passive and active learners". It is interesting to note that this separation disappears in the tabular setting under SA-sampling.

For linear function approximation under SA-sampling a number of authors show related exponential (or infinite) sample complexity when the sampling distribution is chosen in a semi-adversarial way (Amortila et al., 2020; Wang et al., 2021b; Chen et al., 2021) in the sense that it can be chosen to be the worst distribution among those that provide *good coverage in the feature space* (expressed as a condition on the minimum eigenvalue of the feature second moment matrix). The main message of these results is that good coverage in the feature space is insufficient for sample-efficient BPO. In particular, since the hard examples in these works are tabular MDPs with $O(d)$ state-action pairs, the uniform distribution over the state-action space is sufficient to guarantee polynomial sample complexity in the same "hard MDPs". Hence, these

hardness results also have a distinctly different feature than the hardness result we present.

A different line of research can be traced back to the work of Li et al. (2015) who were concerned with statistically efficient batch policy evaluation (BPE) with policy-induced data-collection. The significance of this work for our work is that at the end of the paper the authors added a sidenote stating that the *sample complexity of BPE in finite horizon BPE must be exponential in the horizon*. Their example is a "combination-lock" type MDP, which served as an inspiration for the constructions we use in our lower bound proofs. No arguments are made for the suitability of the lower-bound for BPO, nor is a formal proof given for the exponential sample complexity for BPE. As such, our work can be seen as the careful examination of this remark in this paper and its adoption to BPO. A closely related, but weaker observation, is that the (vanilla) importance sampling estimators for BPE suffer an exponential blow-up of the variance (Guo et al., 2017), a phenomenon that (Liu et al., 2018) call the *curse of horizon* in BPE. This exponential dependence is also pointed out by Jiang & Li (2016), who provide a lower bound on the asymptotic variance of any unbiased estimator in BPE.

Lately, much effort has been devoted to "breaking" this aforementioned curse. The basis of these works is the observation that if sufficient coverage for the state-action space is provided by the logging policy, the curse should not apply (i.e., plug-in estimators should work well). Considering finite-horizon problems for now, the coverage condition is usually expressed as a lower bound d_m on the smallest visit probabilities, $\underline{\nu}_\mu^{\pi_{\log}} := \min_{s,a,t\in[H]} \nu_{\mu,t}^{\pi_{\log}}(s,a) (\leq 1/(SA))$, where $\nu_{\mu,t}^{\pi_{\log}}(s,a) = \mathbb{P}^{\pi_{\log}}(S_t = s, A_t = a | S_0 \sim \mu)$. Much work then is devoted to studying the sample-complexity of learning under the coverage requirement $\underline{\nu}_\mu^{\pi_{\log}} \geq d_m$. Note that for a fixed value of d_m, $SA \leq 1/d_m$ must hold, hence although these results are stated to hold over all combination of finite MDPs and logging policies $\pi_{\log}$ such that $\underline{\nu}_\mu^{\pi_{\log}} \geq d_m$, these MDPs cannot have more than $1/d_m$ state-action pairs. The main result here, due to Yin et al. (2021a), is that the sample-complexity (or, better, episode-complexity) of BPO, with an inhomogeneous H-step MDP and up to constant and logarithmic factors,

is $H^3/(d_m\epsilon^2)$, achieved by the plug-in estimator. According to a result of Yin et al. (2021b), this complexity continues to hold for the discounted setting when it represents the "step complexity" (as opposed to "episode complexity"). The same work also removes a factor of H both from the lower and upper bounds for the finite horizon setting with homogeneous transitions. A further strengthening of the results for the homogeneous setting is due to Ren et al. (2021) who remove an additional H factor under the assumption that the total reward in every episode belongs to the $[0, 1]$ interval. Their lower bound is $\Omega(1/(d_m\epsilon^2))$, while their upper bound is $\tilde{O}(1/(d_m\epsilon^2) + S/(d_m\epsilon))$. These results justify the use of coverage as a way of describing the inherent hardness of BPO. These results are complementary to ours. The lower bound in these works for fixed d_m is achieved by keeping the number of state-action pairs free, while we consider sample complexity for a fixed number of state-action pairs.

An alternative approach to characterize the sample-complexity of BPO is followed by Jin et al. (2021) who, for the inhomogeneous transition kernel, finite-horizon setting, consider a weighted error metric. While their primary interest is in obtaining results for linear function approximation, their result can be simplified back to the tabular setting. If we do this, the new metric that they propose takes the following form: Given a BPO algorithm $\mathcal{L}$ and some data $\mathcal{D}$ composed of a number of full episodes of length H, the weighted error of $\mathcal{L}$ on $\mathcal{D}$ is $Z(\mathcal{L}, \mathcal{D}) = \frac{v^*(\mu) - v^{\mathcal{L}(\mathcal{D})}(\mu)}{\sum_{h=0}^{H-1} \sum_{s,a} \nu^{\pi^*}_{\mu,h}(s,a)/\sqrt{1+N_h(s,a;\mathcal{D})}}$, where π^* is any optimal policy and $N_h(s, a; \mathcal{D})$ counts the number of times state (s, a) is seen at stage h in the episodes in $\mathcal{D}$. Their main result then shows that the minimax expected value of this metric is lower bounded by a universal constant, while the pessimistic algorithm's expected weighted error is upper bound by $\tilde{O}(\mathrm{SA}H)$. Note that the results that are phrased with the help of the minimum coverage probability can also be rewritten as results on the minimax error for a weighted error where the weights would include the minimum coverage probabilities. All these results are complementary to each other.

Average reward BPO with a parametric policy class for finite MDPs using policy-induced data is considered by Liao et al. (2020). The authors derive an "efficient" value estimator, and the policy returned is defined as the one that

achieves the largest estimated value. An upper bound on the suboptimality of the policy returned is given in terms of a number of quantities that relate to the policy parameterization provided that a coverage condition is satisfied similar to the coverage assumption discussed above.

Finally, we note that there is extensive literature on BPE; the reader is referred to the works of (Yin et al., 2021a; Yin & Wang, 2020; Ren et al., 2021; Uehara et al., 2021; Pananjady & Wainwright, 2020) and the references therein. The most relevant works for SA-sampling are concerned with the sample complexity of planning with generative models; see, e.g., (Azar et al., 2013; Agarwal et al., 2020b; Yin & Wang, 2020) and the references therein.

7.7 Conclusion

The main motivation for this chapter is to fill a substantial gap in the literature of batch policy optimization: While the most natural setting for batch policy optimization is when the data is obtained by following some policy, the sample complexity, the minimum number of observations necessary and sufficient to find a good policy, of batch policy optimization with data obtained this way has never been formally studied. Our results characterize how hard BPO under passive data collection exactly is and how the difficulty scales as the problem parameter changes. While our main result that, with a finite planning horizon, the sample complexity scales exponentially is perhaps somewhat expected, this has never been formally established and should therefore be a valuable contribution to the field. In fact, both the lower and the upper bound required considerable work to be rigorously establish and that the sample complexity is finite is less obvious in light of the previous results that involved "minimum coverage" as a superficial argument with these results suggest that the sample complexity could grow without bound if some state-action pairs have arbitrary small visit probabilities. That these results, as far as the details are concerned, are non-obvious is also shown by the gap that we could not close between the upper and lower sample complexity bounds. Another non-obvious insight of our work is that warm starts provably cannot help in reducing the sample

complexity. Our results should be given even more significance by the fact that the tabular setting provides the foundation for most of the insights that lead to better algorithms in RL.

Chapter 8

Understanding and Leveraging Overparameterization in Recursive Value Estimation

8.1 Introduction

Model-free value estimation remains a core method of reinforcement learning (RL), lying at the heart of some of the most prominent achievements in this area (Mnih et al., 2015; Bellemare et al., 2020). Such success appears paradoxical however, given that value estimation is subject to the *deadly triad*: any value update that combines off-policy estimation with Bellman-bootstrapping and function approximation diverges in the worst case (Sutton & Barto, 2018). Without additional assumptions it is impossible to ensure the viability of iterative value estimation schemes, yet this remains a dominant method in RL—its popularity supported by empirical success in many applications. Such a sizable gap between theory and practice reflects limited understanding of such methods, how they behave in practice, and what accounts for their empirical success (van Hasselt et al., 2018; Achiam et al., 2019).

Decomposing the deadly triad indicates that off-policy estimation and bootstrapping are difficult to forego: off-policy estimation is supported by the empirical effectiveness of action value maximization and replay buffers, while Bellman-bootstrapping provides significant advantages over Monte Carlo estimation (Sutton, 1988). On the other hand, our understanding of the third factor, the relationship between function representation and generalization,

has evolved dramatically in recent years. Although it was once thought that *restrictive* function approximation—representations that lack capacity to fit all data constraints—might be essential for generalization, we now know that this view is oversimplified (Belkin et al., 2019). The empirical success of deep learning (Krizhevsky et al., 2012), extremely large models (Brown et al., 2020) and associated theoretical advances (Jacot et al., 2018) have made it clear that gradient-based training of overparameterized models embodies implicit biases that encourage generalization even after all data constraints are fit exactly. This success suggests a new opportunity for breaking the deadly triad: by leveraging overparameterized value representations one can avoid some of the most difficult tradeoffs in value-based RL (Lu et al., 2018).

The use of overparameterized deep models in value-based RL, however, still exhibits mysteries in stability and performance. Although one might expect larger capacity models to improve the stability of Bellman-bootstrapping, in fact the opposite appears to occur (van Hasselt et al., 2018). Our own empirical experience indicates that classical value estimation with deep models always diverges eventually in non-toy problems. It has also been shown that value updating leads to premature rank-collapse in deep models (Kumar et al., 2021), coinciding with instability and degrading generalization. In practice, some form of *early-stopping* is usually necessary to obtain successful results, a fact that is not often emphasized in the literature (Agarwal et al., 2021). Meanwhile, there is a long history of convergent methods being proposed in the RL literature—starting from residual gradient (Baird, 1995), to gradient-TD (Sutton et al., 2008; Maei et al., 2009), prox gradient TD (Liu et al., 2015, 2016), and emphatic TD (Yu, 2015; Sutton et al., 2016)—yet none of these has demonstrated sufficient generalization quality to supplant unstable methods. The current state of development leaves an awkward tradeoff between stability and generalization. A stable recursive value estimation method that ensures generalization quality with overparametrization remains elusive.

In this chapter, we investigate whether overparameterized value representations might allow the stability-generalization tradeoff to be better managed, enabling stable estimation methods that break the deadly triad and generalize

well. To this end, we first consider policy evaluation with *overparameterized linear* value representations, a simplified setting that still imposes the deadly triad (Zhang et al., 2021a). Here we first show that alternative updates, such as temporal difference (TD), fitted value iteration (FVI) and residual minimization (RM) converge to *different* fixed points in the overparameterized case (when they converge), even though these updates share a common fixed point when the approximation error is zero and there are no extra degrees of freedom (Dann et al., 2014). That is, these algorithms embody certain implicit biases that *only* become distinguishable in the overparameterized case. From this result, we observe that the fixed points lie in different bases, which we use to develop a unified view of iterative value estimation as minimizing the Euclidean norm of the weights subject to alternative constraint sets. This unification allows us to formulate alternative updates that share common fixed points with TD and FVI but guarantee stability without requiring regularization or prox constraints (Zhang et al., 2021a). Next, we analyze the generalization performance of these algorithms and provide a per-iterate bound on the value estimation error of FVI, and fixed point bounds on the value estimation error of TD. From these results, we identify two novel regularizers, one that closes the gap between RM and TD and another that quantifies the effect of the feature representation on the generalization bound. We deploy these regularizers in a realistic study of deep model training for optimal value estimation and observe systematic stability and generalization improvements. We also observe that the performance gap between RM and TD/FVI can be closed and in some cases eliminated.

8.2 Related work

Value estimation has a lengthy history throughout RL research. Our main focus is on off-policy value estimation with parametric function representations and iterative (i.e., gradient based) updates. We do not consider exploration nor full planning problems (i.e., approximately solving an entire Markov decision process (MDP)) in the theoretical development, but instead focus on offline

value estimation; however, we do apply the findings to policy improvement experiments in the empirical investigation.

Dann et al. (2014) provide a comprehensive survey of value estimation with parametric function representations. Significant attention has been focused on *under*parameterized representations where backed up values are not necessarily expressible in the function class, however we focus on the overparameterized case where any backed up values can be assumed to be exactly representable with respect to finite data. This change fundamentally alters the conclusions one can draw about algorithm behavior, as we see below. One of the key consequences is that classical distinctions (Scherrer, 2010; Dann et al., 2014) between objectives—e.g., mean squared Bellman error (MSBE), mean squared *projected* Bellman error (MSPBE), mean squared temporal difference error (MSTDE), and the norm of the expected TD update (NEU)—all collapse when the Bellman errors can all be driven to zero. Despite this collapse, we find that algorithms targetting the different objectives—TD and LSTD for MSPBE (Sutton, 1988; Bradtke & Barto, 1996) and RM without double sampling (DS) for MSTDE (Maei et al., 2009; Dann et al., 2014)—converge to different fixed points given overparameterization, even when they ultimately satisfy the same set of temporal consistency constraints.

It is well known that classical value updates can diverge given off-policy data and parametric function representations (Baird, 1995; Tsitsiklis & Van Roy, 1996, 1997). The stability of these methods has therefore been studied extensively with many mitigations proposed, including restricting the function representation (Gordon, 1995; Szepesvári & Smart, 2004) or adjusting the representation to ensure contraction (Kolter, 2011; Ghosh & Bellemare, 2020; Wang et al., 2021c), or modifying the updates to achieve convergent variations, such as LSTD (Bradtke & Barto, 1996; Yu, 2010), FVI (Ernst et al., 2005; Munos & Szepesvári, 2005; Szepesvári & Munos, 2008; Lizotte, 2011) or the introduction of target networks (Mnih et al., 2015; Lillicrap et al., 2016; Zhang et al., 2021a; Carvalho et al., 2020). Others have considered modified the updates to combat various statistical inefficiencies (van Hasselt, 2010; Weng et al., 2020; Konidaris et al., 2011). Another long running trend has been to

consider two time-scale algorithms and analyses. Examples are gradient-TD methods (Sutton et al., 2008; Maei et al., 2009), prox gradient TD (Liu et al., 2015, 2016), primal-dual TD (Dai et al., 2017; Du et al., 2017), and emphatic TD (Yu, 2015; Sutton et al., 2016). Beyond mere convergence, however, we discover a greater diversity in fixed points among algorithms in the overparameterized case, which play a critical but previously unacknowledged role in generalization quality.

The fact that minimizing MSPBE via TD methods still dominates practice appears surprising given the theoretical superiority of other objectives. It has been argued, for example, that direct policy gradient methods (Sutton et al., 1999) dominate minimizing Bellman error objectives (Geist et al., 2017). Even among Bellman based approaches, it is known that MSBE can upper bound the value estimation error (MSE) whereas MSPBE cannot (Kolter, 2011; Dann et al., 2014), yet MSPBE minimization (via TD based methods) empirically dominates minimizing MSBE (via residual methods). This dominance has been thought to be due to the double sampling bias of residual methods (Baird, 1995; Dann et al., 2014), but we uncover a more interesting finding that their fixed points lie in different bases in the overparameterized setting, and that reducing this difference closes the performance gap.

We analyze the convergence of classical updates given offline data and provide associated generalization bounds, with the primary goal of understanding the discrepancy between previous theory and the empirical success of TD/FVI versus RM. Although this theory sheds new light in exploitable ways, it cannot overcome theoretical limits on offline value estimation, such as lower bounds on worst case error that are exponential in horizon length (Wang et al., 2021a,c; Zanette, 2021; Xiao et al., 2021b). We analyze the convergence of the expected updates, extendible to the stochastic case using known techniques (Yu, 2010; Bhandari et al., 2018; Dalal et al., 2018; Prashanth et al., 2021; Patil et al., 2021). We expand the coverage of these earlier works by including alternative updates and focusing on the overparameterized case, uncovering previously unobserved differences in the fixed points.

There is a growing body of work on linear value estimation and planning

that leverages the insight of (Parr et al., 2008; Taylor & Parr, 2009) that linear value estimation is equivalent to linear model approximation. A number of works have strived to obtain provably efficient algorithms for approximating the optimal policy values in this setting, but these generally rely on exploration or strong assumptions about data coverage (Song et al., 2016; Yang & Wang, 2019; Duan et al., 2020; Agarwal et al., 2020a; Jin et al., 2020b; Yang et al., 2020; Hao et al., 2021) that we do not make. Instead we study linear value estimation to gain insight, but rather than focus on linear planning we leverage the findings to improve the empirical performance of value estimation with deep models.

8.3 Preliminaries

Notation We let $\mathbb{R}$ denote the set of real numbers, $\boldsymbol{I}_n$ an $n \times n$ identity matrix, and $\mathbb{I}$ the indicator function. For a finite set $\mathcal{X}$, we use $\Delta(\mathcal{X})$ to denote the set of probability distributions over $\mathcal{X}$. For a vector μ we let $|\text{supp}(\mu)|$ denote the size of the support of μ (i.e., the number of nonzero entries in μ). For a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, we let $\boldsymbol{A}^\dagger$ be the Moore-Penrose pseudoinverse of $\boldsymbol{A}$, $\|\boldsymbol{A}\|$ be its spectral norm, $\lambda_{\max}(\boldsymbol{A})$ and $\lambda_{\min}(\boldsymbol{A})$ be its maximum and minimum *non-zero* eigenvalues. We also use $\Pi_{\boldsymbol{A}} = \boldsymbol{A}^\dagger \boldsymbol{A}$ to denote the projection matrix to the row space of $\boldsymbol{A}$. For a vector $x \in \mathbb{R}^d$, we let $\|x\|$ be its l_2 norm and $\|x\|_{\boldsymbol{A}} = \sqrt{x^\top \boldsymbol{A} x}$ be the associated norm for a positive definite matrix $\boldsymbol{A}$. We also use $\text{diag}(x) \in \mathbb{R}^{d \times d}$ to denote a diagonal matrix whose diagonal elements are x.

8.3.1 Markov reward processes

We consider the problem of predicting the value of a given stationary policy in a Markov Decision Process (MDP) (Section 5.1). For a stationary policy, this problem can be naturally formulated in terms of a Markov reward process $M = \{\mathcal{S}, P, r, \gamma\}$, such that $\mathcal{S}$ is a finite set of states, $r : \mathcal{S} \to \mathbb{R}$ and $P : \mathcal{S} \to \Delta(\mathcal{S})$ are the reward and transition functions respectively, and $\gamma \in [0, 1)$ is the discount factor. For a given state $s \in \mathcal{S}$, the function $r(s)$ gives the immediate

reward incurred at s, while $P(\cdot|s)$ gives the next-state transition probability of s. The value function specifies the future discounted total reward obtained from each state, which is defined as

$$v(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \Big| s_0 = s\right]. \tag{8.1}$$

To simplify the presentation, we identify functions as vectors to allow vector-space operations. In particular, the reward function r is identified as a vector $r \in \mathbb{R}^{|\mathcal{S}|}$, and the transition P is identified as an $|\mathcal{S}| \times |\mathcal{S}|$ transition matrix, where the s-th row P_s specifies the transition probability $P(\cdot|s)$ of state s. These definitions allow the value function to be expressed using Bellman's equation

$$v = r + \gamma P v\,. \tag{8.2}$$

8.3.2 Linear Function Approximation

It is usually not possible to consider tabular value representations in practice, since the state set is usually combinatorial or infinite. In our theoretical development we focus on linear function approximations, where v is approximated by a linear combination of features describing states; i.e., $v(s) \approx \phi(s)^\top \theta$, where $\theta \in \mathbb{R}^d$ is a parameter vector and $\phi : \mathcal{S} \to \mathbb{R}^d$ maps a given state $s \in \mathcal{S}$ to a d-dimensional feature vector $\phi(s) \in \mathbb{R}^d$. We let $\Phi \in \mathbb{R}^{|\mathcal{S}| \times d}$ denote the feature matrix, with the s-th row corresponding to the feature vector $\phi(s)$, so that the value approximation can be written as $v \approx \Phi\theta$. We assume $\|\phi(s)\| \leq 1$ for any $s \in \mathcal{S}$, and for simplicity we also assume that there is no redundant or irrelevant features in the feature map; that is, Φ is full rank.

8.3.3 Batch Value Estimation

We consider *batch mode* ("offline") estimation of the value function. Let $\mu \in \Delta(\mathcal{S})$ be an arbitrary probability distribution over states and $\boldsymbol{D}_\mu = \mathrm{diag}(\mu)$. The data set consists of transition tuples $\{s_i, r_i, s_i'\}_{i=1}^n$, which are generated by $s \sim \mu, r_i = r(s_i), s_i' \sim P(\cdot|s_i)$. Let $n(s) = \sum_{i=1}^n \mathbb{I}(s_i = s)$ be the number of counts of state s. We define the *empirical data distribution matrix* $\boldsymbol{D} =$

diag($\hat{\mu}$), where $\hat{\mu}(s) = n(s)/n$ is the empirical data distribution over states. The goal is to estimate the value function by finding a weight vector $\theta \in \mathbb{R}^d$ that minimizes the *value prediction error*,

$$\mathcal{E}(\theta) = \|\Phi\theta - v\|^2_{\boldsymbol{D}_\mu} = \sum\nolimits_{s\in\mathcal{S}} \mu(s)(\phi(s)^\top\theta - v(s))^2 \,. \tag{8.3}$$

Let $\hat{P}$ be the empirical transition matrix, where the s-th row represents the estimated transition of state s: if $n(s) > 0$, $\hat{P}_s(s') = \sum_{i=1}^n \mathbb{I}(s_i = s, s'_i = s')/n(s)$; if $n(s) = 0$, $\hat{P}_s(s') = 0$ for all $s' \in \mathcal{S}$. The empirical mean squared *Bellman error* on the batch data can be defined as

$$\text{MSBE}(\theta) = \tfrac{1}{2}\big\|r + \gamma\hat{P}\Phi\theta - \Phi\theta\big\|^2_{\boldsymbol{D}} \,. \tag{8.4}$$

8.3.4 Over vs Underparameterized Features

In this chapter we are particularly interested in the *overparameterized* regime $d > |\text{supp}(\hat{\mu})|$ where one can exactly satisfy the temporal consistencies on all transitions in the batch data set, achieving zero Bellman error. (Obviously this would also be possible if $d = |\text{supp}(\hat{\mu})|$ but the strictly overparameterized case is more interesting, as we will see below.) By contrast, in the underparameterized regime $d < |\text{supp}(\hat{\mu})|$, one can only expect to find an approximate solution that in general has nonzero Bellman error.

We consider three core algorithms in our analysis, covering major classical approaches.

8.3.5 Residual Minimization (RM)

RM directly minimizes the empirical mean squared Bellman error Eq. (8.4) (MSBE) (Baird, 1995). The gradient update (Dann et al., 2014) can be expressed as

$$\theta_{t+1} = \theta_t - \eta(\Phi - \gamma\hat{P}\Phi)^\top \boldsymbol{D}\big(\Phi\theta_t - (r + \gamma\hat{P}\Phi\theta_t)\big) \,, \tag{8.5}$$

where θ_t is the estimated weight at step t, and η is the learning rate. As a gradient descent method, the convergence of this update is robust, and applies to both linear and nonlinear function approximation.

8.3.6 Temporal Difference (TD) Learning

The simplest variant of TD (Sutton, 1988), known as TD(0), also updates weights iteratively using transition data to approximate the value function. Let θ_t be the weight vector at step t. Then the so-called "*semi-gradient*" of Eq. (8.4) is used to compute the update,

$$\theta_{t+1} = \theta_t - \eta \Phi^\top \boldsymbol{D}\big(\Phi\theta_t - \big(r + \gamma \hat{P}\Phi\theta_t\big)\big)\,, \tag{8.6}$$

where η is the learning rate. From Eq. (8.6), it is clear that in the underparameterized ($d < |\mathrm{supp}(\hat{\mu})|$) regime, if the system converges, it must converge to parameters $\theta^*_{\boldsymbol{D}}$ such that

$$\Phi^\top \boldsymbol{D} r - \Phi^\top \boldsymbol{D}(\Phi - \gamma \hat{P}\Phi)\theta^*_{\boldsymbol{D}} = 0 \quad \Rightarrow \quad \theta^*_{\boldsymbol{D}} = (\Phi^\top \boldsymbol{D}(\Phi - \gamma \hat{P}\Phi))^{-1}\Phi^\top \boldsymbol{D} r\,, \tag{8.7}$$

where $\theta^*_{\boldsymbol{D}}$ is the *TD fixed point*. That is, given limited representational power, the TD fixed point minimizes the squared projected Bellman error (MSPBE) by solving the *projected Bellman equation*:

$$\Phi\theta^*_{\boldsymbol{D}} = \Pi^{\boldsymbol{D}}_{\Phi}\big(r + \gamma \hat{P}\Phi\theta^*_{\boldsymbol{D}}\big)\,, \tag{8.8}$$

such that $\Pi^{\boldsymbol{D}}_{\Phi} = \Phi(\Phi^\top \boldsymbol{D}\Phi)^{-1}\Phi^\top \boldsymbol{D}$ is a weighted projection matrix. It is well-known that TD(0) can diverge if the data sampling distribution μ is not the stationary distribution of the Markov process. One can still compute the TD fixed point directly using batch data, for example using the LSTD algorithm (Bradtke & Barto, 1996), but this requires computation on the order of $O(d^2)$ compared to $O(d)$ of the iterative update algorithm Eq. (8.6). The value prediction error of TD is discussed in (Tsitsiklis & Van Roy, 1997; Kolter, 2011; Dann et al., 2014; Bhandari et al., 2018).

8.3.7 Fitted Value Iteration (FVI)

FVI iteratively updates the weight vector by solving a regression problem where the target is constructed from the current estimate (Ernst et al., 2005; Dann et al., 2014), which is also known as approximate dynamic programming

(Sutton & Barto, 2018). In particular, given the current weight θ_t at iteration t, the objective Eq. (8.9) is minimized to obtain θ_{t+1},

$$\mathrm{FVI}_t(\theta) = \tfrac{1}{2}\left\|r + \gamma \hat{P}\Phi\theta_t - \Phi\theta\right\|_{\boldsymbol{D}}^2. \tag{8.9}$$

A simple calculation shows the TD fixed point matches the fixed point of FVI whenever θ_0 is in the row-span of $D\Phi$. Although convergence of FVI can be established under strong conditions (Szepesvári & Munos, 2008), the algorithm can be quite unstable in the general batch setting (Chen & Jiang, 2019; Wang et al., 2021c).

8.4 Over-Parameterized Linear Value Function Approximation

In this section, we study the convergence properties of the value estimation algorithms introduced in Section 8.3.3 in the *overparameterized* regime where $d > |\mathrm{supp}(\hat{\mu})|$. To faciliate analysis, we first introduce additional notation to simplify the derivations. Let $\{x_i\}_{i=1}^k$ denote the states in the support of $\hat{\mu}$, such that $n(x_i) > 0$ for all $i = \{1, \ldots, k\}$ and $k = |\mathrm{supp}(\hat{\mu})|$. Define a *mask matrix* $\boldsymbol{H} \in \mathbb{R}^{k \times |\mathcal{S}|}$ and a *truncated empirical data distribution matrix* $\boldsymbol{D}_k \in \mathbb{R}^{k \times k}$ according to

$$\boldsymbol{H} = \begin{bmatrix} \mathbf{1}_{x_1}^\top \\ \vdots \\ \mathbf{1}_{x_k}^\top \end{bmatrix}, \qquad \boldsymbol{D}_k = \begin{bmatrix} \hat{\mu}(x_1) & & \\ & \ddots & \\ & & \hat{\mu}(x_k) \end{bmatrix}, \tag{8.10}$$

where $\mathbf{1}_{x_i} \in \{0,1\}^{|\mathcal{S}|}$ is an indicator vector such that $\phi(x_i) = \Phi^\top \mathbf{1}_{x_i}$. We can then translate between the full distribution and its support via the following.

Proposition 5. *The empirical data distribution matrix $\boldsymbol{D}$ can be decomposed as $\boldsymbol{D} = \boldsymbol{H}^\top \boldsymbol{D}_k \boldsymbol{H}$.*

Let $\boldsymbol{M} = \boldsymbol{H}\Phi$, $\boldsymbol{N} = \boldsymbol{H}\hat{P}\Phi$ and $\boldsymbol{R} = \boldsymbol{H}r$ denote the state features, the expected next state features under the empirical transitions, and the rewards on the support of the data distribution respectively.

8.4.1 Overparameterized Residual Minimization

We first study the convergence of RM given a fixed $\boldsymbol{D}$. First note that the update Eq. (8.5) can be re-written as

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta(\boldsymbol{M} - \gamma\boldsymbol{N})^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N}))\theta_t + \eta(\boldsymbol{M} - \gamma\boldsymbol{N})^\top \boldsymbol{D}_k\boldsymbol{R}\,. \quad (8.11)$$

In the overparameterized regime, one can easily verify that there are infinitely many solutions $\theta \in \mathbb{R}^d$ satisfying $(\boldsymbol{M} - \gamma\boldsymbol{N})\theta = \boldsymbol{R}$. The gradient of Eq. (8.11) is zero at any of these solutions, which implies that RM can have infinitely many fixed points. However, given that RM minimizes the MSBE objective via gradient descent, as we show in the following theorem, the RM update initialized from $\theta_0 = 0$ will converge to a unique fixed point.

Theorem 17. *With $\eta \leq \frac{1}{(1+\gamma)^2}$ and starting from $\theta_0 = 0$, RM converges to $\theta_{RM} = (\boldsymbol{M} - \gamma\boldsymbol{N})^\dagger \boldsymbol{R}$.*

Remark 7. *For simplicity we present the fixed points of RM and TD starting from $\theta_0 = 0$. The fixed points given an arbitrary initial weight vector $\theta_0 \in \mathbb{R}^d$ are shown in Sections E.1.1 and E.1.2.*

This result parallels similar findings in the supervised learning literature, that training overparameterized deep models with gradient descent (or related algorithms) encodes implicit regularization that drives the model solution to particular outcomes in the overparameterized regime (Soudry et al., 2018; Gunasekar et al., 2018; Neyshabur et al., 2019). Moreover, this implicit regularization is often associated with generalization benefits. However, unlike the case for supervised learning, RM solutions do not often generalize well. Below we uncover a key difference between the RM fixed point and those of TD and FVI that sheds new light on the source of generalization differences.

8.4.2 Overparameterized TD Learning

We next consider the convergence properties of the TD(0) update in the overparameterized setting. First, rewrite the TD(0) update formula Eq. (8.6) as

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta\boldsymbol{M}^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N}))\theta_t + \eta\boldsymbol{M}^\top \boldsymbol{D}_k\boldsymbol{R}\,. \quad (8.12)$$

Similar to RM, in the overparameterized regime any solutions $\theta \in \mathbb{R}^d$ that satisfy $(\boldsymbol{M} - \gamma \boldsymbol{N})\theta = \boldsymbol{R}$ are the fixed points of Eq. (8.12), which implies an infinite set of fixed points. This is quite unlike the underparameterized case where there is a unique TD fixed point Eq. (8.7) given by the solution of projected Bellman equation. However, we now show that in the overparameterized setting, similar to solving RM using gradient descent, TD also encodes an implicit bias toward a particular fixed point.

This of course requires TD to converge, which can be assured by a simple condition. Let $\boldsymbol{W} = \boldsymbol{N}\boldsymbol{M}^\dagger$, which has a geometric interpretation that we will exploit later in Section 8.6. Observe that

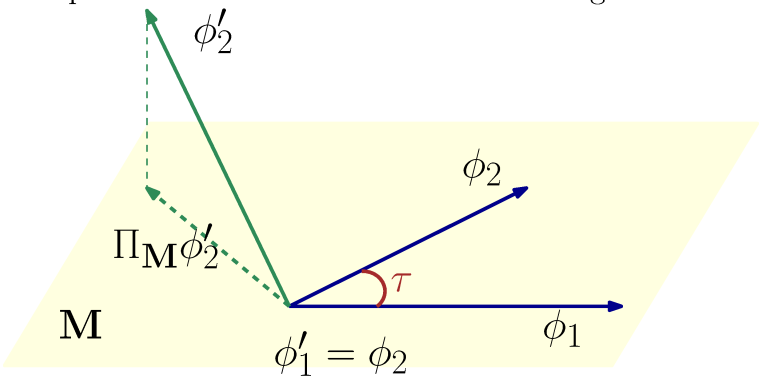
$$\begin{aligned} \boldsymbol{N} =& \boldsymbol{N}\Pi_{\boldsymbol{M}} + \boldsymbol{N}(\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}) = \boldsymbol{N}\boldsymbol{M}^\dagger\boldsymbol{M} + \boldsymbol{N}(\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}) \\ =& \boldsymbol{W}\boldsymbol{M} + \boldsymbol{N}(\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}), \end{aligned} \tag{8.13}$$

i.e., $\boldsymbol{N}$ can be decomposed into its projection onto the row-span of $\boldsymbol{M}$ plus a perpendicular component. Eq. (8.13) shows that $\boldsymbol{W}$ projects $\boldsymbol{N}$ onto the row space of $\boldsymbol{M}$; see Fig. 8.1 for an illustration. We refer to $\boldsymbol{W}$ as the *core matrix* since its norm determines the convergence of TD.

Theorem 18. *Choosing $\eta < \frac{1}{(1+\gamma)\|\Phi\|}$ and starting from $\theta_0 = 0$, if $\|\boldsymbol{W}\| < \frac{1}{\gamma}$, TD(0) converges to $\theta_{TD} = \boldsymbol{M}^\dagger(\boldsymbol{I}_k - \gamma\boldsymbol{W})^{-1}\boldsymbol{R}$. If $\|\boldsymbol{W}\| \geq \frac{1}{\gamma}$ there is an initial θ_0 for which TD(0) does not converge.*

A few key observations. First, note that the RM fixed point in Theorem 17 and the TD fixed point in Theorem 18 are not identical. That is, the different value estimation algorithms continue to demonstrate different preferences for fixed points, but in the overparameterized setting these differences are *implicit* in the algorithms and cannot be captured by the MSBE versus MSPBE objectives, since both are zero for any θ that satisfies $(\boldsymbol{M} - \gamma\boldsymbol{N})\theta = \boldsymbol{R}$. Second, *the fixed point of TD lies in a different basis than RM.* That is, θ_{TD} lies in the row space of the state features $\boldsymbol{M}$, whereas θ_{RM} lies in the row space of the residual features $\boldsymbol{M} - \gamma\boldsymbol{N}$, and these two spaces are not identical in general. We revisit the significance of this difference below, but intuitively, the parameter vector θ is being trained to predict values rather than temporal differences, and the

Figure 8.1: Illustrative example showing the spectrum of $\boldsymbol{W}$ with $k = 2$ and $d = 3$. $\boldsymbol{M} = [\phi_1, \phi_2]^\top$. Without loss of generality, we let $\phi_1 = (\cos\tau, \sin\tau, 0)$ and $\phi_2 = (1, 0, 0)$. $\boldsymbol{N} = [\phi_1', \phi_2']^\top$, where $\phi_1' = \phi_2$, $\phi_2' = (-\cos\tau, \frac{\sqrt{2}}{2}\sin\tau, \frac{\sqrt{2}}{2}\sin\tau)$. Then $\boldsymbol{W} = [[0, 1]^\top, [\frac{\sqrt{2}}{2}, -(1+\frac{\sqrt{2}}{2}\cos\tau)]^\top]$. Clearly, the spectral norm of $\boldsymbol{W}$ increases as the angle τ between ϕ_1 and ϕ_2 decreases.



future test states from which value predictions are made will tend to be closer to the space spanned by $\boldsymbol{M}$ than $\boldsymbol{M} - \gamma\boldsymbol{N}$.

8.4.3 Overparameterized Fitted Value Iteration

Finally, we consider the convergence of FVI. Recall that at iteration t, FVI solves the least squares problem Eq. (8.9) to compute the next weight vector. Using the notation established above, the normal equations for this problem can be expressed as $\boldsymbol{M}^\top\boldsymbol{D}_k\boldsymbol{M}\theta = \boldsymbol{M}^\top\boldsymbol{D}_k(\boldsymbol{R}+\gamma\boldsymbol{N}\theta_t)$, but this system cannot be directly used to compute the solution since $\boldsymbol{M}^\top\boldsymbol{D}_k\boldsymbol{M}$ is not invertible. Furthermore, just like RM and TD, any $\theta \in \mathbb{R}^d$ that satisfies $(\boldsymbol{M} - \gamma\boldsymbol{N})\theta = \boldsymbol{R}$ is a fixed point of FVI. If one solves the least squares problem Eq. (8.9) using gradient descent, it is known (Bartlett et al., 2021; Soudry et al., 2018) that the optimization converges to the minimum norm solution

$$\theta_{t+1} = \boldsymbol{M}^\dagger(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_t)\,. \tag{8.14}$$

Interestingly, by choosing this solution, each iteration of FVI corresponds to applying a linear backup on the current value estimate, where the backup operator is defined by the core matrix.

Definition 6. *Define the core matrix linear operator* $\mathcal{T}_{\boldsymbol{W}}$ *by* $\mathcal{T}_{\boldsymbol{W}}\nu = \boldsymbol{R}+\gamma\boldsymbol{W}\nu$ *for any* $\nu \in \mathbb{R}^{|\mathcal{S}|}$.

Using this operator we can characterize the convergence condition of FVI, reaching the conclusion that whenever $\mathcal{T}_{\boldsymbol{W}}$ is a non-expansion, FVI converges to the same fixed point as TD.

Theorem 19. *Let θ_0 be the initial weight and $\theta_t \in \mathbb{R}^d$ be the output of FVI at iteration t. We have*

$$\theta_{t+1} = \boldsymbol{M}^\dagger \mathcal{T}_{\boldsymbol{W}}^t(\boldsymbol{R} + \gamma \boldsymbol{N}\theta_0)\,. \tag{8.15}$$

Furthermore, given that $\|\boldsymbol{W}\| < 1/\gamma$, the algorithm converges to $\theta_{TD} = \boldsymbol{M}^\dagger(\boldsymbol{I}_k - \gamma \boldsymbol{W})^{-1}\boldsymbol{R}$. If $\|\boldsymbol{W}\| \geq \frac{1}{\gamma}$ there is an initial θ_0 for which FVI does not converge.

8.5 Unified View of Overparameterized Value Estimators

We now show that the convergence points above can be characterized as solutions to related constrained optimization problems, providing a unified perspective on the respective algorithm biases.

Theorem 20. *θ_{RM} is the solution of the following constrained optimization,*

$$\inf_{\theta \in \mathbb{R}^d} \tfrac{1}{2}\|\theta\|^2 \quad s.t.\ \boldsymbol{M}\theta = \boldsymbol{R} + \gamma \boldsymbol{N}\theta\,, \tag{8.16}$$

and θ_{TD} is the solution of the following constrained optimization,

$$\inf_{\theta \in \mathbb{R}^d} \tfrac{1}{2}\|\theta\|^2 \quad s.t.\ \boldsymbol{M}\theta = \boldsymbol{R} + \gamma \boldsymbol{N}\theta\,,\ \mathrm{null}(\boldsymbol{M})\theta = 0\,. \tag{8.17}$$

That is, the convergence points of RM, TD and FVI in the overparameterized case can all be seen as minimizing the Euclidean norm of the weights θ subject to satisfying the Bellman constraints $\boldsymbol{M}\theta = \boldsymbol{R} + \gamma \boldsymbol{N}\theta$, where TD and FVI implicitly add the additional constraint that θ must lie in the row span of $\boldsymbol{M}$; moreover, this is the *only* constraint that differentiates TD from RM. From this perspective, the algorithms can all be seen as iterative procedures for solving a particular form of quadratic program, when they converge. Of course, proper constrained optimization techniques would be able to stably compute solutions in scenarios where TD or FVI diverge (Boyd & Vandenberghe, 2004), but a more direct way to ensure convergence is implied by the following corollary.

Corollary 9. *θ_{TD} is also the solution of the following constrained optimization,*

$$\inf_{\theta\in\mathbb{R}^d} \tfrac{1}{2}\|\theta\|^2 \quad s.t.\ \boldsymbol{M}\theta = \boldsymbol{R} + \gamma \boldsymbol{N}\Pi_{\boldsymbol{M}}\theta\,. \tag{8.18}$$

Note that the right hand side of the constraint simply pre-projects next state value predictions onto the row space of $\boldsymbol{M}$ before determining the Bellman backed up value. This allows a novel objective to be formulated whose minimizer recovers the same fixed point as TD,

$$\text{MSCBE}(\theta) = \tfrac{1}{2}\left\|\boldsymbol{R} + \gamma \boldsymbol{N}\Pi_{\boldsymbol{M}}\theta - \boldsymbol{M}\theta\right\|^2_{\boldsymbol{D}}\,, \tag{8.19}$$

which stands for mean squared *corrected* Bellman error. Note that MSCBE is not identical to MSPBE because the projection is applied before not after the Bellman backup. Gradient descent minimization of MSCBE yields the same fixed point as θ_{TD}, which is essentially equivalent to applying RM to corrected target values while ensuring stability. Note also that in the linear case the projection matrix $\Pi_{\boldsymbol{M}}$ only needs to be precomputed once.

8.5.1 Value Prediction Error Bounds

One can also establish generalization bounds on the value estimation error of these methods in the overparameterized regime. We first provide a finite time analysis of the value prediction error of FVI.

Theorem 21. *Let $\hat{\Sigma} = \boldsymbol{M}^\top \boldsymbol{D}_k \boldsymbol{M}$ be the empirical covariance matrix, and θ_t be the output of FVI starting from θ_0 as defined in Theorem 19. Then for any $\theta^* \in \arg\min_{\theta\in\mathbb{R}^d} \mathcal{E}(\theta)$,*

$$\begin{aligned}&\mathcal{E}(\theta_t) - \mathcal{E}(\theta^*)\\ &\le \tfrac{1}{k\lambda_{\min}(\hat{\Sigma})}\left(\left(\varepsilon^2+\sigma^2\right)\left\|\sum_{i=0}^{t-1}(\gamma\boldsymbol{W})^i\right\|^2 + \left\|(\gamma\boldsymbol{W})^{t-1}\right\|^2\|\Phi\|^2\|\theta_0-\theta^*\|^2\right) + \tfrac{1}{2}\|\theta^*\|^2_{\boldsymbol{I}_d-\Pi_{\boldsymbol{M}}}\,,\end{aligned} \tag{8.20}$$

where $\varepsilon = \|\boldsymbol{N}(\boldsymbol{I}_d - \Pi_{\boldsymbol{M}})\theta^*\|$ and $\sigma = \|\boldsymbol{H}(\hat{P}-P)v\|$.

Intuitively, ε measures the length of next-state features along the direction θ^*, and σ is the expected value prediction error under the empirical transition

model, which can be bounded using standard concentration inequalities. The proof of this theorem is given in Section E.1.5. Observe that for any step $t \geq 1$, the output of FVI θ_t is within the row-span of $\boldsymbol{M}$. This allows us to decompose the prediction error into a component within the row-span, controlled by leveraging the core matrix linear operator $\mathcal{T}_{\boldsymbol{W}}$, and an orthogonal component that can be bounded by $\|\theta^*\|^2_{\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}}$.

Under the convergence conditions of Theorems 18 and 19, we also have the following generalization bound for the value prediction error of θ_{TD}.

Corollary 10. *Suppose that $\|\boldsymbol{W}\| \leq 1$, and the value prediction for any $s \in \mathcal{S}$ is bounded by $v(s) \in [0, v_{\max}]$. For any $\theta^* \in \arg\min_{\theta \in \mathbb{R}^d} \mathcal{E}(\theta)$,*

$$\mathbb{E}[\mathcal{E}(\theta_{\text{TD}})] \leq \frac{\gamma \log(|\mathcal{S}|/\delta)}{n_{\min} \mathbb{E}[\lambda_{\min}(\hat{\Sigma})](1-\gamma)^4} + \frac{4\gamma \mathbb{E}[\|\theta^*\|^2_{\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}}]}{\mathbb{E}[\lambda_{\min}(\hat{\Sigma})](1-\gamma)^2} + \delta v_{\max}\,, \quad (8.21)$$

where $n_{\min} = n \min_{s:\mu(s)>0} \mu(s)$ is the expected minimum counts.

This result automatically implies the requirements for ensuring offline generalization, accounting both for distribution shift (Wang et al., 2021c) and policy completeness (Munos & Szepesvári, 2005; Duan et al., 2020) in feature space. In particular, for Eq. (8.20) and Eq. (8.21), we characterize the distribution shift using well known concentration bounds in Section E.1.6, which leads to the denominators $k\lambda_{\min}(\hat{\Sigma})$ and $n_{\min}\mathbb{E}[\lambda_{\min}(\hat{\Sigma})]$ respectively. In addition, we explicitly characterize the misalignment between the features of current states and next states using the core matrix, which can be used to bound misalignment between values, replacing the feature completeness assumption.

We note that if the convergence condition cannot be satisfied, that is when $\|\boldsymbol{W}\| \geq 1/\gamma$, the estimation error could be arbitrarily large. The sources of value estimation error are explicit in Corollary 10. The first term measures the error due to sampling (statistical error), while the second term considers out-of-span components of the optimal weight vector θ^* with respect to $\boldsymbol{M}$ (approximation error). The smallest eigenvalue of the empirical covariance matrix $\mathbb{E}[\lambda_{\min}(\hat{\Sigma})]$, as well as the length of the orthogonal components $\mathbb{E}[\|\theta^*\|^2_{\boldsymbol{I}_d - \Pi_{\boldsymbol{M}}}]$, can both be controlled using classical techniques for concentration properties of random matrix. In Section E.1.7 we present the exact

approach for bounding these two terms. Furthermore, by Corollary 9, one can also apply Corollary 10 to an algorithm that directly optimizes MSCBE. Although a solution of Eq. (8.18) must exist, its value prediction error can be arbitrarily large given that $\|\boldsymbol{W}\| \geq 1/\gamma$. This also connects to a similar result for the TD fixed point that minimizes MSPBE in the underparameterized regime (Kolter, 2011).

8.6 Regularizers for Deep Reinforcement Learning Algorithms

For tractability, the theory in prior sections assumes fixed representations with a linear parameterization on only the final layer parameters of the value function. However, in practice, deep RL algorithms also learn the representations in an end-to-end fashion. Inspired by the linear case, we now identify two novel regularizers that are applicable more generally—one that closes the gap between RM and TD inspired by the unified view of different fixed points, and another that quantifies the effect of feature representation on the generalization bound.

Two-Part Approximation Most deep RL algorithms rely on approximating the value function with a deep neural network Q_ω that predicts the future outcome of a given state-action pair (Mnih et al., 2015; Kalashnikov et al., 2018; Lillicrap et al., 2016). In practice, Q_ω is trained by TD learning that minimizes the objective $\sum_{s,a}(r(s,a) + \gamma\bar{Q}_\omega(s,a) - Q_\omega(s,a))$, where $\bar{Q}_\omega(s,a)$ is known as the *target network* to increase the learning stability. We view Q_ω as a two part-approximation with $\omega = (\phi, \theta)$, where the output of the penultimate layer is referred as the feature mapping ϕ, the weight of last fully connected layer is referred as θ, and the Q-function is approximated by $Q_\omega(s,a) = \phi(s,a)^\top\theta$. Our goal is to define regularizers on ϕ and θ that can be effectively applied to practical algorithms.

The first regularizer directly takes inspiration from Theorem 20: by restricting the linear weight θ within the row space of $\boldsymbol{M}$ (now defined by exited

(s, a) pairs in the data), RM finds the same fixed point as TD. We implement this idea by penalizing the norm of the perpendicular component of θ,

$$\mathcal{R}_\theta = \|\theta - \Pi_{\boldsymbol{M}}\theta\| \ , \tag{8.22}$$

In practice we compute this regularizer for each minibatch of data. The projection step is computed by a least squares algorithm with an additional l_2 regularization for numerical stability.

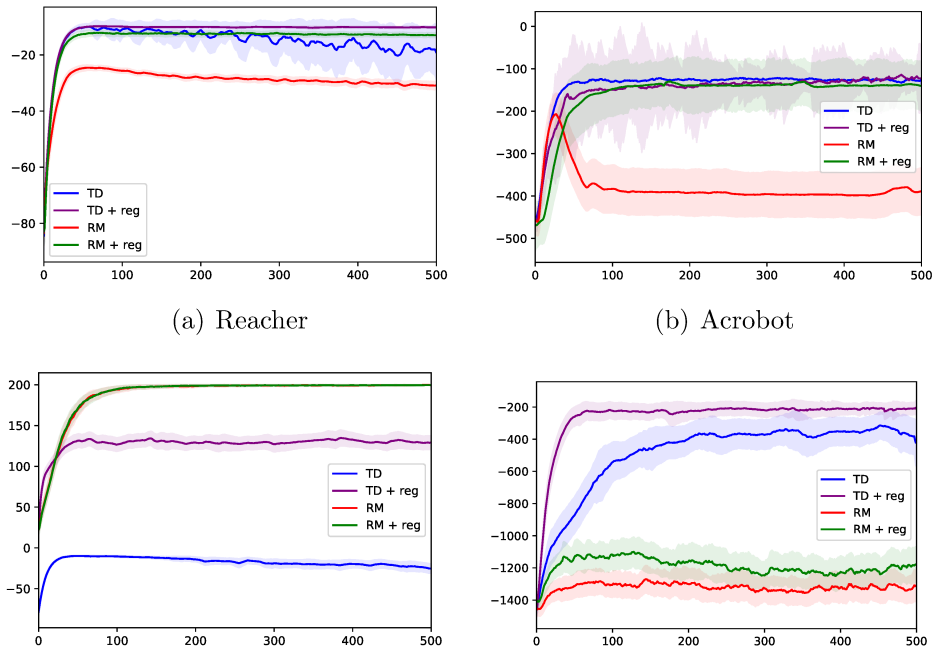
The second regularizer is designed to address the effect of the feature representation on convergence and value prediction error. In particular, as shown by Theorems 18 and 19, a sufficient condition that guarantees the convergence of TD and FVI is that the spectral norm of $\boldsymbol{W}$ be upper bounded by $1/\gamma$, which by Theorem 21 will also reduce the bound on generalization error. Hence, it is natural to penalize the norm of this matrix using standard automatic differentiation tools. However, such an approach is prone to numerical difficulty, as it involves differentiation through a matrix pseudo inverse. We instead propose an alternative regularizer inspired by the geometric interpretation of the core matrix Eq. (8.13): recall from Fig. 8.1 that $\boldsymbol{W}$ can be viewed as the weights that project $\boldsymbol{N}$ onto the row space of $\boldsymbol{M}$. To ensure that an arbitrary feature vector can be well approximated using $\boldsymbol{W}$, it would be ideal if $\boldsymbol{M}$ was orthonomral, which would imply an ideally-behaved basis to represent $\boldsymbol{N}$. This intuition justifies the following regularization:

$$\mathcal{R}_\phi = \left\|\beta \boldsymbol{I}_d - \boldsymbol{M}^\top \boldsymbol{D}_k \boldsymbol{M}\right\| \ , \tag{8.23}$$

where β is a scale parameter designed to approximate the column norm. That is, the regularizer forces the neural network to learn an orthogonal feature embedding by normalizing the empirical feature covariance matrix. The gradient of $\mathcal{R}_\phi$ can also approximated using mini-batches. We augment the original learning objectives by adding both $\mathcal{R}_\theta$ and $\mathcal{R}_\phi$ weighted by hyper-parameters.

8.6.1 Empirical Justification of Regularizers

The goal of our experiments is to assess the applicability of the proposed regularization schemes based on orthogonality and projection operations to



(a) Reacher (b) Acrobot

(c) Cartpole (d) Pendulum

Figure 8.2: We show the results with proposed regularization compared to the baseline algorithms. The algorithms are trained using a fixed offline data set collected by random initialized policies. The x-axis shows the training iterations (in thousands) and y-axis shows the performance. All plots are averaged over 100 runs. The shaded area shows the standard error.

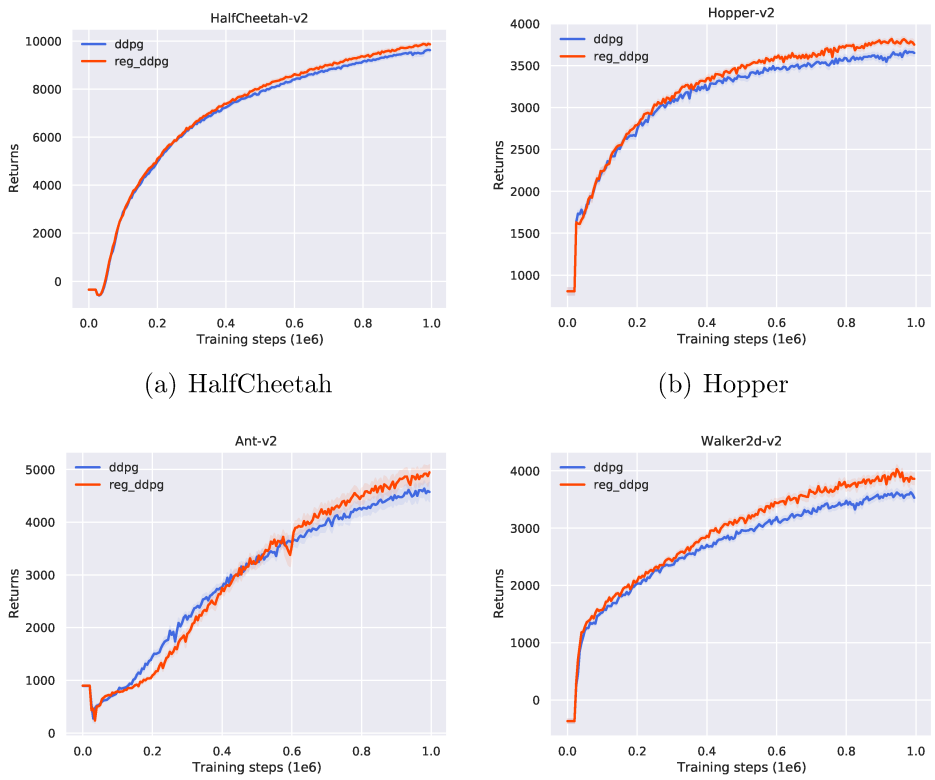
practical deep RL algorithms. To avoid the confounding effects of exploration, we first restrict our study to learning from a frozen batch of data with a fixed number of transitions collected prior to learning. We use a *randomly initialized policy* in this initial collection step. We consider both discrete and continuous control benchmarks in this analysis. For the discrete action environments, we use DQN (Mnih et al., 2015) as the baseline algorithm to add our regularizers. For continuous control environments, we use QT-Opt (Kalashnikov et al., 2018) as the baseline algorithm, which is an actor-critic method that applies the cross-entropy method to perform policy optimization. Our modifications add $\mathcal{R}_\phi$ and $\mathcal{R}_\theta$ to the standard MSBE objective on the critic Q-network. Experimental results contrasting vanilla TD and RM with their regularized variants are summarized in Fig. 8.2. All results are averaged over 100 runs with different random seeds. In all of our experiments, the parameters includ-

ing learning rate, target network moving average, regularization combining weight, and β in Eq. (8.23), are considered as tunable hyper-parameters. We use Google Vizier (Golovin et al., 2017), a black-box optimization parameter tuning engine for tuning. Additional details describing the complete experiment setup for each environment are provided in Section E.2. These findings demonstrate that our regularization schemes can be used to improve the performance of both vanilla TD learning and RM. Note that RM is typically less popular than TD due to its worse empirical performance. On Acrobot and Reacher, our method was able to fully close the gap between RM and TD. On Cartpole, (where vanilla RM dominates vanilla TD), and on Pendulum, our regularizers also deliver significant improvements to the TD learning baseline and modest improvements to the RM baseline.

Finally, we assess the applicability of the proposed regularization R_ϕ in the setting of online RL. We add the regularization R_ϕ to the critic update of DDPG (denoted as reg_ddpg) and compare it with the original algorithm. The results are provided in Fig. 8.3. All results are averaged over 100 runs with different random seeds.

8.7 Conclusion

We have investigated the fixed points of classical updates for value estimation in the overparameterized setting, where there is sufficient capacity to fit all the Bellman constraints in a given data set. We find that TD and FVI have different fixed points than RM, but in the linear case the difference can be entirely attributed to a constraint missing from RM that the solution lie in the row space of the predecessor state features. We devised two novel regularizers based on these findings, which stabilized the performance of TD without sacrificing generalization, while improving the generalization of RM, in the setting of estimating optimal values with a deep model.



(a) HalfCheetah (b) Hopper

(c) Ant (d) Walker2d

Figure 8.3: We show the results with proposed regularization compared to the baseline algorithms. The x-axis shows the training iterations (in thousands) and y-axis shows the performance. All plots are averaged over 100 runs. The shaded area shows the standard error.

Chapter 9

Conclusions and Future Directions

This dissertation introduces two novel techniques to improve the efficiency of Monte Carlo Tree Search (MCTS), and provides three analyses towards a better understanding of the theoretical foundations of batch RL. Below we summarize the contributions and discuss some future directions.

MCTS is an influential planning algorithm that combines tree search with simulations. The performance of MCTS highly replies on the accuracy of value estimation. Inaccurate estimates can mislead building the search tree and severely degrade the performance of the program. To improve the quality of value estimate, Chapter 3 exploits the benefits of generalization during online planning by augmenting MCTS with a memory structure. On the theoretical side, we prove that the memory can be used to provide better value estimates than vanilla Monte Carlo estimation with high probability under mild conditions. On the practical side, we demonstrate the effectiveness of memory-augmented MCTS in the game of Go. An interesting direction for future study is that the feature representation used in memory operations reuses a pre-trained neural network designed for move prediction. Instead, we plan to explore approaches that incorporate feature representation learning in an end-to-end fashion. Chapter 4 exploits the idea of applying maximum entropy policy optimization in MCTS. The proposed algorithm, Maximum Entropy for Tree Search, evaluates each node in the search tree using softmax values backproagated from simulations. It has been proven that the softmax value

can be efficiently back-propagated in the search tree, which enables the search algorithm to achieve faster convergence rate towards finding the optimal action at the root.

In the second part of the thesis, we set out to study the fundamental limits of batch RL. Chapter 6 investigates the optimality of batch policy optimization (BPO) algorithms. Our analysis reveals that any confidence-adjusted index algorithm is nearly minimax optimal, and the instance-dependent optimality cannot be achieved by any algorithm. These observations leave an important open question: There remains a lack of a well-defined theoretical framework that can be used to distinguish between different algorithms. Chapter 7 fills a substantial gap in the literature of batch RL: While the most natural setting for BPO is when the data is obtained by following some policy, the sample complexity of BPO with data obtained this way has never been formally studied. Our results explicitly characterize the hardness of BPO under passive data collection and how the difficulty scales as the problem parameter changes. Another remarkable finding of our work is that even warm starts (when one starts with a policy which is achieving almost as much as the optimal policy) provably cannot help in reducing the sample complexity. Chapter 8 studies the convergence properties of classical value estimation algorithms in the overparameterized setting, where the function approximation is capable to fit all the Bellman constraints in the given batch data. Our main observation is that temporal difference (TD) learning, fitted value iteration and residual minimization embody certain implicit biases that only become distinguishable in the overparameterized regime. We also develop a unified view to characterize the fixed points of these algorithm: Iterative value estimation can be viewed as minimizing the Euclidean norm of the weights subject to alternative constraint sets. This work leaves a number of interesting open questions, including characterizing the implicit bias of other algorithms, such as gradient or emphatic TD variants remains, and identifying new regularizers that further close the performance gap between TD and residual minimization.

Bibliography

Abramowitz, M., Stegun, I. A., and Romer, R. H. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep Q-learning. arXiv preprint arXiv:1903.08894, 2019.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. FLAMBE: Structural complexity and representation learning of low rank MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020a.

Agarwal, A., Kakade, S., and Yang, L. F. Model-based reinforcement learning with a generative model is minimax optimal. In *COLT*, pp. 67–83, 2020b.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020c.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. arXiv preprint arXiv:2108.13264, 2021.

Amortila, P., Jiang, N., and Xie, T. A variant of the Wang-Foster-Kakade lower bound for the discounted setting. *arXiv preprint 2011.01075*, 2020.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Azar, M. G., Munos, R., and Kappen, H. J. Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

Bai, Y., Xie, T., Jiang, N., and Wang, Y.-X. Provably efficient q-learning with low switching cost. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, pp. 30–37, 1995.

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: a statistical viewpoint. *arXiv preprint arXiv:2103.09177*, 2021.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77–82, 2020.

Bellman, R. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

Berger, J. O. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer New York, New York, NY, January 1985. URL http://link.springer.com/10.1007/978-1-4757-4286-2.

Bhandari, J., Russo, D., and Singal, R. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on Learning Theory (COLT)*, pp. 1691–1692, 2018.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge U Press, 2004.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

Brandfonbrener, D., Whitney, W. F., Ranganath, R., and Bruna, J. Offline rl without off-policy evaluation. *arXiv preprint arXiv:2106.08909*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Buckman, J., Gelada, C., and Bellemare, M. G. The importance of pessimism in fixed-dataset policy optimization. In *International Conference on Learning Representations*, 2020.

Carvalho, D., Melo, F. S., and Santos, P. A new convergent variant of Q-learning with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Charikar, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.

Chen, J. and Jiang, N. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 1042–1051. PMLR, 2019.

Chen, L., Scherrer, B., and Bartlett, P. L. Infinite-horizon offline reinforcement learning with linear function approximation: Curse of dimensionality and algorithm. *arXiv preprint 2103.09847*, 2021.

Chen, Z., Yu, P., and Haskell, W. B. Distributionally robust optimization for sequential decision-making. *Optimization*, 68(12):2397–2426, 2019.

Childs, B. E., Brodeur, J. H., and Kocsis, L. Transpositions and move groups in Monte Carlo tree search. In *IEEE Symposium On Computational Intelligence and Games, 2008.*, pp. 389–395, 2008.

Clark, C. and Storkey, A. Training deep convolutional neural networks to play go. In *International conference on machine learning*, pp. 1766–1774. PMLR, 2015.

Coquelin, P.-A. and Munos, R. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Coulom, R. Computing "elo ratings" of move patterns in the game of go. *ICGA journal*, 30(4):198–208, 2007.

Cui, Q. and Yang, L. Is plug-in solver sample-efficient for feature-based reinforcement learning? In *NeurIPS*, volume 33, pp. 6015–6026, 2020.

Dai, B., He, N., Pan, Y., Boots, B., and Song, L. Learning from conditional distributions via dual embeddings. In *Artificial Intelligence and Statistics*, pp. 1458–1467. PMLR, 2017.

Dai, B., Nachum, O., Chow, Y., Li, L., Szepesvári, C., and Schuurmans, D. Coindice: Off-policy confidence interval estimation. *arXiv preprint arXiv:2010.11652*, 2020.

Dalal, G., Szöréni, B., Thoppe, G., and Mannor, S. Finite sample analysis for TD(0) with function approximation. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 6144–6160, 2018.

Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15:809–883, 2014.

Delage, E., Kuhn, D., and Wiesemann, W. "dice"-sion–making under uncertainty: When can a random decision reduce risk? *Management Science*, 65 (7):3282–3301, July 2019.

Derman, E. and Mannor, S. Distributional robustness and regularization in reinforcement learning. March 2020. URL http://arxiv.org/abs/2003.02894.

Du, S. S., Chen, J., Li, L., Xiao, L., and Zhou, D. Stochastic variance reduction methods for policy evaluation. In *International Conference on Machine Learning*, pp. 1049–1058. PMLR, 2017.

Duan, Y., Jia, Z., and Wang, M. Minimax-optimal off-policy evaluation with linear function approximation. In *International Conference on Machine Learning*, pp. 2701–2709. PMLR, 2020.

Duchi, J., Glynn, P., and Namkoong, H. Statistics of robust optimization: A generalized empirical likelihood approach. October 2016. URL https://arxiv.org/abs/1610.03425v3.

Enzenberger, M., Müller, M., Arneson, B., and Segal, R. Fuego - an open-source framework for board games and Go engine based on Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):259–270, 2010. doi: 10.1109/TCIAIG.2010.2083662.

Ernst, D., Guerts, P., and Whenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Faury, L., Tanielian, U., Vasile, F., Smirnova, E., and Dohmatob, E. Distributionally robust counterfactual risk minimization. June 2019. URL http://arxiv.org/abs/1906.06211.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gao, M., Xie, T., Du, S. S., and Yang, L. F. A provably efficient algorithm for linear markov decision process with low switching cost. *arXiv preprint arXiv:2101.00494*, 2021.

Geist, M., Piot, B., and Pietquin, O. Is the Bellman residual a bad proxy? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 3208–3217, 2017.

Gelly, S. and Silver, D. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pp. 273–280, 2007.

Gelly, S. and Silver, D. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

Gelly, S., Wang, Y., Munos, R., and Teytaud, O. Modification of uct with patterns in monte-carlo go. 2006.

Ghosh, D. and Bellemare, M. G. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 3556–3565, 2020.

Gilboa, I. and Schmeidler, D. Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, 18(2):141–153, 1989.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.

Gordon, G. J. Stable function approximation in dynamic programming. In *International Conference on Machine Learning (ICML)*, pp. 261–268, 1995.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning (ICML)*, 2018.

Guo, Z. D., Thomas, P. S., and Brunskill, E. Using options and covariance testing for long horizon off-policy policy evaluation. In *NeurIPS*, pp. 2489–2498, 2017.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hao, B., Duan, Y., Lattimore, T., Szepesvári, C., and Wang, M. Sparse feature selection makes batch reinforcement learning more sample efficient. In *International Conference on Machine Learning (ICML)*, 2021.

Hastie, T., Tibshirani, R., and Friedman, J. The elements of statistical learning. springer series in statistics. In *∴* Springer, 2001.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 8571–8580, 2018.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. 2019.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *ICML*, pp. 652–661, 2016.

Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pp. 4870–4879. PMLR, 2020a.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. *Journal of Machine Learning Research*, 125:1–7, 2020b.

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline RL? 2020c.

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline RL? In *ICML*, 2021.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Karampatziakis, N., Langford, J., and Mineiro, P. Empirical likelihood for contextual bandits. *arXiv preprint arXiv:1906.03323*, 2019.

Karnin, Z., Koren, T., and Somekh, O. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pp. 1238–1246. PMLR, 2013.

Kaufmann, E., Ménard, P., Domingues, O. D., Jonsson, A., Leurent, E., and Valko, M. Adaptive reward-free exploration. In *Algorithmic Learning Theory*, pp. 865–891, 2021.

Kawano, Y. Using similar positions to search game trees. In Nowakowski, R. J. (ed.), *Games of No Chance*, volume 29 of *MSRI Publications*, pp. 193–202. Cambridge University Press, 1996.

Kearns, M., Mansour, Y., and Ng, A. Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49 (2):193–208, 2002.

Khandelwal, P., Liebman, E., Niekum, S., and Stone, P. On the analysis of complex backup strategies in monte carlo tree search. In *International Conference on Machine Learning*, pp. 1319–1328. PMLR, 2016.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based offline reinforcement learning. In *NeurIPS*, 2020.

Kishimoto, A. and Müller, M. A general solution to the graph history interaction problem. In *Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pp. 644–649, San Jose, CA, 2004.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Kolter, J. Z. The fixed points of off-policy TD. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pp. 2169–2177, 2011.

Konidaris, G., Niekum, S., and Thomas, P. S. TD_γ: Re-evaluating complex backups in temporal difference learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pp. 2402–2410, 2011.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. 2020.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Lam, H. Recovering best statistical guarantees via the empirical Divergence-Based distributionally robust optimization. *Oper. Res.*, 67(4):1090–1105, July 2019. URL https://doi.org/10.1287/opre.2018.1786.

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661. PMLR, 2019.

Lattimore, T. and Szepesvári, C. *Bandit algorithms*. Cambridge University Press, 2020.

Lattimore, T., Szepesvári, C., and Weisz, G. Learning with good feature representations in bandits and in RL with a generative model. In *ICML*, pp. 5662–5670, 2020.

LeCam, L. Convergence of estimates under dimensionality restrictions. *The Annals of Statistics*, pp. 38–53, 1973.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. 2020.

Li, G., Wei, Y., Chi, Y., Gu, Y., and Chen, Y. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *NeurIPS*, 33:12861–12872, 2020.

Li, L., Munos, R., and Szepesvári, C. Toward minimax off-policy value estimation. In *AISTATS*, pp. 608–616, 2015.

Liao, P., Qi, Z., and Murphy, S. Batch policy learning in average reward Markov decision processes. *arXiv 2007.11771*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. Finite-sample analysis of proximal gradient TD. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 504–513, 2015.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. Proximal gradient temporal difference learning algorithms. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4195–4199, 2016.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *NeurIPS*, pp. 5361–5371, 2018.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch reinforcement learning without great exploration. 2020.

Lizotte, D. J. Convergent fitted value iteration with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pp. 2537–2545, 2011.

Lu, T., Schuurmans, D., and Boutilier, C. Non-delusional Q-learning and value iteration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 9971–9981, 2018.

Maei, H. R., Szepesvári, C., Bhatnagar, S., Precup, D., and Silver, D. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 22, pp. 1204–1212, 2009.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Müller, M. Computer go. *Artificial Intelligence*, 134(1-2):145–179, 2002.

Munos, R. and Szepesvári, C. Finite time bounds for sampling based fitted value iteration. In *International Conference on Machine Learning (ICML)*, pp. 880–887, 2005.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Neyman, J. and Pearson, E. S. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. The role of over-parameterization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Pananjady, A. and Wainwright, M. J. Instance-dependent bounds for policy

evaluation in tabular reinforcement learning. *IEEE Transactions on Information Theory*, 67(1):566–585, 2020.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. An analysis of linear models, linear value-function approximation, and feature selction for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 752–759, 2008.

Patil, G., Prashanth, L. A., and Precup, D. Finite time analysis of temporal difference learning with linear function approximation: the tail averaged case. 2021.

Pepels, T., Cazenave, T., and Winands, M. H. Sequential halving for partially observable games. In *Computer Games*, pp. 16–29. Springer, 2015.

Prashanth, L. A., Korda, N., and Munos, R. Concentration bounds for temporal difference learning with linear function approximation: The case of batch data and uniform sampling. *Machine Learning*, 110(3):559–618, 2021.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Qin, R., Gao, S., Zhang, X., Xu, Z., Huang, S., Li, Z., Zhang, W., and Yu, Y. Neorl: A near real-world benchmark for offline reinforcement learning. *arXiv preprint arXiv:2102.00714*, 2021.

Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., and Russell, S. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

Ren, T., Li, J., Dai, B., Du, S. S., and Sanghavi, S. Nearly horizon-free offline reinforcement learning. *arXiv preprint arXiv:2103.14077*, 2021.

Scherrer, B. Should one compute the temporal difference fix point or minimize the Bellman residual? In *International Conference on Machine Learning (ICML)*, 2010.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Sidford, A., Wang, M., Wu, X., Yang, L. F., and Ye, Y. Near-optimal time and sample complexities for solving Markov decision processes with a generative model. In *NeurIPS*, pp. 5192–5202, 2018.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Silver, D. and Tesauro, G. Monte-carlo simulation balancing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 945–952, 2009.

Silver, D., Sutton, R., and Müller, M. Temporal-difference search in computer go. *Machine learning*, 87(2):183–219, 2012.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Smith, J. E. and Winkler, R. L. The optimizer's curse: Skepticism and post-

decision surprise in decision analysis. *Manage. Sci.*, 52(3):311–322, March 2006a. URL https://doi.org/10.1287/mnsc.1050.0451.

Smith, J. E. and Winkler, R. L. The optimizer's curse: Skepticism and post-decision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006b.

Smith, S. J. and Nau, D. S. An analysis of forward pruning. In *AAAI*, pp. 1386–1391, 1994.

Song, Z., Parr, R., Liao, X., and Carin, L. Linear feature encoding for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Soudry, D., Hoffer, E., Shpigel Nacson, M., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19:1–57, 2018.

Srinivasan, S., Talvitie, E., Bowling, M. H., and Szepesvári, C. Improving exploration in UCT using local manifolds. In *AAAI*, pp. 3386–3392, 2015.

Sutter, T., Van Parys, B. P. G., and Kuhn, D. A general framework for optimal Data-Driven optimization. October 2020. URL http://arxiv.org/abs/2010.06606.

Sutton, R. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12, 1999.

Sutton, R. S., Szepesvári, C., and Maei, H. R. A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 21, pp. 1609–1616, 2008.

Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 17:1–29, 2016.

Swaminathan, A. and Joachims, T. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16(1):1731–1755, 2015.

Szepesvári, C. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

Szepesvári, C. and Munos, R. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

Szepesvári, C. and Smart, W. D. Interpolation-based Q-learning. In *International Conference on Machine Learning (ICML)*, pp. 100–107, 2004.

Taylor, G. and Parr, R. Kernelized value function approximation for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 4214–4226, 2009.

Tian, Y. and Zhu, Y. Better computer Go player with neural network and long-term prediction. In *International Conference on Learning Representations*, 2015.

Tolpin, D. and Shimony, S. Mcts based on simple regret. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pp. 570–576, 2012.

Tsitsiklis, J. N. and Van Roy, B. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1):59–94, 1996.

Tsitsiklis, J. N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42 (5):674–690, 1997.

Uehara, M., Imaizumi, M., Jiang, N., Kallus, N., Sun, W., and Xie, T. Finite sample analysis of minimax offline reinforcement learning: Completeness, fast rates and first-order efficiency. *arXiv preprint arXiv:2102.02981*, 2021.

van Hasselt, H. Double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 23, 2010.

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. arXiv preprint arXiv:1812.02648, 2018.

Van Parys, B. P. G., Esfahani, P. M., and Kuhn, D. From data to decisions: Distributionally robust optimization is optimal. April 2017. URL http://arxiv.org/abs/1704.04118.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Wang, R., Foster, D., and Kakade, S. M. What are the statistical limits of offline RL with linear function approximation? In *International Conference on Learning Representations (ICLR)*, 2021a.

Wang, R., Foster, D. P., and Kakade, S. M. What are the statistical limits of offline RL with linear function approximation? In *ICLR*, 2021b.

Wang, R., Wu, Y., Salakhutdinov, R., and Kakade, S. M. Instabilities of offline RL with pre-trained neural representation. In *International Conference on Machine Learning (ICML)*, pp. 10948–10960, 2021c.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. Feature hashing for large scale multitask learning. In *Proceedings of the*

26th annual international conference on machine learning, pp. 1113–1120, 2009.

Weng, W., Gupta, H., He, N., and Ying, L. The mean-squared error of double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Weston, J., Chopra, S., and Bordes, A. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Whitehead, S. D. Complexity and cooperation in q-learning. In *International Conference on Machine Learning*, pp. 363–367, 1991.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. 2019.

Xiao, C., Mei, J., and Müller, M. Memory-augmented monte carlo tree search. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xiao, C., Huang, R., Mei, J., Schuurmans, D., and Müller, M. Maximum entropy monte-carlo planning. *Advances in Neural Information Processing Systems*, 32, 2019.

Xiao, C., Dai, B., Mei, J., Ramirez, O. A., Gummadi, R., Harris, C., and Schuurmans, D. Understanding and leveraging overparameterization in recursive value estimation. In *International Conference on Learning Representations*, 2021a.

Xiao, C., Lee, I., Dai, B., Schuurmans, D., and Szepesvari, C. On the sample complexity of batch reinforcement learning with policy-induced data. *arXiv preprint arXiv:2106.09973*, 2021b.

Xiao, C., Wu, Y., Mei, J., Dai, B., Lattimore, T., Li, L., Szepesvari, C., and Schuurmans, D. On the optimality of batch policy optimization algorithms. In *International Conference on Machine Learning*, pp. 11362–11371. PMLR, 2021c.

Xie, T. and Jiang, N. Batch value-function approximation with only realizability. In *International Conference on Machine Learning*, pp. 11404–11413. PMLR, 2021.

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. Bellman-consistent pessimism for offline reinforcement learning. *arXiv preprint arXiv:2106.06926*, 2021a.

Xie, T., Jiang, N., Wang, H., Xiong, C., and Bai, Y. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *arXiv preprint arXiv:2106.04895*, 2021b.

Xu, H. and Mannor, S. Distributionally robust markov decision processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 2*, pp. 2505–2513, 2010.

Yang, I. A convex optimization approach to distributionally robust markov decision processes with wasserstein distance. *IEEE control systems letters*, 1(1):164–169, 2017.

Yang, L. F. and Wang, M. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning (ICML)*, 2019.

Yang, Z., Jin, C., Wang, Z., Wang, M., and Jordan, M. I. On function approximation in reinforcement learning: Optimism in the face of large state spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yin, M. and Wang, Y.-X. Asymptotically efficient off-policy evaluation for tabular reinforcement learning. In *AISTATS*, pp. 3948–3958, 2020.

Yin, M. and Wang, Y.-X. Optimal uniform ope and model-based offline reinforcement learning in time-homogeneous, reward-free and task-agnostic settings. *arXiv preprint arXiv:2105.06029*, 2021.

Yin, M., Bai, Y., and Wang, Y.-X. Near-optimal provable uniform convergence in offline policy evaluation for reinforcement learning. In *AISTATS*, pp. 1567–1575, 2021a.

Yin, M., Bai, Y., and Wang, Y.-X. Near-optimal offline reinforcement learning via double variance reduction. *arXiv preprint arXiv:2102.01748*, 2021b.

Yu, H. Convergence of least squares temporal difference methods under general conditions. In *International Conference on Machine Learning (ICML)*, pp. 1207–1214, 2010.

Yu, H. On convergence of emphatic temporal-difference learning. In *Conference on Learning Theory (COLT)*, pp. 1–28, 2015.

Yu, P. and Xu, H. Distributionally robust counterpart in markov decision processes. *IEEE Transactions on Automatic Control*, 61(9):2538–2543, 2015.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. 2020.

Zanette, A. Exponential lower bounds for batch reinforcement learning: Batch RL can be exponentially harder than online RL. In *ICML*, 2021.

Zhang, S., Yao, H., and Whiteson, S. Breaking the deadly triad with a target network. In *International Conference on Machine Learning (ICML)*, pp. 12621–12631, 2021a.

Zhang, Z., Zhou, Y., and Ji, X. Almost optimal model-free reinforcement learningvia reference-advantage decomposition. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Zhang, Z., Du, S., and Ji, X. Near optimal reward-free reinforcement learning. In *International Conference on Machine Learning*, pp. 12402–12412, 2021b.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Appendix A

Proofs for Chapter 3: Memory-augmented MCTS

A.1 Proof

In this section we prove Theorem 2.

A.1.1 Notation

Similarly to (Coquelin & Munos, 2007), we consider a max search in a tree with branching factor K and depth D in our analysis. Let s_0 be the root of the tree. Each leaf l is assigned a reward distribution $\mathcal{P}_l$, with bounded support included in $[0, 1]$, whose law is unknown. For any node s, let $\mathcal{L}(s)$ be the set of leaves that belong to the sub-tree starting from s, $\mathcal{C}(s)$ be the set of child nodes of s.

We use v^* to denote the optimal value. For a leaf node l, $v^*(l) = \int x P_l(dx)$. For non-leaf node s, $v^*(s) = \max_{s' \in \mathcal{C}(s)} v^*(s')$. Thus $v^*(s) = \max_{l \in \mathcal{L}(s)} v^*(l)$. Let $v^* = v^*(s_0)$ to simplify the notation. For each state s, let $\Delta(s) = v^* - v^*(s)$.

For $n \geq 1$, let $\hat{v}_n(s)$ be the MC evaluation of state s using n simulations

$$\hat{v}_n(s) = \frac{1}{n} \sum_{t=1}^{n} X_{t,s} = \frac{1}{n} \sum_{l \in \mathcal{L}(s)} n(l) \hat{v}_{n(l)}(l)\,, \tag{A.1}$$

where $X_{t,s}$ is the simulation result received at the t-th visit of s, $n(s)$ is the number of times node s has been visited. We also write $\hat{v}_{n(s)}(s)$ as $\hat{v}(s)$ to

simplify the notations. The regret is defined as

$$R_n(s) = nv^*(s) - \sum_{t=1}^{n} X_{t,s} = n(v^*(s) - \hat{v}_n(s)) \,. \tag{A.2}$$

We also define the pseudo-regret as

$$\bar{R}_n(s) = nv^*(s) - \sum_{t=1}^{n} v^*(L_t) = \sum_{l \in \mathcal{L}(s_0)} n(l)\Delta(l) \,. \tag{A.3}$$

We have the following result on the pseudo-regret (Coquelin & Munos, 2007)[1].

Lemma 2. *Let $\beta > 0$. For a given s in the tree and a given $n \geq 1$, with probability at least $1 - \beta$,*

$$\frac{1}{n} \left| R_n(s) - \bar{R}_n(s) \right| \leq \sqrt{\frac{2 \log \frac{2}{\beta}}{n}} \,. \tag{A.4}$$

A.1.2 Flat UCB

At each round t, a bandit-based tree search algorithm runs a simulation by selecting a path from the root to a leaf, and observes a random reward $X_t \sim \mathcal{P}_{L_t}$, where we denote the leaf chosen at round t by L_t. The Flat UCB algorithm considers the following bandit algorithm to select children during the search

- For any leaf $l \in \mathcal{L}$, $B(s) = \hat{v}(s) + c_{n(s)}$, where for $m \geq 1$

$$c_m = \sqrt{\frac{\log(K^D m(m+1)/\beta)}{2m}} \,. \tag{A.5}$$

- For any non-leaf node s, $B(s) = \max_{s' \in \mathcal{C}(s')} B(s')$.

Define $\mathcal{L}'(s) = \{i \in \mathcal{L}(s), v^*(s) - v^*(i) > 0\}$ to be the set of sub-optimal leaves of s, and $\tilde{\Delta}_s = \min_{i \in \mathcal{L}'(s)} v^*(s) - v^*(i)$. We have the following result for Flat UCB (Theorem 2 of (Coquelin & Munos, 2007)).

Lemma 3. *Let $\beta > 0$. Consider the Flat UCB algorithm and a fixed state s and a fixed simulation count $n \geq 1$ through s. Then with probability at least $1 - \beta$, the pseudo-regret is bounded by a constant independent of n*

$$\bar{R}_n(s) \leq \sum_{i \in \mathcal{L}'(s)} \frac{1}{(v^*(s) - v^*(i))^2} \log \left(\frac{2}{(v^*(s) - v^*(i))^2} K^{D+1} \beta^{-1} \right) \leq \frac{6K^D}{\tilde{\Delta}_s} \log \left(\frac{2K^{D+1}}{\tilde{\Delta}_s^2 \beta} \right) \,. \tag{A.6}$$

[1] The result and proofs are given in the middle of Page 4 of (Coquelin & Munos, 2007).

Proof. The proof follows Theorem 2 of (Coquelin & Munos, 2007). Consider the event E under which, for all leaves $i \in \mathcal{L}$, for all $m \geq 1$, we have $|\hat{v}_m(i) - v^*(i)| \leq c_m$. Using a union bound and Hoeffding's inequality,

$$\mathbb{P}\left(|\hat{v}_m(i) - v^*(i)| > c_m, \forall m \geq 1, \forall i \in \mathcal{L}\right) \leq K^D \sum_{m \geq 1} \frac{\beta}{K^D m(m+1)} = \beta\,. \tag{A.7}$$

Since $\mathcal{L}'(s) \subset \mathcal{L}$, we have for all $i \in \mathcal{L}'(s)$, $|\hat{v}_m(i) - v^*(i)| \leq c_m$ for all $m \geq 1$. Given the event E holds, we can provide a regret bound by bounding the number of times each sub-optimal leaf is visited. Let $i \in \mathcal{L}'(s)$ be a sub-optimal leaf such that $v^*(s) - v^*(i) > 0$. Let i^* be the optimal leaf. If at some round the leaf i is chosen, we have $\hat{v}(i) + c_{n(i)} \leq \hat{v}(i^*) + c_{n(i^*)}$. Using the confidence interval bounds for leaves i and i^*, we deduce that $v^*(s) = v^*(i^*) \leq v^*(i) + 2c_{n(i)}$. Let $\Delta_{s,i} = v^*(s) - v^*(i)$ and $\beta_m = \frac{\beta}{K^D m(m+1)}$ for $m \geq 1$. Then

$$\frac{\log \beta^{-1}_{n(i)}}{n(i)} \geq \frac{\Delta^2_{s,i}}{2}\,. \tag{A.8}$$

Hence, $n(i)$ is bounded by the smallest integer n_0 such that $\frac{n_0}{\log(\beta^{-1}_{n_0})} > 2/\Delta^2_{s,i}$. Using the argument of (Coquelin & Munos, 2007), a rough upper bound on n_0 is $\frac{6}{\Delta^2_{s,i}} \log\left(\frac{2}{\Delta^2_{s,i}} K^{D+1}\beta^{-1}\right)$. This gives the number of times a suboptimal leaf i is chosen is at most,

$$n(i) \leq \frac{6}{\Delta^2_{s,i}} \log\left(\frac{2}{\Delta^2_{s,i}} K^{D+1}\beta^{-1}\right)\,. \tag{A.9}$$

The bound on the pseudo-regret immediately follows from the fact that $\bar{R}_n(s) \leq \sum_{i \in \mathcal{L}'(s)} n(i)\Delta_{s,i}$.

□

Combining Lemma 2 and Lemma 3 gives the following result.

Lemma 4. *Let $\beta > 0$. Consider the Flat UCB algorithm and a fixed state s and a fixed simulation count $n \geq 1$ through s. Then there exists constants C_1 and C_2 that only depend on K, D and $\tilde{\Delta}_s$ such that for any $0 < \alpha < 1$,*

$$\mathbb{P}\left(|\hat{v}_n(s) - v^*(s)| \leq \alpha\right) \geq 1 - C_1 e^{-C_2 \alpha^2 n}\,. \tag{A.10}$$

Proof. Let E_1 be the event under which

$$\left|R_n(s) - \bar{R}_n(s)\right| \leq \sqrt{2n \log \frac{2}{\beta}}\,. \tag{A.11}$$

By Lemma 2, we have $\mathbb{P}(E_1) \geq 1 - \beta$. Let E_2 be the event under which

$$\bar{R}_n(s) \leq \frac{6K^D}{\tilde{\Delta}_s} \log\left(\frac{2K^{D+1}}{\tilde{\Delta}_s^2 \beta}\right)\,. \tag{A.12}$$

By Lemma 3, we have $\mathbb{P}(E_2) \geq 1 - \beta$. Let $\bar{E}$ be the opposite event of E. Then

$$\mathbb{P}(E_1 \text{ and } E_2) = 1 - \mathbb{P}(\bar{E}_1 \text{ or } \bar{E}_2) \geq 1 - (\mathbb{P}(\bar{E}_1) + \mathbb{P}(\bar{E}_2)) \geq 1 - 2\beta\,. \tag{A.13}$$

Next we decompose the error by

$$|\hat{v}_n(s) - v^*(s)| = \frac{1}{n}|R_n| = \frac{1}{n}|R_n - \bar{R}_n + \bar{R}_n| \leq \frac{1}{n}|R_n - \bar{R}_n| + \frac{1}{n}|\bar{R}_n| \tag{A.14}$$

By Eq. (A.13), we have the following holds with probability at least $1 - 2\beta$,

$$|\hat{v}_n(s) - v^*(s)| \leq \frac{6K^D}{n\tilde{\Delta}_s} \log\left(\frac{2K^{D+1}}{\tilde{\Delta}_s^2 \beta}\right) + \sqrt{\frac{2\log 2/\beta}{n}}\,. \tag{A.15}$$

Choosing $\beta = \frac{2K^{D+1}}{\tilde{\Delta}_s^2} e^{-\frac{\alpha^2 \tilde{\Delta}_s n}{12K^{D+1}}}$ gives that

$$|\hat{v}_n(s) - v^*(s)| \leq \frac{6K^D}{n\tilde{\Delta}_s} \log\left(\frac{2K^{D+1}}{\tilde{\Delta}_s^2 \frac{2K^{D+1}}{\tilde{\Delta}_s^2} e^{-\frac{\alpha^2 \tilde{\Delta}_s n}{12K^{D+1}}}}\right) + \sqrt{\frac{2}{n} \log \frac{2}{\frac{2K^{D+1}}{\tilde{\Delta}_s^2} e^{-\frac{\alpha^2 \tilde{\Delta}_s n}{12K^{D+1}}}}} \tag{A.16}$$

$$\leq \frac{\alpha^2}{2} + \sqrt{\frac{2}{n} \log \frac{\tilde{\Delta}_s^2}{K^{D+1}} + \frac{\alpha^2 \tilde{\Delta}_s}{6K^{D+1}}} \tag{A.17}$$

$$\leq \frac{\alpha}{2} + \frac{\alpha}{\sqrt{6}} < \alpha\,, \tag{A.18}$$

where the last step we use $0 < \alpha < 1$, $\tilde{\Delta}_s < 1$ and $K, D \geq 1$. This implies that

$$\mathbb{P}\left(|\hat{v}_n(s) - v^*(s)| \leq \alpha\right) \geq 1 - C_1 e^{-C_2 \alpha^2 n}\,, \tag{A.19}$$

where we let $C_1 = \frac{4K^{D+1}}{\tilde{\Delta}_s^2}$ and $C_2 = \frac{\tilde{\Delta}_s}{12K^{D+1}}$.

□

A.1.3 Result

Let $\mathcal{M}$ be the memory, each entry of which contains the statistics for a state $(s, n(s), \hat{v}_{n(s)}(s))$. Given a memory $\mathcal{M}$, the memory based value for any state s is defined as

$$\hat{v}_{\mathcal{M}}(s) = \sum_{i=1}^{M} w_i \hat{v}_{n(s_i)}(s_i)\,, \tag{A.20}$$

where w is the softmax policy defined by $w_i \propto \exp(-(\delta(s_i) + \varepsilon(s, s_i))/\tau)$. We consider the case where we store all states in the memory and $M = K^{D+1} - 1$. We have the following result (restatement of Theorem 2).

Theorem 22. *Let N be the total number of simulations. There exists constants C_1 and C_2 that only depend on K and D, such that for any $\alpha > \varepsilon + \tau(D+1)\log K$,*

$$\begin{aligned}&\mathbb{P}(\forall s, |\hat{v}_{\mathcal{M}}(s) - v^*(s)| \le \alpha)\\ \ge& 1 - \frac{C_1}{(\alpha - \varepsilon - \tau(D+1)\log K)^2} e^{-C_2(\alpha-\varepsilon-\tau(D+1)\log K)^2 \frac{N}{K^{D+1}-1}}\,.\end{aligned}$$

Proof of Theorem 22. Let $M = K^{D+1} - 1$. First consider an arbitrary target states s and recall that the error is decomposed as

$$|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \le \sum_{i=1}^{M} w_i(\delta(s_i) + \varepsilon(s, s_i))\,, \tag{A.21}$$

where $\varepsilon(s_i, s)$ is a deterministic similarity measurement defined beforehand. Recall that $\varepsilon = \max_i \varepsilon(s, s_i)$. The weight w is chosen as a softmax over $-(\delta(s_i) + \varepsilon(s, s_i))$ (note that $\varepsilon(s, s_i)$ is not random.). Then

$$\sum_{i=1}^{M} w_i(\delta(s_i) + \varepsilon(s, s_i)) = -\left(\sum_{i=1}^{M} w_i(-(\delta(s_i) + \varepsilon(s, s_i))) + \tau\mathbb{H}(w)\right) + \tau\mathbb{H}(w) \tag{A.22}$$

$$= -\tau \log \sum_{i=1}^{M} \exp\left(-\frac{\delta(s_i) + \varepsilon(s, s_i)}{\tau}\right) + \tau\mathbb{H}(w) \tag{A.23}$$

$$\le -\tau \log \sum_{i=1}^{M} \exp\left(-\frac{\delta(s_i) + \varepsilon(s, s_i)}{\tau}\right) + \tau \log M \tag{A.24}$$

where the second equation follows from that w is the softmax policy, which gives the optimal solution of maximum entropy optimization. The optimal value is given by the softmax value (log-sum-exp). Then

$$\mathbb{P}\left(|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \alpha\right) \tag{A.25}$$

$$\geq \mathbb{P}\left(-\tau \log \sum_{i=1}^{M} \exp\left(-\frac{\delta(s_i) + \varepsilon(s, s_i)}{\tau}\right) \leq \alpha - \tau \log M\right) \tag{A.26}$$

$$= \mathbb{P}\left(\sum_{i=1}^{M} \exp\left(-\frac{\delta(s_i) + \varepsilon(s, s_i)}{\tau}\right) \geq \exp\left(-\frac{\alpha - \tau \log M}{\tau}\right)\right) \tag{A.27}$$

$$\geq \mathbb{P}\left(\sum_{i=1}^{M} \exp\left(-\frac{\delta(s_i)}{\tau}\right) \geq \exp\left(-\frac{\alpha - \varepsilon - \tau \log M}{\tau}\right)\right) \tag{A.28}$$

$$\geq \mathbb{P}\left(\exists i,\ e^{\delta(s_i)} \leq e^{\alpha - \tau \log M - \varepsilon}\right) \tag{A.29}$$

$$= \mathbb{P}\left(\exists i,\ \delta(s_i) \leq \alpha'\right)\ , \tag{A.30}$$

where in the last step we write $\alpha' = \alpha - \tau \log M - \varepsilon$. Let $i^* = \arg\max_{i \in \{1,\ldots,M\}} n(s_i)$ be a source state with the maximum number of simulations. We know $n(i^*) \geq N/M$. Then

$$\mathbb{P}\left(\exists i,\ \delta(s_i) \leq \alpha'\right) \geq \mathbb{P}\left(\delta(s_{i^*}) \leq \alpha'\right) \tag{A.31}$$

$$\geq \mathbb{P}\left(\forall n(i^*) \in [N/M, N/M + 1, \ldots, N], \delta(i^*) \leq \alpha'\right) \tag{A.32}$$

$$= 1 - \mathbb{P}\left(\exists n(i^*) \in [N/M, N/M + 1, \ldots, N], \delta(i^*) \geq \alpha'\right) \tag{A.33}$$

$$\geq 1 - \sum_{n=N/M}^{N} \mathbb{P}\left(n(i^*) = n, \delta(i^*) \geq \alpha'\right) \tag{A.34}$$

$$\geq 1 - \sum_{n=N/M}^{N} C_1 e^{-C_2 \alpha'^2 n} \tag{A.35}$$

$$\geq 1 - C_1 \int_{n \geq N/M} e^{-C_2 \alpha'^2 n} dn \tag{A.36}$$

$$= 1 - \frac{C_1}{C_2 \alpha'^2} e^{-C_2 \alpha'^2 \frac{N}{M}}\ , \tag{A.37}$$

where Eq. (A.34) follows by a union bound, and Eq. (A.35) follows by Lemma 4 (C_1 and C_2 are the constants defined in Lemma 4). Thus

$$\mathbb{P}\left(|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \alpha\right) \geq 1 - \frac{C_1}{C_2 \alpha'^2} e^{-C_2 \alpha'^2 \frac{N}{M}}\ . \tag{A.38}$$

This gives the value approximation error of a given target state s. Then

$$\mathbb{P}\left(\forall s, |\hat{v}_{\mathcal{M}}(s) - v^*(s)| \leq \alpha\right) = 1 - \mathbb{P}\left(\exists s, |\hat{v}_{\mathcal{M}}(s) - v^*(s)| \geq \alpha\right) \tag{A.39}$$

$$\geq 1 - \sum_s \mathbb{P}\left(|\hat{v}_{\mathcal{M}}(s) - v^*(s)| \geq \alpha\right) \tag{A.40}$$

$$\geq 1 - \frac{MC_1}{C_2 \alpha'^2} e^{-C_2 \alpha'^2 \frac{N}{M}} . \tag{A.41}$$

where the first inequality follows by a union bound. This finishes the proof.

□

Appendix B

Proofs for Chapter 4: Maximum Entropy Monte Carlo Planning

B.1 Experimental Details

We provide the experiment details in this section.

Value estimation in synthetic tree. For all settings, we use $\tau = 0.01$ for the softmax value. The exploration parameters for both MENTS and UCT are tuned from $\{0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0\}$.

Online planning in synthetic tree. The exploration parameters for MENTS and UCT are tuned from $\{0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0\}$. The temperature parameter τ of MENTS is tuned from $\{0.5, 0.1, 0.05, 0.01, 0.005\}$.

Online planning in Atari 2600 games. The exploration parameter for both algorithms are tuned from $\{5.0, 2.0, 1.0, 0.5, 0.1\}$ The temperature parameter τ of MENTS is tuned from $\{0.1, 0.05, 0.01\}$. The results is averaged over ten environment restarts.

In games such as *BeamRider*, one test game will take thousands of environment steps. Therefore, we only test the algorithms within 10,000 environment steps. The search algorithms are used every 10 steps. For the other steps the agent will use the DQN to select action.

B.2 Proofs

B.2.1 Proofs for softmax stochastic bandit

We first introduce a Lemma that approximates the exponential function of empirical estimator using delta method **?**. This Lemma will be used for both lower bound and upper bound analysis.

Lemma 5. *Let $X_1, \ldots, X_n$ be i.i.d. random variables, such that $\mathbb{E}[X_i] = \mu$ and $\mathrm{Var}X_i = \sigma^2 < \infty$, $\bar{X}_n = \sum_{i=1}^n X_i/n$. The first two moment of $\exp\left(\bar{X}_n/\tau\right)$ could be approximated by,*

$$\mathbb{E}\left[\exp\left(\frac{\bar{X}_n}{\tau}\right)\right] = e^{\mu/\tau} + \frac{\sigma^2}{2n}\left(\frac{e^{\mu/\tau}}{\tau^2}\right) + R\left(n\right) \tag{B.1}$$

$$\mathrm{Var}\exp\left(\frac{\bar{X}_n}{\tau}\right) = \frac{\sigma^2}{n}\left(\frac{e^{\mu/\tau}}{\tau}\right)^2 + R'\left(n\right) \tag{B.2}$$

where $|R(n)| \leq O\left(n^{-2}\right), |R'(n)| \leq O\left(n^{-2}\right)$.

Proof. By Taylor's expansion,

$$\exp\left(\frac{\bar{X}_n}{\tau}\right) = e^{\mu/\tau} + \frac{e^{\mu/\tau}}{\tau}\left(\bar{X}_n - \mu\right) + \frac{e^{\mu/\tau}}{2\tau^2}\left(\bar{X}_n - \mu\right)^2 + \frac{e^{\xi/\tau}}{6\tau^3}\left(\bar{X}_n - \mu\right)^3$$

for some ξ between μ and $\bar{X}_n$. Taking the expectation on both sides,

$$\mathbb{E}\left[\exp\left(\frac{\bar{X}_n}{\tau}\right)\right] = e^{\mu/\tau} + 0 + \frac{e^{\mu/\tau}}{2\tau^2}\mathrm{Var}\bar{X}_n + \frac{e^{\xi/\tau}}{6\tau^3}\mathbb{E}\left[\left(\bar{X}_n - \mu\right)^3\right].$$

Let $R(n) = \frac{e^{\xi/\tau}}{6\tau^3}\mathbb{E}\left[\left(\bar{X}_n - \mu\right)^3\right]$. By Lemma 5.3.1 of **?**, $|R(n)| \leq O(n^{-2})$, which gives Eq. (B.1).

Furthermore, note that

$$\begin{aligned}\left(\mathbb{E}\left[\exp\left(\frac{\bar{X}_n}{\tau}\right)\right]\right)^2 &= \left(e^{\mu/\tau} + \frac{\sigma^2}{2n}\left(\frac{e^{\mu/\tau}}{\tau^2}\right) + R(n)\right)^2 \\ &= e^{2\mu/\tau} + \frac{\sigma^2}{n}\left(\frac{e^{\mu/\tau}}{\tau}\right)^2 + \frac{C_1}{n^2} \\ &\quad + C_2 R(n) + C_3\frac{R(n)}{n}\end{aligned}$$

for some constant C_1, C_2, C_3. On the other hand, following the same idea of deriving Eq. (B.1),

$$\mathbb{E}\left[\left(\exp\left(\frac{\bar{X}_n}{\tau}\right)\right)^2\right] = e^{2\mu/\tau} + \frac{2\sigma^2}{n}\left(\frac{e^{\mu/\tau}}{\tau}\right)^2 + \tilde{R}(n)$$

where $|\tilde{R}(n)| \leq O(n^{-2})$. The proof of Eq. (B.2) ends by taking the difference of the above two equations. □

B.2.2 Proof of Theorem 3

We consider the learning problem in a Bayesian setting. In the stochastic bandit problem, we assume the expected reward of each action $r(a)$ is independently sampled from a Gaussian prior $\mathcal{N}(0, \sigma_0^2)$. At time step t, for any action a, a reward $X_{a,t}$ is sampled from $\mathcal{N}(r(A_t), \sigma^2)$, independently to all the previous observations. The learner chooses an action A_t according to some policy and observe $X_t = X_{A_t,t}$. Without loss of generality, we assume that $\sigma^2 = 1$ and $\tau = 1$. Our goal is to prove

$$\lim_{t\to\infty} \mathbb{E}\left[t\left(U - \hat{U}_t\right)^2 - \frac{\sigma^2}{\tau^2}\left(\sum\nolimits_a e^{r(a)/\tau}\right)^2\right] \geq 0\,,$$

where the expectation is taken on the randomness of the algorithm, the expected rewards $\mathbf{r}$, and the observation $X_{a,i}$ given $\mathbf{r}$. Therefore the existence of $\mathbf{r}$ that provides the lower bound is guaranteed since $\mathbf{r}$ satisfies the property in expectation.

We define $\tilde{U}_t$ to be the posterior mean of U, i.e. the conditional expectation of U given the observations $X_{a,t}$. Thus, $\mathbb{E}\left[\left(U - \hat{U}_t\right)^2 - \left(U - \tilde{U}_t\right)^2\right] \geq 0$. The benefit of considering $\tilde{U}_t$ is that $\tilde{U}_t$ can further be decomposed into the Bayes estimator of each action, even without the assumption that $\hat{U}_t$ is decomposable or $\hat{U}_t$ has (asymptotic) unbiased estimator for each arm.

We next introduce two technical lemmas that are useful to prove the lower bound. The first result shows that for an algorithm that performs well on all possible environments, it must pull each arm at least in $\Omega(\log t)$ in t rounds. Note that unlike in the regret analysis for stochastic multi-armed bandits, where one only cares about how many times the suboptimal arms are pulled, the $\Omega(\log t)$ lower bound on $N_t(a)$ for suboptimal arms is not strong enough to provides a tight lower bound of $\mathcal{E}_t$.

Lemma 6. *For any algorithm $\mathcal{A}$ such that $\mathcal{E}_t = O(\frac{1}{t})$, it holds that $N_t(a) = \Omega(\log t)$ for any arm a.*

In the Bayesian learning setting defined above, since $\exp(X_{a,t})$ has a log-normal distribution with a Gaussian prior, its posterior estimation is still log-normal. The second result studies the concentration rate of the posterior estimation.

Lemma 7. *Let* $\Phi(a) = \frac{\sum_{i=1}^{N_t(a)} X_{a,i} + 1/2}{\tau_0 + N_t(a)}$ *be the posterior estimation of* $r(a)$ *and define* $\Delta(a) = e^{r(a)} - e^{\Phi(a)}$. *We have*

$$\mathbb{E}\left[\Delta(a) | N_t(a), \mathbf{r}\right] = O\left(\frac{1}{N_t(a)}\right)$$
$$\mathbb{E}\left[\Delta(a)^2 \,\middle|\, N_t(a), \mathbf{r}\right] = e^{2r(a)}\left(\frac{N_t(a)}{(N_t(a)+\sigma_0)^2} + O\left(\frac{1}{N_t^2(a)}\right)\right).$$

Now we are ready to present the proof of the lower bound.

Proof of Theorem 3. By the tower rule and the fact that $\tilde{U}$ is the minimizer of the mean squared error,

$$\mathbb{E}\left[t\left(U - \hat{U}_t\right)^2\right] \geq \mathbb{E}\left[t\left(U - \tilde{U}_t\right)^2\right] = \mathbb{E}\left[\mathbb{E}\left[t\left(U - \tilde{U}_t\right)^2 \,\middle|\, \mathbf{r}\right]\right],$$

It then suffices to prove that

$$\lim_{t\to\infty} \mathbb{E}\left[t\left(U - \tilde{U}_t\right)^2 \,\middle|\, \mathbf{r}\right] \geq \left(\sum\nolimits_a e^{r(a)}\right)^2$$

for any $\mathbf{r}$. The rest of the proof is always conditioned on $\mathbf{r}$. Let $\mathbf{X}_{a,t} = X_{a,1}, \ldots, X_{a,N_t(a)}$ be the observations of action a up to time step t. We can decompose $\tilde{U}$ by

$$\tilde{U}_t = \mathbb{E}\left[U \,|\, \mathbf{X}_{j,t}, j \in \{1,\ldots,K\}\right] = \sum_{j=1}^{K} \mathbb{E}\left[e^{r(j)} \,\middle|\, \mathbf{X}_{j,t}, j \in \{1,\ldots,K\}\right]$$
$$= \sum_{j=1}^{K} \mathbb{E}\left[e^{r(j)} \,\middle|\, \mathbf{X}_{j,t}\right].$$

Therefore, the Bayesian estimator of U is

$$\tilde{U}_t = \sum_j \exp\left(\frac{\sum_{i=1}^{N_t(j)} X_{j,i} + 1/2}{\tau_0 + N_t(j)}\right).$$

It remains to bound $\left(U - \tilde{U}_t\right)^2$ conditioned on $\mathbf{r}$. Note that

$$\left(U - \tilde{U}_t\right)^2 = \left(\sum_j e^{r(j)} - \exp\left(\frac{\sum_{k=1}^{N_t(j)} X_{j,k} + 1/2}{\tau_0 + N_t(j)}\right)\right)^2 = \sum_j \Delta_j^2 + \sum_{i\neq j} \Delta_j \Delta_i,$$

where $\Delta_j = e^{r(j)} - \exp(\frac{\sum_{k=1}^{N_t(j)} X_{j,k} + 1/2}{\tau_0 + N_t(j)})$. Finally, define $P_t(j) = N_t(j)/t$ and let $\tau_0 \to 0$. By Lemma 7, we have

$$\begin{aligned}
\lim_{t\to\infty} t\mathbb{E}\left[\left(U - \tilde{U}_t\right)^2 \mid \mathbf{r}\right] &= \lim_{t\to\infty} t\mathbb{E}\left[\mathbb{E}\left[\left(U - \tilde{U}_t\right)^2 \,\middle|\, N_t(1), \ldots, N_t(k), \mathbf{r}\right]\right] \\
&= \lim_{t\to\infty} \mathbb{E}\left[\sum_j \frac{e^{2r(j)} + O\left(\frac{1}{N_t(j)}\right)}{P_t(j)}\right] \\
&\geq \left(\sum\nolimits_a e^{r(a)}\right)^2
\end{aligned}$$

where the last inequality follows by Cauchy-Schwarz inequality and Lemma 6. Note that for the inequality to hold there must be for all action $k \in [K]$, $N_t(k) = N_t^*(k)$.

For the general case, where $\sigma, \tau \neq 1$, we can simply scale the reward by τ, then the variance of $X_{j,k}$ is $\frac{\sigma^2}{\tau^2}$. The proof still holds and we obtain the following inequality,

$$\lim_{t\to\infty} t\mathbb{E}\left[\left(U - \tilde{U}_t\right)^2 \mid \mathbf{r}\right] \geq \frac{\sigma^2}{\tau^2}\left(\sum_a \bar{\pi}(a) e^{r(a)/\tau}\right)^2.$$

□

B.2.3 Concentration of $N_t(a)$ in Bandit (Theorem 5)

Define $\tilde{N}_t(a) = \sum_s \pi_s(a)$, where π_s is the policy followed by E2W at time step s. By Theorem 2.3 in **?** or **?**, we have the following concentration result.

$$\mathbb{P}\left(|N_t(a) - \tilde{N}_t(a)| > \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2}{2\sum_{s=1}^t \sigma_s^2}\right) \leq 2\exp\left(-\frac{2\epsilon^2}{t}\right),$$

where $\sigma_s^2 \leq 1/4$ is the variance of Benoulli distribution with $p = \pi_s(k)$ at time step s. Denote the event

$$\widetilde{E}_\epsilon = \{\forall a \in \mathcal{A}, |\tilde{N}_t(a) - N_t(a)| < \epsilon\}.$$

Thus we have

$$\mathbb{P}\left(\widetilde{E}_\epsilon^c\right) \leq 2|\mathcal{A}|\exp\left(-\frac{2\epsilon^2}{t}\right).$$

It remains to bound $\mathbb{P}\left(|\tilde{N}_t(a) - N_t^*(a)| \geq \epsilon\right)$. To prove Theorem 5, we first introduce two technical lemmas, which prove the accuracy of our estimate

on the reward and connect the convergence of the reward estimation to the convergence of policy.

Lemma 8. *For the stochastic softmax bandit problem, E2W can guarantee that, for $t \geq 4$,*

$$\mathbb{P}\left(\|\mathbf{r} - \hat{\mathbf{r}}_t\|_\infty \geq \frac{2\sigma}{\log(2+t)}\right) \leq 4|\mathcal{A}| \exp\left(-\frac{t}{(\log(2+t))^3}\right).$$

Lemma 9. *Given two soft indmax policies, $\pi^{(1)} = \mathbf{f}_\tau(\mathbf{r}^{(1)})$ and $\pi^{(2)} = \mathbf{f}_\tau(\mathbf{r}^{(2)})$, we have*

$$\left\|\pi^{(1)} - \pi^{(2)}\right\|_\infty \leq \left(1 + \frac{1}{\tau}\right) \left\|\mathbf{r}^{(1)} - \mathbf{r}^{(2)}\right\|_\infty$$

Proof of Theorem 5. We denote the following event,

$$E_{\mathbf{r}_t} = \left\{\|\mathbf{r} - \hat{\mathbf{r}}_t\|_\infty < \frac{2\sigma}{\log(2+t)}\right\}.$$

For any time step s and action a, by the definition of $\pi_s(a)$ we have,

$$|\pi_s(a) - \pi^*(a)| \leq |\hat{\pi}_s(a) - \pi^*(a)| + \lambda_s.$$

Thus, to bound $|\tilde{N}_t(a) - N_t^*(a)|$, conditioned on the event $\cap_{i=1}^t E_{\mathbf{r}_t}$ and for $t \geq 4$ there is,

$$\begin{aligned}
|\tilde{N}_t(a) - N_t^*(a)| \leq & \sum_{s=1}^t |\hat{\pi}_s(a) - \pi^*(a)| + \sum_{s=1}^t \lambda_s \\
\leq & \left(1 + \frac{1}{\tau}\right) \sum_{s=1}^t \|\hat{\mathbf{r}}_s - \mathbf{r}\|_\infty + \sum_{s=1}^t \lambda_s \qquad \text{(by Lemma 9)} \\
\leq & \left(1 + \frac{1}{\tau}\right) \sum_{s=1}^t \frac{2\sigma}{\log(2+s)} + \sum_{s=1}^t \lambda_s \qquad \text{(by Lemma 8)} \\
\leq & \left(1 + \frac{1}{\tau}\right) \int_{s=0}^t \frac{2\sigma}{\log(2+s)} \mathrm{d}s + \int_{s=0}^t \frac{|\mathcal{A}|}{\log(1+s)} \mathrm{d}s \\
\leq & \frac{Ct}{\log t},
\end{aligned}$$

for some constant C depending on $|\mathcal{A}|$, σ and τ. Finally,

$$\begin{aligned}\mathbb{P}\left(|\tilde{N}_t(a) - N_t^*(a)| \geq \frac{Ct}{\log t}\right) \leq & \sum_{i=1}^{t} \mathbb{P}\left(E_{\mathbf{r}_t}^c\right) = \sum_{i=1}^{t} 4|\mathcal{A}| \exp\left(-\frac{t}{(\log(2+t))^3}\right) \\ \leq & 4|\mathcal{A}| t \exp\left(-\frac{t}{(\log(2+t))^3}\right).\end{aligned}$$

Therefore,

$$\begin{aligned}&\mathbb{P}\left(|N_t(a) - N_t^*(a)| \geq (1+C)\frac{t}{\log t}\right) \\ \leq & \mathbb{P}\left(|\tilde{N}_t(k) - N_t^*(k)| \geq \frac{Ct}{\log t}\right) + \mathbb{P}\left(|N_t(k) - \tilde{N}_t(k)| > \frac{t}{\log t}\right) \\ \leq & 4|\mathcal{A}| t \exp\left(-\frac{t}{\log(2+t)^3}\right) + 2|\mathcal{A}| \exp\left(-\frac{2t}{\log(2+t)^2}\right) \\ \leq & O\left(t \exp\left(-\frac{t}{(\log t)^3}\right)\right)\end{aligned}$$

□

B.2.4 Proof of Theorem 4

Proof of Theorem 4. Let $\delta_t = Ct/\log t$ with some constant C. Define the following set

$$\mathcal{G}_t = \left\{ s \middle| s \in 1:t, \lceil N_t^*(a) + \delta_t \rceil \geq s \geq \lfloor N_t^*(a) - \delta_t \rfloor \right\},$$

and its complementary set $\mathcal{G}_t^{\mathsf{c}} = \{1, 2, \ldots, t\} \setminus \mathcal{G}_t$.

By Theorem 5, $\forall a \in \{1, \ldots, K\}$, with probability at least $1 - O\left(t \exp\left(-\frac{t}{(\log t)^3}\right)\right)$, $N_t(a) \in \mathcal{G}_t$. By law of total expectation and Lemma 5,

$$\begin{aligned}\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\right] = & \sum_{s=1}^{t} \mathbb{P}\left(N_t(a) = s\right) \mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right) \middle| N_t(a) = s\right] \\ = & \sum_{s=1}^{t} \mathbb{P}\left(N_t(a) = s\right)\left(e^{r(a)/\tau} + \frac{\sigma^2}{2s}\left(\frac{e^{r(a)/\tau}}{\tau^2}\right)\right) + \sum_{s=1}^{t} \mathbb{P}\left(N_t(a) = s\right) O\left(s^{-2}\right) \\ = & \sum_{s=1}^{t} \mathbb{P}\left(N_t(a) = s\right)\left(\frac{\sigma^2}{2s}\left(\frac{e^{r(a)/\tau}}{\tau^2}\right) + O\left(s^{-2}\right)\right) + e^{r(a)/\tau}\end{aligned} \tag{B.3}$$

We divide the summation in two parts. For $s \in \mathcal{G}_t^{\mathsf{c}}$, by Theorem 5,

$$\sum\nolimits_{s \in \mathcal{G}_t^{\mathsf{c}}} \mathbb{P}\left(N_t(a) = s\right) \cdot \left(\frac{\sigma^2}{2s}\left(\frac{e^{r(a)/\tau}}{\tau^2}\right) + O\left(s^{-2}\right)\right) \leq O\left(\frac{1}{t}\right) \tag{B.4}$$

For $s \in \mathcal{G}_t$,

$$\sum\nolimits_{s\in\mathcal{G}_t} \mathbb{P}\left(N_t(a)=s\right)\cdot\left(\frac{\sigma^2}{2s}\left(\frac{e^{r(a)/\tau}}{\tau^2}\right)+O\left(s^{-2}\right)\right)\leq O\left(\left(N_t^*(a)-\delta_t\right)^{-1}\right) \tag{B.5}$$

Combine the above together,

$$\begin{aligned} t\left(U-\mathbb{E}\left[U_t\right]\right)^2 &= t\left(\sum_a \mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\right]-\exp\left(\frac{r_t(a)}{\tau}\right)\right)^2 \\ &= t\left(\sum_a O\left(\frac{1}{t}\right)+O\left(\left(N_t^*(a)-\delta_t\right)^{-1}\right)\right)^2. \end{aligned}$$

Thus,

$$\lim_{t\to\infty} t\left(U^*-\mathbb{E}\left[U_t\right]\right)^2=0,$$

i.e. U_t is a consistent estimate for U^*.

To bound $\mathcal{E}_t$, it remains to bound the variance of U_t since it is unbiased. By the law of total variance,

$$\operatorname{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)=\mathbb{E}\left[\operatorname{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right]+\operatorname{Var}\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right] \tag{B.6}$$

Note that by Lemma 5, the first term is

$$\begin{aligned} &\mathbb{E}\left[\operatorname{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right] \\ &=\sum_{s=1}^{t}\mathbb{P}\left(N_t(a)=s\right)\operatorname{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)=s \\ &=\sum_{s=1}^{t}\mathbb{P}\left(N_t(a)=s\right)\left(\frac{\sigma^2}{s}\left(\frac{e^{r(a)/\tau}}{\tau}\right)^2+O\left(s^{-\frac{3}{2}}\right)\right) \end{aligned}$$

Using the same idea in Eq. (B.4) and Eq. (B.5), we consider the summation in two parts. For $s \in \mathcal{G}_t^c$,

$$\sum\nolimits_{s\in\mathcal{G}_t^c} \mathbb{P}\left(N_t(a)=s\right)\cdot\left(\frac{\sigma^2}{s}\left(\frac{e^{r(a)/\tau}}{\tau}\right)^2+O\left(s^{-\frac{3}{2}}\right)\right)\leq O\left(\frac{1}{t}\right)$$

For $s \in \mathcal{G}_t$,

$$\sum\nolimits_{s\in\mathcal{G}_t} \mathbb{P}\left(N_t(a)=s\right)\cdot\left(\frac{\sigma^2}{s}\left(\frac{e^{r(a)/\tau}}{\tau}\right)^2+O\left(s^{-\frac{3}{2}}\right)\right)\leq \frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t}+O\left(\left(N_t^*(a)-\delta_t\right)^{-\frac{3}{2}}\right)$$

Put these together we have,

$$\mathbb{E}\left[\text{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right] \leq O\left(\frac{1}{t}\right) + \frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t} + O\left(\left(N_t^*(a)-\delta_t\right)^{-\frac{3}{2}}\right) \tag{B.7}$$

For the second term of Eq. (B.6) we have,

$$\text{Var}\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right] = \mathbb{E}\left[\left(\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right]\right)^2\right] - \left(\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\right]\right)^2$$

For the first term, by Lemma 5,

$$\begin{aligned}
&\mathbb{E}\left[\left(\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right]\right)^2\right] \\
&= \sum_{s=1}^{t}\mathbb{P}\left(N_t(a)=s\right)\left(\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right]\right)^2 \\
&= \sum_{s=1}^{t}\mathbb{P}\left(N_t(a)=s\right)\left(e^{2r(a)/\tau}+\frac{\sigma^2}{s}\left(\frac{e^{r(a)/\tau}}{\tau}\right)^2\right) + O\left(s^{-3/2}\right) \\
&\leq e^{2r(a)/\tau} + O\left(\frac{1}{t}\right) + \frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t} + O\left(\left(N_t^*(a)-\delta_t\right)^{-\frac{3}{2}}\right)
\end{aligned}$$

where the last inequality follows by the same idea of proving (B.7). For the second term, combining Eqs. (B.3) to (B.5),

$$\left(\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\right]\right)^2 = \exp\left(\frac{2r(a)}{\tau}\right) + O\left(\frac{1}{t}\right) + O\left(\left(N_t^*(a)-\delta_t\right)^{-1}\right)$$

Then we have,

$$\text{Var}\mathbb{E}\left[\exp\left(\frac{\hat{r}_t(a)}{\tau}\right)\middle|N_t(a)\right] \leq O\left(\frac{1}{t}\right) + \frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t} + O\left(\left(N_t^*(a)-\delta_t\right)^{-1}\right) \tag{B.8}$$

Note that

$$\begin{aligned}
\lim_{t\to\infty} t\cdot\frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t} &= \lim_{t\to\infty}\frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{\pi^*(a)-\frac{\delta_t}{t}} \\
&= \frac{\sigma^2}{\tau^2}\cdot\frac{e^{r(a)/\tau}}{\bar{\pi}(a)}\cdot\left(\sum_a \bar{\pi}(a)\exp(r(a)/\tau)\right)
\end{aligned} \tag{B.9}$$

Combine Eq. (B.7), Eq. (B.8) and Eq. (B.9) together,

$$\begin{aligned}
&\lim_{t\to\infty} t\mathrm{Var}\hat{U}_t \\
&= \lim_{t\to\infty} t\left(\sum_a \bar{\pi}^2(a)\mathrm{Varexp}\left(\frac{\hat{r}_t(a)}{\tau}\right)\right) \\
&\le \lim_{t\to\infty} t\sum_a \bar{\pi}^2(a)\left(O\left(\frac{1}{t}\right) + \frac{\sigma^2}{\tau^2}\cdot\frac{e^{2r(a)/\tau}}{N_t^*(a)-\delta_t}\right) \\
&\quad + t\sum_a \bar{\pi}^2(a)O\left((N_t^*(a)-\delta_t)^{-1}\right) \\
&= \frac{\sigma^2}{\tau^2}\left(\sum_a \bar{\pi}(a)e^{r(a)/\tau}\right)^2
\end{aligned}$$

which ends the proof. □

B.2.5 Technical Lemmas

Proof of Lemma 6. Consider two gaussian environments ν_1 and ν_2 with unit variance. The vector of means of the reward per arm in ν_1 is $(r(1),\ldots,r(K))$ and $(r(1)+2\epsilon, r(2),\ldots,r(K))$ in ν_2. Define

$$U_1 = \sum\nolimits_{i=1}^{K} e^{r_i}, \qquad U_2 = e^{r_1+2\epsilon} + \sum\nolimits_{i=2}^{K} e^{r_i}$$

Let $\mathbb{P}_1$ and $\mathbb{P}_2$ be the distribution induced by ν_1 and ν_2 respectively. Denote the event,

$$E = \left\{|\hat{U}_t - U_1| > e^{r_1}\epsilon\right\},$$

By definition, the error $\mathcal{E}_{t,\nu_1}$ under ν_1 satisfies

$$\mathcal{E}_{t,\nu_1} \ge \mathbb{P}_1(E)\,\mathbb{E}\left[(U_1-\hat{U}_t)^2 \mid E\right] \ge \mathbb{P}_1(E)\,e^{2r_1}\epsilon^2,$$

and the error $\mathcal{E}_{t,\nu_2}$ under ν_2 satisfies

$$\mathcal{E}_{t,\nu_2} \ge \mathbb{P}_2(E^c)\,\mathbb{E}\left[(U_2-\hat{U}_t)^2 \mid E^c\right] \ge \mathbb{P}_2(E^c)\,e^{2r_1}\epsilon^2.$$

Therefore, under the assumption that the algorithm suffers $O(\frac{1}{t})$ error in both environments,

$$\begin{aligned}
O(\frac{1}{t}) &= \mathcal{E}_{t,P_1} + \mathcal{E}_{t,P_2} \ge \mathbb{P}_1(E)\,e^{2r_1}\epsilon^2 + \mathbb{P}_2(E^c)\,e^{2r_1}\epsilon^2 \\
&= e^{2r_1}\epsilon^2\left(\mathbb{P}_1(E) + \mathbb{P}_2(E^c)\right) \ge \frac{1}{2}e^{2r_1}\epsilon^2 e^{-2N_t(k)\epsilon^2}.
\end{aligned}$$

where the last inequality follows by Pinsker's inequality and Divergence decomposition Lemma Lattimore & Szepesvári (2020). Therefore $N_t(k) = \Omega(\log(t))$.

□

Proof of Lemma 7. Define

$$\Gamma(a) = \Phi(a) - r(a) = \frac{N_t(a)}{N_t(a) + \tau_0}(\hat{r}(a) - r(a)) + \frac{1/2 - \tau_0 r(a)}{\tau_0 + N_t(a)}.$$

By the fact that the variance of $X_{a,t}$ given $\mathbf{r}$ is 1,

$$\mathbb{E}\left[\Gamma(a) \,|\, N_t(a), \mathbf{r}\right] = \frac{1/2 - \tau_0 r(a)}{\tau_0 + N_t(a)}.$$

$$\mathbb{E}\left[\Gamma(a)^2 \,\middle|\, N_t(a), \mathbf{r}\right] = \frac{\sigma^2 N_t(a)}{(N_t(a) + \tau_0)^2} + O\left(\frac{1}{N_t^2(a)}\right),$$

Then we have

$$\begin{aligned}\mathbb{E}\left[\Delta(a) | N_t(a), \mathbf{r}\right] &= e^{r(a)} - \mathbb{E}\left[e^{\Phi(a)} | N_t(a), \mathbf{r}\right] \\ &= e^{r(a)}\left(1 - \mathbb{E}\left[e^{\Gamma(a)} | N_t(a), \mathbf{r}\right]\right) = O\left(\frac{1}{N_t(a)}\right)\end{aligned}$$

Similarly,

$$\mathbb{E}\left[\Delta(a)^2 \,\middle|\, N_t(a), \mathbf{r}\right] = e^{2r(a)}\left(\frac{N_t(j)}{(N_t(j) + \sigma_0)^2} + O\left(\frac{1}{N_t^2(j)}\right)\right).$$

□

Proof of Lemma 8. By the choice of $\lambda_s = \frac{|\mathcal{A}|}{\log(1+s)}$, it follows that for all a and $t \geq 4$,

$$\begin{aligned}\tilde{N}_t(a) &= \sum_{s=1}^{t} \pi_s(a) \geq \sum_{s=1}^{t} \frac{1}{\log(1+s)} \\ &\geq \sum_{s=1}^{t} \frac{1}{\log(1+s)} - \frac{s/(s+1)}{(\log(1+s))^2} \\ &\geq \int_1^{1+t} \frac{1}{\log(1+s)} - \frac{s/(s+1)}{(\log(1+s))^2} ds \\ &= \frac{1+t}{\log(2+t)} - \frac{1}{\log 2} \\ &\geq \frac{t}{2\log(2+t)}\end{aligned}$$

Conditioned on the event $\widetilde{E}_\epsilon$ where we set $\epsilon = \frac{t}{4\log(2+t)}$, it follows that $N_t(a) \geq \frac{t}{4\log(2+t)}$. Then, for any action a by the definition of sub-gaussian,

$$\begin{aligned}
&\mathbb{P}\left(|r(a) - \hat{r}_t(a)| > \sqrt{\frac{8\sigma^2 \log(\frac{2}{\delta}) \log(2+t)}{t}}\right) \\
\leq &\mathbb{P}\left(|r(a) - \hat{r}_t(a)| > \sqrt{\frac{2\sigma^2 \log(\frac{2}{\delta})}{N_t(a)}}\right) \leq \delta.
\end{aligned}$$

Let δ satisfy that $\log(2/\delta) = \frac{t}{(\log(2+t))^3}$,

$$\mathbb{P}\left(|r(a) - \hat{r}_t(a)| > \frac{2\sigma}{\log(2+t)}\right) \leq 2\exp\left(-\frac{t}{(\log(2+t))^3}\right)$$

Therefore for $t \geq 2$

$$\begin{aligned}
&\mathbb{P}\left(\|\mathbf{r}_t - \hat{\mathbf{r}}_t\|_\infty \geq \frac{2\sigma}{\log(2+t)}\right) \\
\leq &\mathbb{P}\left(\|\mathbf{r}_t - \hat{\mathbf{r}}_t\|_\infty \geq \frac{2\sigma}{\log(2+t)} \,\middle|\, \widetilde{E}_\epsilon\right) + \mathbb{P}\left(\widetilde{E}_\epsilon^c\right) \\
\leq &\sum_k \mathbb{P}\left(|r(a) - \hat{r}_t(a)| > \frac{2\sigma}{\log(2+t)} \,\middle|\, \widetilde{E}_\epsilon\right) + \mathbb{P}\left(\widetilde{E}_\epsilon^c\right) \\
\leq &2|\mathcal{A}|\exp\left(-\frac{t}{(\log(2+t))^3}\right) + 2|\mathcal{A}|\exp\left(-\frac{t}{2(\log(t+2))^2}\right) \\
\leq &4|\mathcal{A}|\exp\left(-\frac{t}{(\log(2+t))^3}\right)
\end{aligned}$$

□

Proof of Lemma 9. Note that

$$\begin{aligned}
\left\|\pi^{(1)} - \pi^{(2)}\right\|_\infty &\leq \left\|\log \pi^{(1)} - \log \pi^{(2)}\right\|_\infty \\
&\leq \frac{1}{\tau}\left\|\mathbf{r}^{(1)} - \mathbf{r}^{(2)}\right\|_\infty + \left|\mathcal{F}_\tau(\mathbf{r}^{(1)}) - \mathcal{F}_\tau(\mathbf{r}^{(2)})\right|
\end{aligned}$$

The proof ends by using the fact $\left|\mathcal{F}_\tau(\mathbf{r}^{(1)}) - \mathcal{F}_\tau(\mathbf{r}^{(2)})\right| \leq \left\|\mathbf{r}^{(1)} - \mathbf{r}^{(2)}\right\|_\infty$, which follows Lemma 8 of Nachum et al. (2017). □

B.3 Proofs for Tree

This section contains the detailed proof for theorems in the tree setting, in particular, Theorem 6 and Theorem 7.

B.3.1 Proof of Theorem 6

Proof. We prove this using induction on the depth D of tree. For the base case (D=0), the result directly follows by the fact ν is sub-gaussian. Now, at some internal node $n(s) \in \mathcal{T}$, assume the result holds for all its children, we prove the result still holds.

For any state s, we define $\mathrm{EV}(s) = \exp(v_{\mathrm{sft}}(s)/\tau)$ and $\mathrm{EV}^*(s) = \exp(v^*_{\mathrm{sft}}(s)/\tau)$. Note that

$$\mathrm{EV} - \mathrm{EV}^* \geq \epsilon \mathrm{EV}^* \Leftrightarrow V \geq \tau \log(1+\epsilon) + V^*$$
$$\mathrm{EV}^* - \mathrm{EV} \geq \epsilon \mathrm{EV}^* \Leftrightarrow V \leq \tau \log(1-\epsilon) + V^*$$

Therefore it is equivalent to prove for any node in tree,

$$\mathbb{P}\left(|\mathrm{EV}(s) - \mathrm{EV}^*(s)| \geq \epsilon \mathrm{EV}^*(s) | E_s\right) \leq \tilde{C} \exp\left\{-\frac{\epsilon^2 N(s)}{C\sigma^2}\right\}$$

for some constant C and $\tilde{C}$. Note that by the definition of U we have

$$\mathrm{EV}(s) = \sum_a \exp(q_{\mathrm{sft}}(s,a)/\tau) = \sum_a \exp\{(r(s,a) + v_{\mathrm{sft}}(s_a))/\tau\}$$

where s_a is the state reached by taking action a at state s. Since the reward is deterministic and bounded which only affects the scale, we can then only consider the convergence of $v_{\mathrm{sft}}(s_a)$. Consider a decompose vector α such that $\sum_a \alpha_a \mathrm{EV}^*(s_a) = \epsilon \mathrm{EV}^*(s)$.

$$\begin{aligned}\mathbb{P}\left(|\mathrm{EV}(s) - \mathrm{EV}^*(s)| \geq \epsilon \mathrm{EV}^*(s) \,|\, E_s\right) &\leq \sum_a \mathbb{P}\left(|\mathrm{EV}(s_a) - \mathrm{EV}^*(s_a)| \geq \alpha_a \mathrm{EV}^*(s_a) \,|\, E_s\right) \\ &\leq \sum_a \tilde{C}_a \exp\left(-\frac{\alpha_a^2 N(s) \pi^*_{\mathrm{sft}}(a|s)}{2C_a\sigma^2}\right),\end{aligned}$$

where the last inequality is by the induction hypothesis. Let $\alpha_a^2 \pi^*_{\mathrm{sft}}(a|s) = M$ where $\sqrt{M} = \frac{\epsilon \mathrm{EV}^*(s)}{\sum_a \mathrm{EV}^*(s_a)/\sqrt{\pi^*_{\mathrm{sft}}(a|s)}}$. One can verify that $\sum_a \alpha_a \mathrm{EV}^*(s_a) =$

$\epsilon \mathrm{EV}^*(s)$. Therefore,

$$\begin{aligned}
\mathbb{P}\left(|\mathrm{EV}(s) - \mathrm{EV}^*(s)| \geq \epsilon \mathrm{EV}^*(s)\right) \leq & \sum_a \tilde{C}_a \exp\left(-\frac{N(s)}{2C_a\sigma^2}\left(\frac{\epsilon \mathrm{EV}^*(s)}{\sum_a \mathrm{EV}^*(s_a)/\sqrt{\pi^*_{\mathrm{sft}}(a|s)}}\right)^2\right) \\
\leq & |\mathcal{A}|\tilde{C}\exp\left(-\frac{\epsilon^2 N(s)}{2C\sigma^2}\frac{\mathrm{EV}^*(s)^2}{\left(\sum_a \sqrt{\mathrm{EV}^*(s)\mathrm{EV}^*(s_a)}\right)^2}\right) \\
\leq & |\mathcal{A}|\tilde{C}\exp\left(-\frac{\epsilon^2 N(s)}{2C\sigma^2}\frac{\mathrm{EV}^*(s)}{\left(\sum_a \sqrt{\mathrm{EV}^*(s_a)}\right)^2}\right) \\
\leq & |\mathcal{A}|\tilde{C}\exp\left(-\frac{1}{|\mathcal{A}|}\frac{\epsilon^2 N(s)}{2C\sigma^2}\right) \\
\leq & \tilde{C}_1 \exp\left(-\frac{\epsilon^2 N(s)}{\tilde{C}_2\sigma^2}\right).
\end{aligned}$$

Picking $\tilde{C} = \max\{\tilde{C}_1, \tilde{C}_2\}$ leads to the conclusion. □

B.3.2 Proof of Theorem 7

Proof. Let a^* be the action with largest softmax value and s be the root state. Moreover, let $U(s_a) = \exp\left(q_{\mathrm{sft}}(s,a)/\tau\right)$ and $U^*(s_a) = \exp\left(q^*_{\mathrm{sft}}(s,a)/\tau\right)$. The event E_s is defined as in Theorem 6. The probability that MENT selects an sub-optimal arm at s is,

$$\begin{aligned}
\mathbb{P}\left(\exists a \in \mathcal{A}, U(s_a) > U(s_{a^*})\right) \leq & \mathbb{P}\left(\exists a \in \mathcal{A}, U(s_a) > U(s_{a^*}) \,|\, E_s\right) + \mathbb{P}\left(E_s^c\right) \\
\leq & \sum_a \mathbb{P}\left(U(s_a) > U(s_{a^*}) \,|\, E_s\right) + \mathbb{P}\left(E_s^c\right).
\end{aligned}$$

Since we can upper bound $\mathbb{P}\left(E_s^c\right)$ by Theorem 5, it remains to bound $\mathbb{P}\left(U(s_a) > U(s_{a^*}) \,|\, E_s\right)$.

$$\begin{aligned}
& \mathbb{P}\left(U(s_a) > U(s_{a^*}) \,|\, E_s\right) \\
= & \mathbb{P}\left(U(s_a) - U(s_{a^*}) - U^*(s_a) + U^*(s_{a^*}) > U^*(s_{a^*}) - U^*(s_a) \,|\, E_s\right) \\
\leq & \mathbb{P}\left(|U(s_{a^*}) - U^*(s_{a^*})| > \alpha U^*(s_{a^*}) \,|\, E_s\right) + \mathbb{P}\left(|U(s_a) - U^*(s_a)| > \beta U^*(s_a) \,|\, E_s\right) \\
\leq & \tilde{C}_{a^*} \exp\left\{-\frac{N^*(s,a^*)\alpha^2}{2C_{a^*}\sigma^2}\right\} + \tilde{C}_a \exp\left\{-\frac{N^*(s,a)\beta^2}{2C_a\sigma^2}\right\}
\end{aligned}$$

where $\alpha U^*(s_{a^*}) + \beta U^*(s_a) = U^*(s_{a^*}) - U^*(s_a)$. The last inequality follows by Theorem 6, since $U(s_a) - U^*(s_a) = \exp(r(s,a))\left(\exp\left(v_{\mathrm{sft}}(s')\right) - \exp\left(v^*_{\mathrm{sft}}(s')\right)\right)$,

where s' is the state of the child of $n(s)$ taking action a. Recall that for any action a, $N^*(s,a) = t \cdot \pi^*_{\text{sft}}(a|s)$. We can choose α and β similarly as in the proof,

$$\begin{aligned}\alpha =& \frac{(U^*(s_{a^*}) - U^*(s_a))/\sqrt{\pi^*_{\text{sft}}(a^*|s)}}{U^*(s,a)/\sqrt{\pi^*_{\text{sft}}(a|s)} + U^*(s,a^*)/\sqrt{\pi^*_{\text{sft}}(a^*|s)}} \\ \beta =& \frac{(U^*(s_{a^*}) - U^*(s_a))/\sqrt{\pi^*_{\text{sft}}(a|s)}}{U^*(s,a)/\sqrt{\pi^*_{\text{sft}}(a|s)} + U^*(s,a^*)/\sqrt{\pi^*_{\text{sft}}(a^*|s)}} \quad .\end{aligned}$$

Then, there exists some constant C_a and C'_a such that

$$\mathbb{P}\left(U(s_a) > U(s_{a^*}) \,|\, E_s\right) \leq C'_a \exp\left(-\frac{t}{2C_a\sigma^2}\frac{(U^*(s_{a^*}) - U^*(s_a))^2}{U^*(s)(\sqrt{U^*(s,a)} + \sqrt{U^*(s,a^*)})^2}\right) .$$

We can omit the terms depending on U^* since they only affect the scale (we can switch to a new constant C'_a.) Finally, by Theorem 5,

$$\begin{aligned}\mathbb{P}\left(\exists a \in \mathcal{A}, U(s_a) > U(s_{a^*})\right) &\leq \sum_a \mathbb{P}\left(U(s_a) > U(s_{a^*}) \,|\, E_s\right) + \mathbb{P}\left(E_s^c\right) \\ &\leq \sum_a C'_a \exp\left\{-\frac{t}{2C_a\sigma^2}\right\} + C't\exp\left\{-\frac{t}{(\log t)^3}\right\} \\ &\leq Ct\exp\left\{-\frac{t}{(\log t)^3}\right\}\end{aligned}$$

for some constant C not depending on t. □

Appendix C

Proofs for Chapter 6: On the Optimality of Batch Policy Optimization Algorithms

C.1 Proof of Minimax Results

C.1.1 Proof of Theorem 8

Let $m \geq 2$ and $\mu^1, \ldots, \mu^m$ be a collection of vectors in $\mathbb{R}^k$ with $\mu_a^b = \Delta \mathbb{I}\{a = b\}$ where $\Delta > 0$ is a constant to be chosen later. Next, let θ_b be the environment in $\Theta_{\mathbf{n}}$ with P_a a Gaussian distribution with mean μ_a^b and unit variance. Let B be a random variable uniformly distributed on $[m]$ where $m \in [k]$. The Bayesian regret of an algorithm $\mathcal{A}$ is

$$\mathcal{BR}^* = \inf_{\mathcal{A}} \mathbb{E}\left[\mathcal{R}(\mathcal{A}, \theta_B)\right] = \Delta \mathbb{E}\left[\mathbb{I}\{A \neq B\}\right] ,$$

where $A \in [k]$ is the $\sigma(X)$-measurable random variable representing the decision of the Bayesian optimal policy, which is $A = \arg\max_{b \in [k]} \mathbb{P}\{B = b | X\}$. By Bayes' law and the choice of uniform prior,

$$\begin{aligned}
\mathbb{P}\{B = b | X\} &\propto \exp\left(-\frac{1}{2}\sum_{a=1}^{k} n_a(\hat{\mu}_a - \mu_a^b)^2\right) \\
&= \exp\left(-\frac{1}{2}\sum_{a=1}^{k} n_a(\hat{\mu}_a - \Delta \mathbb{I}\{a = b\})^2\right) .
\end{aligned}$$

Therefore, the Bayesian optimal policy chooses

$$A = \arg\min_{b \in [k]} n_b(\Delta/2 - \hat{\mu}_b) .$$

On the other hand,

$$\mathcal{BR}^* = \Delta \mathbb{P}\{A \neq B\} = \frac{\Delta}{k} \sum_{b=1}^{k} \mathbb{P}_b(A \neq b)\,,$$

where $\mathbb{P}_b = \mathbb{P}\{\cdot | B = b\}$. Let $b \in [m]$ be arbitrary. Then,

$$\begin{aligned}
&\mathbb{P}_b\{A \neq b\} \\
&\geq \mathbb{P}_b\left\{\hat{\mu}_b \leq \Delta \text{ and } \max_{a \in [m]\setminus\{b\}} \hat{\mu}_a \geq \frac{\Delta}{2}\left(1 + \frac{n_b}{n_a}\right)\right\} \\
&\geq \frac{1}{2}\left(1 - \prod_{a \in [m]\setminus\{b\}} \left(1 - \mathbb{P}_b\left\{\hat{\mu}_a \geq \frac{\Delta}{2}\left(1 + \frac{n_b}{n_a}\right)\right\}\right)\right) \\
&\geq \frac{1}{2}\left(1 - \prod_{a > b} \left(1 - \mathbb{P}_b\left\{\hat{\mu}_a \geq \Delta\right\}\right)\right)\,,
\end{aligned}$$

where in the second inequality we used independence and the fact that the law of $\hat{\mu}_b$ under $\mathbb{P}_b$ is Gaussian with mean Δ and variance $1/n_b$. The first inequality follows because

$$\left\{\hat{\mu}_b \leq \Delta \text{ and } \max_{a \neq b} \hat{\mu}_a \geq \frac{\Delta}{2}\left(1 + \frac{n_b}{n_a}\right)\right\} \subset \{A \neq b\}\,.$$

Let $b < a \leq m$ and

$$\delta_a(\Delta) = \frac{1}{\Delta\sqrt{n_a} + \sqrt{4 + n_a\Delta^2}} \sqrt{\frac{2}{\pi}} \exp\left(-\frac{n_a\Delta^2}{2}\right)\,.$$

Since for $a \neq b$, $\hat{\mu}_a$ has law $\mathcal{N}(0, 1/n_a)$ under $\mathbb{P}_b$, by standard Gaussian tail inequalities (Abramowitz et al., 1988, §26),

$$\mathbb{P}_b\{\hat{\mu}_a \geq \Delta\} = \mathbb{P}_b\{\hat{\mu}_a\sqrt{n_a} \geq \Delta\sqrt{n_a}\} \geq \delta_a(\Delta) \geq \delta_m(\Delta)\,,$$

where the last inequality follows from our assumption that $n_1 \leq \cdots \leq n_k$. Therefore, choosing Δ so that $\delta_m(\Delta) = 1/(2m)$,

$$\begin{aligned}
\mathcal{BR}^* &\geq \frac{\Delta}{2m} \sum_{b \in [m]} \left(1 - (1 - \delta_m(\Delta))^{m-b}\right) \\
&\geq \frac{\Delta}{2m} \sum_{b \in [m]} \left(1 - \left(1 - \frac{1}{2m}\right)^{m-b}\right) \\
&\geq \frac{\Delta}{2m} \sum_{b \leq m/2} \left(1 - \left(1 - \frac{1}{2m}\right)^{m/2}\right) \\
&\geq \frac{\Delta(m-1)}{20m} \geq \frac{\Delta}{40}\,.
\end{aligned}$$

A calculation shows there exists a universal constant $c > 0$ such that

$$\Delta \geq c\sqrt{\frac{\log(m)}{n_m}}\,,$$

which shows there exists a (different) universal constant $c > 0$ such that

$$\inf_{\mathcal{A}} \sup_{\theta} \mathcal{R}(\mathcal{A}, \theta) \geq \mathcal{BR}^* \geq \max_{m \geq 2} c\sqrt{\frac{\log(m)}{n_m}}\,.$$

The argument above relies on the assumption that $m \geq 2$. A minor modification is needed to handle the case where n_1 is much smaller than n_2. Let B be uniformly distributed on $\{1, 2\}$ and let $\theta_1, \theta_2 \in \Theta_{\mathbf{n}}$ be defined as above, but with $\mu^1 = (\Delta, 0)$ and $\mu^2 = (-\Delta, 0)$ for some constant $\Delta > 0$ to be tuned momentarily. As before, the Bayesian optimal policy has a simple closed form solution, which is

$$A = \begin{cases} 1 & \text{if } \hat{\mu}_1 \geq 0 \\ 2 & \text{otherwise}\,. \end{cases}$$

The Bayesian regret of this policy satisfies

$$\begin{aligned}
\mathcal{BR}^* &= \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_1) + \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_2) \geq \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_1) \\
&\geq \frac{1}{2}\mathbb{P}_1\{A = 2\} \geq \frac{\Delta}{2}\mathbb{P}_1\{\hat{\mu}_1 < 0\} \\
&\geq \sqrt{\frac{2}{\pi}} \frac{\Delta}{2\Delta\sqrt{n_1} + 2\sqrt{4 + n_1\Delta^2}} \exp\left(-\frac{n_1\Delta^2}{2}\right) \\
&\geq \frac{1}{13}\sqrt{\frac{1}{n_1}}\,,
\end{aligned}$$

where the final inequality follows by tuning Δ.

C.1.2 Proof of Theorem 9

Proof. Let $i' = \arg\max_i \tilde{\mu}_i$. Then, given that (6.2) is true for all arms, which is with probability at least $1-\delta$, we have

$$\begin{aligned}
\mu^* - \mu_{i'} &= \mu^* - \tilde{\mu}_{a^*} + \tilde{\mu}_{a^*} - \tilde{\mu}_{i'} + \tilde{\mu}_{i'} - \mu_{i'} \\
&\leq \mu^* - \tilde{\mu}_{a^*} + \tilde{\mu}_{i'} - \mu_{i'} \\
&\leq \mu^* - \hat{\mu}_{a^*} + \hat{\mu}_{i'} - \mu_{i'} + 2\sqrt{\frac{2\log(k/\delta)}{\min_i n_i}} \\
&\leq \sqrt{\frac{32\log(k/\delta)}{\min_i n_i}},
\end{aligned}$$

where the first two inequalities follow from the definition of the index algorithm, and the last follows from (6.2). Using the tower rule gives the desired result. □

C.2 Proof of Instance-dependent Results

C.2.1 Instance-dependent Upper Bound

Proof of Theorem 10. Assuming $\mu_1 \geq \mu_2 \geq ... \geq \mu_k$, if we have $\mathbb{P}(\mathcal{A}(X) \geq i) \leq b_i$, then we can write

$$\begin{aligned}
\mathcal{R}(\mathcal{A}) &= \sum_{2\leq i\leq k} \Delta_i \mathbb{P}(\mathcal{A}(X) = i) \\
&= \sum_{2\leq i\leq k} \Delta_i \left(\mathbb{P}(\mathcal{A}(X) \geq i) - \mathbb{P}(\mathcal{A}(X) \geq i+1)\right) \\
&= \sum_{2\leq i\leq k} (\Delta_i - \Delta_{i-1}) \mathbb{P}(\mathcal{A}(X) \geq i) \\
&\leq \sum_{2\leq i\leq k} (\Delta_i - \Delta_{i-1}) b_i \\
&= \sum_{2\leq i\leq k} \Delta_i (b_i - b_{i+1}).
\end{aligned}$$

To upper bound $\mathbb{P}(\mathcal{A}(X) \geq i)$, let I_i be the index used by algorithm $\mathcal{A}$, i.e., $\mathcal{A}(X) = \arg\max_i I_i$. Then

$$\mathbb{P}(\mathcal{A}(X) \geq i) \leq \mathbb{P}\left(\max_{j\geq i} I_j \geq \max_{j<i} I_j\right).$$

Hence we can further write

$$\begin{aligned}\mathbb{P}\left(\mathcal{A}(X) \geq i\right) &\leq \mathbb{P}\left(\max_{j\geq i} I_j \geq \max_{j<i} I_i, \max_{j<i} I_j \geq \eta\right) \\ &\quad + \mathbb{P}\left(\max_{j\geq i} I_j \geq \max_{j<i} I_i, \max_{j<i} I_j < \eta\right) \\ &\leq \mathbb{P}\left(\max_{j\geq i} I_j \geq \eta\right) + \mathbb{P}\left(\max_{j<i} I_j < \eta\right) . \end{aligned} \tag{C.1}$$

Next we optimize the choice of η according to the specific choice of the index. For this let $I_i = \hat{\mu}_i + b_i$.

Continuing with equation C.1, for the first term, by the union bound we have

$$\mathbb{P}\left(\max_{j\geq i} I_j \geq \eta\right) \leq \sum_{j\geq i} \mathbb{P}\left(I_j \geq \eta\right) .$$

For each $j \geq i$, by Hoeffding's inequality we have

$$\mathbb{P}\left(I_j \geq \eta\right) \leq e^{-\frac{n_j}{2}(\eta-\mu_j-b_j)_+^2} .$$

For the second term in equation C.1, we have $\mathbb{P}\left(\max_{j<i} I_j < \eta\right) \leq \mathbb{P}\left(I_j < \eta\right)$ for each $j < i$.

By Hoeffding's inequality we have

$$\mathbb{P}\left(I_j < \eta\right) \leq e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2} ,$$

and thus

$$\mathbb{P}\left(\max_{j<i} I_j < \eta\right) \leq \min_{j<i} e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2} .$$

Define

$$g_i(\eta) = \sum_{j\geq i} e^{-\frac{n_j}{2}(\eta-\mu_j-b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2}$$

and $g_i^* = \min_\eta g_i(\eta)$. Then we have

$$\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\} .$$

Putting everything together, we bound the expected regret as

$$\mathcal{R}(\mathcal{A}) \leq \sum_{2\leq i\leq k} \Delta_i \left(\min\{1, g_i^*\} - \min\{1, g_{i+1}^*\}\right)$$

where we define $g_{k+1}^* = 0$.

□

Proof of Remark 3. Recall the definition of $g_i(\eta)$:

$$g_i(\eta) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2}.$$

Let $\eta = \mu_1 - 2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}$. Then, for the second term of $g_i(\eta)$,

$$\min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2} \leq e^{-\frac{n_1}{2}\left(2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}} - \sqrt{\frac{2}{n_1} \log \frac{k}{\delta}}\right)_+^2} \leq \frac{\delta}{k}.$$

For the first term,

$$\sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} = \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\mu_1 - 2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}} - \mu_j - b_j\right)_+^2} \leq \sum_{j \geq i} e^{-\frac{n_{\min}}{2}\left(\Delta_j - 3\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}\right)_+^2}.$$

Thus,

$$g_i^* \leq \sum_{j \geq i} e^{-\frac{n_{\min}}{2}\left(\Delta_j - 3\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}\right)_+^2} + \frac{\delta}{k}.$$

For arm i such that $\Delta_i \geq 4\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}$, by Theorem 10 we have $P(\mathcal{A}(X) \geq i) \leq g_i^* \leq \delta$. The result then follows by the tower rule. □

Proof of Corollary 1. For each i, let $\eta_i = \max_{j<i} L_j$. Then,

$$g_i(\eta_i) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\max_{j<i} L_j - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \max_{j<i} L_j)_+^2}.$$

Let $s = \arg\max_{j<i} L_j$. For the second term we have,

$$\min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \max_{j<i} L_j)_+^2} \leq e^{-\frac{n_s}{2}(\mu_s + b_s - L_s)_+^2} \leq \frac{\delta}{k}.$$

Next we consider the first term. Recall that $h = \max\{i \in [k] : \max_{j<i} L_j < \max_{j' \geq i} U_{j'}\}$. Then for any $i > h$, we have $\max_{j<i} L_j \geq U_{j'}$ for all $j' \geq i$. Therefore,

$$\begin{aligned}
&\sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - \mu_j - b_j\right)_+^2} \\
=&\sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j + \sqrt{\frac{2}{n_j} \log \frac{k}{\delta}}\right)_+^2} \\
\leq&\frac{\delta}{k} \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2}.
\end{aligned}$$

Note that for $i \le h$, $\Delta_i \le \Delta_h$. Thus we have,

$$\begin{aligned}\mathcal{R}(\mathcal{A}) &\le \Delta_h + \sum_{i>h}(\Delta_i - \Delta_{i-1})\mathbb{P}(\mathcal{A}(X) \ge i) \\ &\le \Delta_h + \frac{\delta}{k}\Delta_{\max} + \frac{\delta}{k}\sum_{i>h}(\Delta_i - \Delta_{i-1})\sum_{j\ge i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2},\end{aligned}$$

which concludes the proof. □

Proof of Corollary 2. Considering the greedy algorithm, for each $i \ge 2$,

$$g_i(\eta) = \sum_{j\ge i} e^{-\frac{n_j}{2}(\eta-\mu_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j-\eta)_+^2}.$$

Define $h_i = \arg\max_{j<i}\mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$ and $\eta_i = \mu_{h_i} - \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}}$. Then we have $e^{-\frac{n_{h_i}}{2}\left(\mu_{h_i}-\eta_i\right)_+^2} = \delta/k$. Then for $j \ge i$ we have

$$e^{-\frac{n_j}{2}(\eta_i-\mu_j)_+^2} = e^{-\frac{n_j}{2}\left(\mu_{h_i}-\mu_j-\sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}}\right)_+^2}.$$

When $\mu_{h_i} - \mu_j \ge \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}} + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$ we have $e^{-\frac{n_j}{2}(\eta_i-\mu_j)_+^2} \le \delta/k$.

Define

$$U_i = \mathbb{I}\left\{\forall j \ge i, \mu_{h_i} - \mu_j \ge \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}} + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right\}.$$

Then we have $g_i^* U_i \le \frac{k-i+2}{k}\delta \le \delta$. According to Theorem 10 we have $\mathbb{P}\left(\mathcal{A}(X) \ge i\right) \le \min\{1, g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X) \ge i\right) > \delta$, we must have $U_i = 0$, which is equivalent to

$$\max_{j<i}\mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}} < \max_{j\ge i}\mu_j + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}. \tag{C.2}$$

Let $\hat{i}$ be the largest index i that satisfies Eq. (C.2). Then $\mathbb{P}\left(\mathcal{A}(X) \ge \hat{i}+1\right) \le \delta$. Thus we have $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \le \Delta_{\hat{i}}\right) \ge 1-\delta$, it remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \le i$ then $\Delta_{\hat{i}} \le \Delta_i$. If $\hat{i} > i$ we have

$$\max_{j<\hat{i}}\mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}} \ge \mu_i - \sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}$$

and

$$\max_{j\geq\hat{i}}\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\leq\mu_{\hat{i}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\,.$$

Applying equation C.2 gives

$$\Delta_{\hat{i}}-\Delta_i=\mu_i-\mu_{\hat{i}}\leq\sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\,,$$

so

$$\Delta_{\hat{i}}\leq\Delta_i+\sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$$

holds for any $i\in[k]$, concluding the proof. □

Proof of Corollary 3. Let $\eta=\max_i\mu_i-\sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}$. Considering the LCB algorithm, for each $i\geq 2$, we have

$$g_i(\eta)=\sum_{j\geq i}e^{-\frac{n_j}{2}\left(\eta-\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}+\min_{j<i}e^{-\frac{n_j}{2}\left(\mu_j-\eta-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}.$$

Define $h_i=\arg\max_{j<i}\mu_j-\sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}$ and $\eta_i=\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}$. Then we have $e^{-\frac{n_{h_i}}{2}\left(\mu_{h_i}-\eta_i-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}=\delta/k$. Now, consider $j\geq i$. Then,

$$e^{-\frac{n_j}{2}\left(\eta_i-\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}\leq\frac{\delta}{k}$$

whenever $\eta_i-\mu_j\geq 0$, i.e. $\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}\geq\mu_j$.

Define

$$U_i=\mathbb{I}\left\{\forall j\geq i,\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}\geq\mu_j\right\}.$$

Then we have $g_i^*U_i\leq\frac{k-i+2}{k}\delta\leq\delta$. According to Theorem 10 we have $\mathbb{P}\left(\mathcal{A}(X)\geq i\right)\leq\min\{1,g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X)\geq i\right)>\delta$, we must have $U_i=0$, which is equivalent to that there exists some $s\geq i$ such that

$$\mu_s>\max_{j<i}\mu_j-\sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}\,. \tag{C.3}$$

Let $\hat{i}$ be the largest index i that satisfies equation C.3. Then we have $\mathbb{P}\left(\mathcal{A}(X) \geq \hat{i}+1\right) \leq \delta$ and thus $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \leq \Delta_{\hat{i}}\right) \geq 1-\delta$. It remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \leq i$ then $\Delta_{\hat{i}} \leq \Delta_i$. If $\hat{i} > i$ we have

$$\mu_{\hat{i}} > \max_{j<\hat{i}} \mu_j - \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}} \geq \mu_i - \sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}\,.$$

Therefore,

$$\Delta_{\hat{i}} = \Delta_i + \mu_i - \mu_{\hat{i}} \leq \Delta_i + \sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}\,,$$

which concludes the proof. □

Proof of Corollary 4. Consider now the UCB algorithm. Then, for each $i \geq 2$,

$$g_i(\eta) = \sum_{j\geq i} e^{-\frac{n_j}{2}\left(\eta-\mu_j-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}\left(\mu_j-\eta+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}.$$

Pick $\eta = \mu_1$ then the second term in $g_i(\eta)$ becomes δ/k. For j such that $\Delta_j \geq \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}$ we have

$$e^{-\frac{n_j}{2}\left(\eta-\mu_j-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2} \leq \frac{\delta}{k}\,.$$

Define

$$U_i = \mathbb{I}\left\{\forall j \geq i, \Delta_j \geq \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}\right\}\,.$$

Then we have $g_i^* U_i \leq \frac{k-i+2}{k}\delta \leq \delta$. According to Theorem 10 we have $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) > \delta$, we must have $U_i = 0$, which is equivalent to

$$\max_{j\geq i} \mu_j + \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}} > \mu_1\,. \tag{C.4}$$

Let $\hat{i}$ be the largest index i that satisfies equation C.4. Then we have $\mathbb{P}\left(\mathcal{A}(X) \geq \hat{i}+1\right) \leq \delta$. Therefore, we have $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \leq \Delta_{\hat{i}}\right) \geq 1-\delta$. It remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \leq i$ then $\Delta_{\hat{i}} \leq \Delta_i$. If $\hat{i} > i$, we have

$$\max_{j \geq \hat{i}} \mu_j + \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \leq \mu_{\hat{i}} + \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \,.$$

Applying equation C.4 gives

$$\Delta_{\hat{i}} = \mu_1 - \mu_{\hat{i}} \leq \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \,.$$

Therefore,

$$\Delta_{\hat{i}} \leq \max \left\{ \Delta_i, \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \right\} \leq \Delta_i + \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \,.$$

for any $i \in [k]$, which concludes the proof. □

Proof of Proposition 2. Fixing $S \subset [k]$, we take $\{n_i\}_{i \in S} \to \infty$ and $\{n_i\}_{i \notin S} = 1$. The upper bound for LCB in Corollary 3 can be written as

$$\begin{aligned} \hat{\mathcal{R}}_S(\text{LCB}) &= \min \left\{ \min_{i \in S} \Delta_i, \min_{i \notin S} \left(\Delta_i + \sqrt{8 \log \frac{k}{\delta}} \right) \right\} + \delta \\ &= \min_{i \in S} \Delta_i + \delta \\ &= \Delta_{\min\{i \in [k] : i \in S\}} + \delta \,. \end{aligned}$$

Similarly, we have

$$\hat{\mathcal{R}}_S(\text{UCB}) = \min_{i \in [k]} \left(\Delta_i + \max_{j > i, j \notin S} \sqrt{8 \log \frac{k}{\delta}} \right) + \delta$$

and

$$\hat{\mathcal{R}}_S(\text{greedy}) \geq \min_{i \in [k]} \left(\Delta_i + \max_{j > i, j \notin S} \sqrt{2 \log \frac{k}{\delta}} \right) + \delta \,.$$

Note that for $\delta \in (0, 1)$, $\sqrt{2 \log \frac{k}{\delta}} > 1 \geq \Delta_{\max}$. So we can further lower bound $\hat{\mathcal{R}}_S(\text{UCB})$ and $\hat{\mathcal{R}}_S(\text{greedy})$ by $\Delta_h + \delta$ where $h = \min\{i \in [k] : \forall j > i, j \in S\}$. Let $m = |S|$. Notice that unless $S = \{k - m + 1, ..., k\}$, we always have $\min\{i \in [k] : i \in S\} < \min\{i \in [k] : \forall j > i, j \in S\}$. So we have $\hat{\mathcal{R}}_S(\text{LCB}) < \hat{\mathcal{R}}_S(\text{UCB})$ (or $\hat{\mathcal{R}}_S(\text{greedy})$) whenever $S \neq \{k-m+1, ..., k\}$. Under the uniform distribution over all possible subsets for S, the event $S = \{k - m + 1, ..., k\}$ happens with probability $\binom{k}{m}^{-1}$, which concludes the proof.

□

C.2.2 Instance-dependent Lower Bounds

Proof of Theorem 11. We first derive an upper bound for $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$. Assuming $X = (X_1, X_2, \mathbf{n})$ with $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$, for any $\beta \in \mathbb{R}$, we define algorithm $\mathcal{A}_\beta$ as

$$\mathcal{A}_\beta(X) = \begin{cases} 1, & \text{if } X_1 - X_2 \geq \frac{\beta}{\sqrt{n_{\min}}}; \\ 2, & \text{otherwise } . \end{cases}$$

We now analyze the regret for $\mathcal{A}_\beta$. By Hoeffding's inequality we have the following instance-dependent regret upper bound:

Proposition 6. *Consider any $\beta \in \mathbb{R}$ and $\theta \in \Theta_{\mathbf{n}}$. Let $\Delta = |\mu_1 - \mu_2|$. If $\mu_1 \geq \mu_2$ then*

$$\mathcal{R}(\mathcal{A}_\beta, \theta) \leq \mathbb{I}\left\{\Delta \leq \frac{\beta}{\sqrt{n_{\min}}}\right\} \frac{\beta}{\sqrt{n_{\min}}} + \mathbb{I}\left\{\Delta > \frac{\beta}{\sqrt{n_{\min}}}\right\} e^{-\frac{n_{\min}}{4}\left(\Delta - \frac{\beta}{\sqrt{n_{\min}}}\right)_+^2}.$$

Furthermore, if $\mu_1 < \mu_2$, we have

$$\mathcal{R}(\mathcal{A}_\beta, \theta) \leq \mathbb{I}\left\{\Delta \leq \frac{-\beta}{\sqrt{n_{\min}}}\right\} \frac{-\beta}{\sqrt{n_{\min}}} + \mathbb{I}\left\{\Delta > \frac{-\beta}{\sqrt{n_{\min}}}\right\} e^{-\frac{n_{\min}}{4}\left(\Delta + \frac{\beta}{\sqrt{n_{\min}}}\right)_+^2}.$$

Maximizing over Δ gives our worst case regret guarantee:

Proposition 7. *For any $\beta \in \mathbb{R}$,*

$$\sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}_\beta, \theta) \leq \frac{|\beta| + 2}{\sqrt{n_{\min}}}.$$

$\mathcal{A}_\beta(X)$ is minimax optimal for a specific range of β:

Proposition 8. *If $|\beta| \leq cc_0 - 2$ then $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$.*

Given $\theta \in \Theta_{\mathbf{n}}$, to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$, we pick β such that $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$ and $\mathcal{A}_\beta$ performs well on θ. For θ where $\mu_1 \geq \mu_2$, we set $\beta = 2 - cc_0$ thus $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_{2-cc_0}, \theta)$. For θ where $\mu_1 < \mu_2$, we set $\beta = cc_0 - 2$ thus $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_{cc_0-2}, \theta)$.

We now construct two instances $\theta_1, \theta_2 \in \Theta_{\mathbf{n}}$ and show that no algorithm can achieve regret close to $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}$ on both instances. Fixing some $\lambda \in \mathbb{R}$ and $\eta > 0$, we define

$$\theta_1 = (\mu_1, \mu_2) = (\lambda + \frac{\eta}{n_1}, \lambda - \frac{\eta}{n_2})$$

and

$$\theta_2 = (\mu_1', \mu_2') = (\lambda - \frac{\eta}{n_1}, \lambda + \frac{\eta}{n_2}) .$$

On instance θ_1 we have $X_1 - X_2 \sim \mathcal{N}((\frac{1}{n_1} + \frac{1}{n_2})\eta, \frac{1}{n_1} + \frac{1}{n_2})$ while on instance θ_2 we have $X_1 - X_2 \sim \mathcal{N}(-(\frac{1}{n_1} + \frac{1}{n_2})\eta, \frac{1}{n_1} + \frac{1}{n_2})$. Let Φ be the CDF of the standard normal distribution $\mathcal{N}(0,1)$, $\Delta = (\frac{1}{n_1} + \frac{1}{n_2})\eta$, and $\sigma^2 = \frac{1}{n_1} + \frac{1}{n_2}$. Then we have

$$\begin{aligned}
\mathcal{R}(\mathcal{A}_\beta, \theta_1) &= \Delta \mathbb{P}_{\theta_1} (\mathcal{A}_\beta = 2) \\
&= \Delta \mathbb{P}_{\theta_1} \left(X_1 - X_2 < \frac{\beta}{\sqrt{n_{\min}}} \right) \\
&= \Delta \Phi \left(\frac{\beta - \Delta\sqrt{n_{\min}}}{\sigma \sqrt{n_{\min}}} \right) ,
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{R}(\mathcal{A}_{-\beta}, \theta_2) &= \Delta \mathbb{P}_{\theta_2} (\mathcal{A}_{-\beta} = 1) \\
&= \Delta \mathbb{P}_{\theta_2} \left(X_1 - X_2 \geq -\frac{\beta}{\sqrt{n_{\min}}} \right) \\
&= \Delta \Phi \left(\frac{\beta - \Delta\sqrt{n_{\min}}}{\sigma \sqrt{n_{\min}}} \right) .
\end{aligned}$$

It follows that our upper bound on $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}$ is the same for both instances, i.e., $\mathcal{R}(\mathcal{A}_{2-cc_0}, \theta_1) = \mathcal{R}(\mathcal{A}_{cc_0-2}, \theta_2)$. Next we show that the greedy algorithm $\mathcal{A}_0$ is optimal in terms of minimizing the worse regret between θ_1 and θ_2.

Lemma 10. *Let $\mathcal{A}_0$ be the greedy algorithm where $\mathcal{A}_0(X) = 1$ if $X_1 \geq X_2$ and $\mathcal{A}_0(X) = 2$ otherwise. Then we have*

$$\mathcal{R}(\mathcal{A}_0, \theta_1) = \mathcal{R}(\mathcal{A}_0, \theta_2) = \min_{\mathcal{A}} \max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\} .$$

Proof of Lemma 10. The first step is to show that by applying the Neyman-Pearson Lemma, thresholding algorithms on $X_1 - X_2$ perform the most powerful hypothesis tests between θ_1 and θ_2.

Let f_θ be the probability density function for the observation (X_1, X_2) under instance θ. Then, the likelihood ratio function can be written as

$$\frac{f_{\theta_1}(X_1, X_2)}{f_{\theta_2}(X_1, X_2)} = \frac{e^{-\frac{n_1}{2}(X_1 - \lambda - \eta/n_1)^2 - \frac{n_2}{2}(X_2 - \lambda + \eta/n_2)^2}}{e^{-\frac{n_1}{2}(X_1 - \lambda + \eta/n_1)^2 - \frac{n_2}{2}(X_2 - \lambda - \eta/n_2)^2}} = e^{2\eta(X_1 - X_2)} .$$

Applying the Neyman-Pearson Lemma to our scenario gives the following statement:

Proposition 9 (Neyman-Pearson Lemma)**.** *For any $\gamma > 0$ let $\mathcal{A}^\gamma$ be the algorithm where $\mathcal{A}^\gamma(X) = 1$ if $\frac{f_{\theta_1}(X_1,X_2)}{f_{\theta_2}(X_1,X_2)} \geq \gamma$ and $\mathcal{A}^\gamma(X) = 2$ otherwise. Let $\alpha = \mathbb{P}_{\theta_1}(\mathcal{A}^\gamma(X) = 2)$. Then for any algorithm $\mathcal{A}'$ such that $\mathbb{P}_{\theta_1}(\mathcal{A}'(X) = 2) = \alpha$, we have $\mathbb{P}_{\theta_2}(\mathcal{A}'(X) = 1) \geq \mathbb{P}_{\theta_2}(\mathcal{A}^\gamma(X) = 1)$.*

Note that $\frac{f_{\theta_1}(X_1,X_2)}{f_{\theta_2}(X_1,X_2)} \geq \gamma$ is equivalent to $X_1 - X_2 \geq (2\eta)^{-1} \log \gamma$. Returning to the proof of Lemma 10, consider an arbitrary algorithm $\mathcal{A}'$ and let $\alpha = \mathcal{R}(\mathcal{A}', \theta_1)/\Delta = \mathbb{P}_{\theta_1}(\mathcal{A}'(X) = 2)$. Let γ be the threshold that satisfies $\mathbb{P}_{\theta_1}(\mathcal{A}^\gamma(X) = 2) = \alpha$. This exists because X_1, X_2 follow a continuous distribution. According to Proposition 9 we have $\mathbb{P}_{\theta_2}(\mathcal{A}'(X) = 1) \geq \mathbb{P}_{\theta_2}(\mathcal{A}^\gamma(X) = 1)$. Therefore, we have shown that $\mathcal{R}(\mathcal{A}^\gamma, \theta_1) = \mathcal{R}(\mathcal{A}', \theta_1)$ and $\mathcal{R}(\mathcal{A}^\gamma, \theta_2) \leq \mathcal{R}(\mathcal{A}', \theta_2)$, which means that for any algorithm $\mathcal{A}'$ there exists some γ such that

$$\max\{\mathcal{R}(\mathcal{A}^\gamma, \theta_1), \mathcal{R}(\mathcal{A}^\gamma, \theta_2)\} \leq \max\{\mathcal{R}(\mathcal{A}', \theta_1), \mathcal{R}(\mathcal{A}', \theta_2)\}.$$

It remains to show that $\gamma = 1$ is the minimizer of $\max\{\mathcal{R}(\mathcal{A}^\gamma, \theta_1), \mathcal{R}(\mathcal{A}^\gamma, \theta_2)\}$. This comes from the fact that $\mathcal{R}(\mathcal{A}^\gamma, \theta_1)$ is a monotonically increasing function of γ while $\mathcal{R}(\mathcal{A}^\gamma, \theta_2)$ is a monotonically decreasing function of γ and $\gamma = 1$ makes $\mathcal{R}(\mathcal{A}^\gamma, \theta_1) = \mathcal{R}(\mathcal{A}^\gamma, \theta_2)$, which means that $\gamma = 1$ is the minimizer. □

We now continue with the proof of Theorem 11. Applying Lemma 10 gives

$$\begin{aligned}
\sup_{\theta \in \Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} &\geq \max\left\{ \frac{\mathcal{R}(\mathcal{A}, \theta_1)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)}, \frac{\mathcal{R}(\mathcal{A}, \theta_2)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)} \right\} \\
&\geq \max\left\{ \frac{\mathcal{R}(\mathcal{A}, \theta_1)}{\mathcal{R}(\mathcal{A}_{2-cc_0}, \theta_1)}, \frac{\mathcal{R}(\mathcal{A}, \theta_2)}{\mathcal{R}(\mathcal{A}_{cc_0-2}, \theta_2)} \right\} \\
&= \frac{\max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\}}{\mathcal{R}(\mathcal{A}_{2-cc_0}, \theta_1)} \\
&\geq \frac{\mathcal{R}(\mathcal{A}_0, \theta_1)}{\mathcal{R}(\mathcal{A}_{2-cc_0}, \theta_1)} \\
&= \frac{\Phi\left(-\frac{\Delta}{\sigma}\right)}{\Phi\left(-\frac{cc_0-2}{\sigma\sqrt{n_{\min}}} - \frac{\Delta}{\sigma}\right)}.
\end{aligned} \tag{C.5}$$

Now we apply the fact that for $x > 0$, $\frac{x}{1+x^2}\phi(x) < \Phi(-x) < \frac{1}{x}\phi(x)$ to lower bound equation C.5, where ϕ is the probability density function of the standard normal distribution. Choosing $\beta = cc_0 - 2$, we have

$$\frac{\Phi\left(-\frac{\Delta}{\sigma}\right)}{\Phi\left(-\frac{\beta}{\sigma\sqrt{n_{\min}}} - \frac{\Delta}{\sigma}\right)} \geq \frac{\beta + \Delta\sqrt{n_{\min}}}{\sigma\sqrt{n_{\min}}} \frac{\Delta/\sigma}{1 + (\Delta/\sigma)^2} e^{\frac{1}{2}\left(\frac{\beta^2}{\sigma^2 n_{\min}} + \frac{\beta\Delta}{\sigma^2\sqrt{n_{\min}}}\right)} \geq \frac{\eta^2}{n_{\min} + \eta^2} e^{\frac{\beta^2}{4} + \frac{\beta\eta}{2\sqrt{n_{\min}}}}.$$

Picking $\lambda = 1/2$ and $\eta = n_{\min}/2$ such that $\theta_1, \theta_2 \in [0,1]^2$, we have

$$\sup_{\theta \in \Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} \geq \frac{n_{\min}}{n_{\min} + 4} e^{\frac{\beta^2}{4} + \frac{\beta}{4}\sqrt{n_{\min}}},$$

which concludes the proof. □

C.3 Proof for Section 6.5

For any θ, let μ_1 and n_1 be the reward mean and sample count for the optimal arm. We first prove that $\mathcal{E}^*(\theta)$ is at the order of $1/\sqrt{n_1}$ for any θ. Our proof uses the following result.

Proposition 10. *Let p and q be two Bernoulli distributions with parameter p and p'. If $p \geq 1/2$ we have $\mathrm{KL}(p, p') \geq \frac{1}{2}\log\frac{1}{4q}$.*

Proof of Proposition 10.

$$\begin{aligned}
\mathrm{KL}(p, p') &= p\log\frac{p}{p'} + (1-p)\log\frac{1-p}{1-p'} \\
&= p\log p + (1-p)\log(1-p) - p\log p' - (1-p)\log(1-p') \\
&\geq \log\frac{1}{2} + \frac{1}{2}\log\frac{1}{p'} + (1-p)\log\frac{1}{1-p'} \\
&\geq \frac{1}{2}\log\frac{1}{4} + \frac{1}{2}\log\frac{1}{p'} \\
&= \frac{1}{2}\log\frac{1}{4p'}.
\end{aligned}$$

□

Proposition 11. *There exist universal constants c_0 and c_1 such that, for any $\theta \in \Theta_{\mathbf{n}}$, $c_0/\sqrt{n_1} \leq \mathcal{E}^*(\theta) \leq c_1/\sqrt{n_1}$.*

Proof of Proposition 11. Let c be the constant in the definition of $\mathcal{V}_{\mathbf{n}}^*$. We define $\theta' \in \Theta$ such that the only difference between θ' and θ is the mean for the optimal arm: θ' has $\mu_1' = \mu_1 + \frac{4c}{\sqrt{n_1}}$. Then $\mathrm{KL}(\theta, \theta') = 8c^2$.

We first apply the fact that the empirical mean estimator $\nu = \hat{\mu}_1$ has $\mathbb{E}_\theta\left[|\mu_1 - \nu|\right] \leq \frac{1}{\sqrt{n_1}}$ for any θ, which implies that $\inf_\nu \sup_\theta \mathbb{E}_\theta\left[|\mu_1 - \nu|\right] \leq \frac{1}{\sqrt{n_1}}$. Thus, for any estimator $\nu \in \mathcal{V}_{\mathbf{n}}^*$, $\mathbb{E}_{\theta'}\left[|\mu_1' - \nu|\right] \leq \frac{c}{\sqrt{n_1}}$. By Markov inequality, we have $\mathbb{P}_{\theta'}\left(|\mu_1' - \nu| \geq \frac{2c}{\sqrt{n_1}}\right) \leq \frac{1}{2}$. Thus $\mathbb{P}_{\theta'}\left(\nu \leq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \leq \frac{1}{2}$. Let $A = \mathbb{I}\{\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\}$ be a Bernoulli random variable, $p = P_\theta(A = 1)$ and $p' = P_{\theta'}(A = 1)$. Let $\mathrm{KL}(p', p)$ be the KL divergence between two Bernoulli distribution with parameters p' and p. By Proposition 10, we have

$$\mathrm{KL}(p', p) \geq \frac{1}{4} \log \frac{1}{4p}\,.$$

Then

$$\mathbb{P}_\theta\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) = \mathbb{P}_\theta(A = 1) \geq \frac{1}{4} e^{-2\mathrm{KL}(p', p)} \tag{C.6}$$

Let f_θ and $f_{\theta'}$ be the densities of $\mathbb{P}_\theta$ and $\mathbb{P}_{\theta'}$. By the log sum inequality, we have

$$\begin{aligned}
\mathrm{KL}(p', p) &= \mathbb{P}_{\theta'}(A = 1) \log \frac{\mathbb{P}_{\theta'}(A = 1)}{\mathbb{P}_\theta(A = 1)} + \mathbb{P}_{\theta'}(A = 0) \log \frac{\mathbb{P}_{\theta'}(A = 0)}{\mathbb{P}_\theta(A = 0)} \\
&= \int_A f_{\theta'}(x)dx \log \frac{\int_A f_{\theta'}(x)dx}{\int_A f_\theta(x)dx} + \int_{A^c} f_{\theta'}(x)dx \log \frac{\int_{A^c} f_{\theta'}(x)dx}{\int_{A^c} f_\theta(x)dx} \\
&\leq \int_A f_{\theta'}(x) \log \frac{f_{\theta'}(x)}{f_\theta(x)}dx + \int_{A^c} f_{\theta'}(x) \log \frac{f_{\theta'}(x)}{f_\theta(x)}dx \\
&= \int f_{\theta'}(x) \log \frac{f_{\theta'}(x)}{f_\theta(x)}dx = \mathrm{KL}(\theta', \theta)\,.
\end{aligned}$$

Combining this with Eq. (C.6) shows that

$$\mathbb{P}_\theta\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \geq \frac{1}{4} e^{-2\mathrm{KL}(p', p)} \geq \frac{1}{4} e^{-2\mathrm{KL}(\theta, \theta')} = \frac{1}{4} e^{-16c^2}\,. \tag{C.7}$$

Therefore, we have

$$\mathbb{E}_\theta\left[|\mu_1 - \nu|\right] \geq \frac{2c}{\sqrt{n_1}} \mathbb{P}_\theta\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \geq \frac{c e^{-16c^2}}{2\sqrt{n_1}}\,.$$

Therefore, $\frac{c e^{-16c^2}}{2\sqrt{n_1}}$ is a lower bound on $\mathcal{E}^*(\theta)$ for any θ. An upper bound on $\mathcal{E}^*(\theta)$ can be obtained by the empirical estimator and the definition of $\mathcal{V}_{\mathbf{n}}^*$. In

conclusion, there exist universal constants c_0 and c_1 such that, for any $\theta \in \Theta_{\mathbf{n}}$, $c_0/\sqrt{n_1} \leq \mathcal{E}^*(\theta) \leq c_1/\sqrt{n_1}$.

□

Proof of Proposition 3. Picking $\delta = \frac{1}{\sqrt{|\mathbf{n}|}}$ for the LCB algorithm, according to Corollary 3 gives that there exists a universal constant c (which may contain the term $\log k$) such that $\mathcal{R}(\text{LCB}, \theta) \leq \frac{c\sqrt{\log|\mathbf{n}|}}{\sqrt{n_1}}$. Applying Proposition 11 concludes the proof.

□

Proof of Proposition 4. Consider a sequence of counts $\mathbf{n}_1, \mathbf{n}_2, ...$ with $n_2 = 1$ and $n_1 = 2, 3, ..., +\infty$. Fix $\mu_1 = \mu_2 + 0.1$ and let $\Delta = \mu_1 - \mu_2$. For the UCB algorithm, we have

$$\begin{aligned}
\mathcal{R}(\text{UCB}, \theta) &= \Delta \mathbb{P}_\theta \left(\hat{\mu}_2 + \frac{\beta_\delta}{\sqrt{n_2}} \geq \hat{\mu}_1 + \frac{\beta_\delta}{\sqrt{n_1}} \right) \\
&= 0.1 \mathbb{P}_\theta \left(\hat{\mu}_1 - \hat{\mu}_2 \leq \frac{\beta_\delta}{\sqrt{n_2}} - \frac{\beta_\delta}{\sqrt{n_1}} \right) \\
&\geq 0.1 \mathbb{P}_\theta \left(\hat{\mu}_1 - \hat{\mu}_2 \leq \left(1 - \frac{1}{\sqrt{2}}\right) \beta_\delta \right) \\
&\geq 0.1 \mathbb{P}_\theta \left(\hat{\mu}_1 - \hat{\mu}_2 \leq \left(1 - \frac{1}{\sqrt{2}}\right) \right) \\
&\geq 0.1 \mathbb{P}_\theta \left(\hat{\mu}_1 - \hat{\mu}_2 \leq \Delta \right) \\
&= 0.05
\end{aligned}$$

where we applied the fact that $\beta_\delta \geq 1$ for any $\delta \in (0, 1)$ and the random variable $\hat{\mu}_1 - \hat{\mu}_2$ follows a Gaussian distribution with mean Δ. Applying Proposition 11 gives

$$\limsup_{j \to \infty} \sup_{\theta \in \Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\text{UCB}, \theta)}{\sqrt{\log|\mathbf{n}_j|} \cdot \mathcal{E}^*(\theta)} \geq \limsup_{j \to \infty} \frac{0.05\sqrt{j+1}}{c_1\sqrt{\log(j+2)}} = +\infty$$

For the greedy algorithm, we have

$$\mathcal{R}(\text{greedy}, \theta) = 0.1 \mathbb{P}_\theta \left(\hat{\mu}_1 - \hat{\mu}_2 \leq 0 \right) .$$

The random variable $\hat{\mu}_1 - \hat{\mu}_2$ follows a Gaussian distribution with mean $\Delta > 0$ and variance $\frac{1}{n_1} + \frac{1}{n_2} \geq 1$. Since shrinking the variance of $\hat{\mu}_1 - \hat{\mu}_2$ will lower the

probability $\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq 0\right)$, we have $\mathcal{R}(\text{greedy}, \theta) \geq 0.1\Phi(-0.1)$ where Φ is the CDF for the standard normal distribution. Now using a similar statement as for the UCB algorithm gives the result.

□

Appendix D

Proofs for Chapter 7: The Curse of Passive Data Collection in Batch ReinforcementLearning

D.1 An absolute bound on the state-action probability ratios under the uniform logging policy and the uniform mix of deterministic policies

For a policy π, $t \geq 0$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ let

$$\nu^{\pi}_{\mu,t}(s,a) := \mathbb{P}^{\pi}(S_t = s, A_t = a | S_0 \sim \mu)\,.$$

As noted beforehand, ratios of these marginal probabilities appear in previous upper (and lower) bounds on how well the value of a target policy π_{trg} can be estimated given data from a logging policy π_{log}. To minimize clutter, let $\nu^{\mathrm{trg}}_{\mu,t}$ stand for $\nu^{\pi_{\mathrm{trg}}}_{\mu,t}$ and, similarly, let $\nu^{\mathrm{log}}_{\mu,t}$ stand for $\nu^{\pi_{\mathrm{log}}}_{\mu,t}$. The purpose of this section is to present a short calculation that bounds $\frac{\nu^{\mathrm{trg}}_{\mu,t}(s,a)}{\nu^{\mathrm{log}}_{\mu,t}(s,a)}$, which is the ratio that appears in the previously mentioned bounds. First, we bound this ratio for the uniform logging policy when $\pi_{\mathrm{log}}(a|s) = 1/\mathrm{A}$.

Proposition 12. *When π_{log} is the uniform policy, for any $t \geq 0$, $(s,a) \in \mathcal{S} \times \mathcal{A}$*

and π_{trg} *is any target policy,*

$$\frac{\nu^{\text{trg}}_{\mu,t}(s,a)}{\nu^{\log}_{\mu,t}(s,a)} \le \mathrm{A}^{t+1}\,. \tag{D.1}$$

Furthermore, there exists a MDP and $(s,a) \in \mathcal{S}\times\mathcal{A}$ *such that* $\nu^{\text{trg}}_{\mu,t}(s,a)/\nu^{\log}_{\mu,t}(s,a) \ge \mathrm{A}^{\mathrm{S}}$ *for* $t+1 \ge \mathrm{S}$.

Proof. Fix an arbitrary pair $(s_t,a_t) \in \mathcal{S}\times\mathcal{A}$. Let $s_{0:t}$ denote a sequence $(s_0,\dots,s_t)$ of states and let $a_{0:t}$ denote a sequence $(a_0,\dots,a_t)$ of actions. We have

$$\begin{aligned}
\nu^{\text{trg}}_{\mu,t}(s_t,a_t) &= \sum_{\substack{s_{0:t-1}\\ a_{0:t-1}}} \mu(s_0)\pi_{\text{trg}}(a_0|s_0)p(s_1|s_0,a_0)\dots\pi_{\text{trg}}(a_{t-1}|s_{t-1})p(s_t|s_{t-1},a_{t-1})\pi_{\text{trg}}(a_t|s_t)\\
&\le \sum_{\substack{s_{0:t-1}\\ a_{0:t-1}}} \mathrm{A}^{t+1}\mu(s_0)\pi_{\log}(a_0|s_0)p(s_1|s_0,a_0)\dots\pi_{\log}(a_{t-1}|s_{t-1})p(s_t|s_{t-1},a_{t-1})\pi_{\log}(a_t|s_t)\\
&= \mathrm{A}^{t+1}\nu^{\log}_{\mu,t}(s_t,a_t)\,.
\end{aligned}$$

Dividing both sides by $\nu^{\log}_{\mu,t}(s_t,a_t)$ gives the desired bound. The inequality is tight when there is only one possible path $(s_0,a_0,s_1,a_1,\dots,s_t,a_t)$ to (s_t,a_t) in an MDP and the target policy is the deterministic policy taking the actions in the unique path.

We now show an example to prove the second part of the claim. Consider a MDP with two states $\mathcal{S}=\{s_1,s_2\}$ and two actions $\mathcal{A}=\{a_1,a_2\}$. The MDP always starts at state s_1 at the beginning of an episode, that is $\mu(s_1)=1,\mu(s_2)=0$. At state s_1 under action a_2, the MDP transits to s_2, while it stays at s_1 under a_1. State s_2 is absorbing under any action. Let π_{trg} be a deterministic policy $\pi_{\text{trg}} \in \mathrm{DET}$ such that $\pi_{\text{trg}}(s_1)=a_1$, it can be verified that $\nu^{\pi_{\text{trg}}}_{\mu,t}(s_1,a_1)=1$ for any $t\ge 1$. First consider the uniform logging policy. For $t\ge 1$,

$$\nu^{\log}_{\mu,t}(s_1,a_1) = \nu^{\log}_{\mu,t-1}(s_1,a_1)\mathbb{P}(s_1,a_1|s_1,a_1) = \frac{1}{2}\nu^{\log}_{\mu,t-1}(s_1,a_1) = \dots = \frac{1}{2^{t+1}}\,.$$

Thus for $t+1\ge \mathrm{S}$, $\nu^{\pi_{\text{trg}}}_{\mu,t}(s_1,a_1)/\nu^{\log}_{\mu,t}(s_1,a_1) = 2^{t+1} \ge \mathrm{A}^{\mathrm{S}}$.

Now consider using the mixture of deterministic policies as $\pi_{\log}$. One can see that $\nu^{\log}_{\mu,t}(s_1,a_1)=\frac{1}{2}$ since it will always stay at s_1 as long as a policy always selects a_1 at s_1, and there are 2 out of 4 such deterministic policies.

□

From the proof it is clear that the result continues to hold even if the target policy depends on the full history.

The uniform mix of all deterministic policies selects one among all A^{S} deterministic policies at the beginning where each of them can be chosen with probability $1/\mathrm{A}^{\mathrm{S}}$. Then, it uses the chosen deterministic policy in all time periods. A key difference between the two logging policies we consider is that the uniform policy randomly chooses an action every time the system reaches a state whereas the uniform mix of all deterministic policies randomly selects a deterministic policy at the beginning and then uses the deterministic policy afterwards, thus, a random selection happens only once. Now we prove a counterpart of Proposition 12 for the uniform mix of all deterministic policies.

Proposition 13. *When $\pi_{\log}$ is the uniform mix of deterministic policies, for any $t \geq 0$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ and π_{trg} is any deterministic target policy,*

$$\frac{\nu_{\mu,t}^{\mathrm{trg}}(s,a)}{\nu_{\mu,t}^{\log}(s,a)} \leq \mathrm{A}^{\min(t+1,S)}. \tag{D.2}$$

Proof. Let DET be the set of stationary deterministic policies over $\mathcal{S}$ and $\mathcal{A}$. Fix an arbitrary pair $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. We have

$$\begin{aligned}
\nu_{\mu,t}^{\log}(s_t,a_t) &= \frac{1}{A^S} \sum_{\pi \in \mathrm{DET}} \nu_{\mu,t}^{\pi}(s_t,a_t) \\
&= \frac{1}{A^S} \sum_{\pi \in \mathrm{DET}} \sum_{\substack{s_{0:t-1} \\ a_{0:t-1}}} \mu(s_0)p(s_1|s_0,a_0)\ldots p(s_t|s_{t-1},a_{t-1})\mathbf{1}(a_0 = \pi(s_0),\ldots,a_t = \pi(s_t)) \\
&= \frac{1}{A^S} \sum_{\substack{s_{0:t-1} \\ a_{0:t-1}}} \sum_{\pi \in \mathrm{DET}} \mu(s_0)p(s_1|s_0,a_0)\ldots p(s_t|s_{t-1},a_{t-1})\mathbf{1}(a_0 = \pi(s_0),\ldots,a_t = \pi(s_t)) \\
&= \frac{1}{A^S} \sum_{\substack{s_{0:t-1} \\ a_{0:t-1}}} \mu(s_0)p(s_1|s_0,a_0)\ldots p(s_t|s_{t-1},a_{t-1}) \sum_{\pi \in \mathrm{DET}} \mathbf{1}(a_0 = \pi(s_0),\ldots,a_t = \pi(s_t)) \\
&\geq \frac{1}{A^S} \sum_{\substack{s_{0:t-1} \\ a_{0:t-1}}} \mu(s_0)p(s_1|s_0,a_0)\ldots p(s_t|s_{t-1},a_{t-1})A^{S-t-1} \\
&\geq \frac{1}{A^{t+1}} \nu_{\mu,t}^{\mathrm{trg}}(s_t,a_t).
\end{aligned}$$

Also, for $(s,a) \in \mathcal{S} \times \mathcal{A}$ we have

$$\nu^{\text{trg}}_{\mu,t}(s,a) \leq \sum_{\pi \in \text{DET}} \nu^{\pi}_{\mu,t}(s,a) = \text{A}^{\text{S}} \frac{1}{\text{A}^{\text{S}}} \sum_{\pi \in \text{DET}} \nu^{\pi}_{\mu,t}(s,a) = \text{A}^{\text{S}} \nu^{\log}_{\mu,t}(s,a),$$

and this completes the proof. The inequality is tight when there is only one possible path $(s_0, a_0, s_1, a_1, \ldots, s_t, a_t)$ to (s_t, a_t) in an MDP, the target policy is the deterministic policy taking the actions in the unique path, and the path does not repeat any state. □

D.2 Lower Bound Proofs

Before these proofs, an equivalent form of (ϵ, δ)-soundness will be useful to consider. Recall that $\mathcal{L}$ is (ϵ, δ)-sound on instance (M, G) if

$$\mathbb{P}_{\mathcal{D} \sim G} \left(v^{\mathcal{L}(\mathcal{D})}(\mu) > v^*(\mu) - \varepsilon \right) > 1 - \delta \,,$$

Now, $\mathbb{P}_{\mathcal{D} \sim G} \left(v^{\mathcal{L}(\mathcal{D})}(\mu) > v^*(\mu) - \varepsilon \right) = 1 - \mathbb{P}_{\mathcal{D} \sim G} \left(v^{\mathcal{L}(\mathcal{D})}(\mu) \leq v^*(\mu) - \varepsilon \right)$. Hence, $\mathcal{L}$ is (ϵ, δ)-sound on instance (M, G) if and only if

$$\mathbb{P}_{\mathcal{D} \sim G} \left(v^{\mathcal{L}(\mathcal{D})}(\mu) \leq v^*(\mu) - \varepsilon \right) < \delta \,.$$

Finally, by reordering, this last display is equivalent to

$$\mathbb{P}_{\mathcal{D} \sim G} \left(v^*(\mu) - v^{\mathcal{L}(\mathcal{D})}(\mu) \geq \varepsilon \right) < \delta \,.$$

Thus, $\mathcal{L}$ is *not* (ϵ, δ) sound on (M, G) if

$$\mathbb{P}_{\mathcal{D} \sim G} \left(v^*(\mu) - v^{\mathcal{L}(\mathcal{D})}(\mu) \geq \varepsilon \right) \geq \delta \,. \tag{D.3}$$

We will need some basic concepts, definitions, and results from information theory. For two probability measures, P and Q over a common measurable space, we use $D_{\text{KL}}(P, Q)$ to denote the relative entropy (or Kullback-Leibler divergence) of P with respect to Q, which is infinite when P is not absolutely continuous with respect to Q, and otherwise it is defined as $D(P||Q) = \int \log(\frac{dP}{dQ}) dP$, where dP/dQ is the Radon-Nikodym derivative of P with respect to Q. By abusing notation, we will use $P(X)$ to denote the probability distribution $P(X \in \cdot)$ of a random element X under probability measure P.

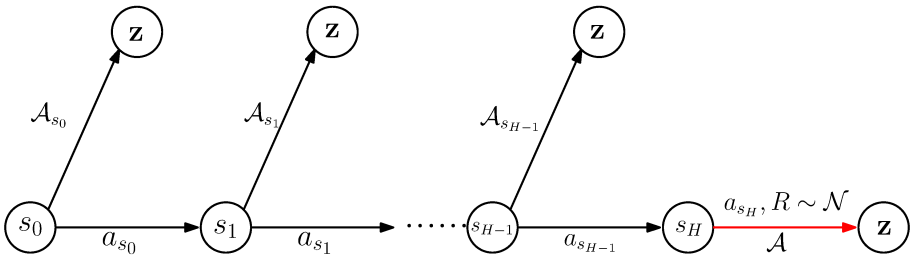


Figure D.1: Illustration of the MDPs used in the proof of Theorem 12. For $\varepsilon > 0$, Let $H = H_{\gamma,2\varepsilon}$. The state space consists of two parts $\mathcal{S} = \{s_0, s_1, \ldots, s_H\} \cup \{z\}$, where s_0 is the initial state, z is a self-absorbing state. For any $s \in \mathcal{S}$, let $a_s = \arg\min_a \pi_{\log}(a|s)$ be the action with minimal chance of being selected by $\pi_{\log}$, and $\mathcal{A}_s = \mathcal{A} \setminus \{a_s\}$. The transitions and rewards are as follows: State z is absorbing under any action. For $i \in \{0, \ldots, H-1\}$, at state s_i under action a_{s_i} the MDP transits to s_{i+1}, while it transits to z under any other actions. From s_H, the next state is also z under any action. The rewards are deterministically zero for any state-action pair except when the state is s_H, and action a_{S_H} is taken, when it is random with either a positive or negative mean.

For jointly distributed random elements X and Y, we let $P(X|Y)$ denote the conditional distribution of X given Y, $P(X \in \cdot|Y)$, which is Y-measurable. With this, the *chain rule* for relative entropy states that

$$D_{\mathrm{KL}}(P(X,Y), Q(X,Y)) = \int D_{\mathrm{KL}}(P(X|Y), Q(X|Y))dP + D_{\mathrm{KL}}(P(Y), Q(Y))\,,$$

which, of course, extends to any number of jointly distributed random elements.

We will also need the following result, which is given, for example, as Theorem 14.2 in the book of Lattimore & Szepesvári (2020).

Lemma 11 (Bretagnolle–Huber inequality)**.** *Let P and Q be probability measures on the same measurable space $(\Omega, \mathcal{F})$, and let $A \in \mathcal{F}$ be an arbitrary event. Then,*

$$P(A) + Q(A^c) \geq \frac{1}{2}\exp\left(-D_{\mathrm{KL}}(P,Q)\right)\,, \tag{D.4}$$

where $A^c = \Omega \setminus A$ is the complement of A.

Proof of Theorem 12. We first consider the case where $\mathrm{S} \geq H := H_{\gamma,2\varepsilon} + 2$.

We let $\{s_0, s_1, \ldots, s_H, z\}$ be arbitrary, distinct states and choose μ to be the distribution that is concentrated at s_0. Let $\pi_{\log}$ be the distribution used to construct the batch (which could depend on μ). We now define two MDPs, $M_1, M_{-1} \in \mathcal{M}(S, A)$ (cf. Fig. D.1). For any $s \in \mathcal{S}$, let $a_s = \arg\min_a \pi_{\log}(a|s)$ be the action with the minimal chance of being selected by $\pi_{\log}$. Note that $\pi_{\log}(a_s|s) \leq 1/\mathrm{A}$.

The transition structure in the two MDPs are identical, the transitions are deterministic and almost all the rewards are also the same with the exception of one transition. The details are as follows. State z is absorbing: For any action taken at z, the next state is z. For $i < H$, s_i is followed by s_{i+1} when a_{s_i} is taken, while the next state is z when any other action is taken at this state. At s_H under any action, the next state is z. The rewards are deterministically zero for any state-action pair except when the state is s_H and action a_{s_H} is taken at this state. In this case, the reward R is drawn from a Gaussian with mean $\alpha \in \{-1, +1\}$ in MDP M_α.

We will use v^π_α, v^*_α and $\nu_{\mu,\alpha}$ to denote the value function of a policy π on M_α, the optimal value function on M_α, and the discounted occupancy measure on M_α with μ as the initial state distribution, respectively. Note that $v^*_1(s_0) = \gamma^H \geq \gamma^{\frac{\ln(1/(2\epsilon))}{\ln(1/\gamma)}} = 2\epsilon$, where the first inequality is because $\gamma \leq 1$ and $H \leq \frac{\ln(1/(2\epsilon))}{\ln(1/\gamma)}$ by its definition. Note also that $v^*_{-1}(s_0) = 0$.

Fix $\pi_{\log}$ and the episode lengths $\mathbf{h} = (h_0, \ldots, h_{m-1})$. We show that if the number of episodes m is too small, then no algorithm will be sound both on M_1 and M_{-1}.

For this fix an arbitrary BPO algorithm $\mathcal{L}$. Let the data collected by following the logging policy $\pi_{\log}$ be $\mathcal{D} = (S_i, A_i, R_i, S'_i)_{i=0}^{n-1}$. Let π be the output of $\mathcal{L}$. Let $\mathbb{P}_\alpha$ be the distribution over $(\mathcal{D}, \pi)$ induced by using $\pi_{\log}$ on M_α with episode lengths $\mathbf{h}$ and μ and then running $\mathcal{L}$ on $\mathcal{D}$ to produce π. Note that both $\mathbb{P}_1$ and $\mathbb{P}_{-1}$ share the same measure space. Let $\mathbb{E}_\alpha$ be the expectation operator for $\mathbb{P}_\alpha$.

Define the event $E = \{v^\pi_1(s_0) < \varepsilon\}$. Let E^c be the complement of E. Let $a \vee b$ denote the maximum of a and b. We first prove the following claim:

Claim: If

$$\mathbb{P}_1(E) \vee \mathbb{P}_{-1}(E^c) \geq \delta \tag{D.5}$$

then $\mathcal{L}$ is not (ϵ, δ)-sound.

Proof of the claim. By Eq. (D.3), $\mathcal{L}$ is not (ϵ, δ)-sound if

$$\mathbb{P}_1(v_1^*(s_0) - v_1^\pi(s_0) \geq \epsilon) \vee \mathbb{P}_{-1}(v_{-1}^*(s_0) - v_{-1}^\pi(s_0) \geq \epsilon) \geq \delta\,.$$

By $v_1^*(s_0) \geq 2\varepsilon$, we have

$$\mathbb{P}_1(v_1^*(s_0) - v_1^\pi(s_0) \geq \epsilon) \geq \mathbb{P}_1(v_1^\pi(s_0) \leq \epsilon) \geq \mathbb{P}_1(v_1^\pi(s_0) < \epsilon) = \mathbb{P}_1(E)\,.$$

Similarly, by $v_{-1}^*(s_0) = 0$, we have

$$\mathbb{P}_{-1}(v_{-1}^*(s_0) - v_{-1}^\pi(s_0) \geq \epsilon) = \mathbb{P}_{-1}(v_{-1}^\pi(s_0) \leq -\epsilon) \geq \mathbb{P}_{-1}(v_1^\pi(s_0) \geq \epsilon) = \mathbb{P}_{-1}(E^c)\,,$$

where the inequality follows because if $v_1^\pi(s_0) \geq \epsilon$ holds then since $v_1^\pi(s_0) = \langle \nu_1^\pi, r_1^\pi \rangle = \nu_1^\pi(s_H, a_{s_H}) r_1^\pi(s_H, a_{s_H}) = \nu_1^\pi(s_H, a_{s_H})$ and since the transitions in M_1 and M_{-1} are same, we have $\nu_{-1}^\pi(s_H, a_{s_H}) = \nu_1^\pi(s_H, a_{s_H}) \geq \varepsilon$ and therefore $v_{-1}^\pi(s_0) = -\nu_{-1}^\pi(s_H, a_{s_H}) \leq -\varepsilon$.

Putting things together, we get that

$$\mathbb{P}_1(v_1^*(s_0) - v_1^\pi(s_0) \geq \epsilon) \vee \mathbb{P}_{-1}(v_{-1}^*(s_0) - v_{-1}^\pi(s_0) \geq \epsilon) \geq \mathbb{P}_1(E) \vee \mathbb{P}_{-1}(E^c) \geq \delta,$$

where the last inequality follows by our assumption. □

It remains to prove that Eq. (D.5) holds. For this, note that by the Bretagnolle-Huber inequality (Lemma 11) we have,

$$\mathbb{P}_1(E) \vee \mathbb{P}_{-1}(E^c) \geq \frac{\mathbb{P}_1(E) + \mathbb{P}_{-1}(E^c)}{2} \geq \frac{1}{4} \exp(-D_{\mathrm{KL}}(\mathbb{P}_1, \mathbb{P}_{-1}))\,. \tag{D.6}$$

It remains to upper bound $D_{\mathrm{KL}}(\mathbb{P}_1, \mathbb{P}_{-1})$. Let $U_0 = S_0$, $U_1 = A_0$, $U_2 = R_0$, $U_3 = S_0'$, $U_4 = S_1$, $\ldots$, $U_{4(n-1)} = S_{n-1}'$. Further, for $0 \leq j \leq 4n-1$ let $U_{0:j} = (U_0, \ldots, U_j)$ and let $U_{0:-1}$ stand for a "dummy" (trivial) random element. By

the chain rule for relative entropy,[1]

$$D_{\mathrm{KL}}(\mathbb{P}_1, \mathbb{P}_{-1}) = \mathbb{E}_1[D_{\mathrm{KL}}(\mathbb{P}_1(\pi|U_{0:4(n-1)}), \mathbb{P}_{-1}(\pi|U_{0:4(n-1)}))] + \sum_{j=0}^{4(n-1)} \mathbb{E}_1[D_{\mathrm{KL}}(\mathbb{P}_1(U_j|U_{0:j-1}), \mathbb{P}_{-1}(U_j|U_{0:j-1}))] .$$

Note that, $\mathbb{P}_1$-almost surely, $\mathbb{P}_1(\pi|U_{0:4(n-1)}) = \mathbb{P}_{-1}(\pi|U_{0:4(n-1)})$ since, by definition, $\mathcal{L}$ assigns a fixed probability distribution over the policies to any possible dataset. For $0 \le j \le 4(n-1)$, let $D_j = D_{\mathrm{KL}}(\mathbb{P}_1(U_j|U_{0:j-1}), \mathbb{P}_{-1}(U_j|U_{0:j-1}))$. Since the only difference between M_1 and M_{-1} is in the reward distribution corresponding to taking action a_{s_H} in state s_H, unless $j = 4i+2$ for some $i \in [n]$ and $S_i = s_H, A_i = a_{s_H}$, we have $D_j = 0$ $\mathbb{P}_1$-almost surely. Further, when $j = 4i+2$, $\mathbb{P}_1$-almost surely we have $D_j = \mathbb{I}\{S_i = s_H, A_i = a_{s_H}\}(1-(-1))^2/2 = 2\mathbb{I}\{S_i = s_H, A_i = a_{s_H}\}$ by the formula for the relative entropy between $\mathcal{N}(1,1)$ and $\mathcal{N}(-1,1)$. Therefore,

$$D_{\mathrm{KL}}(\mathbb{P}_1, \mathbb{P}_{-1}) = 2\mathbb{E}_1\left[\sum_{i=0}^{n-1} \mathbb{I}\{S_i = s_H, A_i = a_{s_H}\}\right] \le 2m\mathbb{P}_1(S_H = s_H, A_H = a_{s_H}) \le \frac{2m}{\mathrm{A}^{H+1}} ,$$

where the first inequality follows from that, by the construction of M_1, s_H can be visited only in the Hth step of *any* episode, the data in distinct episodes are identically distributed, and there are at most m episodes. The second inequality follows because

$$\begin{aligned}
&\mathbb{P}_1(S_H = s_H, A_H = a_{s_H}) = \mathbb{P}_1(A_H = a_{s_H}|S_H = s_H)\mathbb{P}_1(S_H = s_H) \\
&\quad = \mathbb{P}_1(A_H = a_{s_H}|S_H = s_H)\mathbb{P}_1(A_{H-1} = a_{s_{H-1}}, S_{H-1} = s_{H-1}) \\
&\quad = \mathbb{P}_1(A_H = a_{s_H}|S_H = s_H)\mathbb{P}_1(A_{H-1} = a_{s_{H-1}}|S_{H-1} = s_{H-1}) \dots \mathbb{P}_1(A_0 = a_{s_0}|S_0 = s_0) \\
&\quad = \pi_{\log}(a_{s_0}|s_0) \dots \pi_{\log}(a_{s_H}|s_H) \le \frac{1}{\mathrm{A}^{H+1}} ,
\end{aligned}$$

where the last inequality follows by the choice of a_{s_i}, $i \in [H+1]$. Plugging the upper bound on $D_{\mathrm{KL}}(\mathbb{P}_1, \mathbb{P}_{-1})$ into Eq. (D.6), we get that

$$\mathbb{P}_1(E) \vee \mathbb{P}_{-1}(E^c) \ge \frac{1}{4}\exp(-2m\mathrm{A}^{-(H+1)})$$

[1]Here, we use a notation common in information theory, which uses $P(X)$ ($P(X|Y)$) to denote the distribution of X induced by P (the conditional distribution of X, given Y, induced by P, respectively).

which is larger than δ if $m \leq (\mathrm{A}^{H+1} \ln \frac{1}{4\delta})/2$. The result then follows by our previous claim.

To prove the result for $\mathrm{S} < H_{\gamma,2\varepsilon} + 2$, we use the same construction as described above with $H_{\gamma,2\varepsilon'} = \mathrm{S} - 2 < H_{\gamma,2\varepsilon}$ for some $\varepsilon' \geq \varepsilon$. Then any learning algorithm $\mathcal{L}$ needs at least $(\mathrm{A}^{H_{\gamma,2\varepsilon'}+1} \ln \frac{1}{4\delta})/2$ episodes to be (ε', δ)-sound. To be (ε, δ)-sound it needs at least the same amount of data. This finishes the proof. □

Proof of Corollary 7. The result directly follows from the lower bound construction in Theorem 12. □

Proof of Corollary 8. We first consider the case $\mathrm{S} \geq H + 2$. Recall that $H = H_{\gamma,2\varepsilon}$. Define $\mathcal{S}_{\min}$ to be the set of $H+1$ states that have the smallest $\pi_{\log,\min}(s)$ values, where we let $\pi_{\log,\min}(s) = \min_{a \in \mathcal{A}} \pi_{\log}(a|s)$. Construct the same MDPs as in the proof of Theorem 12 using the states in $\mathcal{S}_{\min}$ to form the chain. Then, the same proof holds with A^{H+1} replaced by

$$\prod_{s \in \mathcal{S}_{\min}} \frac{1}{\pi_{\log,\min}(s)} \,. \tag{D.7}$$

Since $\pi_{\log}$ is not uniform, the above value is strictly greater than A^{H+1}.

For the case $S < H + 2$, let $\mathcal{S}_{\min}$ to be the set of $S - 1$ states that have the smallest $\pi_{\log,\min}(s)$ values. Construct the same MDPs as above. Then the same arguments hold as in the last part of the proof of Theorem 12 except that $\mathrm{A}^{\mathrm{S}-1}$ is replaced by equation D.7, which is strictly greater than $\mathrm{A}^{\mathrm{S}-1}$. This concludes the proof of the corollary. □

Proof of Theorem 13. This proof is similar to the proof of Theorem 12. We first consider the case where $\mathrm{S} \geq H+1$. We construct the same MDPs as in the proof of Theorem 12 except that the chain consists of H states, that is, ending at s_{H-1} and the hidden reward R is at $(s_{H-1}, a_{s_{H-1}})$. The logging policy $\pi_{\log}$ collects m trajectories with length H as the dataset $\mathcal{D} = (S_i, A_i, R_i, S_i')_{i=0}^{mH-1}$, where $S_0 = S_H = \cdots = S_{(m-1)H} = s_0$. Now we consider two MDPs $M_\alpha \in \mathcal{M}, \alpha \in \{2\varepsilon, -2\varepsilon\}$, where the reward $R \sim \mathcal{N}(\alpha, 1)$ on M_α.

We use the same notation as in the proof of Theorem 12. Define the event $E = \{v^{\pi}_{2\varepsilon}(s_0) < \varepsilon\}$. Then, by following the same arguments we can show that $\mathcal{L}$ is not (ε, δ)-sound on $M_{2\varepsilon}$ if $\mathbb{P}_{2\varepsilon}(E) \geq \delta$ and that $\mathcal{L}$ is not (ε, δ)-sound on $M_{-2\varepsilon}$ if $\mathbb{P}_{-2\varepsilon}(E^c) \geq \delta$.

By the Bretagnolle–Huber inequality, we have

$$\max\{\mathbb{P}_{2\varepsilon}(E), \mathbb{P}_{-2\varepsilon}(E^c)\} \geq \frac{\mathbb{P}_{2\varepsilon}(E) + \mathbb{P}_{-2\varepsilon}(E^c)}{2} \geq \frac{1}{4}\exp(-D_{\mathrm{KL}}(\mathbb{P}_{2\varepsilon}, \mathbb{P}_{-2\varepsilon}))\,.$$

Similarly as in the proof of Theorem 12, we obtain

$$\begin{aligned} D_{\mathrm{KL}}(\mathbb{P}_{2\varepsilon}, \mathbb{P}_{-2\varepsilon}) &= 8\varepsilon^2 \mathbb{E}_{2\varepsilon}\left[\sum_{i=0}^{mH-1} \mathbb{I}\{S_i = s_{H-1}, A_i = a_{s_{H-1}}\}\right] \\ &= 8m\varepsilon^2 \mathbb{E}_{2\varepsilon}\left[\sum_{i=0}^{H-1} \mathbb{I}\{S_i = s_{H-1}, A_i = a_{s_{H-1}}\}\right] \\ &= 8m\varepsilon^2 \mathbb{P}_{2\varepsilon}(S_{H-1} = s_{H-1}, A_{H-1} = a_{s_{H-1}}) \leq \frac{8m\varepsilon^2}{\mathrm{A}^H}\,, \end{aligned}$$

where the second equality is obtained by the fact that the episodes are independently sampled. Combining the above together we have that if $m \leq \frac{\mathrm{A}^H \ln \frac{1}{4\delta}}{8\varepsilon^2}$, $\max\{\mathbb{P}_{2\varepsilon}(E), \mathbb{P}_{-2\varepsilon}(E^c)\} \geq \delta$, which means that $\mathcal{L}$ is not (ε, δ)-sound on either $M_{2\varepsilon}$ or $M_{-2\varepsilon}$.

To prove the result for $\mathrm{S} \leq H$, we use the same construction as described above with $H' = \mathrm{S} - 1 < H$. Then any learning algorithm $\mathcal{L}$ needs at least $\frac{\mathrm{A}^{H'} \ln \frac{1}{4\delta}}{8\varepsilon^2}$ trajectories to be (ε, δ)-sound. This finishes the proof. □

Proof of Theorem 14. We use MDPs similar to those in the proof of Theorem 12 but with some key differences. Let the state space consist of three parts $\mathcal{S} = \{s_0, s_1, \ldots, s_{H-1}\} \cup \{y\} \cup \{z\}$, where $H = \mathrm{S} - 2$. Consider μ concentrated on s_0. For any $s \in \mathcal{S}$, let $a_s = \arg\min_a \pi_{\log}(a|s)$. At s_i for $i \in \{0, \ldots, H-2\}$, it transits to s_{i+1} by taking a_{s_i} and transits to z by taking any other actions, where z is an absorbing state. At s_{H-1}, by taking any action it transits to y with probability $p > 0$ and goes back to s_0 with probability $1 - p$. y is also an absorbing state, but there is a reward R for any action in y. The rewards are deterministically zero for any other state-action pairs.

Now consider two such MDPs $M_\alpha \in \mathcal{M}, \alpha \in \{2\varepsilon, -2\varepsilon\}$, where the reward $R \sim \mathcal{N}(\alpha, 1)$ on M_α. We keep using the same notation v_α and $\nu_{\mu,\alpha}$, the latter

of which denotes the occupancy measure on M_α with μ as the initial state distribution. Also, we use the rest of notation in the proof of Theorem 12. Recall that π is the output policy of a learning algorithm $\mathcal{L}$.

Define the event $E = \{v^\pi_{2\varepsilon}(s_0) < \varepsilon\}$.

Claim: If

$$\mathbb{P}_{2\varepsilon}(E) \vee \mathbb{P}_{-2\varepsilon}(E^c) \geq \delta \tag{D.8}$$

then $\mathcal{L}$ is not (ϵ, δ)-sound.

Proof of the claim. By Eq. (D.3), $\mathcal{L}$ is not (ϵ, δ)-sound if

$$\mathbb{P}_{2\varepsilon}(v^*_{2\varepsilon}(s_0) - v^\pi_{2\varepsilon}(s_0) \geq \epsilon) \vee \mathbb{P}_{-2\varepsilon}(v^*_{-2\varepsilon}(s_0) - v^\pi_{-2\varepsilon}(s_0) \geq \epsilon) \geq \delta\,.$$

By the definition of $M_{2\varepsilon}$, the optimal policy is choosing a_{s_i} at s_i for $i \in \{0, \dots, H-2\}$. We have $v^*_{2\varepsilon}(s_0) = 2\varepsilon$, because p is positive, and thus, the optimal policy reaches y in finite steps with probability one. Thus, we have

$$\mathbb{P}_{2\varepsilon}(v^*_{2\varepsilon}(s_0) - v^\pi_{2\varepsilon}(s_0) \geq \epsilon) = \mathbb{P}_{2\varepsilon}(2\varepsilon - v^\pi_{2\varepsilon}(s_0) \geq \epsilon) \geq \mathbb{P}_{2\varepsilon}(v^\pi_{2\varepsilon}(s_0) < \epsilon) = \mathbb{P}_{2\varepsilon}(E)\,.$$

Similarly, by $v^*_{-2\varepsilon}(s_0) = 0$, we have

$$\mathbb{P}_{-2\varepsilon}(v^*_{-2\varepsilon}(s_0) - v^\pi_{-2\varepsilon}(s_0) \geq \epsilon) = \mathbb{P}_{-2\varepsilon}(v^\pi_{-2\varepsilon}(s_0) \leq -\epsilon) \geq \mathbb{P}_{-2\varepsilon}(v^\pi_{2\varepsilon}(s_0) \geq \epsilon) = \mathbb{P}_{-2\varepsilon}(E^c)\,,$$

where the inequality follows because if $v^\pi_{2\varepsilon}(s_0) \geq \epsilon$ holds then since $v^\pi_{2\varepsilon}(s_0) = \langle \nu^\pi_{2\varepsilon}, r^\pi_{2\varepsilon} \rangle = 2\varepsilon \nu^\pi_{2\varepsilon}(y)$ and since the transitions in $M_{2\varepsilon}$ and $M_{-2\varepsilon}$ are same, we have $\nu^\pi_{-2\varepsilon}(y) = \nu^\pi_{2\varepsilon}(y) \geq 1/2$ and therefore $v^\pi_{-2\varepsilon}(s_0) = -2\varepsilon \nu^\pi_{-2\varepsilon}(y) \leq -\varepsilon$.

Putting things together, we get that

$$\mathbb{P}_{2\varepsilon}(v^*_{2\varepsilon}(s_0) - v^\pi_{2\varepsilon}(s_0) \geq \epsilon) \vee \mathbb{P}_{-2\varepsilon}(v^*_{-2\varepsilon}(s_0) - v^\pi_{-2\varepsilon}(s_0) \geq \epsilon) \geq \mathbb{P}_{2\varepsilon}(E) \vee \mathbb{P}_{-2\varepsilon}(E^c) \geq \delta,$$

where the last inequality follows by our assumption. □

Following the same arguments in the proof of Theorem 12, we have

$$\begin{aligned} D_{\mathrm{KL}}(\mathbb{P}_{2\varepsilon}, \mathbb{P}_{-2\varepsilon}) &= 8\varepsilon^2 \mathbb{E}_{2\varepsilon}\left[\sum_{i=0}^{n-1} \mathbb{I}\{S_i = y\}\right] = 8\varepsilon^2 \sum_{i=0}^{n-1} \mathbb{P}_{2\varepsilon}\{S_i = y\} \\ &= 8\varepsilon^2 p \sum_{i=1}^{n-1} \mathbb{P}_{2\varepsilon}\{S_{i-1} = s_{H-1}\} \leq \frac{8\varepsilon^2 np}{\mathrm{A}^{H-1}}\,. \end{aligned}$$

Combining the above together and using the Bretagnolle–Huber inequality (Lemma 11) as we did in the proof of Theorem 12, we have that if $n \leq \frac{\mathrm{A}^{H-1} \ln \frac{1}{4\delta}}{8\varepsilon^2 p}$, then $\mathcal{L}$ is not (ε, δ)-sound on either $M_{2\varepsilon}$ or $M_{-2\varepsilon}$. We obtain the result by sending p to zero from the right hand side. □

For the proof of Theorem 15, we will need some results on the relative entropy between Bernoulli distributions, which we present now. Let $\mathrm{Ber}(p)$ denote the Bernoulli distribution with parameter $p \in [0,1]$. As it is well known (and not hard to see from the definition),

$$D(\mathrm{Ber}(p), \mathrm{Ber}(q)) = d(p,q)$$

where $d(p,q)$ is the so-called *binary relative entropy function*, which is defined as

$$d(p,q) = p \log(p/q) + (1-p) \log((1-p)/(1-q)) \,.$$

Proposition 14. *For $p, q \in (0,1)$, defining p^* to be p or q depending on which is further away from $1/2$,*

$$d(p,q) \leq \frac{(p-q)^2}{2p^*(1-p^*)} \,. \tag{D.9}$$

Proof. Let R be the unnormalized negentropy over $[0,\infty)^2$. Then, by Theorem 26.12 of the book of Lattimore & Szepesvári (2020), for any $x, y \in (0,\infty)^2$,

$$D_R(x,y) = \frac{1}{2}\|x-y\|^2_{\nabla R(z)}$$

for some z on the line segment connecting x to y. We have $R(z) = z_1 \log(z_1) + z_2 \log(z_2) - z_1 - z_2$. Hence, $\nabla R(z) = [\log(z_1), \log(z_2)]^\top$ and $\nabla R(z) = \mathrm{diag}(1/z_1, 1/z_2)$, both defined for $z \in (0,\infty)^2$. Thus,

$$D_R(x,y) = \frac{(x_1-y_1)^2}{2z_1} + \frac{(x_2-y_2)^2}{2z_2} \,.$$

Now choosing $x = (p, 1-p)$, $y = (q, 1-q)$, we see that $x, y \in (0,\infty)^2$ if $p, q \in (0,1)$. In this case, with some $\alpha \in [0,1]$, $z = \alpha x + (1-\alpha) y = (\alpha p + (1-\alpha) q, \alpha(1-p) + (1-\alpha)(1-q))^\top = (\alpha p + (1-\alpha) q, 1 - (\alpha p + (1-\alpha) q))^\top$. Hence, $z_2 = 1 - z_1$ and

$$d(p,q) = \frac{(p-q)^2}{2z_1} + \frac{(p-q)^2}{2(1-z_1)} = \frac{(p-q)^2}{2z_1(1-z_1)} \,.$$

Now, $z_1(1-z_1) \geq p^*(1-p^*)$ (the function $z \mapsto z(1-z)$ has a maximum at $z = 1/2$ and is decreasing on "either side" of the line $z = 1/2$). Putting things together, we get

$$d(p,q) = \frac{(p-q)^2}{2z_1(1-z_1)} \leq \frac{(p-q)^2}{2p^*(1-p^*)}\,.$$

□

Now we re-state Theorem 15 and prove it.

Theorem 23 (Restatement of Theorem 15)**.** *Fix any $\gamma_0 > 0$. Then, there exist some constants $c_0, c_1 > 0$ such that for any $\gamma \in [\gamma_0, 1)$, any positive integers S and A, $\delta \in (0,1)$, and $0 < \varepsilon \leq c_0/(1-\gamma)$, the sample size n needed by any (ε,δ)-sound algorithm that produces as output a memoryless policy and works with SA-sampling for MDPs with* S *states and* A *actions under the γ-discounted expected reward criterion must be so that is at least $c_1 \frac{\mathrm{SA}\ln(1/(4\delta))}{\varepsilon^2(1-\gamma)^3}$.*

Proof of Theorem 23. The proof also uses Le Cam's method, just like Theorem 12. At the heart of the proof is a gadget with a self-looping state which was introduced by Azar et al. (2013) to give a lower bound on the sample complexity of estimating the optimal value function in the simulation setting where the estimate's error is measured with its worst-case error.

The idea of the proof is illustrated by Fig. D.2. Fix an initial state distribution μ concentrated on an arbitrary state $s_0 \in \mathcal{S}$. Let $\mu_{\log}$ be the logging distribution chosen based on μ and let (s', a') be any state-action pair that has the minimum sampling probability under $\mu_{\log}$. Note that $\mu_{\log}(s', a') \leq 1/(\mathrm{SA})$. Assume that $s' \neq s_0$. As we shall see by the end of the proof, there is no loss of generality in making this assumption (when $s' = s_0$, the lower bound would be larger).

We construct two MDPs as follows. The reward structures of the two MDPs are completely identical and the transition structures are also identical except for when action a' is taken at state s'. In particular, in both MDPs, the rewards are identically zero except at state s', where for any action the reward incurred is one. The transition structures are as follows: Let $p_0 < \bar{p} < p_1$ be in

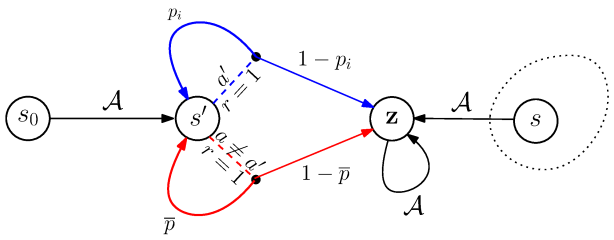


Figure D.2: Illustration of the MDPs used in the proof of Theorem 15. The initial distribution μ concentrates on state $\{s_0\}$. The pair (s', a') is the one where $\mu_{\log}$ takes on the smallest value (which is below $1/(SA)$) and without loss of generality $s' \neq s_0$ and taking any action in s_0 makes the next state s'. We have $p_0 < \bar{p} < p_1$, all in the $[1/2, 1)$ interval. In MDP M_i with $i \in \{0, 1\}$, the probability of transitioning under action a' from s' to z, an absorbing state, is p_i, while with probability $1 - p_i$, the next state is s'. All other actions use probability $\bar{p}$ at this state. All other states under any action lead to z. The rewards are deterministically zero except at state s', when all actions yield a reward of one, regardless of the identity of the next state.

$(0, 1)$, to be determined later. At s_0, by taking any action the system transits to s' deterministically. At s', for any $a \in \mathcal{A} \setminus \{a'\}$, taking action a leads to s' as the next state with probability $\bar{p}$ and to z with probability $1-\bar{p}$, where z is an absorbing state. The transition under a' at s' is similar, except that in M_i ($i \in \{0, 1\}$), the probability that the next state is s' is p_i (and the probability that the next state is z is $1 - p_i$). At any state $s \in \mathcal{S} \setminus \{s_0, s'\}$, taking any action moves the system to z deterministically. The optimal policy at s' in M_1 is to pull the action a', while in M_0 all the other actions are optimal. It is easy to see that in any of these MDPs, for $a \in \mathcal{A}$,

$$q^*(s', a) = \frac{1}{1 - \gamma P(s'|s', a)} \ .$$

Now we select $p_0, \bar{p}$, and p_1. Let b be a constant such that $1 < b < \frac{1-\gamma/2}{1-\gamma}$. We will choose a specific value for b at the end of the proof. The values of $p_0, \tilde{p}$ and p_1 will depend on b. In particular, we choose $\tilde{p} \in (1/2, 1)$ so that

$$b = \frac{1 - \gamma\tilde{p}}{1 - \gamma},$$

while we set $p_0 = \tilde{p}$. Then $p_0 = (1 - b + \gamma b)/\gamma$. Note that $p_0 > 1/2$ by its choice. Let $f(p) = \frac{\gamma}{1-\gamma p}$. Note that for any deterministic policy π, $v^\pi(\mu) =$

$f(P(s'|s', \pi(s')))$ and also $v^*(\mu) = f(\bar{p})$ in MDP M_0 and $v^*(\mu) = f(p_1)$ in MDP M_1.

By Taylor's theorem, for some $p \in [p_0, p_1]$, we have

$$f(p_1) = f(p_0) + f'(p)(p_1 - p_0) \geq f(p_0) + f'(p_0)(p_1 - p_0)\,,$$

where the inequality follows by $p_1 > p_0$ and the fact that f' is increasing. Thus, if $p_1 - p_0 \geq 4\varepsilon/f'(p_0)$, we have $f(p_1) \geq f(p_0) + 4\varepsilon$. Because of the choice of p_0,

$$f'(p_0) = \frac{\gamma^2}{(1-\gamma p_0)^2} = \frac{\gamma^2}{(1-\gamma)^2 b^2}\,.$$

We let $p_1 = p_0 + 4\varepsilon/f'(p_0)$. Then, we have $p_1 < 1$ given that $\varepsilon \leq c_0/(1-\gamma) := \frac{\gamma(b-1)}{8(1-\gamma)b^2}$, because

$$\begin{aligned} p_1 - 1 &= p_0 + 4\varepsilon/f'(p_0) - 1 = \frac{1-b+\gamma b}{\gamma} + \frac{4(1-\gamma)^2\varepsilon b^2}{\gamma^2} - 1 \\ &= \frac{4(1-\gamma)^2\varepsilon b^2 + \gamma(\gamma-1)(b-1)}{\gamma^2} \leq \frac{(\gamma-1)(b-1)}{2\gamma} < 0\,, \end{aligned} \tag{D.10}$$

where the first inequality is due to the choice of ε. Lastly, we set $\bar{p}$ so that $f(\bar{p}) = [f(p_0) + f(p_1)]/2$ (such $\bar{p}$ uniquely exists because f is increasing and continuous). Note that $f(p_1) - f(\bar{p}) \geq 2\varepsilon$ and $f(\bar{p}) - f(p_0) \geq 2\varepsilon$.

Let $\mathbb{P}_0$ and $\mathbb{P}_1$ be the joint probability distribution on the data and the output policy of any given learning algorithm $\mathcal{L}$, induced by μ, $\mu_{\log}$, $\mathcal{L}$, and the two MDPs M_0 and M_1, respectively. For any algorithm $\mathcal{L}$, let $E = \{\pi(a'|s') \geq 1/2\}$, where π is the output of $\mathcal{L}$.

If E is true, in M_0,

$$v^\pi(\mu) = \pi(a'|s')f(p_0) + (1 - \pi(a'|s'))f(\bar{p}) \leq \frac{f(p_0) + f(\bar{p})}{2} \leq \frac{(f(\bar{p}) - 2\varepsilon) + f(\bar{p})}{2} = f(\bar{p}) - \varepsilon\,.$$

Thus, $\mathcal{L}$ is not (ε, δ)-sound for M_0 if $\mathbb{P}_0(E) \geq \delta$. If E^c holds, in M_1,

$$v^\pi(\mu) = \pi(a'|s')f(p_1) + (1 - \pi(a'|s'))f(\bar{p}) \leq \frac{f(p_1) + f(\bar{p})}{2} \leq \frac{f(p_1) + (f(p_1) - 2\varepsilon)}{2} = f(p_1) - \varepsilon\,.$$

Therefore, if $\mathbb{P}_1(E^c) \geq \delta$, then $\mathcal{L}$ is not (ε, δ)-sound for M_1.

By the Bretagnolle-Huber inequality (Lemma 11) we have,

$$\max\{\mathbb{P}_0(E), \mathbb{P}_1(E^c)\} \geq \frac{\mathbb{P}_0(E) + \mathbb{P}_1(E^c)}{2} \geq \frac{1}{4}\exp\left(-D_{\mathrm{KL}}(\mathbb{P}_0||\mathbb{P}_1)\right)\,.$$

Recall that n is the number of samples. Since M_0 and M_1 differ only in the state transition from (s', a'), by the chain rule of relative entropy, with a reasoning similar to that used in the proof of Theorem 12, we derive

$$\begin{aligned} D_{\mathrm{KL}}(\mathbb{P}_0, \mathbb{P}_1) &= n\mathbb{P}_0(S_i = s', A_i = a') D_{\mathrm{KL}}(\mathrm{Ber}(p_0), \mathrm{Ber}(p_1)) \leq \frac{n}{\mathrm{SA}} \cdot \frac{(p_0 - p_1)^2}{2p_1(1-p_1)} \\ &= \frac{n}{\mathrm{SA}} \cdot \frac{16\varepsilon^2(1-\gamma)^4 b^4}{2\gamma^4 p_1(1-p_1)} \\ &< \frac{n}{\mathrm{SA}} \cdot \frac{16\varepsilon^2(1-\gamma)^3 b^4}{\gamma^3(b-1)p_0}\,, \end{aligned}$$

where the first inequality is due to Proposition 14 and the second inequality is due to Eq. (D.10) and the fact that $p_0 < p_1$.

Now fix $\gamma_0 \in (0,1)$ and let $\gamma \geq \gamma_0$ and choose $b = 0.5(1 + \frac{1-\gamma_0/2}{1-\gamma_0}) \in (1, \frac{1-\gamma/2}{1-\gamma})$. Then, combining the above together and reordering show that if $n \leq c_1 \frac{\mathrm{SA}\ln(1/(4\delta))}{\varepsilon^2(1-\gamma)^3}$ where $c_1 = \frac{\gamma_0^3(b-1)p_0}{16b^4}$, we can guarantee that $\mathcal{L}$ is not (ε, δ)-sound on either M_0 or M_1, concluding the proof. □

D.3 Upper bound proofs

We start with some extra notation. We identify the transition function P as an $\mathrm{SA} \times \mathrm{S}$ matrix, whose entries $P_{sa,s'}$ specify the conditional probability of transitioning to state s' starting from state s and taking action a, and the reward function r as an $\mathrm{SA} \times 1$ reward vector. We use $\|x\|_1$ to denote the 1-norm $\sum_i |x_i|$ of $x \in \mathbb{R}^n$.

Recall first that we defined P^π to be the transition matrix on state-action pairs induced by the policy π. Define the H-step action-value function for $H > 0$ by

$$q_H^\pi = \sum_{h=0}^{H-1} (\gamma P^\pi)^h r\,.$$

We let v_H^π denote the H-step state-value function. In what follows we will need quantities for $\hat{M}$, which, in general could be any MDP that differs from M from only its transition kernel. Quantities related to $\hat{M}$ receive a "hat". For example, we use $\hat{P}$ for the transition kernel of $\hat{M}$, $\hat{P}^\pi$ for the state-action transition matrix induced by a policy π and $\hat{P}$, etc.

In subsequent proofs, we will need the following lemma, which gives two decompositions of the difference between the action-value functions on two MDPs, M and $\hat{M}$:

Lemma 12. *For any policy π, transition model $\hat{P}$, and $H > 0$,*

$$q_H^\pi - \hat{q}_H^\pi = \gamma \sum_{h=0}^{H-1} (\gamma P^\pi)^h (P - \hat{P}) \hat{v}_{H-h-1}^\pi \,, \tag{D.11}$$

$$\hat{q}_H^\pi - q_H^\pi = \gamma \sum_{h=0}^{H-1} (\gamma \hat{P}^\pi)^h (\hat{P} - P) v_{H-h-1}^\pi \,. \tag{D.12}$$

Proof. By symmetry, it suffices to prove Eq. (D.11). For convenience, we re-express a policy π as an $\mathrm{S} \times \mathrm{SA}$ matrix/operator Π. In particular, as a left linear operator, Π maps $q \in \mathbb{R}^{\mathrm{SA}}$ to $\sum_a \pi(a|\cdot) q(\cdot, a) \in \mathbb{R}^{\mathrm{S}}$. Note that with this $P^\pi = P\Pi$, $\hat{P}^\pi = \hat{P}\Pi$, $v_h^\pi = \Pi q_h^\pi$ and $\hat{v}_h^\pi = \Pi \hat{q}_h^\pi$. To reduce clutter, as π is fixed, for the rest of the proof we drop the upper indices and just use v_h, $\hat{v}_h$, q_h and $\hat{q}_h$.

Note that for $H > 0$,

$$\begin{aligned} q_H &= r + \gamma P \Pi q_{H-1} \,, \qquad \text{and} \\ \hat{q}_H &= r + \gamma \hat{P} \Pi \hat{q}_{H-1} \,. \end{aligned}$$

Hence,

$$\begin{aligned} q_H - \hat{q}_H &= \gamma (P \Pi q_{H-1} - \hat{P} \Pi \hat{q}_{H-1}) \\ &= \gamma (P - \hat{P}) \Pi \hat{q}_{H-1} + \gamma P \Pi (q_{H-1} - \hat{q}_{H-1}) \,. \end{aligned}$$

Then using $\Pi \hat{q}_{H-1} = \hat{v}_{H-1}$ and recursively expanding $q_{H-1} - \hat{q}_{H-1}$ in the same way gives the result, noting that $q_0 = r = \hat{q}_0$. □

We need two standard results from the concentration of binomial random variables.

Lemma 13 (Multiplicative Chernoff Bound for the Lower Tail, Theorem 4.5 of **?**)**.** *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables with parameter p, $S_n = \sum_{i=1}^n X_i$. Then, for any $0 \le \beta < 1$,*

$$\mathbb{P}\left(\frac{S_n}{n} \le (1 - \beta) p \right) \le \exp\left(-\frac{\beta^2 n p}{2} \right) .$$

Lemma 14. *Let n be a positive integer, $p > 0$, $\delta \in (0,1)$ such that*

$$\frac{2}{np} \ln \frac{1}{\delta} \leq \frac{1}{4}\,. \tag{D.13}$$

Let S_n be as in the previous lemma, $\hat{p} = S_n/n$. Then, with probability at least $1-\delta$, it holds that

$$\hat{p} \geq p/2 > 0$$

while we also have

$$\frac{1}{\hat{p}} \leq \frac{1}{p} + \frac{2}{p}\sqrt{\frac{2}{np} \ln \frac{1}{\delta}}\,.$$

on the same $(1-\delta)$-probability event.

In what follows, we will only need the first lower bound, $\hat{p} \geq p/2$ from above; the second is useful to optimize constants only.

Proof. According to the multiplicative Chernoff bound for the low tail (cf. Lemma 13), for any $0 < \delta \leq 1$, with probability at least $1-\delta$, we have

$$\hat{p} \geq p - \sqrt{\frac{2p}{n} \ln \frac{1}{\delta}}.$$

Denote by $\mathcal{E}_\delta$ the event when this inequality holds. Using Eq. (D.13), on $\mathcal{E}_\delta$ we have

$$\hat{p} \geq p - \sqrt{\frac{2p}{n} \ln \frac{1}{\delta}} = p\left(1 - \sqrt{\frac{2}{pn} \ln \frac{1}{\delta}}\right) \geq p\left(1 - \frac{1}{2}\right) = \frac{p}{2} > 0\,,$$

and thus, thanks to $1/(1-x) \leq 1 + 2x$ which holds for any $x \in [0, 1/2]$,

$$\frac{1}{\hat{p}} \leq \frac{1}{p} \frac{1}{1 - \sqrt{\frac{2}{np} \ln \frac{1}{\delta}}} \leq \frac{1}{p} + \frac{2}{p}\sqrt{\frac{2}{np} \ln \frac{1}{\delta}}\,.$$

□

Our next lemma bounds the deviation between the empirical transition kernel and the "true" one:

Lemma 15. *With probability* $1-\delta$*, for any* $(s,a) \in \mathcal{S} \times \mathcal{A}$*,*

$$\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\|_1 \leq \beta(N(s,a),\delta) \tag{D.14}$$

where for $u \geq 0$*,*

$$\beta(u,\delta) = 2\sqrt{\frac{\mathrm{S}\ln 2 + \ln \frac{u_+(u+1)\mathrm{SA}}{\delta}}{2u_+}}\,,$$

where $u_+ = u \vee 1$*.*

Proof. By abusing notation, for $u \geq 0$, let $\beta(u) = 2\sqrt{\frac{\mathrm{S}\ln 2 + \ln \frac{u_+(u+1)}{\delta}}{2u_+}}$, where $u_+ = u \vee 1$. We will prove below the following claim:
Claim: For any fixed state-action pair (s,a), with probability $1-\delta$,

$$\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\|_1 \leq \beta(N(s,a))\,.$$

Clearly, from this claim the lemma follows by a union bound over the state-action pairs. Hence, it remains to prove the claim.

For this fix $(s,a) \in \mathcal{S} \times \mathcal{A}$. Recall that the data $\mathcal{D} = ((S_i, A_i, R_i, S_i')_{i\in[n]})$ that is used to estimate $\hat{P}(\cdot|s,a)$ consists of m trajectories of length H obtained by following the uniform policy $\pi_{\log}$ while the initial state is selected from μ at random. In particular, for $j \in [m]$, the jth trajectory is

$$(S_0^{(j)}, A_0^{(j)}, R_0^{(j)}, \ldots, S_{h_j-1}^{(j)}, A_{H-1}^{(j)}, R_{H-1}^{(j)}, S_H^{(j)})\,,$$

where $S_0^{(j)} \sim \mu$, $A_t^{(j)} \sim \pi_{\log}(\cdot|S_t^{(j)})$, $(R_t^{(j)}, S_{t+1}^{(j)}) \sim Q(\cdot|S_t^{(j)}, A_t^{(j)})$. Clearly, if $q := \mathbb{P}\left(\exists 0 \leq i \leq H-1 \,:\, S_i^{(0)} = s, A_i^{(0)} = a\right) = 0$ then $N(s,a) = 0$ holds with probability one. The claim then follows since when $N(s,a) = 0$, $\hat{P}(\cdot|s,a)$ is identically zero, hence,

$$\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\|_1 = \|P(\cdot|s,a)\|_1 = 1 \leq 1.177\ldots \leq \beta(0)\,. \tag{D.15}$$

Hence, it remains to prove the claim for the case when $q > 0$, which we assume from now on. For convenience, append to the data infinitely many further trajectories, giving rise to the infinite sequence $(S_i, A_i, R_i, S_i')_{i\geq 0}$. Let $\tau_0 = 0$ and for $u \geq 1$, let $\tau_u = \min\{i \in \mathbb{N} \,:\, i > \tau_{u-1} \text{ and } S_i = s, A_i = a\}$ be

the "time" indices when (s,a) is visited, where we define the minimum of an empty set to be infinite. Since $q > 0$, almost surely $(\tau_u)_{u\geq 0}$ is a well-defined sequence of *finite* random variables. Now let $X_u = S'_{\tau_u}$ be the "next state" upon the uth visit of (s,a). Let $\hat{p}_u(s') = \frac{\sum_{v=1}^{u} \mathbb{I}\{X_v = s'\}}{u}$. Note that

$$\hat{P}(\cdot|s,a) = \hat{p}_{N(s,a)}(\cdot) \tag{D.16}$$

provided that $N(s,a) > 0$. By the Markov property, it follows that $(X_v)_{v\geq 1}$ is an i.i.d. sequence of categorical variables with common distribution $p(\cdot) := P(\cdot|s,a)$.

Now,

$$\|\hat{p}_u - p\|_1 = \max_{y\in\{-1,+1\}^{\mathcal{S}}} \langle \hat{p}_u - p, y\rangle \,,$$

while

$$\langle \hat{p}_u - p, y\rangle = \frac{1}{u}\sum_{v=1}^{u} \underbrace{y(X_v) - \sum_{s'} p(s')y(s')}_{\Delta_v} \,.$$

Now, $(\Delta_v)_{1\leq v\leq u}$ is an i.i.d. sequence, $|\Delta_v| \leq 2$ for any v and $\mathbb{E}\Delta_v = 0$. Hence, by Hoeffding's inequality, with probability $1-\delta$,

$$\frac{1}{u}\sum_{v=1}^{u} \Delta_v \leq 2\sqrt{\frac{\ln\frac{1}{\delta}}{2u}} \,.$$

Since the cardinality of $\{-1,+1\}^{\mathcal{S}}$ is 2^{S}, applying a union bound over $y \in \{-1,+1\}^{\mathcal{S}}$, we get that with probability $1-\delta$,

$$\|\hat{p}_u - p\|_1 \leq 2\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{1}{\delta}}{2u}} \,.$$

Applying another union bound over u, owing to that $\sum_{u=1}^{\infty} \frac{1}{u(u+1)} = 1$, we get that with probability $1-\delta$, for any $u \geq 1$,

$$\|\hat{p}_u - p\|_1 \leq 2\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{u(u+1)}{\delta}}{2u}} = \beta(u) \,.$$

Since $\|\hat{p}_0 - p\|_1 \leq 1 \leq \beta(0)$ (cf. Eq. (D.15)), the claim follows by Eq. (D.16).

□

We now state a lemma that bounds, with high probability, the error of predicting the value of some fixed policy when the prediction is based on a transition kernel P' which is "close" to the true transition kernel P, where closedness is based on how often the individual state-action pairs have been visited. This notion of closedness is motivated by Lemma 15; this lemma can be used when $P' = \hat{P}$, or some other transition kernel in the vicinity of $\hat{P}$. The former will be needed in the analysis of the plug-in method presented here; while the latter will be used in the next section where we analyze the pessimistic algorithm.

Lemma 16. *Let $\delta \in (0,1)$ and m be the number of episodes collected by the logging policy and fix any policy π. For any P' such that for any $(s,a) \in \mathcal{S}\times\mathcal{A}$,*

$$\|P'(\cdot|s,a) - P(\cdot|s,a)\|_1 \le \frac{C}{\sqrt{N(s,a)\vee 1}},$$

with probability at least $1-\delta$ for $C > 0$, we have

$$v^\pi(\mu) - v^\pi_{P'}(\mu) \le \frac{4\gamma C \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2\sqrt{m}} + \frac{8\gamma \mathrm{SA}}{(1-\gamma)^2}\frac{\ln\frac{\mathrm{SA}}{\delta}}{m} + \varepsilon\,. \tag{D.17}$$

Proof. Note that

$$\gamma^{H_{\gamma,\epsilon}} \le \gamma^{1+\frac{\ln(1/\epsilon)}{\ln(1/\gamma)}} = \gamma\epsilon\,.$$

Hence, for $H = H_{\gamma,(1-\gamma)\varepsilon/(2\gamma)}$,

$$\gamma^H \le \tfrac{1}{2}\,\epsilon(1-\gamma)\,.$$

Owning to that the immediate rewards belong to $[-1,1]$, it follows that for any policy π,

$$q^\pi - q^\pi_{P'} \le q^\pi_H - q^\pi_{P',H} + \varepsilon\mathbf{1}\,, \tag{D.18}$$

where we use $q^\pi_{P',H}$ to denote the H-step value function under transition model P'. Define $N_h(s,a)$ as the number of episodes when the hth state-action pair in the episode is (s,a). Note that $N(s,a) \ge N_h(s,a)$. Let $Z_h = \{(s,a) \in \mathcal{S}\times\mathcal{A} : \nu^\pi_{\mu,h}(s,a) > \frac{8}{m}\ln\frac{\mathrm{SA}}{\delta}\}$ and let $\mathcal{F}$ be the event when

$$\frac{N_h(s,a)}{m} \ge \frac{\nu^{\pi_{\log}}_{\mu,h}(s,a)}{2}$$

holds for any $(s,a) \in Z_h$.[2] By Lemma 14, $\mathbb{P}(\mathcal{F}) \geq 1-\delta$.

Assume that $\mathcal{F}$ holds. Combining Eq. (D.18) with Lemma 12, we get that on this event

$$
\begin{aligned}
v^\pi(\mu) - \hat{v}^\pi(\mu) &\leq (\mu^\pi)^\top (q^\pi_H - q^\pi_{P',H}) + \varepsilon \\
&= \gamma \sum_{h=0}^{H-1} \gamma^h (\nu^\pi_{\mu,h})^\top (P - P') v^\pi_{P',H-h-1} + \varepsilon && \text{(by Eq. (D.11))} \\
&\leq \frac{\gamma}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{s,a} \nu^\pi_{\mu,h}(s,a) \|P(\cdot|s,a) - P'(\cdot|s,a)\|_1 + \varepsilon \\
& && \text{(by } \|\hat{v}^\pi_{H-h-1}\|_\infty \leq 1/(1-\gamma)) \\
&\leq \frac{\gamma}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{(s,a)\in Z_h} \nu^\pi_{\mu,h}(s,a) \|P(\cdot|s,a) - P'(\cdot|s,a)\|_1 + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \\
& && \text{(by the definition of } Z_h) \\
&\leq \frac{\gamma}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{(s,a)\in Z_h} \sqrt{\nu^\pi_{\mu,h}(s,a)} \|P(\cdot|s,a) - P'(\cdot|s,a)\|_1 + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \\
& && \text{(by } \nu^\pi_{\mu,h}(s,a) \leq 1) \\
&\leq \frac{\gamma \mathrm{A}^{\min(H,\mathrm{S})/2}}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{(s,a)\in Z_h} \sqrt{\nu^{\pi_{\log}}_{\mu,h}(s,a)} \|P'(\cdot|s,a) - P(\cdot|s,a)\|_1 + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \\
& && \text{(by Proposition 12)} \\
&\leq \frac{2\gamma \mathrm{A}^{\min(H,\mathrm{S})/2} C}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{(s,a)\in Z_h} \sqrt{\nu^{\pi_{\log}}_{\mu,h}(s,a)} \frac{1}{\sqrt{N_h(s,a) \vee 1}} + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \\
& && \text{(by the definition of } P') \\
&\leq \frac{2\gamma \mathrm{A}^{\min(H,\mathrm{S})/2} C}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \sum_{(s,a)\in Z_h} \sqrt{\nu^{\pi_{\log}}_{\mu,h}(s,a)} \sqrt{\frac{2}{m \nu^{\pi_{\log}}_{\mu,h}(s,a)}} + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \\
& && \text{(by the definitions of } \mathcal{F} \text{ and } Z_h) \\
&\leq \frac{4\gamma \mathrm{A}^{\min(H,\mathrm{S})/2} C}{(1-\gamma)^2 \sqrt{m}} \mathrm{SA} + \frac{8\gamma \mathrm{SA}}{m(1-\gamma)^2} \ln \frac{\mathrm{SA}}{\delta} + \varepsilon \,.
\end{aligned}
$$

This finishes the proof. □

For the plug-in method we use the previous lemma with $P' = \hat{P}$, resulting in the following corollary:

[2] Note that Z_h, and thus also $\mathcal{F}$ depends on π, which is the reason that the result, as stated, holds only for a fixed policy.

Corollary 11. *Let $\delta \in (0,1)$ and m be the number of episodes collected by the logging policy and fix any policy π. With probability at least $1-\delta$, we have*

$$v^\pi(\mu) - v^\pi_{\hat{P}}(\mu) \le \frac{8\gamma \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2}\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{2n(n+1)\mathrm{SA}}{\delta}}{2m}} + \frac{8\gamma\mathrm{SA}}{(1-\gamma)^2}\frac{\ln\frac{2\mathrm{SA}}{\delta}}{m} + \varepsilon\,.$$

Proof. Fix $\delta \in (0,1)$. Let $\mathcal{E}_\delta$ be the event when for any $(s,a) \in \mathcal{S}\times\mathcal{A}$,

$$\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\|_1 \le \beta(N(s,a),\delta)\,, \tag{D.19}$$

where β is defined in Lemma 15, which also gives that $\mathbb{P}(\mathcal{E}_\delta) \ge 1-\delta$. Further, defining

$$C_\delta = 2\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{n(n+1)\mathrm{SA}}{\delta}}{2}}\,,$$

note that $\beta(u,\delta) \le C_\delta/\sqrt{u \vee 1}$. Now, let $\mathcal{F}_\delta$ be the event when the conclusion of Lemma 16 holds. Then, on the one hand, by a union bound, $\mathbb{P}(\mathcal{E}_{\delta/2}\cap\mathcal{F}_{\delta/2}) \ge 1-\delta$, while on the other hand on $\mathcal{E}_{\delta/2}\cap\mathcal{F}_{\delta/2}$, the condition of Lemma 16 holds for P' defined so that

$$P'(\cdot|s,a) = \begin{cases} \hat{P}(\cdot|s,a)\,, & \text{if } \|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\|_1 \le \beta(N(s,a),\delta/2)\,; \\ P(\cdot|s,a)\,, & \text{otherwise}\,. \end{cases}$$

with $C := C_{\delta/2}$.

Furthermore, on $\mathcal{E}_{\delta/2}$, $\hat{P}(\cdot|s,a) = P(\cdot|s,a)$ holds for any (s,a) pair. Hence, the result follows by replacing δ with $\delta/2$ in Eq. (D.17) and plugging in $C_{\delta/2}$ in place of C. □

We now are ready to prove the upper bound of plug-in algorithm.

Theorem 24 (Restatement of Theorem 16)**.** *Fix* S, A, *an MDP $M \in \mathcal{M}(\mathrm{S},\mathrm{A})$ and a distribution μ on the state space of M. Suppose $\delta > 0$, $\varepsilon > 0$, and $\varepsilon_{opt} > 0$. Assume that the data is collected by following the uniform policy and it consists of m episodes, each of length $H = H_{\gamma,(1-\gamma)\varepsilon/(2\gamma)}$. Let $\hat{\pi}$ be any deterministic, ε_{opt}-optimal policy for $\hat{M} = (\hat{P}, r)$ where $\hat{P}$ is the sample-mean based empirical estimate of the transition probabilities based on the data collected. Then if*

$$m = \tilde{\Omega}\left(\frac{\mathrm{S}^3\mathrm{A}^{\min(H,\mathrm{S})+2}\ln\frac{1}{\delta}}{(1-\gamma)^4\epsilon^2}\right),$$

where $\tilde{\Omega}$ *hides log factors of* S, A *and* H*, we have* $v^{\hat{\pi}}(\mu) \geq v^*(\mu) - 4\varepsilon - \varepsilon_{opt}$ *with probability at least* $1 - \delta$*.*

Proof. We upper bound the suboptimality gap of $\hat{\pi}$ as follows:

$$\begin{aligned} v^*(\mu) - v^{\hat{\pi}}(\mu) &= v^*(\mu) - \hat{v}^{\pi^*}(\mu) + \hat{v}^{\pi^*}(\mu) - \hat{v}^{\hat{\pi}}(\mu) + \hat{v}^{\hat{\pi}}(\mu) - v^{\hat{\pi}}(\mu) \\ &\leq v^*(\mu) - \hat{v}^{\pi^*}(\mu) + \hat{v}^{\hat{\pi}}(\mu) - v^{\hat{\pi}}(\mu) + \varepsilon_{\text{opt}} \,. \qquad (\ \hat{\pi} \text{ is } \varepsilon_{\text{opt}}\text{-optimal in } \hat{M}) \end{aligned}$$

By Corollary 11 and a union bound, with probability at least $1 - \delta$, for any deterministic policy π obtained from the data $\mathcal{D}$ we have

$$\begin{aligned} &v^{\pi}(\mu) - \hat{v}^{\pi}(\mu) \\ &\quad \leq \frac{8\gamma \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2} \sqrt{\frac{\mathrm{S}\ln 2 + \ln \frac{2n(n+1)\mathrm{SA}}{\delta} + \mathrm{S}\ln \mathrm{A}}{2m}} + \frac{8\gamma \mathrm{SA}}{(1-\gamma)^2} \frac{\ln \frac{2\mathrm{SA}}{\delta} + \mathrm{S}\ln \mathrm{A}}{m} + \varepsilon \\ &\quad \leq \frac{8\gamma \mathrm{S}^{\frac{3}{2}} \mathrm{A}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2} \sqrt{\frac{\ln 2 + \ln \frac{H^2\mathrm{SA}}{\delta} + \ln 2m + \ln \mathrm{A}}{2m}} + \frac{8\gamma \mathrm{SA}}{(1-\gamma)^2} \frac{\ln \frac{2\mathrm{SA}}{\delta} + \mathrm{S}\ln \mathrm{A}}{m} + \varepsilon \,, \end{aligned}$$

Thus, given that

$$m = \tilde{\Omega}\left(\frac{\mathrm{S}^3 \mathrm{A}^{\min(H,\mathrm{S})+2} \ln \frac{1}{\delta}}{(1-\gamma)^4 \epsilon^2} \right) ,$$

where $\tilde{\Omega}$ hides log-factors, with probability at least $1 - \delta$ we have,

$$v^*(\mu) - v^{\hat{\pi}}(\mu) \leq v^*(\mu) - \hat{v}^{\pi^*}(\mu) + \hat{v}^{\hat{\pi}^*}(\mu) - v^{\hat{\pi}^*}(\mu) + \varepsilon_{\text{opt}} \leq 4\varepsilon + \varepsilon_{\text{opt}} \,.$$

□

D.3.1 Pessimistic Algorithm

We present a result in this section for the "pessimistic algorithm" in the discounted total expected reward criterion to complement the results in the main text (Jin et al., 2021; Buckman et al., 2020; Kidambi et al., 2020; Yu et al., 2020; Kumar et al., 2020; Liu et al., 2020). The sample complexity we show is the same as for the plug-in method. While this may be off by a polynomial factor, we do not expect the pessimistic algorithm to have a major advantage over the plug-in method in the worst-case setting. In fact, the recent work of Xiao et al. (2021c) established this in a rigorous fashion for the bandit setting

by showing an algorithm independent lower bound that matched the upper bound for both the plug-in method and the pessimistic algorithm. As argued by Xiao et al. (2021c) (and proved by Jin et al. (2021) in the context of linear MDPs, which includes tabular MDPs), the advantage of the pessimistic algorithm is that it is weighted minimax optimal with respect to a special criterion.

The pessimistic algorithm with parameters $\delta \in (0,1)$ and $\epsilon_{\text{opt}} > 0$ chooses a deterministic ϵ_{opt} policy $\tilde{\pi}$ of the MDP with reward r and transition kernel $\tilde{P}$, the latter of which is obtained via

$$\tilde{P} = \operatorname*{arg\,min}_{P' \in \mathcal{P}_\delta} v^*_{P'}(\mu)\,,$$

where for a transition kernel P' we use $v^*_{P'}$ to denote the optimal value function in the MDP with immediate rewards r and transition kernel P', and $\mathcal{P}_\delta$ is is defined as

$$\mathcal{P}_\delta = \left\{ P' \,:\, \text{for any } (s,a) \in \mathcal{S} \times \mathcal{A}\,, \|\hat{P}(\cdot|s,a) - P'(\cdot|s,a)\|_1 \le \beta(N(s,a),\delta) \right\}\,,$$

where β is defined in Lemma 15. Recall that the same result ensures that P, the "true" transition kernel belongs to $\mathcal{P}_\delta$ with probability at least $1-\delta$.

Theorem 25 (Pessimistic algorithm)**.** *Fix* S, A, *an MDP* $M \in \mathcal{M}(\mathrm{S},\mathrm{A})$ *and a distribution* μ *on the state space of* M*. Suppose* $\delta > 0$*,* $\varepsilon > 0$*, and* $\epsilon_{\text{opt}} > 0$*. Assume that the data is collected by following the uniform policy and it consists of* m *episodes, each of length* $H = H_{\gamma,(1-\gamma)\varepsilon/(2\gamma)}$*. Then, if*

$$m = \tilde{\Omega}\left(\frac{\mathrm{S}^3 \mathrm{A}^{\min(H,\mathrm{S})+2} \ln \frac{1}{\delta}}{(1-\gamma)^4 \epsilon^2} \right)\,,$$

where $\tilde{\Omega}$ *hides log factors of* S, A *and* H*, we have* $v^{\tilde{\pi}}(\mu) \ge v^*(\mu) - 2\varepsilon - \varepsilon_{opt}$ *with probability at least* $1-\delta$*, where* $\tilde{\pi}$ *is the output of the pessimistic algorithm run with parameters* $(\delta, \epsilon_{\text{opt}})$*.*

Proof. Let us denote by $v^\pi_{P'}$ the value function of policy π in the MDP with reward r and transition kernel P'. Let $\Delta^\pi = v^\pi_{\tilde{P}}(\mu) - v^\pi(\mu)$ and let π^* is an

deterministic optimal policy in the "true" MDP. Such a policy exists (e.g., see Theorem 6.2.10 of Puterman (2014)). We have

$$\begin{aligned}
v^*(\mu) - v^{\tilde{\pi}}(\mu) &= v^*(\mu) - v^{\tilde{\pi}}_{\tilde{P}}(\mu) + \Delta^{\tilde{\pi}} \\
&\le v^*(\mu) - v^*_{\tilde{P}}(\mu) + \Delta^{\tilde{\pi}} + \epsilon_{\text{opt}} && \text{(by the definition of } \tilde{\pi}) \\
&\le v^*(\mu) - v^{\pi^*}_{\tilde{P}}(\mu) + \Delta^{\tilde{\pi}} + \epsilon_{\text{opt}} && \text{(because } v^{\pi^*}_{\tilde{P}} \le v^*_{\tilde{P}}) \\
&\le \Delta^{\pi^*} + \Delta^{\tilde{\pi}} + \epsilon_{\text{opt}} \,. && \text{(because } v^*(\mu) = v^{\pi^*}(\mu))
\end{aligned}$$

Hence, it remains to upper bound Δ^{π^*} and $\Delta^{\tilde{\pi}}$. For this, we make the following claim:

Claim: Fix any policy π. Then, with probability at least $1-\delta$, we have

$$v^{\pi}(\mu) - v^{\pi}_{\tilde{P}}(\mu) \le \frac{16\gamma \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2}\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{2n(n+1)\mathrm{SA}}{\delta}}{2m}} + \frac{8\gamma\mathrm{SA}}{(1-\gamma)^2}\frac{\ln\frac{2\mathrm{SA}}{\delta}}{m} + \varepsilon \,.$$

Proof of Claim. For the latter, note that the proof of Corollary 11 can be repeated with the only change that now instead of Eq. (D.19), we have

$$\|P(\cdot|s,a) - \tilde{P}(\cdot|s,a)\|_1 \le \|P(\cdot|s,a) - \hat{P}(\cdot|s,a)\|_1 \|\hat{P}(\cdot|s,a) - \tilde{P}(\cdot|s,a)\|_1 \le 2\beta(N(s,a),\delta) \,.$$

□

From this claim, by a union bound over all the A^{S} deterministic policies, we get that with probability $1-\delta$, for any deterministic policy π,

$$\begin{aligned}
&v^{\pi}(\mu) - v^{\pi}_{\tilde{P}}(\mu) \\
&\quad\le \frac{16\gamma \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2}\sqrt{\frac{\left(\mathrm{S}\ln 2 + \ln\frac{2n(n+1)\mathrm{SA}}{\delta} + \mathrm{S}\ln\mathrm{A}\right)}{2m}} + \frac{8\gamma\mathrm{SA}}{(1-\gamma)^2}\frac{\ln\frac{2\mathrm{SA}}{\delta} + \mathrm{S}\ln\mathrm{A}}{m} + \varepsilon \,.
\end{aligned}$$

Since $\tilde{\pi}$, by definition is also a deterministic policy, the last display holds with probability $1-\delta$ for $\tilde{\pi}$ as well. Putting things together gives that

$$\begin{aligned}
&v^*(\mu) - v^{\tilde{\pi}}(\mu) \\
&\quad\le \frac{32\gamma \mathrm{SA}^{\frac{\min(H,\mathrm{S})}{2}+1}}{(1-\gamma)^2}\sqrt{\frac{\mathrm{S}\ln 2 + \ln\frac{2n(n+1)\mathrm{SA}}{\delta} + \mathrm{S}\ln\mathrm{A}}{2m}} + \frac{16\gamma\mathrm{SA}}{(1-\gamma)^2}\frac{\ln\frac{2\mathrm{SA}}{\delta} + \mathrm{S}\ln\mathrm{A}}{m} + 2\varepsilon + \epsilon_{\text{opt}} \,.
\end{aligned}$$

The proof is finished by a calculation similar to that done at the end of the proof of Theorem 24. □

Appendix E

Proofs for Chapter 8: Understanding and Leveraging Overparameterization in Recursive Value Estimation

E.1 Proofs

E.1.1 Proof of Theorem 17

Theorem 26 (Restatement of Theorem 17)**.** *Let $\theta_0 \in \mathbb{R}^d$ be the initial weight vector. With $\eta \leq \frac{1}{(1+\gamma)^2}$, RM converges to $\theta_{\mathrm{RM}} = (\boldsymbol{M} - \gamma \boldsymbol{N})^\dagger \boldsymbol{R} + (\boldsymbol{I}_d - \Pi_{\boldsymbol{M}-\gamma\boldsymbol{N}})\theta_0$.*

Proof. Let $\boldsymbol{A} = \boldsymbol{M} - \gamma \boldsymbol{N}$ for simplicity. First recall the residual minimization update,

$$\theta_{t+1} = \left(\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right) \theta_t + \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{R}\,. \tag{E.1}$$

Let $\theta^* = \boldsymbol{A}^\dagger \boldsymbol{R}$. It can be verified θ^* is one of the feasible solution as $\boldsymbol{A}\theta^* = \boldsymbol{R}$. Then we use induction to show that for any $\theta_0 \in \mathbb{R}^d$ and $t \geq 0$

$$\theta_{t+1} - \theta^* = (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} (\theta_0 - \theta^*)\,. \tag{E.2}$$

The base case holds by the update rule Eq. (E.1). Suppose that the statement

holds for t, then we have

$$\theta_{t+1} - \theta^* = \left(\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right) \theta_t + \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{R} - \theta^* \tag{E.3}$$

$$= \left(\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right) \theta_t - (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})\theta^* \tag{E.4}$$

$$= \left(\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right) (\theta_t - \theta^*) \tag{E.5}$$

$$= \left(\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right)^{t+1} (\theta_0 - \theta^*) \,. \tag{E.6}$$

Thus,

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} \theta_0 + (\boldsymbol{I}_d - (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1}) \theta^* \,. \tag{E.7}$$

We let $\boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\top$ be its eigendecomposition of $\boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}$, which is the empirical covariance matrix of residual features. Let $\boldsymbol{V}_-$ be the null space of $\boldsymbol{V}$. Then

$$\boldsymbol{I}_d - (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} \tag{E.8}$$

$$= \boldsymbol{I}_d - (\boldsymbol{V}\boldsymbol{V}^\top - \eta \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\top + \boldsymbol{V}_- \boldsymbol{V}_-^\top)^{t+1} \tag{E.9}$$

$$= \boldsymbol{I}_d - (\boldsymbol{V}(\boldsymbol{I}_k - \eta \boldsymbol{\Lambda}) \boldsymbol{V}^\top + \boldsymbol{V}_- \boldsymbol{V}_-^\top)^{t+1} \tag{E.10}$$

$$= \boldsymbol{I}_d - (\boldsymbol{V}_- \boldsymbol{V}_-^\top)^{t+1} - \boldsymbol{V}(\boldsymbol{I}_k - \eta \boldsymbol{\Lambda})^{t+1} \boldsymbol{V}^\top \tag{E.11}$$

$$= \boldsymbol{V}\boldsymbol{V}^\top - \boldsymbol{V}(\boldsymbol{I}_k - \eta \boldsymbol{\Lambda})^{t+1} \boldsymbol{V}^\top \tag{E.12}$$

$$= \boldsymbol{V} \left(\boldsymbol{I}_k - (\boldsymbol{I}_k - \eta \boldsymbol{\Lambda})^{t+1}\right) \boldsymbol{V}^\top \tag{E.13}$$

Let $\lambda_{\max}$ be the largest eigenvalue of $\boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}$. We now show that $\lambda_{\max} \leq 1 + \gamma$.

$$\lambda_{\max}\left(\boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A}\right) \leq \sum_{i=1}^{k} \hat{\mu}(s_i) \lambda_{\max}\Big((\phi(s_i) - \gamma \bar{\phi}(s_i'))(\phi(s_i) - \gamma \bar{\phi}(s_i'))^\top\Big) \leq (1+\gamma)^2 \,, \tag{E.14}$$

where we use the fact that $\lambda_{\max}$ is a convex function and we assume $\|\phi(s)\| \leq 1$ for all $s \in \mathcal{S}$. Thus, given that $\eta \leq \frac{1}{1+\gamma}$, $\eta \leq \frac{1}{\lambda_{\max}}$. Then $\boldsymbol{I}_d - (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} = \boldsymbol{V}\boldsymbol{V}^\top$ as $t \to \infty$. Thus

$$\lim_{t\to\infty} \theta_t = \lim_{t\to\infty} (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} \theta_0 + \boldsymbol{V}\boldsymbol{V}^\top \theta^* = \lim_{t\to\infty} (\boldsymbol{I}_d - \eta \boldsymbol{A}^\top \boldsymbol{D}_k \boldsymbol{A})^{t+1} \theta_0 + \theta^* \,, \tag{E.15}$$

where the last equality follows by that θ^* is in the row space of $\boldsymbol{A}$ by definition. When $\theta_0 = 0$, we have the algorithm converges to θ^*.

We next show the result for general θ_0. Let $\theta_0 = \theta_0^1 + \theta_0^2$, where $\theta_0^1 = \Pi_{\boldsymbol{M}-\gamma\boldsymbol{N}}\theta_0$ is the component of θ_0 that is in the row space of $\boldsymbol{A}$, $\theta_0^2 = (\boldsymbol{I}_d - \Pi_{\boldsymbol{M}-\gamma\boldsymbol{N}})\theta_0$ is the perpendicular residual. Then,

$$\lim_{t\to\infty}(\boldsymbol{I}_d - \eta\boldsymbol{A}^\top\boldsymbol{D}_k\boldsymbol{A})^{t+1}\theta_0 \tag{E.16}$$

$$= \lim_{t\to\infty}(\boldsymbol{V}_-\boldsymbol{V}_-^\top + \boldsymbol{V}(\boldsymbol{I}_k - \eta\boldsymbol{\Lambda})^{t+1}\boldsymbol{V}^\top)(\theta_0^1 + \theta_0^2) \tag{E.17}$$

$$= \theta_0^2 + \lim_{t\to\infty}\boldsymbol{V}(\boldsymbol{I}_k - \eta\boldsymbol{\Lambda})^{t+1}\boldsymbol{V}^\top\theta_0^1 = \theta_0^2\,, \tag{E.18}$$

where the last step follows by the choice of η. This finishes the proof.

□

E.1.2 Proof of Theorem 18

We will need the matrix binomial theorem in the proof.

Lemma 17 (Matrix Binomial Theorem)**.** *For $n \geq 0$ and two matrices $\boldsymbol{X}, \boldsymbol{Y}$*

$$(\boldsymbol{I} + \boldsymbol{X}\boldsymbol{Y})^n\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{I} + \boldsymbol{Y}\boldsymbol{X})^n\,. \tag{E.19}$$

Proof.

$$(\boldsymbol{I} + \boldsymbol{X}\boldsymbol{Y})^n\boldsymbol{X} = \sum_{k=0}^{n}\binom{n}{k}(\boldsymbol{X}\boldsymbol{Y})^k\boldsymbol{X} = \boldsymbol{X}\sum_{k=0}^{n}\binom{n}{k}(\boldsymbol{Y}\boldsymbol{X})^k = \boldsymbol{X}(\boldsymbol{I} + \boldsymbol{Y}\boldsymbol{X})^n\,. \tag{E.20}$$

□

Theorem 27 (Restatement of Theorem 18)**.** *Assuming that $\boldsymbol{M}^\top\boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N})$ is diagonalizable Let $\theta_0 \in \mathbb{R}^d$ be the initial weight vector. With $\eta < \frac{1}{(1+\gamma)\|\Phi\|}$, if $\|\boldsymbol{W}\| < \frac{1}{\gamma}$, TD(0) converges to $\theta_{TD} = \boldsymbol{M}^\dagger(\boldsymbol{I}_k - \gamma\boldsymbol{W})^{-1}\boldsymbol{R} + \beta$, where $\beta = \boldsymbol{Q}_0\boldsymbol{Q}_0^{-1}\theta_0$, $\boldsymbol{Q}_0$ are eigenvectors of $\boldsymbol{M}^\top\boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N})$ with zero eigenvalues. If $\|\boldsymbol{W}\| \geq \frac{1}{\gamma}$ there is an initial θ_0 for which TD(0) does not converge.*

Proof. We first rewrite the TD update formulate as

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta\boldsymbol{M}^\top\boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N}))\theta_t + \eta\boldsymbol{M}^\top\boldsymbol{D}_k\boldsymbol{R} \tag{E.21}$$

A simple recursive argument shows that for any $\theta_0 \in \mathbb{R}^d$,

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{M} - \gamma \boldsymbol{N}))^{t-1} \theta_0 + \eta \sum_{i=0}^{t} (\boldsymbol{I}_d - \eta \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{M} - \gamma \boldsymbol{N}))^i \boldsymbol{M}^\top \boldsymbol{D}_k \boldsymbol{R}\,. \tag{E.22}$$

By the matrix binomial theorem (Lemma 17),

$$(\boldsymbol{I}_d - \eta \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{M} - \gamma \boldsymbol{N}))^i \boldsymbol{M}^\top \boldsymbol{D}_k = \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{I}_k - \eta (\boldsymbol{M} - \gamma \boldsymbol{N}) \boldsymbol{M}^\top \boldsymbol{D}_k)^i\,. \tag{E.23}$$

By writing $\boldsymbol{N}$ as the projection to the row-span of $\boldsymbol{M}$ and the perpendicular component, we have

$$\begin{aligned}
&(\boldsymbol{M} - \gamma \boldsymbol{N}) \boldsymbol{M}^\top && \text{(E.24)}\\
=&(\boldsymbol{M} - \gamma \boldsymbol{N} \boldsymbol{M}^\dagger \boldsymbol{M} - \gamma \boldsymbol{N} (\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M})) \boldsymbol{M}^\top && \text{(E.25)}\\
=&(\boldsymbol{I}_k - \gamma \boldsymbol{W}) \boldsymbol{M} \boldsymbol{M}^\top\,, && \text{(E.26)}
\end{aligned}$$

where the last step follows by $(\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}) \boldsymbol{M}^\top = 0$. Thus we can rewrite θ_{t+1} as

$$\theta_{t+1} = (\boldsymbol{I}_d - \eta \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{M} - \gamma \boldsymbol{N}))^{t-1} \theta_0 + \eta \boldsymbol{M}^\top \boldsymbol{D}_k \sum_{i=0}^{t} (\boldsymbol{I}_k - \eta (\boldsymbol{M} - \gamma \boldsymbol{N}) \boldsymbol{M}^\top \boldsymbol{D}_k)^i \boldsymbol{R} \tag{E.27}$$

$$= (\boldsymbol{I}_d - \eta \boldsymbol{M}^\top \boldsymbol{D}_k (\boldsymbol{M} - \gamma \boldsymbol{N}))^{t-1} \theta_0 + \eta \boldsymbol{M}^\top \boldsymbol{D}_k \sum_{i=0}^{t} (\boldsymbol{I}_k - \eta (\boldsymbol{I}_k - \gamma \boldsymbol{W}) \boldsymbol{M} \boldsymbol{M}^\top \boldsymbol{D}_k)^i \boldsymbol{R}\,, \tag{E.28}$$

Given $\|\boldsymbol{W}\| < 1/\gamma$, we have that all eigenvalues of $\boldsymbol{I}_k - \gamma \boldsymbol{W}$ are positive. Let $\eta < \frac{1}{(1+\gamma)\|\Phi\|}$, then

$$\|\eta (\boldsymbol{I}_k - \gamma \boldsymbol{W}) \boldsymbol{M} \boldsymbol{M}^\top \boldsymbol{D}_k\| < \eta \|(\boldsymbol{I}_k - \gamma \boldsymbol{W})\| \|\boldsymbol{M} \boldsymbol{M}^\top \boldsymbol{D}_k\| < 1\,, \tag{E.29}$$

otherwise the matrix power series diverges. Thus

$$\eta \boldsymbol{M}^\top \boldsymbol{D}_k \sum_{i=0}^{t} (\boldsymbol{I}_k - \eta (\boldsymbol{I}_k - \gamma \boldsymbol{W}) \boldsymbol{M} \boldsymbol{M}^\top \boldsymbol{D}_k)^i \boldsymbol{R} = \boldsymbol{M}^\dagger (\boldsymbol{I}_k - \gamma \boldsymbol{W})^{-1} \boldsymbol{R}\,. \tag{E.30}$$

Therefore, given that $\theta_0 = 0$, we have the algorithm converge to $\boldsymbol{M}^\dagger (\boldsymbol{I}_k - \gamma \boldsymbol{W})^{-1} \boldsymbol{R}$.

We now show the convergence point for an arbitrary θ_0. Let $\boldsymbol{Q}\Lambda\boldsymbol{Q}^{-1}$ be the eigen decomposition of $\boldsymbol{M}^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N})$. By the low rank structure of this matrix, it has at most $h \leq k$ non-zero eigenvalues. Let $\boldsymbol{Q}_0$ be the eigenvectors with eigenvalue *zero*. Then

$$\lim_{t\to\infty}(\boldsymbol{I}_d - \eta\boldsymbol{M}^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N}))^t\theta_0 \tag{E.31}$$

$$= \lim_{t\to\infty} \boldsymbol{Q}(\boldsymbol{I}_d - \eta\Lambda)^t\boldsymbol{Q}^{-1}\theta_0 \tag{E.32}$$

$$= \boldsymbol{Q}_0\boldsymbol{Q}_0^{-1}\theta_0\,, \tag{E.33}$$

where the last step follows by the choice of η.

Characterization for Non-diagonalizable Case

In the above analysis, we assume that the matrix $\boldsymbol{M}^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N})$ is diagonalizable. We now characterize the convergent point for the general case using Jordan normal form of the matrix. Let $\boldsymbol{Z} = \boldsymbol{M}^\top \boldsymbol{D}_k(\boldsymbol{M} - \gamma\boldsymbol{N})$ and $\boldsymbol{Z} = \boldsymbol{J}\boldsymbol{J}\boldsymbol{Q}^{-1}$ be the jordan normal form of $\boldsymbol{Z}$. We still denote $\boldsymbol{Q}_0$ the eigenvectors with eigenvalue zero. Then there is

$$\lim_{t\to\infty}(\boldsymbol{I} - \eta\boldsymbol{Z})^t = \lim_{t\to\infty} \boldsymbol{Q}(\boldsymbol{I} - \eta\boldsymbol{J})^t\boldsymbol{Q}^{-1} \tag{E.34}$$

Since $\boldsymbol{I} - \eta\boldsymbol{J}$ has a block diagonal structure, its power can be obtained by first computing the power of each block. Let $\boldsymbol{J}_i$ be the jordan block with eigenvalue λ_i. We write $\boldsymbol{J}_i = \lambda_i\boldsymbol{I} + \boldsymbol{L}$, where $\boldsymbol{L}$ is a matrix such that the only non-zero entries of $\boldsymbol{L}$ are on the first off-diagonal. Then we can write the i-th block of $\boldsymbol{J}$ as $(1 - \eta\lambda_i)\boldsymbol{I} - \eta\boldsymbol{L}$. Using the binomial theorem we get

$$((1 - \eta\lambda_i)\boldsymbol{I} - \eta\boldsymbol{L})^t = \sum_{s=0}^{t} \tbinom{t}{s}(1 - \eta\lambda_i)^{t-s}(-\eta\boldsymbol{L})^s\,. \tag{E.35}$$

Note that $\boldsymbol{L}^s$ is the matrix with ones on the s-th diagonal away from the main diagonal, and $\boldsymbol{L}^s = 0$ for s larger than the size of $\boldsymbol{L}$. Therefore, $((1 - \eta\lambda_i)\boldsymbol{I} - \eta\boldsymbol{L})^t$ is a triangular matrix with $(1-\eta\lambda_i)^t$ on the main diagonal, $-\eta t(1-\eta\lambda_i)^{t-1}$ on the first off-diagonal, and so on. Therefore, the eigenvalues of this matrix are all $(1 - \eta\lambda_i)^t$. Then given a learning rate that $\eta < 1/\lambda_{\max}$, for any jordan block with $\lambda_i > 0$, we have that the matrix power converges. For $\lambda_i = 0$, the

jordan block corresponds to eigenvectors that are in the kernel space of $\boldsymbol{Z}$. Thus, suppose that all eigenvalues of $\boldsymbol{Z}$ are non-negative, we have

$$\lim_{t\to\infty} \boldsymbol{Q}(\boldsymbol{I} - \eta\boldsymbol{J})^t\boldsymbol{Q}^{-1}\theta_0 = \boldsymbol{Q}_0\boldsymbol{Q}_0^{-1}\theta_0\,. \tag{E.36}$$

Note that if a negative λ_i exists, the above derivations can still be used to characterize the convergent sub-component of θ_0. The non-convergent sub-component of θ_0 will diverge with an exponential rate as shown above.

□

E.1.3 Proof of Theorem 19

Proof. We first prove the update formula. Recall the FVI update,

$$\theta_t = \boldsymbol{M}^\dagger(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_{t-1})\,.$$

For $t = 1$ the result holds by definition. Suppose that

$$\theta_t = \boldsymbol{M}^\dagger\left(\sum_{i=0}^{t-2}(\gamma\boldsymbol{N}\boldsymbol{M})^i\boldsymbol{R} + (\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^{t-1}(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_0)\right) \tag{E.37}$$

We now prove the result for $t+1$ by induction.

$$\begin{aligned}
\theta_{t+1} &= \boldsymbol{M}^\dagger(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_t) && \text{(E.38)}\\
&= \boldsymbol{M}^\dagger\left(\boldsymbol{R} + \gamma\boldsymbol{N}\boldsymbol{M}^\dagger\left(\sum_{i=0}^{t-2}(\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^i\boldsymbol{R} + (\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^{t-1}(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_0)\right)\right) && \text{(E.39)}\\
&= \boldsymbol{M}^\dagger\left(\boldsymbol{R} + \left(\sum_{i=1}^{t-1}(\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^i\boldsymbol{R} + (\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^{t}(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_0)\right)\right) && \text{(E.40)}\\
&= \boldsymbol{M}^\dagger\left(\sum_{i=0}^{t-1}(\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^i\boldsymbol{R} + (\gamma\boldsymbol{N}\boldsymbol{M}^\dagger)^{t}(\boldsymbol{R} + \gamma\boldsymbol{N}\theta_0)\right) && \text{(E.41)}
\end{aligned}$$

Clearly the convergence of this algorithm depends on the spectral norm of $\boldsymbol{N}\boldsymbol{M}^\dagger$. In particular, given that $\|\boldsymbol{N}\boldsymbol{M}^\dagger\| < 1/\gamma$, we have the algorithm converges to

$$\boldsymbol{M}^\dagger(\boldsymbol{I}_k - \gamma\boldsymbol{W})^{-1}\boldsymbol{R} \tag{E.42}$$

as $t \to \infty$. This finishes the proof. □

E.1.4 Proof of Theorem 20 and Corollary 9

Proof. We first prove the result for residual minimization fixed point θ_{RM}. The proof is adopted from characterizing the minimum norm solution of solving least square (Boyd & Vandenberghe, 2004). Let $\boldsymbol{A} = \boldsymbol{M} - \gamma \boldsymbol{N}$ for simplicity. We write the Lagrange of the optimization problem,

$$\mathcal{L}(\theta, \alpha) = \inf_{\theta \in \mathbb{R}^d} \sup_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|\theta\|^2 + \alpha^\top (\boldsymbol{R} - \boldsymbol{A}\theta) \tag{E.43}$$

$$= \sup_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|\boldsymbol{A}^\top \alpha\|^2 + \alpha^\top \boldsymbol{R} - \alpha^\top \boldsymbol{A}\boldsymbol{A}^\top \alpha \tag{E.44}$$

$$= \sup_{\alpha \in \mathbb{R}^k} \alpha^\top \boldsymbol{R} - \frac{1}{2}\alpha^\top \boldsymbol{A}\boldsymbol{A}^\top \alpha \,. \tag{E.45}$$

Solving for α^* and add it to $\theta^* = \boldsymbol{A}^\top \alpha^*$ gives that $\theta^* = \boldsymbol{A}^\dagger \boldsymbol{R}$.

We next prove Corollary 9, which characterizes the TD and FVI fixed point θ_{TD}. Let $\boldsymbol{W} = \boldsymbol{N}\boldsymbol{M}^\dagger$. We write the Lagrange of the optimization problem,

$$\mathcal{L}(\theta, \alpha) = \inf_{\theta \in \mathbb{R}^d} \sup_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|\theta\|^2 + \alpha^\top (\boldsymbol{R} - (\boldsymbol{I}_k - \gamma \boldsymbol{W})\boldsymbol{M}\theta) \tag{E.46}$$

$$= \sup_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|\boldsymbol{M}^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})^\top \alpha\|^2 + \alpha^\top \boldsymbol{R} - \alpha^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})\boldsymbol{M}\boldsymbol{M}^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})^\top \alpha \tag{E.47}$$

$$= \sup_{\alpha \in \mathbb{R}^k} \alpha^\top \boldsymbol{R} - \frac{1}{2}\alpha^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})\boldsymbol{M}\boldsymbol{M}^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})^\top \alpha \,. \tag{E.48}$$

Solving for α^* and add it to $\theta^* = \boldsymbol{M}^\top (\boldsymbol{I}_k - \gamma \boldsymbol{W})^\top \alpha^*$ gives that $\theta^* = \boldsymbol{M}^\dagger (\boldsymbol{I}_k - \gamma \boldsymbol{W})^{-1} \boldsymbol{R}$. The second part of Theorem 20 is immediately followed by this.

□

E.1.5 Proof of Theorem 21

Lemma 18. *Let θ_t be the output of FVI at iteration t with θ_0 as the initial parameter. We have that $\boldsymbol{M}^\dagger \boldsymbol{M} \theta_t = \theta_t$ for any $t \geq 1$.*

Proof. The claim is implied by the fact that θ_t is in the row space of $\boldsymbol{M}$. In particular, by Theorem 19, $\theta_t = \boldsymbol{M}^\dagger \alpha$ for some $\alpha \in \mathbb{R}^n$. Thus,

$$\boldsymbol{M}^\dagger \boldsymbol{M} \theta_t = \boldsymbol{M}^\dagger \boldsymbol{M}\boldsymbol{M}^\dagger \alpha = \boldsymbol{M}^\dagger \alpha = \theta_t \,. \tag{E.49}$$

This finishes the proof. □

Lemma 19. *$\mathcal{E}(\theta)$ is 1-smoothness.*

Proof. Recall the prediction error of $\theta \in \mathbb{R}^d$

$$\mathcal{E}(\theta) = \frac{1}{2} \|\Phi\theta - v\|^2_{\boldsymbol{D}_\mu} = \frac{1}{2} \|\theta - \theta^*\|^2_\Sigma \,, \tag{E.50}$$

where $\Sigma = \Phi^\top \boldsymbol{D}_\mu \Phi$. The gradient of θ is $\mathcal{E}'(\theta) = \Sigma(\theta - \theta^*)$. Then

$$\|\mathcal{E}'(\theta_1) - \mathcal{E}'(\theta_2)\| = \|\Sigma(\theta_1 - \theta_2)\| \leq \lambda_{\max}(\Sigma) \|(\theta_1 - \theta_2)\| \leq \|\theta_1 - \theta_2\| \,, \tag{E.51}$$

where we use $\|\phi(s)\| \leq 1$ for all $s \in \mathcal{S}$ and $\lambda_{\max}(\Sigma) \leq \sum_s \mu(s) \lambda_{\max}(\phi(s)\phi(s)^\top) \leq 1$. □

Lemma 20. *Let $\varepsilon_{app} = \boldsymbol{N}\Pi^\perp_{\boldsymbol{M}}\theta^*$ and $\sigma_{stat} = \boldsymbol{H}(P - \hat{P})\Phi\theta^*$. We have*

$$\boldsymbol{M}\theta^* = \boldsymbol{R} + \gamma(\varepsilon_{app} + \varepsilon_{stat}) + \gamma \boldsymbol{W}\boldsymbol{M}\theta^* \,. \tag{E.52}$$

Proof. Using the definitions we have,

$$\begin{aligned}
\boldsymbol{M}\theta^* &= \boldsymbol{R} + \gamma \boldsymbol{H} P \Phi\theta^* && \text{(E.53)} \\
&= \boldsymbol{R} + \gamma \boldsymbol{N}\theta^* + \gamma \boldsymbol{H}(P - \hat{P})\Phi\theta^* && \text{(E.54)} \\
&= \boldsymbol{R} + \gamma(\boldsymbol{W}\boldsymbol{M} + \boldsymbol{N}\Pi^\perp_{\boldsymbol{M}})\theta^* + \gamma \boldsymbol{H}(P - \hat{P})\Phi\theta^* && \text{(E.55)} \\
&= \boldsymbol{R} + \gamma(\varepsilon_{\text{app}} + \varepsilon_{\text{stat}}) + \gamma \boldsymbol{W}\boldsymbol{M}\theta^* \,. && \text{(E.56)}
\end{aligned}$$

□

Proof. By Theorem 19, $\theta_t = \boldsymbol{M}^\dagger \mathcal{T}^{t-1}(\boldsymbol{R} + \gamma \boldsymbol{N}\theta_0)$ is the output of FVI at iteration t. By noting that $\mathcal{E}(\theta^*) = 0$ and Lemma 19, for any $\theta \in \mathbb{R}^d$.

$$\mathcal{E}(\theta) \leq \frac{1}{2} \|\theta - \theta^*\|^2 = \frac{1}{2} \left(\|\theta - \theta^*\|^2_{\boldsymbol{M}^\dagger \boldsymbol{M}} + \|\theta - \theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} \right) . \tag{E.57}$$

We first consider the second term. By Lemma 18,

$$\|\theta_t - \theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} = (\theta_t - \theta^*)^\top \left(\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M} \right) (\theta_t - \theta^*) = \|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} \,. \tag{E.58}$$

We now consider the first term. By Lemma 20,

$$\boldsymbol{M}(\theta^* - \theta_t) = \boldsymbol{M}\theta^* - \mathcal{T}^{t-1}(\boldsymbol{R} + \gamma \boldsymbol{N}\theta_0) \tag{E.59}$$

$$= \sum_{i=0}^{t-2} (\gamma \boldsymbol{W})^i \left(\boldsymbol{R} + \gamma(\varepsilon_{\text{app}} + \varepsilon_{\text{stat}}) \right) + (\gamma \boldsymbol{W})^{t-1} (\boldsymbol{M}\theta^*) - \sum_{i=0}^{t-2} (\gamma \boldsymbol{W})^i \boldsymbol{R} - (\gamma \boldsymbol{W})^{t-1} (\boldsymbol{R} + \gamma \boldsymbol{N}\theta_0) \tag{E.60}$$

$$= \gamma \left(\sum_{i=0}^{t-2} (\gamma \boldsymbol{W})^i \varepsilon_{\text{app}} + \sum_{i=0}^{t-1} (\gamma \boldsymbol{W})^i \varepsilon_{\text{stat}} + (\gamma \boldsymbol{W})^{t-1} \boldsymbol{N}(\theta^* - \theta) \right) \tag{E.61}$$

Let $\hat{\Sigma} = \boldsymbol{M}^\top \boldsymbol{D}_k \boldsymbol{M}$ be the empirical covariance matrix. Note that $\lambda_{\min}(\boldsymbol{M}^\top \boldsymbol{M})/k \geq \lambda_{\min}(\hat{\Sigma})$. To show this, let $\bar{\boldsymbol{D}} = \operatorname{diag}(\frac{1}{k}, \ldots, \frac{1}{k})$, and $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ be the SVD of $\boldsymbol{M}$. Then

$$\begin{aligned}
\lambda^+_{\min}(\boldsymbol{M}^\top \bar{\boldsymbol{D}} \boldsymbol{M}) &= \min_{\|x\|=1} x^\top \boldsymbol{M}^\top \bar{\boldsymbol{D}} \boldsymbol{M} x && \text{(E.62)} \\
&= \min_{\|\alpha\|=1} \alpha^\top \boldsymbol{S}\boldsymbol{U}^\top \bar{\boldsymbol{D}} \boldsymbol{U}\boldsymbol{S}\alpha && \text{(E.63)} \\
&\geq \min_{\|\alpha\|=1} \alpha^\top \boldsymbol{S}\boldsymbol{U}^\top \boldsymbol{D}_k \boldsymbol{U}\boldsymbol{S}\alpha && \text{(E.64)} \\
&= \lambda^+_{\min}(\boldsymbol{M}^\top \boldsymbol{D}_k \boldsymbol{M}) && \text{(E.65)}
\end{aligned}$$

where we replace $x = \boldsymbol{V}\alpha$ since $\boldsymbol{V}$ are orthonormal bases, and $\min_{i\in[k]} \hat{\mu}_i \leq 1/k$. Therefore,

$$\|\boldsymbol{M}^\dagger\| = 1/\sqrt{\lambda^+_{\min}(\boldsymbol{M}^\dagger \boldsymbol{M})} \leq 1/\sqrt{k\lambda^+_{\min}(\hat{\Sigma})}.$$

Combining the results above we have,

$$\begin{aligned}
\|\theta_t - \theta^*\|^2_{\boldsymbol{M}^\dagger \boldsymbol{M}} &= \left\|\boldsymbol{M}^\dagger \boldsymbol{M}(\theta_t - \theta^*)\right\|^2 && \text{(E.66)} \\
&\leq \|\boldsymbol{M}^\dagger\|^2 \left\|\boldsymbol{M}(\theta_t - \theta^*)\right\|^2 && \text{(E.67)} \\
&\leq \frac{\gamma}{k\lambda_{\min}(\hat{\Sigma})} \left\| \sum_{i=0}^{t-1} (\gamma \boldsymbol{W})^i (\varepsilon_{\text{app}} + \varepsilon_{\text{stat}}) + (\gamma \boldsymbol{W})^{t-1} \boldsymbol{N}(\theta^* - \theta) \right\|^2 && \text{(E.68)} \\
&\leq \frac{4\gamma}{k\lambda_{\min}(\hat{\Sigma})} \left((\varepsilon^2 + \sigma^2) \left\| \sum_{i=0}^{t-1} (\gamma \boldsymbol{W})^i \right\|^2 + \left\|(\gamma \boldsymbol{W})^{t-1}\right\|^2 \|\Phi\|^2 \|\theta_0 - \theta^*\|^2 \right) . && \text{(E.69)}
\end{aligned}$$

Combine this with Eq. (E.58) finishes the proof.

□

E.1.6 Proof of Corollary 10

Proof. Recall that in the proof of Theorem 21 we have

$$\|\theta_t - \theta^*\|^2_{\boldsymbol{M}^\dagger \boldsymbol{M}} \leq \frac{4\gamma}{k\lambda_{\min}(\hat{\Sigma})} \left((\varepsilon^2 + \sigma^2) \left\| \sum_{i=0}^{t-1} (\gamma \boldsymbol{W})^i \right\|^2 + \left\|(\gamma \boldsymbol{W})^{t-1}\right\|^2 \|\Phi\|^2 \|\theta_0 - \theta^*\|^2 \right) . \quad \text{(E.70)}$$

Given that $\|\boldsymbol{W}\| < 1$, we have for the fixed point θ_∞,

$$\|\theta_t - \theta^*\|^2_{\boldsymbol{M}^\dagger \boldsymbol{M}} \leq \frac{4\gamma(\varepsilon^2 + \sigma^2)}{k\lambda_{\min}(\hat{\Sigma})(1-\gamma)^2} \tag{E.71}$$

We first consider $\sigma^2 = \|\boldsymbol{H}(P - \hat{P})v\|^2$. By Hoeffding's inequality and a union bound we have with probability at least $1 - \delta$, for any $s \in \text{supp}(\boldsymbol{D})$,

$$\left|(\hat{P}_s - P_s)^\top v\right| \leq \frac{1}{1-\gamma}\sqrt{\frac{\log(|\mathcal{S}|/\delta)}{2n(s)}}\,. \tag{E.72}$$

Thus, let $n_{\min} = \min_{s:n(s)>0} n(s)$, we have

$$\frac{\sigma^2}{k} \leq \frac{\log(|\mathcal{S}|/\delta)}{2(1-\gamma)^2 n_{\min}}\,. \tag{E.73}$$

Now we consider $\varepsilon^2 = \|\boldsymbol{N}\Pi^\perp_{\boldsymbol{M}}\theta^*\|^2$. Since $\boldsymbol{N}\Pi^\perp_{\boldsymbol{M}}$ is perpendicular to $\boldsymbol{M}$, and all features have norm bounded by one,

$$\frac{\varepsilon^2}{k} \leq \|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}}\,. \tag{E.74}$$

Combine the above we have,

$$\begin{aligned}
\mathcal{E}(\theta) &\leq \frac{1}{2}\|\theta - \theta^*\|^2_{\boldsymbol{M}^\dagger \boldsymbol{M}} + \frac{1}{2}\|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} && \text{(E.75)}\\
&\leq \frac{2\gamma}{\lambda_{\min}(\hat{\Sigma})(1-\gamma)^2}\left(\frac{\log(|\mathcal{S}|/\delta)}{2(1-\gamma)^2 n_{\min}} + \|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}}\right) + \frac{1}{2}\|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} && \text{(E.76)}\\
&= \frac{\gamma \log(|\mathcal{S}|/\delta)}{\lambda_{\min}(\hat{\Sigma})(1-\gamma)^4 n_{\min}} + \frac{4\gamma}{\lambda_{\min}(\hat{\Sigma})(1-\gamma)^2}\|\theta^*\|^2_{\boldsymbol{I}_d - \boldsymbol{M}^\dagger \boldsymbol{M}} && \text{(E.77)}
\end{aligned}$$

Finally, using the tower rule gives the desired result.

□

E.1.7 Concentration of Eigenvalues and Bounding the Orthogonal Complement

We will need the following result (**?**, Theorem 6), which is concerned with the magnitude of projection onto the eigenspace of a covariance matrix. The result is based on (**?**, Theorem 1)

Lemma 21. *Let $\hat{\Sigma} = \frac{1}{n}\sum_i x_i x_i^\top$ be the covariance matrix of i.i.d. data $x_i \in \mathbb{R}^d$. Denote the h-"tail" of eigenvalues of a covariance matrix $\hat{\Sigma} =$ as*

$$\lambda^{>h} = \sum_{i=h+1}^{n} \lambda_i \,. \tag{E.78}$$

Let $\boldsymbol{U}_r$ be the first r eigenbasis for $r \in [n]$. Then for any $z \in \mathbb{R}^d$, with probability at least $1-\delta$,

$$\mathbb{E}\left[\|\Pi^{\perp}_{\boldsymbol{U}_r} z\|_2^2\right] \leq \min_{h\in[r]} \left\{ \frac{1}{n}\lambda^{>h} + \frac{1+\sqrt{h}}{\sqrt{n}} \sqrt{\frac{2}{n}\sum_{i=1}^{n} \|x_i\|^2} \right\} + \|z\|_2^2 \sqrt{\frac{18}{n} \ln\left(\frac{2n}{\delta}\right)} \,. \tag{E.79}$$

The next lemma gives a non-asymptotic result to understand the behaviour of $\hat{\lambda}_{\min}$ (**?**, Lemma 1).

Lemma 22. *Let $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n] \in \mathbb{R}^{d\times n}$ be a random matrix with i.i.d. columns, such that $\max_i \|\boldsymbol{X}_i\|_2 \leq K$, and let $\hat{\Sigma} = \boldsymbol{X}\boldsymbol{X}^\top/n$, and $\Sigma = \mathbb{E}[\boldsymbol{X}_1 \boldsymbol{X}_1^\top]$. Then, for every $\alpha \geq 0$, with probability at least $1 - 2e^{-\alpha}$, we have*

$$\lambda^{+}_{\min}(\hat{\Sigma}) \geq \lambda^{+}_{\min}(\Sigma) \left(1 - K^2 \left(c\sqrt{\frac{d}{n}} + \sqrt{\frac{\alpha}{n}}\right)\right)_{+}^{2} \quad \text{for } n \geq d \,, \tag{E.80}$$

and furthermore, assuming that $\|\boldsymbol{X}_i\|_{\Sigma^\dagger} = \sqrt{d}$, for all $i \in [n]$, we have

$$\lambda^{+}_{\min}(\hat{\Sigma}) \geq \lambda^{+}_{\min}(\Sigma) \left(\sqrt{\frac{d}{n}} - K^2 \left(c + 6\sqrt{\frac{\alpha}{n}}\right)\right)_{+}^{2} \quad \text{for } n < d \,, \tag{E.81}$$

where we have an absolute constant $c = 2^{3.5}\sqrt{\ln 9}$.

E.2 Experiment setup

In this section, we provide additional details about the experimental setup and hyper-parameters used for each of the environments. For all of these environments the regularization weights were considered as tunable hyper-parameters. In addition, for R_ϕ (see Eq 8.23), the scale factor β was also considered as a parameter to be tuned in order to approximate the feature matrix norm. We use Google Vizier to automatically tune the hyper parameters (Golovin et al., 2017).

E.2.1 Acrobot

- Replay buffer with 10k tuples sampled using a random policy across trajectories with maximum episode length of 64.
- A DQN with hidden units consisting of fully connected layers with $(100, 100)$ units.
- Batch size 64.
- Learning rate of 1e-3.
- Regularized RM with weight of 2e-2 on $\mathcal{R}_\phi$ and 1e-4 on $\mathcal{R}_w$.
- Regularized TD with weight of 0 on $\mathcal{R}_\phi$ and 1e-4 on $\mathcal{R}_w$.

E.2.2 Reacher

- Replay buffer with 10k tuples sampled from a random policy across trajectories with maximum steps per episode of length 50.
- Learning rate 1e-4.
- A value network for the continuous action inputs with a fc observation layer with params (50,), a fc action layer with params (50,) and a joint fc layer with params (100,).
- Batch size 64.
- Gradient clipping with a norm of 10.0
- Regularized RM with weight of 0.15 on $\mathcal{R}_w$ and 0 on $\mathcal{R}_\phi$.
- Regularized TD with weight of 2e-2 on $\mathcal{R}_w$ and 7e-3 on $\mathcal{R}_\phi$.

E.2.3 Cartpole

- Replay buffer with 10k tuples sampled using a random policy across trajectories with maximum steps per episode of length 50.

- A DQN with hidden units consisting of fully connected layers with $(100, 100)$ units.
- Batch size 64.
- Learning rate 1e-3.
- Regularized RM with weight of 0.29 on $\mathcal{R}_w$ and 0 on $\mathcal{R}_\phi$.
- Regularized TD with weight of 1.5e-3 on $\mathcal{R}_w$ and 5e-3 on $\mathcal{R}_\phi$.

E.2.4 Pendulum

- Replay buffer with 1k tuples obtained by sampling directly from a fixed initial state distribution using a random policy.
- A value network for the continuous action inputs with a fc observation layer with params (50,), a fc action layer with params (50,) and a joint fc layer with params (100,).
- Batch size 64.
- Learning rate 1e-3.
- Regularized RM with weight of 1.0 on $\mathcal{R}_w$ and 5.4e-4 on $\mathcal{R}_\phi$.
- Regularized TD with weight of 0 on $\mathcal{R}_w$ and 1.0 on $\mathcal{R}_\phi$.

E.2.5 Mujoco

- The Q-function is approximated by two hidden layer fully neural networks, where the hidden layer size is 256.
- Batch size 256.
- Learning rate 3e-4.
- Regularized TD with weight of 0 on $\mathcal{R}_w$ and 1.0 on $\mathcal{R}_\phi$.