POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

EXPLOITING STRUCTURE FOR TRANSFER IN REINFORCEMENT LEARNING

Doctoral Dissertation of:
Andrea Tirinzoni

Supervisor:
Prof. Marcello Restelli

Tutor:
Prof. Nicola Gatti

The Chair of the Doctoral Program:
Prof. Barbara Pernici

2020 – XXXIII

Acknowledgements

First of all, I would like to thank my advisor Marcello Restelli for making me a researcher. I am grateful for his constant guidance throughout my PhD studies, for the many opportunities that he offered to me, and for really teaching more than I thought I could learn. The passion for doing research that he transmitted to me is probably the most valuable skill that I learned in these years.

Special thanks go to Alessandro Lazaric, who has been a second advisor to me. In particular, I would like to thank Alessandro for letting me collaborate with him on many amazing research problems, for the many insightful discussions that we had, and for giving me the opportunity to do an internship at Facebook.

I would like to thank Professor Emma Brunskill and Professor Emilie Kaufmann for accepting to review this thesis. Besides being honored to have my thesis reviewed by two of the most renowned RL researchers world-wide, I am really grateful for their hard work (especially given that this document is admittedly very long) and for their precious feedback.

My gratitude also goes to Matteo Pirotta, who became a friend as well as a colleague. Starting from the first paper that he co-authored at the beginning of my PhD, I really learned a lot from Matteo. I thank him for being always available to answer my questions, to help me when I needed, and for the many discussions that we had.

Special thanks go to my office mates Matteo, Alberto, and Giorgia with whom I spent the majority of time throughout this PhD. I thank them for the many experiences that we shared, including conferences, schools, meals,

co-authored papers, etc. I am really proud that I had the opportunity to work with such amazing people and I believe that sharing my PhD journey with them really made me a better person and researcher.

Of course, I did not forget about all the other colleagues at Politecnico and at the AirLab who were part of my everyday life during the last years. I thank (in alphabetic order) Alberto, the two Alessandro, Alessio, Amarildo, Andrea, Carlo, Davide, Francesco, Giulia, Giuseppe, Lorenzo, Luca, Matteo, Mattia, Mirco, Nico, and Simone. I must also thank all the exceptional master students that I supervised in the last years. In particular, I thank Andrea, Rafael, Mattia, and Riccardo, with whom I co-authored some of the works presented in this thesis.

Moving beyond my university life, I thank all my friends for the few moments that we shared in these years. Sadly, due to distance and busy work, we have not been able to see each other much in the last years. However, I feel extremely lucky to have people that I know since I was born and that are always there for me.

I am infinitely grateful to my family, whose constant support and help are really invaluable to me. In particular, I thank my parents, Tiziana and Stefano, for sharing my successes, my difficulties, and for always believing in me. I believe it is mostly thanks to them if I am writing this thesis.

Finally, my biggest and sincere thanks go to Giorgia for literally sharing our entire lives in the last two years. I am grateful for her continuous support and motivation, for being always present (and tolerant) in all my difficult moments, for teaching me to never give up and always fight to overcome our limits, and of course for her feedback on this whole thesis. Meeting Giorgia is certainly the best thing that happened to me during this PhD.

Abstract

Recent advancements have allowed reinforcement learning algorithms to achieve outstanding results in a variety of complex sequential decision-making problems, from playing board and video games to the control of sophisticated robotic systems. However, current techniques are still very inefficient, in the sense that they require a huge amount of experience before learning near-optimal behavior. One solution to mitigate this limitation is knowledge transfer, i.e., the process of reusing experience obtained while facing previous tasks to speed-up the learning process of new related problems. In this thesis, we offer a number of contributions to the field of transfer in reinforcement learning, from practical to theoretical aspects. We do so in the context of *structured domains*, a concept that we introduce to model problems with similarities that enable knowledge transfer. We start by studying how to reuse old experience from a set of *source tasks* to reduce the sample complexity for learning a *target task*. For this problem, we derive two novel algorithms for batch and online settings, respectively. We then study the problem of generating new experience, i.e., of *exploration* in the target task given knowledge from previous tasks. We first design a *practical* algorithm that explores the target task driven by a prior distribution over its solution that is learned from the source tasks. We then study this problem from a theoretical perspective under the assumption that the underlying task structure, or an approximation of it, is known. For both multi-armed bandits and Markov decision processes, we design different algorithms for which we formally establish the *benefits* of exploiting structure, while ensuring *optimality* in specific cases. All together, these results advance our understanding of knowledge transfer, one of the key components towards the deployment of reinforcement learning agents to the real world.

Contents

Mathematical Notation

I have tried to keep the notation close to the one adopted in the literature, while being as consistent as possible throughout the thesis. Unfortunately, staying close to the literature required different notation for similar problems (e.g., bandits vs MDPs). Here I report a quick summary.

General notation. For a measurable space $(\Omega, \mathcal{F})$, we denote by $\mathcal{P}(\Omega)$ the set of probability measures over Ω and by $\mathcal{B}(\Omega, b)$ the space of measurable functions over Ω bounded by $0 < b < \infty$. That is, $\forall f \in \mathcal{B}(\Omega, b), \forall x \in \Omega,\ |f(x)| \leq b$. Given a probability measure ν, the ℓ_p-norm of a measurable function f is $\|f\|_{p,\mu} = \left(\int |f|^p d\mu\right)^{1/p}$. Let $\mathcal{D}_n$ be a sequence $(x_1, \dots, x_n)$ with values in Ω. The empirical norm of f is $\|f\|^p_{p,\mathcal{D}_n} := \frac{1}{n}\sum_{i=1}^n |f(x_i)|^p$. We consider the ℓ_2-norm whenever omitting the subscript p.

General notation	
$\mathcal{P}(\Omega)$	Set of probability measures over set Ω
$\mathcal{B}(\Omega)$	Set of bounded measurable functions over Ω
$\mathcal{B}(\Omega, b)$	Set of measurable functions over Ω bounded by $b > 0$
$\|x\|_p^p = \sum_{i=1}^d |x_i|^p$	l_p-norm of a d-dimensional vector $x \in \mathbb{R}^d$
$\|f\|_{p,\nu}^p = \int \|f\|^p \mathrm{d}\nu$	Weighted l_p-norm of a function f
$\|f\|^p_{p,\mathcal{D}_n} := \frac{1}{n}\sum_{i=1}^n \|f(x_i)\|^p$	Empirical norm of f on dataset $\mathcal{D}_n = \{x_i\}_{i=1}^n$
Divergences	
D_{KL}	Kullback-Leibler (KL) divergence
D_{TV}	Total variation
d_2	Exponentiated Renyi divergence

MDPs and bandits. The table below summarizes the main symbols, though we shall introduce more specific terms in each chapter. Moreover, some direct variants of these symbols will be used. For instance, we shall consider settings with multiple MDPs or

parametrized models, for which the corresponding symbols below will have an MDP-index/parameter subscript. Similarly, the notation for structured bandits will be overloaded in contextual problems (Chapter 10) by including the specific context dependencies.

Structured Domains	
$\mathfrak{E} = (\mathfrak{M}, \mathfrak{D})$	Structured domain (Chapter 3)
$\mathfrak{E}_\Theta = (\Theta, \mathfrak{M}_\Theta, \mathfrak{D})$	Parametrized structured domain (Chapter 8)
$\mathfrak{M}$ (or $\mathfrak{M}_\Theta$)	Set of realizable tasks (MDPs or bandits)
$\mathfrak{D}$	Task generation process
Θ	Set of task parameters (only for parameterized domains)
Markov decision processes	
$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, U, \rho, \gamma)$	MDP model
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S} \times \mathbb{R})$	Joint transition kernel
$P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$	Marginal state-transition kernel
$U : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathbb{R})$	Marginal reward kernel
$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Mean-reward function
$\rho \in \mathcal{P}(\mathcal{S})$	Initial state distribution
$\gamma \in [0, 1]$	Discount factor
$r_{\max} > 0$	Bound on the absolute rewards
$V^\pi : \mathcal{S} \to \mathbb{R}$	Value function of policy π
$V^\star : \mathcal{S} \to \mathbb{R}$	Optimal value function
$Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Action-value function of policy π
$Q^\star : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Optimal action-value function
$T^\pi : \mathcal{S} \to \mathbb{R}$	Bellman operator for policy π
$T^\star : \mathcal{S} \to \mathbb{R}$	Bellman optimality operator
$J(\pi)$	Expected return of policy π
Structured Multi-armed bandits	
$\mathcal{M}_\theta = (\mathcal{A}, \mu_\theta)$	Structured bandit model
$\mathcal{A}$	Finite set of arms
K	Number of arms
Θ	Set of possible parameters
$\mathfrak{M}_\Theta = \{\{\mu_\theta(a)\}_{a \in \mathcal{A}} \mid \theta \in \Theta\}$	Hypothesis space
$\nu_\theta(a)$	Reward distribution of arm a
$\mu_\theta(a)$	Mean reward of arm a
$\mu_\theta^\star = \max_{a \in \mathcal{A}} \mu_\theta(a)$	Optimal reward
$a_\theta^\star = \operatorname{argmax}_{a \in \mathcal{A}} \mu_\theta(a)$	Optimal arm
$R_n^\pi(\theta, \Theta)$	Expected regret over n steps of algorithm π in bandit θ with structure $(\Theta, \mathfrak{M}_\Theta)$
$\Delta_\theta(a) = \mu_\theta^\star - \mu_\theta(a)$	Sub-optimality gap of arm $a \in \mathcal{A}$

CHAPTER 1

Introduction

Suppose you are about to learn how to drive a car for the first time. Unfortunately, neither your parents nor an older sibling were so kind to teach you how to do that and left you alone in the car. You must figure out how to drive it by yourself. Imagine you have driven another motor vehicle before, like a motorbike, so that you already know the pattern: you have to start the car, then accelerate to make it move, steer, break, and so on. You push random buttons until you eventually manage to start the car. Then, you notice there are two pedals and try to push one. Fortunately, it is the gas pedal. Sadly, the car starts moving backwards. You see a wall behind you and realize the situation is not looking good. So you immediately release the accelerator and push the other pedal; the car abruptly stops. You now realize that the first pedal you pushed did not work as you expected: it should have moved the car forwards, not backwards. You must have messed up with some buttons while you were trying to start the car. You carry on this trial-and-error procedure, pushing buttons and observing their effect, and eventually manage to switch the car back to forward mode, accelerate, and start driving. So far everything seems quite reasonable, right? Just an amateur trying to learn how to drive without supervision. However, if this was really the first time you were to drive a car, how did you know that turning on the ignition was the first thing to do? Why did you feel danger when the car was moving towards a wall? How did you know that the gas pedal was not behaving correctly? The answer might seem trivial: you have never driven a car before, but you have experienced other related situations, such as driving a motorbike; so you know that a motor vehicle must be started, that hitting a wall is bad, and that there must be some pedal to make the car move forwards. More generally, what you did is *knowledge transfer*, one of the fundamental features of

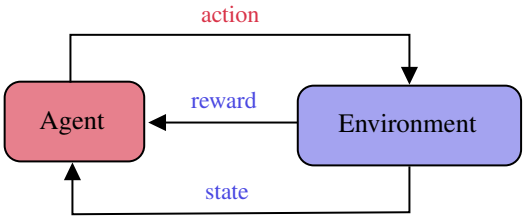

Figure 1.1: *The agent-environment interaction in reinforcement learning. Inspired by Figure 3.1 of Sutton and Barto (2018).*

human learning. That is, you extrapolated knowledge of the problem of driving beyond the specific vehicle and reused it for the specific task of driving cars. Without this, the only thing you could do is to press buttons and pedals randomly, which would require years of practice to control the car instead of a few hours, possibly with the vehicle crashing many times.

This "trivial" feature of human learning, the ability to reuse and generalize knowledge from a lifetime of experience, is all this thesis is about. Our main focus is to build artificial learning agents, framed into the reinforcement learning paradigm, that posses this capability. In particular, artificial agents, exactly as humans, should be able to *reuse knowledge* obtained in tasks faced throughout their "lives" to *quickly learn* new related problems, hence facilitating their deployment to the real world.

The Reinforcement Learning Paradigm

Reinforcement learning (Sutton and Barto, 2018) is a learning paradigm where the decision-maker, called the *agent*, interacts with the external world, called the *environment*, by taking actions in order to maximize a scalar *reward* signal. Differently from other learning paradigms, like supervised learning, the learner is not explicitly told which actions lead to high rewards, nor is it provided with examples of good actions. Instead, the agent has to figure out the optimal actions in each possible situation by sole interaction with the environment in a trial-and-error fashion. Each "situation" is described by a *state*, i.e., a set of variables that summarize the current configuration of the environment as perceived by the agent. The interaction works as sketched in Figure 1.1. At each decision step, the agent takes an action based on the currently-observed state of the environment. The environment, in response, produces a new state and a scalar reward. Both are communicated to the agent, but the process according to which they are generated is internal to the environment and unknown. This interaction is then repeated over and over, with the agent taking actions and the environment producing new states and rewards. Using only this feedback, the agent aims at maximizing the total reward collected over time. This is motivated by Sutton and Barto's reward hypothesis: *"that all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)"* (Sutton and Barto, 2018, Section 3.2).

The fact that the agent does not know how the environment responds to its actions poses several challenges which are peculiar of the reinforcement learning paradigm. The *exploration-exploitation* dilemma is perhaps the most important; on the one hand, the agent

is tempted to take those actions that are known to be good, e.g., because they performed best in the past. In such case, we say that the agent *exploits* the current knowledge. On the other hand, the agent should take less-played actions, for which poor information is available, in order to assess whether better decisions exist. In such case, we say that the agent *explores* actions to gain information. These two objectives, exploration and exploitation, are often very competing and how to trade them off is one of the long-standing problems in the reinforcement learning literature. The other key challenge regards *delayed rewards*. While it is quite simple to figure out whether some decision leads to high immediate reward, there might be actions with low immediate reward but that drive, possibly far into the future, the system to certain states where high rewards can be obtained. This also opens the *credit assignment* problem, i.e., the problem of understanding what decisions in the past led to a certain outcome in the present (such as a very high reward). How to deal with these problems (and others, as we discuss later) is a central part in the design of reinforcement learning agents.

Despite the simplicity of this paradigm, recent advances, in particular the combination with deep neural networks, have allowed reinforcement learning methods to achieve impressive results in a wide variety of complex tasks. These include beating the world champions at the game of Go (Silver et al., 2016), obtaining super-human performance in video-games from the Atari suite (Mnih et al., 2015; Badia et al., 2020) and in the game of Dota (Berner et al., 2019), and controlling sophisticated robotic systems (Kober and Peters, 2009; Levine et al., 2016), just to name a few.

Why Transfer in Reinforcement Learning?

While it was popularized first in the supervised learning literature (see, e.g., Pan and Yang (2009); Weiss et al. (2016); Tan et al. (2018); Da Silva and Costa (2019) and references therein), *knowledge transfer* has become a key ingredient in modern reinforcement learning (Taylor and Stone, 2009; Lazaric, 2012). This is mainly because of two reasons: a *problem* and an *opportunity*. The problem is that, despite the recent outstanding results, reinforcement learning algorithms are still very *sample inefficient*. For instance, although reinforcement learning agents have now mastered games in the Atari suite (Mnih et al., 2015), a recent empirical study (Tsividis et al., 2017) shows that they require hundreds of hours of experience to learn skills that a human would learn in a few minutes. This is because the artificial agent learns each task *from scratch*, while a human player, though he/she might have never played Atari before, is able to generalize and reuse the experience of his/her entire life to quickly learn new tasks. This is exactly what knowledge transfer aims at achieving in artificial agents:

> *Knowledge transfer aims at reducing the amount of experience needed for learning new tasks by reusing knowledge from previously-solved tasks.*

The key insight, arising from psychology (e.g., Woodworth and Thorndike, 1901), is that intelligent agents are able to generalize across different tasks rather than only within the same task. Let us go back to our car driving example and suppose that, after years of usage, you sell your old car to buy a new one. Consider driving it for the first time. It might take some time to perfectly master its control, yet anyone who has driven a car before is likely to decently drive the new one without any complication. This is because a human can

generalize the skill of "car driving" beyond the specific car, exactly as one can generalize "playing a video-game" beyond Atari.

So what is the opportunity? The *opportunity* is that reinforcement learning agents deployed to the real world are often required to face multiple different tasks. This might be, for instance, because the environment evolves over time due to natural phenomena or due to human intervention; or because the agent's desiderata (i.e., its goals) change. In our car driving example, an agent might be required to drive under different weather conditions (the environment changes naturally), or to drive different cars (the environment changes due to human intervention), or to drive for different purposes, such as taking a passenger to a desired location or racing with other cars (the agent's desiderata change). Therefore, there is indeed an opportunity for an artificial agent to have prior knowledge from different (related) tasks, thus making knowledge transfer a primary concern.

Main Research Problems

If we now go back to the goal of transfer in reinforcement learning, using the right terminology (Taylor and Stone, 2009; Lazaric, 2012), the aim is to reuse the knowledge gathered while facing a set of *source tasks* to simplify and speed up the learning process of a related *target task*. At this point, some questions might arise in the reader's mind. (1) *What knowledge should be transferred?* "Knowledge" is a general concept which does not specify what components of the past information have to be reused. (2) *How should knowledge transfer be performed?* Injecting prior knowledge into a learning process might be achieved in a multitude of different ways. (3) *When should knowledge be transferred?* In other words, is it always safe and beneficial to reuse previous knowledge or are there cases where it is better not to transfer? It turns out that an answer to these questions lies at the core of all transfer methods and one of the aims of this thesis is indeed to provide a better understanding on this matter.

The literature on transfer in reinforcement learning is vast (Taylor and Stone, 2009; Lazaric, 2012), as we shall throughly survey in Chapter 3. While a variety of approaches have been proposed with promising results, many open problems, concerning both practical and theoretical aspects, still exist. On the practical side, one of the long-standing problems is to build transfer methods that are not tied to a specific setting or learning algorithm, a property that does not hold for the majority of existing works. A related problem is that of building *scalable* approaches. Recent advances in deep neural networks have made progress on this matter by allowing methods with remarkable performance on complex tasks (Zhu et al., 2020). For instance, meta-learning approaches (e.g., Duan et al., 2016; Wang et al., 2016a; Finn et al., 2017) learn the reinforcement learning agent itself, and hence its capabilities to transfer knowledge across domains. These neural-network-based approaches often have at least two limitations. One is *interpretability*, i.e., it is often difficult to understand how and when knowledge is reused. The second is that, though there is empirical evidence of reduced sample complexity for learning new tasks, this gain is often reduced to a *"task complexity"*, i.e., the agent needs to face a large number of tasks before successfully generalizing to new ones.

On the theoretical side, only few existing works provide formal guarantees on the performance of the proposed methods (e.g., Mahmud et al., 2013; Zhan et al., 2016; Abel et al., 2018, just to name a few). This is a primary concern in our setting since, as we shall

see, careless knowledge transfer might even harm performance rather then improving it (the so-called *negative transfer* issue). More generally, while theoretical studies exist (e.g., Lazaric and Restelli, 2011; Brunskill and Li, 2013; Azar et al., 2013a), the literature still lacks a sufficient theoretical understanding of this problem. Several questions remain open, including: what are the fundamental limits of transfer in reinforcement learning? How can the agent extrapolate knowledge from solved tasks so as to ease its reuse in new problems? How should an agent interact with a new task when provided with prior knowledge from related problems?

Motivated by these open problems, in this thesis we offer a number of contributions to the field of knowledge transfer in reinforcement learning. The common pattern behind all the approaches we shall propose, and possibly behind the majority of transfer methods in the literature, is the concept of *structured domain* that we introduce in Chapter 3 and better discuss in Chapter 8. Informally, a structured domain is defined by a *collection of tasks* together with a *process* generating them. Tasks in the same domain possibly share some underlying *structural properties*, i.e., they have some similarities that enable knowledge transfer. An agent that faces multiple tasks from the same domain should be able to understand their hidden structure, so as to transfer this knowledge to improve the learning process of new problems. [1]

While framing everything in the context of structured domains, we study two complementary problems in the transfer literature:

> (P1) *How can the agent reuse experience samples collected while facing a set of source tasks to speed-up the learning process of a new target task?*
>
> (P2) *How should the agent generate new experience in the target task (i.e., how should it explore the new environment) given the available prior knowledge?*

We said that knowledge transfer aims at reducing the amount of experience needed for learning new tasks. These problems describe perhaps the two main techniques to achieve this: directly reuse experience from the source tasks to *augment* the one from the target, or use it to figure out how to *generate* better experience from the target itself. In particular, the first problem is motivated by real-world scenarios where we need to learn some target task with very *limited access* to the corresponding environment, while we have a considerable amount of *data* from related problems/environments at our disposal. For instance, we might be interested in building a self-driving car and possess navigation data from many different vehicles; or we might want to control a power plant and have access to the historical operations of other plants. In this context, one naturally wonders whether it is really necessary to learn the target task from scratch or whether it is possible to *reuse* the available data.

The second problem relates to the exploration-exploration dilemma lying at the core of all reinforcement learning agents. While the dilemma is well-studied for agents that learn tasks from *scratch*, how to properly explore a new environment when provided with *prior knowledge* remains one of the key open questions. In particular, optimal exploration strategies might significantly change under prior knowledge. For instance, in our car driving example, the agent does not need to explore the consequences of crashing the car to a

[1] Hence the title "exploiting structure for transfer in reinforcement learning".

wall since such situation is already known to be bad from past experience. On the contrary, an agent learning the task from scratch necessarily needs to experience such situation in order to label it as undesirable.

In this thesis, we study both practical and theoretical aspects related to these two problems. On the practical side, our aim is to build knowledge-transfer methods that can be applied to high-dimensional *continuous control* problems without imposing restrictive assumptions on the underlying reinforcement learning agents. On the theoretical side, our aim is to advance the understanding of *exploration under prior knowledge*. This prior knowledge might derive from exact knowledge of the underlying structured domain or from its estimation from previous tasks.

Outline of Contributions

The thesis is organized in three parts, each studying the problem of knowledge transfer in reinforcement learning from a different perspective. Before diving into our contributions, we provide the foundations of reinforcement learning in Chapter 2, while in Chapter 3 we introduce the knowledge transfer problem and summarize the current literature.

Part I. The first part deals with problem (P1) above. In particular, we design methods to transfer experience samples (i.e., states, actions, and rewards collected from the interaction with different environments) across tasks. We start by formalizing and motivating the problem in Chapter 4, while discussing its main challenges. We then propose algorithms for transferring experience samples in batch reinforcement learning (Chapter 5) and online policy search (Chapter 6). For both settings, we follow the same pattern: we transfer all the available samples, while weighting their contribution to the learning process using importance sampling to compensate for the distribution mismatch between source and target domains. This is in contrast to existing works, which either transfer samples directly without accounting for distribution mismatch (Taylor et al., 2008), make strong assumptions Laroche and Barlier (2017), or carry out an expensive sample selection process (Lazaric et al., 2008b). Since computing the importance weights requires knowledge of the unknown source/target environment models, we propose a sound technique to estimate these quantities by directly reducing the mean square error of the resulting estimators. In both settings, we provide theoretical insights one the performance of our methods and report good empirical performance on continuous control tasks.

Part II. In the second part, we begin studying problem (P2) above from a practical perspective. In Chapter 7, we use ideas from randomized value functions (Osband et al., 2014, 2019) to design an algorithm that estimates a distribution over the optimal value functions of tasks in the underlying structured domain and uses it as a prior to enable exploration via posterior sampling in new tasks. Since computing the corresponding posterior given target samples is intractable in most cases of practical interest, we propose an efficient variational approximation (Blei et al., 2017). Notably, the generality of our design allows our algorithm to be combined with complex value-function approximators (such as neural networks) and posterior distribution classes. We theoretically study the finite-sample properties of this approximation and report good results on increasingly-complex continuous domains.

Part III. In Part III we address (P2) from a theoretical perspective. In Chapter 8, we start by better formalizing the problem of transfer in structured domains. We use existing ideas (Brunskill and Li, 2013; Azar et al., 2013a) to decompose the problem in two sub-problems: (1) learning structure from previous tasks and (2) exploiting structure to quickly learn new tasks. We focus on the second sub-problem for the case where the structure is exactly known and for the one where it is only approximate (e.g., learned from experience). In Chapter 9, we consider a multi-armed bandit problem with known structure in a general form and design an arm-elimination strategy that exploits the given structure to quickly discard sub-optimal arms. Our theoretical results, differently from those of previous works (Azar et al., 2013a; Lattimore and Munos, 2014), clearly show the performance gain obtained by using prior knowledge while still certifying that our algorithm never performs worse than an unstructured baseline (UCB, Auer et al. (2002a)). In Chapter 10, we restrict our attention to structures where the rewards are linear functions of given features and unknown parameters. We design a computationally-efficient incremental algorithm that is asymptotically optimal (in a problem-dependent sense) for this specific structure, while providing finite-time guarantees on its performance. In Chapter 11, we consider the problem of best policy identification in Markov decision processes with approximate structure. Under the assumption that a generative model of the target task is accessible, we design an algorithm that actively demands samples from state-action pairs that yield high information for finding a near-optimal policy. We derive an upper bound to its sample complexity that certifies how the algorithm exploits the approximate structure while never resulting worse than baselines not using structure at all.

We conclude in Chapter 12 with a brief summary of our main results and with possible directions for future research.

List of Publications

This thesis is based on 6 first-author publications in the main machine learning venues. The list of these publications, together with the chapters of this document in which they are presented, is as follows. In all these works, the leading author contributed to the design of the algorithms, their theoretical analysis, the empirical evaluation, and the realization of the manuscript.

- Chapter 5. “Importance Weighted Transfer of Samples in Reinforcement Learning”. Co-authored with Andrea Sessa, Matteo Pirotta, and Marcello Restelli. Published at ICML 2018.

- Chapter 6. “Transfer of Samples in Policy Search via Multiple Importance Sampling”. Co-authored with Mattia Salvini and Marcello Restelli. Published at ICML 2019.

- Chapter 7. “Transfer of Value Functions via Variational Methods”. Co-authored with Rafael Rodriguez Sanchez and Marcello Restelli. Published at NeurIPS 2018.

- Chapter 9. “A Novel Confidence-Based Algorithm for Structured Bandits”. Co-authored with Alessandro Lazaric and Marcello Restelli. Published at AISTATS 2020.

- Chapter 10. “An Asymptotically Optimal Primal-Dual Incremental Algorithm for Linear Contextual Bandits” . Co-authored with Matteo Pirotta, Marcello Restelli, and Alessandro Lazaric. Published at NeurIPS 2020.
- Chapter 11. “Sequential Transfer in Reinforcement Learning with a Generative Model”. Co-authored with Riccardo Poiani and Marcello Restelli. Published at ICML 2020.

CHAPTER 2

Reinforcement Learning

In this chapter, we provide the foundations of reinforcement learning and discuss the main algorithms. Of course, there is not enough space to discuss such a vast literature, so we shall focus on the main topics that are relevant to this thesis. The presentation follows those of the most popular books on this topic (Szepesvári, 2010; Sutton and Barto, 2018). In Section 2.1, we start by describing Markov decision processes, the mathematical model for the agent-environment interaction described in the introduction. Then, we directly dive into reinforcement learning algorithms, from exact solutions (Section 2.3 to sample-based tabular (Section 2.4) and approximate (Section 2.5) methods. We conclude in Section 2.6 with an introduction of stochastic multi-armed bandits (Lattimore and Szepesvári, 2020), a class of simpler reinforcement learning problems which will be the focus of the final part of the thesis.

Markov Decision Processes

Markov decision processes (MDPs) are the core mathematical model of the sequential decision-making problems introduced in the previous chapter. Here we provide their explicit formulation together with the basic results on top of which most reinforcement learning algorithms are built. We shall use standard notation and concepts from the main books on this topic (Puterman, 2014; Bertsekas, 2001; Sutton and Barto, 2018; Szepesvári, 2010).

Definition 2.1.1 (Markov decision process (MDP))**.** *A Markov decision process is a tuple* $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, \rho, \gamma)$*, where*

- $\mathcal{S}$ *is a measurable* state space*;*
- $\mathcal{A}$ *is a measurable* action space*;*[1]
- $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S} \times \mathbb{R})$ *is the* transition probability kernel. *When evaluated at* $(s, a) \in \mathcal{S} \times \mathcal{A}$*, the mapping* P *yields a probability measure over the possible next state and reward;*[2]
- $\rho \in \mathcal{P}(\mathcal{S})$ *is the* initial state distribution *which provides the probability of the system starting in any state;*
- $\gamma \in [0, 1]$ *is the discount factor.*

The state and action spaces can be either finite or infinite (see also Chapter 2 of Puterman (2014)). In the first case, we say that the MDP is *finite*. In the second case, the most common assumption involves compact subsets of some Euclidean space, and we say that the MDP is *continuous*. We shall work under the latter assumption in the first two parts of this thesis. Note, however, that most of our results hold for more general Borel-measurable spaces. Occasionally, especially when discussing existing literature, we use notation and results for finite MDPs, which will be stated explicitly unless clear from the context.

MDPs model discrete-time stochastic processes that are partially controlled by the agent. In fact, the agent is allowed to take actions, while the evolution of states and rewards, as a function of the chosen actions, is governed by the environment. The process starts at time $t = 0$, where the initial state $S_0 \sim \rho$ is drawn from the initial state distribution ρ. After observing S_0, the agent takes the first action A_0. Then, the next state and reward $(S_1, R_1) \sim P(\cdot|S_0, A_0)$ are drawn from the transition kernel P, and the process is repeated. The sequence of states, actions, and rewards $H_t := (S_0, A_0, S_1, R_1, \ldots, S_t, R_t)$ is called t-step *history*. The set of such histories is denoted by $\mathcal{H}_t$. Note that, in an MDP, the distribution of the next state and reward is fully characterized given the current state and action, and it does not depend on any variable before the current time step. This is referred to as the *Markov property* (hence the name "Markov decision process").

Given a sequence of states, actions, and rewards as above, we can define the *discounted return* up to some horizon T as the random variable

$$G_T := \sum_{t=0}^{T-1} \gamma^t R_{t+1}. \tag{2.1}$$

The decision-maker is concerned with maximizing this variable through the choice of its actions, as we shall discuss in more details later. The *discount factor* γ gives more importance to rewards obtained in earlier steps as opposed to later ones. More precisely, the contribution of each single reward to the discounted return decays exponentially with t if $\gamma \in (0, 1)$ and remains constant if $\gamma = 1$. The *horizon* T can be either *finite* ($T < \infty$), in which case the MDP is called *finite-horizon*, or infinite ($T = \infty$), and the MDP is called *infinite-horizon*. There exists a popular class of MDPs called *episodic* in which each sequence of states eventually ends up in a *terminal* or *absorbing* state, i.e., a state from

[1] According to the most general formulation (Puterman, 2014), the space of available actions might depend on the current state. For the sake of simplicity, we use the standard formulation with a fixed action space.

[2] We recall that $\mathcal{P}(\Omega)$ denotes the set of probability measures over a generic set Ω.

which the system cannot escape and where the reward is always zero. More formally, if the agent is in state S_t at time t and S_t is terminal, then $R_{t+u} = 0$ and $S_{t+u} = S_t$ for all $u \in \{1, 2, \dots\}$. Here the term *episode* (or *trajectory*) refers to a sequence of states, actions, and rewards terminating into an absorbing state. Let $\bar{\mathcal{S}} \subset \mathcal{S}$ be the subset of absorbing states. Then, with some abuse of notation, we can define the random variable

$$T := \min_{t \geq 0} \{t | S_t \in \bar{\mathcal{S}}\}, \tag{2.2}$$

namely the first time step in which the agent ends up in an absorbing state. This acts as an effective horizon in episodic MDPs, such that $G_\infty = G_T$. In this thesis, we focus on infinite horizon MDPs and, in some cases, we restrict our attention to episodic ones. In the first case, we require $\gamma \in [0, 1)$ to make sure the discounted return is well-defined.

The *transition kernel* plays a fundamental role in this framework as it directly governs the system dynamics (i.e., the temporal evolution of the process). From the joint state-reward formulation of Definition 2.1.1, we can immediately extract several useful components. First, by marginalizing over rewards we obtain the *state-transition probability kernel*,

$$P(\cdot | s, a) := \int_{\mathbb{R}} P(\cdot, \mathrm{d}x | s, a), \tag{2.3}$$

where, with some abuse of notation, we use the same symbol as the full transition kernel. The distinction is clear from the missing reward argument. Similarly, the other marginal yields the *reward kernel*,

$$U(\cdot | s, a) := \int_{\mathcal{S}} P(\mathrm{d}s', \cdot | s, a). \tag{2.4}$$

The immediate expected reward received after taking action a in state s is

$$r(s, a) := \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right] = \int_{\mathbb{R}} U(\mathrm{d}x | s, a) x. \tag{2.5}$$

We shall consider the following standard assumption on the boundedness of rewards.

Assumption 2.1.1. *The random rewards are bounded almost surely, i.e., there exists a constant $r_{\max} > 0$ such that $|R_{t+1}| \leq r_{\max}$ holds with probability one for each $t \in \{0, 1, 2, \dots\}$.*

An immediate implication is that $\|r\|_\infty = \sup_{s \in \mathcal{S}, a \in \mathcal{A}} |r(s, a)| \leq r_{\max}$ also holds.

Policies and Decision Rules

The behavior of the agent (i.e., how actions are chosen) is specified by means of a *control policy* (or just policy), i.e., a set of decision rules specifying what actions to take at each time steps. Formally, a *decision rule* at time t is a mapping $\pi_t : \mathcal{H}_t \to \mathcal{P}(\mathcal{A})$ that, given a t-step history, produces a probability measure over actions. A decision rule can be *history-dependent*, if the action distribution depends on the history up to time t, or *Markov*, if it only depends on the last observed state. Similarly, a decision rule can be *deterministic*, if

it assigns probability mass to a single action in $\mathcal{A}$, or *stochastic* otherwise. A policy π is simply a collection of decision rules, one for each time step, $\pi := \{\pi_t\}_{t\geq 0}$. Depending on the type of its decision rules, a policy inherits the same classification (history-dependent vs Markov and deterministic vs stochastic). Furthermore, a policy is *stationary* if its decision rules are all equivalent, in which case we write π to directly denote the mapping from histories/states to actions. We say that the agent follows or executes policy π when its actions are chosen according to π, i.e., $A_t \sim \pi(\cdot|H_t)$. In this thesis, we only consider stationary Markov policies. When deterministic, with some abuse of notation, we write $\pi(s)$ to denote the action prescribed in state $s \in \mathcal{S}$. Finally, we use Π^{SD} (Π^{SR}) to denote the set of all stationary Markov deterministic (randomized) policies.

Value Functions and Optimality

Value functions play a fundamental role in solving MDPs. Specifically, they provide the expected discounted return that the agent obtains when following a certain policy π starting from some state or state-action pair.

Definition 2.1.2 (Value function)**.** *Let π be a policy and $s \in \mathcal{S}$ be any state. The value function of π when starting from s is*

$$V^{\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \;\middle|\; S_0 = s\right], \tag{2.6}$$

where the expectation is taken under $A_t \sim \pi(\cdot|S_t)$ and $(R_{t+1}, S_{t+1}) \sim P(\cdot,\cdot|S_t, A_t)$ for all $t \geq 0$.

We say that the value function V^{π} provides an *evaluation* of policy π, in the sense that it provides the expected performance of an agent following π for any starting state. Similarly, we can define the action-value function as follows.

Definition 2.1.3 (Action-value function)**.** *Let π be a policy, $s \in \mathcal{S}$ be any state, and $a \in \mathcal{A}$ be any action. The action-value function of π when starting from s and taking a in the first step is*

$$Q^{\pi}(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \;\middle|\; S_0 = s, A_0 = a\right], \tag{2.7}$$

where the expectation is taken under $A_t \sim \pi(\cdot|S_t)$, for all $t \geq 1$, and $(R_{t+1}, S_{t+1}) \sim P(\cdot,\cdot|S_t, A_t)$ for $t \geq 0$.

Similarly to $V^{\pi}(s)$, $Q^{\pi}(s,a)$ provides the expected discounted reward of an agent that starts in state s, takes action a, and follows π thereafter. The action-value function is also referred to as Q-function. We shall use this alternative nomenclature later on.

Generally speaking, solving an MDP requires finding an *optimal policy* $\pi^{\star}$, i.e., a policy that maximizes the expected return at all states (if there exists one), so that

$$\forall s \in \mathcal{S} : V^{\pi^{\star}}(s) = \sup_{\pi \in \Pi^{\mathrm{SR}}} V^{\pi}(s).$$

To this purpose, it is convenient to define the *optimal value function* and the *optimal action-value function* respectively as

$$V^\star(s) := \sup_{\pi \in \Pi^{\mathrm{SR}}} V^\pi(s), \tag{2.8}$$

$$Q^\star(s,a) := \sup_{\pi \in \Pi^{\mathrm{SR}}} Q^\pi(s,a). \tag{2.9}$$

The suprema above are with respect to the general set of stationary stochastic policies, though it is known that deterministic ones actually suffice (Bertsekas, 2001; Puterman, 2014). In particular, acting greedily with respect to $Q^\star$ induces optimal behavior. More formally, we say that a deterministic policy π is *greedy* with respect to some action-value function Q if

$$\pi(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a)$$

holds for all $s \in \mathcal{S}$ (and under the assumption that the maximum is attained). Then, we have that $\pi^\star$ is greedy with respect to $Q^\star$.

While the one mentioned above is the standard optimality criterion for infinite-horizon discounted MDPs, there exists a weaker and widely-adopted notion of optimality for the specific case of *episodic* MDP, where each episode/trajectory terminates almost surely. Instead of seeking a policy that maximizes the value function uniformly at all states, we seek one that maximizes the expected value function under the initial state distribution.

Definition 2.1.4 (Expected discounted return)**.** *The expected discounted return of a policy π is*

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t R_{t+1}\right], \tag{2.10}$$

where the expectation is taken with respect to $S_0 \sim \rho$, $A_t \sim \pi(\cdot|S_t)$, and $(R_{t+1}, S_{t+1}) \sim P(\cdot,\cdot|S_t, A_t)$ for $t \geq 0$. By the tower rule of expectation, we also have that $J(\pi) := \mathbb{E}_{S_0 \sim \rho}[V^\pi(S_0)]$.

Then, the optimal policy $\pi^\star$ according to the alternative optimality criterion is such that

$$J(\pi^\star) = \sup_{\pi \in \Pi^{\mathrm{SR}}} J(\pi).$$

In this thesis, we shall consider both optimality criteria and switch between them according to the specific context.

Bellman Equations and Operators

The Bellman equations (Bellman, 1954) constitute one of the fundamental results towards the solution of MDPs. We start from the equations for policy evaluation, which allow to find the value function of a given policy.

Proposition 2.1.1 (Bellman equations)**.** *Let π be any policy, then*

$$Q^\pi(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) V^\pi(s'), \tag{2.11}$$

$$V^\pi(s) = \int_{\mathcal{A}} \pi(\mathrm{d}a|s) Q^\pi(s,a). \tag{2.12}$$

Intuitively, the action-value function of π evaluated at (s,a) can be computed by summing the immediate expected reward obtained when executing a in s and the expected (discounted) value achieved by π in the next random state. Similarly, the value function of π evaluated at s is simply the expectation of $Q^\pi(s,a)$ when a is randomly chosen by π. These two equations can be rewritten in a more convenient form using the notion of Bellman operator.

Definition 2.1.5 (Bellman operators)**.** *Let π be any policy and $V \in \mathcal{B}(\mathcal{S})$ be any value function. The Bellman operator $T^\pi : \mathcal{B}(\mathcal{S}) \to \mathcal{B}(\mathcal{S})$ for the value function is defined as*[3]

$$(T^\pi V)(s) := \int_{\mathcal{A}} \pi(\mathrm{d}a|s) \left(r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) V(s') \right). \tag{2.13}$$

Similarly, for $Q \in \mathcal{B}(\mathcal{S} \times \mathcal{A})$ and with some abuse of notation, the Bellman operator $T^\pi : \mathcal{B}(\mathcal{S} \times \mathcal{A}) \to \mathcal{B}(\mathcal{S} \times \mathcal{A})$ for the action-value function is defined as

$$(T^\pi Q)(s,a) := r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) \int_{\mathcal{A}} \pi(\mathrm{d}a'|s') Q(s',a'). \tag{2.14}$$

Using these operators, it is easy to see that the Bellman equations of Proposition 2.1.1 can be rewritten as $T^\pi V^\pi = V^\pi$ and $T^\pi Q^\pi = Q^\pi$. In other words, the value function V^π and the action-value function Q^π are *fixed-points* of their respective operators. Note that the notation $T^\pi V$ can be interpreted as the function resulting from the application of T^π to V. One of the key properties of these operators is that, for $\gamma < 1$, they are γ-contractions with respect to the l_∞-norm. That is, if $V, V' : \mathcal{S} \to \mathbb{R}$ are any two value functions, then

$$\|T^\pi V - T^\pi V'\|_\infty \leq \gamma \|V - V'\|_\infty, \tag{2.15}$$

and the same holds for the action-value functions.

Analogously to the case of a fixed policy π, we can define the Bellman optimality equations and operators, which allow computing the optimal policy and value functions of an MDP.

Proposition 2.1.2 (Bellman optimality equations)**.** *The optimal value function $V^\star$ and action-value function $Q^\star$ satisfy*

$$Q^\star(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) V^\star(s'), \tag{2.16}$$

$$V^\star(s) = \sup_{a \in \mathcal{A}} Q^\star(s,a). \tag{2.17}$$

[3]Note that, in order to keep consistent notation with the literature, we use the symbol T to denote both Bellman operators and the learning horizon. The distinction is clear from the fact that the former ones always come with a superscript.

Definition 2.1.6 (Bellman optimality operators)**.** *Let $V \in \mathcal{B}(\mathcal{S})$ and $Q \in \mathcal{B}(\mathcal{S} \times \mathcal{A})$ be any value function and action-value function, respectively. The Bellman optimality operators $T^\star : \mathcal{B}(\mathcal{S}) \to \mathcal{B}(\mathcal{S})$ and $T^\star : \mathcal{B}(\mathcal{S} \times \mathcal{A}) \to \mathcal{B}(\mathcal{S} \times \mathcal{A})$ are respectively defined as*

$$(T^\star V)(s) = \sup_{a \in \mathcal{A}} \left(r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) V(s') \right), \tag{2.18}$$

$$(T^\star Q)(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) \sup_{a' \in \mathcal{A}} Q(s',a'). \tag{2.19}$$

As before, we have that $V^\star$ and $Q^\star$ are fixed-points of the corresponding operator, i.e., $V^\star = T^\star V^\star$ and $Q^\star = T^\star Q^\star$. Moreover, these operators are still γ-contractions with respect to the l_∞-norm.

Taxonomy of Reinforcement Learning Methods

As we already mentioned before, solving an MDP means finding an optimal (or approximately optimal) policy. When the MDP is fully known (i.e., all of its components are known), this problem is often referred to as *planning*. It is important to note that planning is *not* a learning problem since no experience/data is involved, while the focus is mostly on the computational aspect. The majority of planning algorithms fall under the class of *dynamic programming* methods (Bellman, 1954). We shall review some of them in the next section as they provide the basic ideas at the core of many reinforcement learning algorithms.

Differently from planning, *reinforcement learning* is concerned with learning an optimal policy when the MDP dynamics are not known a priori but it is possible to interact with the environment. In general, the agent knows the state and action spaces and the discount factor, but it does not have access to the (joint) transition probability kernel P and the initial state distribution ρ. In some cases, the immediate expected reward $r(s,a)$, or $r(s,a,s')$, might be known, yet the state-transition probabilities are always unknown. The possibility to *interact* with the environment means that the agent, at each step t, is in some S_t, takes an action A_t (e.g., by executing a policy), and observes its effect, i.e., a new state S_{t+1} and a reward R_{t+1}. Though the agent does not fully know the MDP model, it can use these *experience samples* to understand how the MDP works and, consequently, to compute an optimal policy.

In the introduction, we briefly mentioned some of the fundamental problems that are faced by a reinforcement learning agent, including the exploration-exploitation dilemma, delayed rewards, and credit assignment. There exist two additional challenges, which are more easily understood now that we provided the basics of MDPs. (1) *Approximation*: when dealing with MDPs with large or continuous state-action spaces, it might not be possible to exactly represent the quantities of interest (e.g., a policy or a value function). In this case, the agent has to rely on function approximation. The choice of the specific functional space plays a fundamental role in determining the quality of the resulting solution (Szepesvári, 2010, Section 3.2.4). (2) *Estimation*: since solving an MDP mostly involves the computation of certain statistics (in particular, expected values) under the distributions induced by the MDP, this cannot be done exactly when the latter is unknown. In this case, the agent has to rely on sample-based approximations and finding good estimators for the relevant quantities is crucial.

The literature on reinforcement learning is extremely wide and the next sections only introduce those settings/algorithms that are relevant to this thesis. Nevertheless, before diving in, it is important to become familiar with the main terminology and with the taxonomy of reinforcement learning methods. Though later chapters only focus on very specific settings, these terms will be recurrent throughout the whole document.

Evaluation vs control problems The problem of *policy evaluation* consists in assessing the performance of a given policy π. More precisely, the aim is to compute either the value function V^π (equivalently, the action-value function Q^π) or the expected return $J(\pi)$. *Control*, on the other hand, refers to the main problem of finding an optimal policy (i.e., of solving the MDP).

Tabular vs approximate methods Reinforcement learning methods for finite MDPs that exactly represent policies and/or value functions are called *tabular*. The name comes from the fact that these quantities can indeed be stored in a table (e.g., a value function is a $|\mathcal{S}|$-dimensional array while a policy is a $|\mathcal{S}| \times |\mathcal{A}|$ matrix). Approximate methods, on the other hand, do not represent these quantities explicitly but rather use *function approximation*. For instance, an action-value function Q could be linearly parametrized, e.g., $Q(s,a) = \phi(s,a)^T\omega$ for features ϕ and parameters ω; alternatively, it is possible to use non-parametric estimators, such as nearest-neighbors or trees. In all cases, it is no longer necessary to store a single value for each state-action pair, even when the MDP is finite.

Value-based vs policy search vs actor-critic methods *Value-based* algorithms focus on computing/approximating a value function and typically do not explicitly model a policy. For instance, in a control problem a common goal is to find the optimal action-value function $Q^\star$, from which optimal behavior can be extracted by acting greedily as described before. On the other hand, *policy search* methods directly look in the space of policies, usually without modeling any value function. Here it is common to consider parametrized stochastic policies $\pi_\theta(a|s)$, where $\theta \in \Theta$ is some parameter, and seek the optimal parameters maximizing the objective function under consideration (e.g., the expected return). Finally, *actor-critic* algorithms refer to a combination of these two settings. In particular, they both learn a (parametrized) policy, which is known as the *actor*, and a (parametrized) value function, which is known as the *critic*. The actor is constantly improved towards higher returns, while the critic evaluates its performance. Therefore, the learning processes of these two components are tightly connected.

On-policy vs off-policy algorithms An algorithm is said *on-policy* if the policy that is being learned is also the one that is used to interact with the environment. On the other hand, in an *off-policy* algorithm these two policies differ. For instance, in the case of policy evaluation, the agent might interact with the environment using some *behavioral* policy π_b with the aim of evaluating a *target* policy π. When $\pi_b \neq \pi$, we say that we are performing off-policy evaluation, and conversely when $\pi_b = \pi$.

Online vs batch methods In an *online* algorithm, learning and interaction are interleaved. In particular, the agent collects some experience by interacting with the environment (no matter whether on- or off-policy), uses it to learn something (e.g., improve or

evaluate its policy), and repeats this process over and over. In a *batch* setting, on the other hand, the agent is only provided with a finite batch of data, typically in the form of tuples (S, A, S', R), where $(S', R) \sim P(\cdot, \cdot|S, A)$, and cannot interact with the environment or request more data. Therefore, learning must be carried out without explicit interaction.

Model-based vs model-free algorithms A *model-based* algorithm learns the dynamic model (e.g., the transition kernel) of the underlying MDP. These models can then be used for planning, to simulate experience, and more. Conversely, a *model-free* algorithm learns entirely from the collected experience samples, without ever modeling/approximating the MDP components.

Fully-observable vs partially-observable environments At the beginning of this chapter, when we first introduced MDPs, we mentioned that the agent directly observes the state S_t at all time steps. This is the case of a *fully-observable* environment. However, this property is not always verified in practice and one often observes only part of the state. For instance, in a self-driving car, the state might include the positions of the other vehicles nearby, but not all of them might be directly observed due to the presence of obstacles, such as buildings, trees, and so on. Environments where this happens are called *partially observable* and they are modeled by means of a partially observable Markov decision process (POMDP, Monahan, 1982; Cassandra, 1998; Spaan, 2012). In this thesis, we shall only consider fully-observable environments.

Exact Solution Methods

In this section, we describe some of the basic *exact* solution methods for MDPs. The term "exact" refers to the fact that we know all the MDP components (in particular, the transition kernel P) and our goal is to either compute the value function of a given policy (for evaluation problems) or the optimal policy/value function (for control problems).

Policy Evaluation

Policy evaluation is the problem of computing the value function (or the action-value function) of a given policy π. Suppose, for simplicity, that we are interested in computing V^π (the computation of Q^π follows straightforwardly). Since the MDP is known, we can use the fixed-point characterization of V^π. This property, in conjunction with the contraction of the Bellman operator T^π for $\gamma < 1$, allows us to utilize the following iterative scheme. We start from an arbitrarily initialized value function V_0. At each iteration k, we apply the Bellman operator to obtain the next value function,

$$V_{k+1} = T^\pi V_k. \tag{2.20}$$

Banach's fixed-point theorem ensures that the sequence $\{V_k\}_{k\geq 0}$ converges to V^π (i.e., the fixed-point of T^π) at a geometric rate. To see this,

$$\begin{aligned}\|V_k - V^\pi\|_\infty &\overset{(a)}{=} \|T^\pi V_{k-1} - T^\pi V^\pi\|_\infty \overset{(b)}{\leq} \gamma \|V_{k-1} - V^\pi\|_\infty \\ &\overset{(c)}{\leq} \cdots \leq \gamma^k \|V_0 - V^\pi\|_\infty,\end{aligned}$$

where (a) uses the update rule for V_k and the fixed-point property of V^π, (b) uses the contraction of T^π, and (c) applies these steps recursively. Besides working beyond finite MDPs, this approach only requires the iterative application of the Bellman operator. Furthermore, one can freely decide when to stop and obtain a desired level of accuracy in the solution. Using the same properties of V_k and T^π as above, it is easy to show that

$$\|V_k - V^\pi\|_\infty \leq \frac{1}{1-\gamma}\|V_k - V_{k+1}\|_\infty,$$

so that one can use the deviation $\|V_k - V_{k+1}\|_\infty$ between the value functions computed at two consecutive iterations (which is directly observable) to assess how far V_k is from V^π. This, in turn, can be used as a stopping condition to ensure the desired accuracy.

Policy and Value Iteration

Policy iteration. We now consider control problems, where the goal is to find an optimal policy or value function. The policy evaluation discussed before constitutes one of the two building blocks of the *policy iteration* algorithm. The idea is quite simple. We start from an arbitrary policy π_0 and we evaluate it to find V^{π_0} (or Q^{π_0}). We use the value function to figure out how to change π_0 to increase its performance (i.e., its value at all states) and obtain a new policy π_1. This is called the *policy improvement* step. Then, this process is repeated until convergence to an optimal policy. We already discussed the policy evaluation step in the previous section, so let us focus on policy improvement. Consider two deterministic policies π and π'. The *policy improvement theorem* (Sutton and Barto, 2018) states that, if $Q^\pi(s, \pi'(s)) \geq V^\pi(s)$ holds for all $s \in \mathcal{S}$, then $V^{\pi'}(s) \geq V^\pi(s)$ must also hold for all $s \in \mathcal{S}$. Intuitively, this result states that, if the actions prescribed by π' are globally no worse than those prescribed by π (according to the evaluation of π), then π' itself is globally no worse than π. Suppose that π_k is the current policy to be improved and we have access to Q^{π_k}. The policy improvement theorem suggests that π_{k+1} should maximize $Q^{\pi_k}(s, \cdot)$ at all states. This can be achieved by setting π_{k+1} to the *greedy* policy with respect to Q^{π_k}, which is what policy iteration does. Overall, policy iteration computes a sequence of policies and value functions

$$\pi_0 \xrightarrow{\text{eval}} Q^{\pi_0} \xrightarrow{\text{greedy}} \pi_1 \xrightarrow{\text{eval}} \dots \xrightarrow{\text{greedy}} \pi_k$$

such that π_k is never worse than the previous policy π_{k-1}.

Value iteration. One of the limitations of policy iteration is that it requires to evaluate multiple policies. In practice, this can be a quite expensive procedure and often unnecessary. In fact, since each policy evaluation step is itself an iterative procedure, it is often possible to stop evaluating after few iterations without severely affecting the policy improvement. Algorithms that interleave the policy evaluation and improvement processes without necessarily waiting for their convergence are classified as *generalized policy iteration*.

The extreme case is when we stop the evaluation after a single iteration. The resulting algorithm is known as *value iteration* and its update rule, which combines one step of policy evaluation and policy improvement, is equivalent as applying the Bellman optimality

operator $T^\star$ to the current value function. Thus, value iteration starts from an arbitrarily initialized action-value function Q_0 (equivalently, value function V_0) and, at each iteration $k \geq 0$, uses the update rule

$$Q_{k+1} = T^\star Q_k \quad (\text{or } V_{k+1} = T^\star V_k). \tag{2.21}$$

Note that this procedure is exactly the one presented for policy evaluation, except that we apply the Bellman optimality operator. Differently from policy iteration, here there is no explicit policy being computed at each step. However, the final (optimal) policy can be extracted as the greedy one with respect to the returned action-value function Q_{k+1}. The quality of the policies returned by policy iteration is in general better than the one of those returned by value iteration after the same number of iterations and when starting from the same initial value functions (Szepesvári, 2010). However, the latter algorithm is computationally much more efficient.

Tabular Learning Methods

We now review some of the basic tabular methods which, together with the exact methods described in the previous section, lie at the core of most modern reinforcement learning algorithms. We recall that tabular methods are designed only for finite MDPs, which are thus considered in the rest of this section.

Monte Carlo Methods

Monte Carlo (MC) methods are perhaps the simplest form of reinforcement learning algorithms. The main idea is to estimate value functions by averaging the *returns* of trajectories/episodes collected while interacting with the environment. As such, Monte Carlo methods are suitable for episodic problems (where each episode indeed terminates).

Evaluation problems Let us start from a simple *policy evaluation* problem, where the goal is to estimate the expected return of a given policy π. Suppose we execute π in the environment for n episodes and observe the resulting trajectories $\{\tau_i\}_{i=1}^n$, where $\tau_i = (S_{i,0}, A_{i,0}, S_{i,1}, R_{i,1}, \ldots, S_{i,T_i}, R_{i,T_i})$. Then, the Monte Carlo estimator for the expected return $J(\pi)$ is

$$\hat{J}_n(\pi) := \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T_i - 1} \gamma^t R_{i,t+1}. \tag{2.22}$$

This is effectively the average of n i.i.d. samples, i.e., the returns $G_i := \sum_{t=0}^{T_i-1} \gamma^t R_{i,t+1}$. As such, it is unbiased, i.e., $\mathbb{E}[\hat{J}_n(\pi)] = J(\pi)$, and has variance $\mathbb{V}\mathrm{ar}[\hat{J}_n(\pi)] = \frac{\mathbb{V}\mathrm{ar}[G]}{n}$. Therefore, by the law of large numbers, $\hat{J}_n(\pi)$ converges to $J(\pi)$ as n tends to infinity.

Let us now complicate the problem a little and suppose we want to find the value function V^π instead of J^π. The idea remains the same: if we want to estimate $V^\pi(s)$ for some state $s \in \mathcal{S}$, we simply average the sample returns starting from those time steps where s is observed (i.e., $S_t = s$). Since s might occur more than once in each episode, we have two possibilities. (1) In *first-visit* Monte Carlo, we only average the returns after

the *first* occurrence (also known as visit) of s in each trajectory. Any other occurrence is discarded. Similarly to the previous problem, first-visit MC is unbiased since it averages i.i.d. random variables. (2) In *every-visit* Monte Carlo, the returns after *all* occurrences of s are averaged. The resulting estimator is no longer unbiased in general, but might have lower variance since it averages more variables than first-visit MC. Both first-visit and every-visit MC have been shown to converge to $V^\pi(s)$ when the number of visits to s goes to infinity (Sutton and Barto, 2018).

Control Problems Generalizing Monte Carlo evaluation to control problems, where the aim is to find the optimal policy, is relatively easy by using the *generalized policy iteration* (GPI) scheme of the previous section. First, notice that Monte Carlo policy evaluation can be easily generalized to estimate action-value functions simply by averaging the returns after each visit to a specific state-action pair. Then, a Monte Carlo GPI scheme could work as follows. We start from some policy π_0 and execute it in the environment for one or more episodes. Then, we use the observed returns to get a rough estimation of its action-value function Q^{π_0}, find the new policy π_1 by greedy policy improvement, and repeat this process until convergence. However, here we face a subtle complication that was not present in exact GPI: once the policy becomes deterministic (e.g., after one step of greedy improvement), only one action is taken in each state (e.g., the greedy one) and so it is not possible to get a reliable estimate of Q^π for the other actions. Unfortunately, estimating the value of *all* actions is fundamental to perform policy improvement and so an approach in this form cannot work. This is indeed the first of a myriad of problems where the *exploration-exploitation dilemma* arises: on the one hand, we would like the policy to be highly stochastic, so that we can guarantee sufficient *exploration* to reliably estimate the value of all actions; on the other hand, we would like the policy to be nearly deterministic, so that it can *exploit* the current knowledge and collect high rewards. This conflict is typically resolved by forcing some stochasticity in the policy that is being learned. Two common ways to achieve this are ϵ-greedy and Boltzmann exploration. Let Q be the current action-value function estimate. For $\epsilon \in [0, 1]$, an ϵ-greedy policy with respect to Q is

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} & \text{if } a = \operatorname{argmax}_{a' \in \mathcal{A}} Q(s, a'), \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise.} \end{cases} \tag{2.23}$$

That is, π chooses a random action with probability ϵ and the greedy action with respect to Q otherwise. A Boltzmann policy is defined as

$$\pi(a|s) = \frac{e^{Q(s,a)/\eta}}{\sum_{a' \in \mathcal{A}} e^{Q(s,a')/\eta}}, \tag{2.24}$$

where $\eta > 0$ is called *temperature*. In this case, the policy assigns more probability to actions with larger values and converges to the greedy one for $\eta \to 0$. In both types of exploration, the policy ensures that all actions are taken with non-zero probability. A common practice is then to run the Monte Carlo control algorithm with either of these policies and make the corresponding exploration parameter (ϵ or η) slowly approach zero with the number of iterations.

Temporal-Difference Learning

One of the limitations of Monte Carlo methods is that they cannot learn from incomplete episodes, i.e., they must wait until the end of the episode to get a sample return and perform an update. Temporal-difference (TD) methods resolve this issue by allowing updates after each single transition. Once again, let us consider first an evaluation problem for policy π. Suppose the agent is in state S_t, takes action $A_t \sim \pi$, and observes S_{t+1} and R_{t+1}. If V is the current guess for the value function of π, the TD update rule is

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)), \tag{2.25}$$

where α is a tunable parameter that can be regarded as a *learning rate*. Intuitively, the current value $V(S_t)$ is moved towards the *target* $R_{t+1} + \gamma V(S_{t+1})$. The term inside the round brackets in 2.25 is called the *TD error* and roughly indicates how well the current value estimate matches its target.

It is known that the MC update could be rewritten in the same form as 2.25 with the return G_t until the end of the episode as target. From the Bellman equations, we have that the MC and TD target match in expectation if $V = V^\pi$, i.e.,

$$V^\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s]. \tag{2.26}$$

However, since in general $V \neq V^\pi$, the TD target is *biased*. Nevertheless, the TD update has often much lower variance since it only involves the randomicity of a one-step transition, while MC suffers that of a full episode.

The extension of the evaluation method to control problems is straightforward given the discussion of the previous section. Here we discuss the two main algorithms, the first on-policy (Sarsa) and the second off-policy (Q-learning).

Sarsa Sarsa is an on-policy control method that directly implements the GPI scheme as we have seen for MC control, except that the evaluation step is performed by an incremental TD update. In particular, Sarsa maintains and estimate Q of the current policy's action-value function. Given a tuple $(S_t, A_t, S_{t+1}, R_{t+1}, A_{t+1})$, where actions are taken by the policy π that is being learning, the TD update is

$$g_t \leftarrow R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t), \tag{2.27}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha g_t. \tag{2.28}$$

As before, the policy can be anything that guarantees sufficient exploration while slowly converging to deterministic behavior, such as ϵ-greedy or Boltzmann. Convergence have been established under the condition that all state-action pairs are visited an infinite number of times and that the policy becomes greedy in the limit (Singh et al., 2000).

Note that the update rule of Equation 2.27-2.28 involves the random action A_{t+1} at the next time-step. Since the Bellman equation, which ideally establishes the target, involves an expectation of Q^π over the next action and the policy is known, it is possible to directly calculate such expectation in the update rule,

$$g_t \leftarrow R_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t), \tag{2.29}$$

which removes the necessity of knowing A_{t+1}. The resulting algorithm is known as *expected Sarsa*.

Q-Learning Q-learning (Watkins, 1989) is an off-policy TD control method. Differently from Sarsa, its aim is to directly estimate the optimal action-value function $Q^\star$ rather than the value function of the adopted policy. The update rule is, regardless of how actions are chosen,

$$g_t \leftarrow R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t), \tag{2.30}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha g_t. \tag{2.31}$$

This rule is similar to the one used by Sarsa, except for the maximization of Q over actions in the next state, which resembles an application of the Bellman optimality operator. Despite its simplicity, several convergence results have been established (Jaakkola et al., 1994; Tsitsiklis, 1994; Szepesvári, 1998; Melo, 2001). More recently, it has been shown that Q-learning in combination with classic exploration techniques achieves sub-linear regret (Jin et al., 2018; Yang et al., 2020).

Approximate Methods

So far we have seen how to deal with one of the two challenges introduced in Section 2.2, estimation, for tabular methods that work in small finite MDPs. Here we relax this assumption and go back to the general case of continuous MDPs, which requires dealing with the second challenge, *approximation*. We start from value-based algorithms since most of them directly extend the (exact) dynamic programming methods of Section 2.3. We shall discuss modern policy search and actor-critic algorithms in the sections thereafter.

Value-Based Methods

In general, value-based methods fall under the class of approximate dynamic programming (ADP), a family of algorithms that combine the exact solution methods of Section 2.3 with function approximation. The idea is to represent value functions in some functional space $\mathcal{F}$, with the main advantage being that large or even infinite state spaces can be easily handled for properly-chosen $\mathcal{F}$. The goal is to find a function $f \in \mathcal{F}$ that approximates well the optimal value function $V^\star$ or action-value function $Q^\star$.

Fitted Q-Iteration

Fitted Q-Iteration (FQI, Ernst et al., 2005) is perhaps the most popular algorithm in the family of ADP. FQI is a *batch* model-free approach for control that seeks an approximation to the optimal action-value function $Q^\star$ given only a finite dataset of n sample transitions $\mathcal{D} = \{(S_i, A_i, R_i, S_i')\}_{i=1}^n$ and with no environment interaction allowed. The idea is quite simple. Given the hypothesis space $\mathcal{F}$, FQI starts from an initial value function $Q_0 \in \mathcal{F}$. At each iteration $k \geq 0$, FQI approximates the application of the Bellman optimality operator to Q_k by solving a supervised learning problem. More precisely, this is achieved by building a regression problem from $\mathcal{D}$ and Q_k, where the i-th covariate is (S_i, A_i) and the i-th target is $Y_i := R_i + \gamma \max_{a \in \mathcal{A}} Q_k(S_i', a)$. The next approximate value function

Algorithm 1 Fitted Q-Iteration (FQI)

Require: Number of iterations K, dataset $\mathcal{D} = \{(S_i, A_i, R_i, S'_i) | i = 1, \dots, n\}$, hypothesis space $\mathcal{F}$
Ensure: Greedy policy π_K
  Initialize value function: $Q_0 \leftarrow 0$
  for $k = 0, \dots, K-1$ **do**
    Build regression targets: $Y_i \leftarrow R_i + \gamma \max_{a \in \mathcal{A}} Q_k(S'_i, a), \quad \forall i = 1, \dots, n$
    Solve regression problem: $Q_{k+1} \leftarrow \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(f(S_i, A_i) - Y_i\right)^2$
  end for
  Compute greedy policy: $\pi_K(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a), \ \ \forall s \in \mathcal{S}$

Q_{k+1} is then

$$Q_{k+1} = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(f(S_i, A_i) - Y_i\right)^2. \tag{2.32}$$

Here we framed the problem as a minimization of the mean-square error, but other losses can be freely chosen. Abstracting a little, the construction of the regression targets can be seen as an application of an *empirical Bellman operator* to Q_k.

Definition 2.5.1 (Empirical Bellman Optimality Operator)**.** *Let $\mathcal{D}$ be a dataset of n transitions, $\mathcal{D} = \{(S_i, A_i, R_i, S'_i) | i = 1, \dots, n\}$, and $Q \in \mathcal{F}$ be any approximate action-value function. The empirical Bellman optimality operator $\hat{T}^\star$ using $\mathcal{D}$ is*

$$(\hat{T}^\star Q)(S_i, A_i) := R_i + \gamma \max_{a \in \mathcal{A}} Q(S'_i, a). \tag{2.33}$$

The empirical Bellman operator for policy evaluation can be defined analogously. Thus, the k-th FQI update step can be described as an application of $\hat{T}^\star$ to Q_k and then a *projection* of $\hat{T}^\star Q_k$ onto the hypothesis space $\mathcal{F}$. The projection step constitutes the key difference with respect to exact methods. In fact, in exact methods no projection is required since the value function resulting from an application of the Bellman operator always lies in the assumed functional space (i.e., the space of all possible value functions). Here this property does not necessarily hold and hence projection is required. The complete pseudo-code of FQI is provided in Algorithm 1.

Several studies of FQI-based strategies are available in the literature. Ernst et al. (2005) use tree-based methods to approximate action-value functions and show their good practical performance, especially with *extremely randomized trees* (Geurts et al., 2006). A similar empirical study was conducted by Riedmiller (2005) using neural networks. These two approximators have become the most commonly adopted for this setting. Other techniques have been proposed, including regularization (Farahmand et al., 2009), advantage-weighted regression (Neumann and Peters, 2009), and boostingtosatto2017boosted. Though FQI is intrinsically defined for finite actions, an extension to continuous action spaces was designed by Antos et al. (2008).

Q-Learning with Function Approximation

Several approximate variants of Q-learning have been proposed. Suppose that the hypothesis space $\mathcal{F}$ for approximating the action-value function Q is a set of functions parametrized by some vector $\omega \in \mathbb{R}^d$, i.e., $\mathcal{F} = \{Q_\omega : \omega \in \mathbb{R}^d\}$. Furthermore, assume that these functions are differentiable with respect to ω. The natural extension of Q-learning to the approximate setting uses the following update rule:

$$\omega_{t+1} \leftarrow \omega_t + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q_\omega(S_{t+1}, a) - Q_\omega(S_t, A_t))\nabla_\omega Q_\omega(S_t, A_t).$$

This can be easily interpreted as a stochastic gradient update when the overall objective is to minimize the average squared TD error. The convergence of this scheme has only been established in specific settings, e.g., by Melo and Ribeiro (2007) for linear approximators and under other very restrictive assumptions.

Deep Q-networks When using neural networks, Q-learning with function approximation is often called deep Q-learning or deep Q-network (DQN, Mnih et al., 2015). While neural networks provide high representation power, their combination with an online off-policy algorithm such as Q-learning introduces many complications. These complications, in turns, require several tricks to stabilize and improve learning with respect to the basic approximate scheme described above. The first is the concept of *experience replay*. As mentioned above, approximate Q-learning can be seen as minimizing the average TD error (over the observed transitions) by stochastic gradient descent. This resembles a supervised learning problem with the exception that samples are not i.i.d. since states are sequentially correlated. To break this correlation, a DQN agent stores all the observed transitions (as the usual four-tuples) into a *replay buffer*. At each step, n transitions are sampled uniformly from this buffer and used to estimate the average squared TD error,

$$f(\omega) := \frac{1}{n} \sum_{i=1}^{n} \left(R_i + \gamma \max_{a \in \mathcal{A}} Q_\omega(S_i', a) - Q_\omega(S_i, A_i) \right)^2, \tag{2.34}$$

and its gradient

$$\nabla_\omega f(\omega) = -\frac{1}{n} \sum_{i=1}^{n} \left(R_i + \gamma \max_{a \in \mathcal{A}} Q_\omega(S_i', a) - Q_\omega(S_i, A_i) \right) \nabla_\omega Q_\omega(S_i, A_i).$$

Note that, in the computation of the gradient, the targets are assumed fixed and not dependent on ω, which avoids differentiating through the maximum. Then, the Q-network is updated as described above. Finally, new experience is collected by running a sufficiently-exploring policy with respect to the current action-value function. The most common choice is an ϵ-greedy policy with ϵ decaying at an appropriate rate, but other techniques (like Boltzmann exploration) could be adopted.

To avoid the targets from changing too fast (another violation of the supervised learning assumptions), a *target network* is frequently used to further stabilize learning. The idea is to update the parameters of the Q-network used to compute the targets at a slower rate than the main Q-network used to compute the gradient and for exploration. Another issue with the computation of the targets is the overestimation of the maximum expected

value. This can be handled by double Q-learning (Hasselt, 2010) which led to the double DQN algorithm (Van Hasselt et al., 2015). Other improvements include non-uniform sampling from the replay buffer, called *prioritized experience replay* (Schaul et al., 2015b), and the usage of dueling architectures (Wang et al., 2016c) to better model the value/advantage function structure of the problem. An empirical study of the combination of all these tricks (and more) was carried out by Hessel et al. (2017).

Policy-Gradient Methods

While value-based algorithms aim at estimating/approximating value functions, *policy search* methods directly parametrize control policies and seek the optimal parameters. They have been shown effective in problems with high-dimensional and continuous state-action spaces, especially in the field of robotics (Kober and Peters, 2009; Deisenroth et al., 2013; Mülling et al., 2013; Chatzilygeroudis et al., 2017). These methods are also called *actor-only* since only the actor component (i.e., the policy) is learned, while the critic (i.e., the value function) is missing. We shall discuss actor-critic algorithms, which combine the two, in the next section.

Let $\pi : \mathcal{S} \times \Theta \to \mathcal{P}(\mathcal{A})$ be a parametrized stochastic control policy, where $\Theta \subseteq \mathbb{R}^d$ is the parameter space. We denote by $\pi_\theta(a|s)$ the conditional probability density function evaluated at action $a \in \mathcal{A}$ given state $s \in \mathcal{S}$ and parameters $\theta \in \Theta$. Following Deisenroth et al. (2013), we shall adopt a trajectory-based notation in an *episodic* setting. Recall that a trajectory τ is a sequence of states and actions, $\tau = (s_0, a_0, s_1, \dots, s_T)$. With some abuse of notation, we denote by $r(\tau) := \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)$ the expected reward associated to trajectory τ. Each policy π_θ induces a probability distribution over trajectories. Let $p_\theta(\tau)$ denote the corresponding probability density function, which can be factored as

$$p_\theta(\tau) = \rho(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t). \tag{2.35}$$

Then, the expected return $J(\theta) = J(\pi_\theta)$ of π_θ can be expressed as

$$J(\theta) = \int r(\tau)p_\theta(\tau)\mathrm{d}\tau, \tag{2.36}$$

where the integral is over the space of all possible trajectories. Policy search methods aim at maximizing this quantity over Θ. Among the class of policy search methods, here we focus on those based on *policy gradients*, i.e., those that update parameters iteratively by gradient ascent. Starting from an arbitrary $\theta_0 \in \Theta$, at each iteration $k \geq 0$ the ideal update rule is

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta) \tag{2.37}$$

where $\nabla_\theta J(\theta)$ is the *policy gradient*. Since the transition kernel of the underlying MDP is unknown, the exact policy gradient, similarly to expected returns, cannot be computed and has to be estimated from sample trajectories. We now discuss the most common techniques to achieve this.

REINFORCE

The REINFORCE algorithm is based on the so-called *likelihood-ratio* trick and was originally proposed by Williams (1992). The idea is that, using the chain rule, $\nabla_\theta p_\theta(\tau) = p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)$. This allows to rewrite the policy gradient in a more convenient form.

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \int r(\tau) p_\theta(\tau) \mathrm{d}\tau = \int r(\tau) \nabla_\theta p_\theta(\tau) \mathrm{d}\tau \\ &= \int r(\tau) p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) \mathrm{d}\tau = \mathbb{E}_{\tau\sim p_\theta}\left[r(\tau)\nabla_\theta \log p_\theta(\tau)\right].\end{aligned}$$

Using (2.35),

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \left(\log \rho(s_0) + \sum_{t=0}^{T-1} \left(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)\right)\right).$$

The gradients of ρ and P are zero since these functions do not depend on θ. Therefore, we arrive at the following important result.

Proposition 2.5.1. *The policy gradient can be expressed as*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]. \tag{2.38}$$

This formulation expresses the policy gradient as an expectation over trajectories and as such can be easily estimated by averaging n i.i.d. episodes collected under policy π_θ. To reduce the variance of the resulting estimate, it is possible to use *control variates* (Hammersley and Handscomb, 1964), often referred to as *baselines* in the policy gradient literature (Peters and Schaal, 2008b; Deisenroth et al., 2013). For a fixed vector b, it is easy to show that

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) - b\right)\right], \tag{2.39}$$

and thus it is possible to find a baseline b that minimizes (component-wise) the variance of the gradient estimator without changing its expected value. The optimal baseline is given by

$$b = \frac{\mathbb{E}_{\tau\sim p_\theta}\left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)^2 \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]}{\mathbb{E}_{\tau\sim p_\theta}\left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)^2\right]}, \tag{2.40}$$

where the expectations can be once again estimated from samples.

G(PO)MDP

G(PO)MDP (Baxter and Bartlett, 2001), which stands for gradient of a (partially observable) Markov decision process, uses the intuition that future actions do not influence past

rewards to reduce the variance of the REINFORCE gradient estimator. This intuition is formalized by the following expression of the policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\sum_{h=0}^{t}\nabla_\theta \log \pi_\theta(a_h|s_h)\left(\gamma^t r(s_t,a_t) - b_t\right)\right]. \tag{2.41}$$

Similarly to REINFORCE, the optimal (time-dependent) baseline is

$$b_t = \frac{\mathbb{E}_{\tau\sim p_\theta}\left[\left(\sum_{h=0}^{t}\nabla_\theta \log \pi_\theta(a_h|s_h)\right)^2 \gamma^t r(s_t,a_t)\right]}{\mathbb{E}_{\tau\sim p_\theta}\left[\left(\sum_{h=0}^{t}\nabla_\theta \log \pi_\theta(a_h|s_h)\right)^2\right]}. \tag{2.42}$$

Policy Gradient Theorem

The policy gradient theorem (PGT) (Sutton et al., 2000) states yet another formulation of the policy gradient. For any function $b_t : \mathcal{S} \to \mathbb{R}$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\gamma^t\left(Q^{\pi_\theta}(s_t,a_t) - b_t(s_t)\right)\right]. \tag{2.43}$$

The baseline $b_t(s_t)$ is now allowed to depend on the current state. A common choice is the value function $V^{\pi_\theta}(s_t)$. The action value function can be estimated by Monte Carlo methods, $Q^{\pi_\theta}(s_t,a_t) \simeq \sum_{h=t}^{T-1}\gamma^{h-t} r(s_h,a_h)$. More sophisticated techniques involve the usage of function approximation for Q as well. In such case, the resulting PGT-based algorithm belongs to the class of actor-critic methods, discussed in the next section.

Actor-Critic Algorithms

Actor-critic algorithms implement generalized policy iteration (see Section 2.3). They are composed of two learned components: (1) the *actor*, i.e., a parametrized control policy $\pi_\theta(a|s)$ which describes the behavior of the agent; (2) the *critic*, i.e., a parametrized action-value function $Q_\omega(s,a)$ (or value function $V_\omega(s)$) which evaluates the current agent's policy. These two components are learned simultaneously, either synchronously or asynchronously.

Actor-critic algorithms have become perhaps the most adopted and successful reinforcement learning methods. In particular, when combined with neural networks, they are at the core of the recent field of *deep reinforcement learning*. Here we briefly review the main techniques. We refer the reader to recent surveys (Li, 2017; Arulkumaran et al., 2017; François-Lavet et al., 2018) for more details.

In an early study, Sutton et al. (2000) provided insights on how to design an actor-critic scheme based on the policy gradient theorem. The key result is that, when the function approximation used for the action-value function is *compatible* with the one used for the policy, in the sense that $\nabla_\omega Q_\omega(s,a) = \nabla_\theta \log \pi_\theta(a|s)$, and ω has converged to a stationary point of the corresponding objective, then $Q_\omega(s,a)$ can be safely used instead of $Q^{\pi_\theta}(s,a)$ to compute the policy gradient $\nabla_\theta J(\theta)$ without introducing any error. Similar results were obtained concurrently by Konda and Tsitsiklis (2000). These results and algorithms were

refined by Peters and Schaal (2008a) in combination with natural gradients Amari (1998). More recently, Schulman et al. (2015a) proposed trust-region policy optimization (TRPO), an approach for optimizing parametrized policy that is derived from a theoretically-sound monotonically improving (in terms of returns of the computed policies) method. Similarly to natural gradient methods, TRPO constrains each policy update to stay close (in KL divergence) to the previous policy. Experimentally, the algorithm was shown very effective in many high-dimensional tasks in which stable and sample-efficient learning are primary concerns. An algorithm following similar, perhaps simpler, ideas, called proximal policy optimization (PPO), was proposed by Schulman et al. (2017). Despite its simplicity, PPO enjoys similar, and sometimes better, performance than TRPO. Schulman et al. (2015b) derived an approach to estimate the advantage function to be used in conjunction with PGT (Sutton et al., 2000) or other policy optimization approaches. While all these algorithms optimize stochastic policies, the optimization of deterministic policies was investigated by Silver et al. (2014). The authors derived a deterministic version of the policy gradient theorem, which basically reduces to the expected gradient of the action-value function, and proposed an off-policy actor-critic method based on this result. Its extension to deep neural networks, called deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), was demonstrated very effective in several high-dimension continuous control tasks. Mnih et al. (2016) proposed advantage actor-critic (A3C), a standard actor-critic algorithm with multiple asynchronous actor learners, where the policy is updated by the policy-gradient theorem (with the advantage function), and the advantage function is updated by a value-function-based critic, the latter estimated by TD methods. Its synchronous variant (called A2C) was also shown effective. Recently, Haarnoja et al. (2018a,b) proposed soft actor-critic (SAC), a method based on maximum entropy reinforcement learning. Differently from previous works, the aim is to maximize the expected return plus the entropy of the chosen policy (with the trade-off controlled by a suitable tunable parameter). The algorithm achieved state-of-the-art results in several continuous control tasks, while leading to more stable learning and better exploration than previous methods.

Multi-Armed Bandits

The stochastic multi-armed bandit (MAB) is one of the simplest reinforcement learning problems. A stochastic bandit can be described by an MDP with only one state (or, equivalently, no states) and, thus, no state-transition dynamics. Although MDPs and bandits are tightly connected, the literatures studying these two problems often use slightly different notation. At the price of generality, and to favor readability, we shall introduce bandits using the standard notation adopted in the literature (e.g., Bubeck et al., 2012; Lattimore and Szepesvári, 2020). Formally, a (finite) stochastic bandit can be described by a finite set of K actions $\mathcal{A}$ (also called *arms*) and a set of reward distributions $\nu = \{\nu(a) : a \in \mathcal{A}\}$. The learning agent interacts with the environment for n rounds. At each round $t \in \{1, \dots, n\}$, the learner chooses an action $A_t \in \mathcal{A}$ and it receives a random reward $Y_t \sim \nu(A_t)$. The actions are chosen by a a *bandit algorithm*, i.e., a possibly-randomized history-dependent policy $\pi = \{\pi_t\}_{t=1}^n$, where π_t maps $(t-1)$-step histories $H_t = (A_1, Y_1, \dots, A_{t-1}, Y_{t-1})$ into (probability distributions over) actions. Let $\mu(a) := \mathbb{E}_{Y \sim \nu(a)}[Y]$ be the mean reward

of arm $a \in \mathcal{A}$. The learner aims at minimizing the *expected regret*,

$$\mathrm{R}_n^\pi(\nu) := n \max_{a \in \mathcal{A}} \mu(a) - \mathbb{E}_{\nu,\pi}\left[\sum_{t=1}^n Y_t\right], \tag{2.44}$$

that is, the difference between the expected cumulative reward obtained by an oracle that always pulls the optimal arm and the expected cumulative reward obtained by the learner playing policy π. Minimizing the regret requires the agent to trade off between *exploring* arms to understand their uncertain outcomes and *exploiting* those that have performed best in the past. In fact, the learner often has only partial information about the underlying bandit problem ν. The learner is typically provided with a set $\mathfrak{M}$ of realizable bandit problems (let us call it a *problem class*), so that $\nu \in \mathfrak{m}$. The most common assumption involves sets of the form $\mathfrak{M} = \mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_K$, where $\mathcal{V}_a \subseteq \mathcal{P}(\mathbb{R})$ is a subset of possible reward distributions for arm $a \in \mathcal{A}$. These could be, for instance, Gaussian distributions with fixed variance, Bernoulli distributions, sub-Gaussian distributions, and so on. We call problem classes of this form *disjoint*.[4] In this case, samples from one arm do not provide any information about the rewards of other arms since the corresponding distributions are effectively disjoint. On the other hand, a non-disjoint problem class is such that different arms might be correlated. Non-disjoint problem classes, which relate to our concept of structured domain, are thoroughly discussed in Part III. Here we briefly discuss the main strategies and results for disjoint classes.

Disjoint problems. The classic disjoint MAB problem, in which the rewards of the different arms are uncorrelated, is theoretically well understood. In particular, asymptotic problem-dependent lower bounds on the regret have been derived by Lai and Robbins (1985); Burnetas and Katehakis (1996). These characterize the minimum regret that any "good" bandit algorithm must suffer on a given problem class. Formally, we use the following definition of "good" bandit algorithm.

Definition 2.6.1 (Lai and Robbins (1985))**.** *A bandit algorithm π is uniformly consistent on a problem class $\mathfrak{M}$ if, for all $\nu \in \mathfrak{M}$ and $p > 0$,*

$$\limsup_{n \to \infty} \frac{R_n^\pi(\nu)}{n^p} = 0. \tag{2.45}$$

That is, a uniformly consistent algorithm suffers sub-polynomial regret in all bandit instances in $\mathfrak{M}$.

Theorem 2.6.1 (Problem-dependent lower bound (Burnetas and Katehakis, 1996))**.** *Let $\mathfrak{M} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_K$ be a disjoint problem class with K arms and π be a uniformly consistent bandit algorithm over $\mathfrak{M}$. Then, for all $\nu \in \mathfrak{M}$ with $\nu = \{\nu(a) : a \in \mathcal{A}\}$,*

$$\liminf_{n \to \infty} \frac{R_n^\pi(\nu)}{\log n} \geq \sum_{a \in \mathcal{A} : \Delta(a) > 0} \frac{\Delta(a)}{\inf_{\nu' \in \mathcal{V}_a} \{D_{\mathrm{KL}}(\nu(a) \| \nu') : \mu(\nu') > \mu^\star\}},$$

where $\Delta(a) = \max_{a' \in \mathcal{A}} \mu(a') - \mu(a)$ is the sub-optimality gap of arm a, $\mu^\star = \max_{a \in \mathcal{A}} \mu(a)$ is the optimal reward, and $\mu(\nu')$ is the expected value of random rewards drawn from ν'.

[4] In the literature (Lattimore and Szepesvári, 2020), disjoint models are called unstructured, while non-disjoint models are called structured. Here we avoid these terms since later we shall consider a slightly more general concept of "structured problem".

This bound can be instantiated for any distribution class. For instance, for Gaussian bandits with variance equal to one, we recover the lower bound of Lai and Robbins (1985),

$$\liminf_{n\to\infty} \frac{R_n^\pi(\nu)}{\log n} \geq \sum_{a\in\mathcal{A}:\Delta(a)>0} \frac{2}{\Delta(a)}. \tag{2.46}$$

For this setting, the UCB algorithm (that we describe shortly) is asymptotically optimal, i.e., its regret upper bound matches (2.46). Similarly, the KL-UCB (Garivier and Cappé, 2011) is optimal for Bernoulli distributions. While problem-dependent lower bounds focus on instance-specific regret, another line of works derive finite-time *worst-case* (or minimax) lower bounds (see e.g., Chapter 15 of Lattimore and Szepesvári (2020)), which focus on the hardest instance in a given problem class. For instance, it can be shown that, for Gaussian bandits with K arms, for any horizon $n \geq K$, there exists a bandit problem on which the regret of *any* policy (not only the consistent ones) is at least $\Omega(\sqrt{kn})$.

We now describe two of the most common algorithms in the bandit literature, UCB and Thompson sampling, whose design relies on fundamental exploration principles.

Upper confidence bound. The upper confidence bound (UCB) algorithm uses the principle of *optimism in face of uncertainty* to trade off exploration and exploitation. The idea is quite simple: whenever we are uncertain about the expected reward of some arm, we assume the maximum value that is realizable according to our uncertainty (in some sense, the best possible world) and pull the optimal arm of the resulting bandit problem. Formally, UCB constructs independent confidence intervals for the mean reward of each arm. At each step, the algorithm chooses

$$A_t = \operatorname*{argmax}_{a\in\mathcal{A}} \left\{ \hat{\mu}_{t-1}(a) + \sqrt{\frac{\alpha \log t}{N_{t-1}(a)}} \right\}, \tag{2.47}$$

where $N_{t-1}(a) = \sum_{s=1}^{t-1} \mathbb{1}\{A_t = a\}$ is the number of times $a \in \mathcal{A}$ has been chosen prior to round t, $\hat{\mu}_{t-1}(a) = \frac{1}{N_{t-1}(a)} \sum_{s=1}^{t} Y_t \mathbb{1}\{A_t = a\}$ is the empirical mean of the same arm, and α is a parameter. In other words, UCB chooses the arm with the highest *optimistic* mean reward. The algorithm enjoys logarithmic regret for certain values of α (Auer et al., 2002a) and is asymptotically optimal for Gaussian bandits when combined with slightly refined confidence intervals (see, e.g., Chapter 8 of Lattimore and Szepesvári (2020)). UCB also enjoys a worst-case bound of $\mathcal{O}(kn \log n)$, which matches the minimax lower bound except for the logarithmic term.

Thompson sampling. Thompson sampling (Thompson, 1933) is another very popular exploration principle. It works in Bayesian bandits, i.e., where the learner has a prior distribution $p_0 \in \mathcal{P}(\mathfrak{M})$ over possible bandit problems and updates the corresponding posterior p_t as more experience is collected. In disjoint problems, this distribution reduces to an independent distribution for each arm. Similarly to optimism, the idea is quite simple: at each round t, the algorithm samples a bandit problem ν_t from the current posterior p_t and takes its optimal arm, $A_t = \operatorname{argmax}_{a\in\mathcal{A}} \mathbb{E}_{Y\sim\nu_t(a)}[Y]$. While simple, the algorithm enjoys sub-linear problem dependent and worst-case regret bounds, both in the frequentist and Bayesian settings (Kaufmann et al., 2012; Agrawal and Goyal, 2012; Russo and Van Roy, 2016; Agrawal and Goyal, 2017).

CHAPTER 3

Transfer in Reinforcement Learning

The purpose of this chapter is two-fold. First, we intend to provide a general *taxonomy* of transfer methods that unifies the existing literature and, in particular, the multitude of problem settings in which knowledge transfer is studied and applied. While we shall mostly follow the surveys of Taylor and Stone (2009) and Lazaric (2012), we also need to adapt the taxonomies presented there to cover the vast literature that appeared in the recent years. Second, we intend to provide a brief *survey* of transfer methods and related problem settings, again with a focus on modern research.

The chapter is organized as follows. We start by discussing the main problem settings where knowledge transfer is applied, and, in particular, we formally define the concept of structured domain, in Section 3.1. Then, we present the main dimensions according to which transfer methods can be classified in Section 3.2, while in Section 3.3 we discuss how these methods can be evaluated, with a focus on the main performance measures. We conclude in Section 3.4 with a review of existing approaches classified by problem settings.

Problem Settings

Knowledge transfer is a core component in a multitude of reinforcement learning problems. In general, the invariant factor among these settings is the presence of multiple tasks with some commonalities, which makes the adoption of transfer methods possible. Throughout this thesis, we shall refer to this setting as a *structured domain*, which is formalized as follows.

Definition 3.1.1 (Structured domain)**.** *A structured domain is a tuple* $\mathfrak{E} = (\mathfrak{M}, \mathfrak{D})$*, where* $\mathfrak{M}$ *is the task family, i.e., a set of Markov decision processes, and* $\mathfrak{D}$ *is the task generation process.*

We use structured domains to model all kind of settings in which knowledge transfer is applied. Informally, $\mathfrak{M}$ encodes the set of possible tasks that the agent might face, while $\mathfrak{D}$ describes the process that generates these tasks. The term "structured" comes from the fact that tasks in $\mathfrak{M}$ might share some hidden structure (e.g., a common representation) that enables knowledge transfer. That is, an agent facing multiple tasks from $\mathfrak{E}$ might be capable of understanding their similarities, thus using them to improve the learning process in new problems from the same domain. Definition 3.1.1 generalizes the one of Lazaric (2012) since the task generation process $\mathfrak{D}$ is kept general so that it can be instantiated in different ways and, thus, capture the majority of existing problem settings.[1]

Depending on the specific learning objectives, on how tasks are generated/faced, and potentially on other factors, the literature can be clustered in different problem settings.

Transfer Learning. By "transfer learning" we refer to the basic problem setting where knowledge transfer is the primary focus (Taylor and Stone, 2009; Lazaric, 2012). Here the agent is assumed to have some kind of knowledge from a set of $m \geq 1$ *source tasks* $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\} \subseteq \mathfrak{M}$. Each task is an MDP $\mathcal{M}_j := (\mathcal{S}_j, \mathcal{A}_j, P_j, \rho_j, \gamma_j)$ with potentially different components. For instance, the agent might have solved these tasks, thus computing approximately-optimal policies/value functions, or it might have estimated their transition models, or it might have simply collected some experience samples while interacting with them. Given a *target task* $\mathcal{M}_0 := (\mathcal{S}_0, \mathcal{A}_0, P_0, \rho_0, \gamma_0)$, the goal of the agent is to transfer whatever knowledge is available from the source tasks to improve the learning process of the target. While the final purpose is to perform well on the unknown target task, here the focus is on the challenges faced when performing transfer, as discussed in the introduction and throughout this chapter.

Multi-Task Reinforcement Learning. In multi-task reinforcement learning, it is common to assume that the task generation process $\mathfrak{D}$ is some unknown probability distribution and the goal is to generalize to tasks drawn from $\mathfrak{D}$. More precisely, after training on m source tasks, which are drawn i.i.d. from $\mathfrak{D}$, the goal is to perform well on target tasks drawn from the same distribution. Though knowledge transfer is a fundamental component, here the focus is mainly on learning single models (e.g., policies) that solve tasks from $\mathfrak{D}$ without further learning (Espeholt et al., 2018; Hessel et al., 2019)[2] or on learning separate models jointly, each for a different task (Lazaric and Ghavamzadeh, 2010; Calandriello et al., 2014). For instance, in the first case, one could consider the following optimization problem:

$$\max_{\pi} \mathbb{E}_{\mathcal{M}\sim\rho}\left[\mathbb{E}_{\mathcal{M},\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\right]\right]. \tag{3.1}$$

Using the intuition that the multi-task objective can be reduced to a POMDP where the only hidden variable is the task identity, it is possible to optimize this by using history-

[1] Lazaric (2012) explicitly model $\mathfrak{D}$ as a probability distribution generating i.i.d. tasks.

[2] This is also known as zero-shot generalization/adaptation.

dependent policies (e.g., using recurrent neural networks). Alternatively, if tasks are parameterized and their parameters, say ω, are known, it is possible to optimize a task-conditioned policy $\pi(a|s, \omega)$.

Meta-Reinforcement Learning. Meta-reinforcement learning is an instance of meta-learning, or learning to learn (see, e.g., Schmidhuber (1987); Thrun and Pratt (2012); Vanschoren (2018)), where the goal is to directly learn a reinforcement learning agent that is customized to learning related tasks, i.e., tasks from the same structured domain. The meta-reinforcement learning setting is conceptually similar to that of multi-task learning; the agent still aims at performing well on tasks drawn from some unknown distribution. The main difference is that focus is on the learning process in new tasks rather than computing a single model that solves multiple tasks. More specifically, the goal is to have an agent that quickly adapts to new tasks drawn from $\mathfrak{D}$ in a few learning steps. The meta-reinforcement learning process involves two steps. (1) At *meta-training*, the agent faces multiple source tasks drawn from $\mathfrak{D}$ and optimizes its learning algorithm so that it quickly adapts to new instances. In this step, it is typically assumed that the agent can freely sample tasks from $\mathfrak{D}$ and interact with them. (2) At *meta-testing*, the agent is evaluated on tasks drawn from the same distribution. Typically, the agent is allowed to perform only few learning steps on these tasks and evaluated based on its final performance. This is in contrast to the multi-task evaluation protocol, which typically focuses on the "zero-shot" performance in new tasks, without further learning/adaptation allowed.

Lifelong Reinforcement Learning. Lifelong reinforcement learning typically refers to a setting where the agent faces tasks *online* and in *sequence*. In a common lifelong scenario, the agent interacts with each task for a limited number of steps/episodes and then the environment changes. The agent is typically informed of the change. The setting is related to that of *non-stationary* reinforcement learning, where, under the most general assumptions, the environment might change arbitrarily without the agent knowing the change points. Tasks could be generated i.i.d. from a fixed distribution or they might be sequentially correlated, for instance through a hidden Markov chain (Choi et al., 2000b) or more general stochastic processes. The sequential and online nature of the lifelong problem introduce many additional challenges with respect to the previous settings. These include accumulating and refining knowledge over time without forgetting, understanding temporal correlations between tasks, optimizing for future problems, and more.

Curriculum Learning. Curriculum learning is mostly based on the idea that solving simpler tasks might help in learning complex ones through knowledge transfer. Given some target task, the agent generates and solves a sequence of source tasks, called *curriculum*, with the final purpose of transferring knowledge to speed up the learning process of the target. Therefore, differently from the previous settings, the agent does not receive tasks from some distribution but rather explicitly chooses them, typically from a given set of alternatives. That is, the task generation process $\mathfrak{D}$ is explicitly *controlled* by the agent. Since curriculum learning is out of the scope of this thesis, we refer the reader to Narvekar et al. (2020) for a thorough survey.

Taxonomy

Transfer learning methods can be classified along multiple dimensions (Taylor and Stone, 2009; Lazaric, 2012). Here we highlight the most relevant ones. Since evaluation is often orthogonal to the specific transfer method, we defer a discussion of performance measures to the next section.

Task difference. Tasks in the same structured domain might differ in multiple MDP components. In the most common case, the state and action spaces are fixed, while the reward and transition probabilities vary between tasks. Occasionally, one of these two components is assumed fixed and transfer is entirely focused on the other. The most general setting, typically referred to as *cross-domain*, involves tasks with different state-action spaces. Here the agent is required to learn (or it is provided with) some *inter-task mappings* (Taylor et al., 2007), i.e., functions that map different states/actions to allow efficient knowledge transfer. In this thesis we only focus on transfer learning with shared state-action space. We refer the reader to the survey of Taylor and Stone (2009) for an overview of cross-domain approaches.

Task generation process. The task generation process $\mathfrak{D}$ is tightly connected to the problem setting. We distinguish four different processes which cover all those encountered in the literature. (1) *i.i.d.* Tasks are generated from a fixed unknown distribution, either in batch mode or online. This is the most common assumption, encountered in the transfer, multi-task, and lifelong learning settings. (2) *Sequential*. Tasks are sequentially generated online from a general stochastic process that allows temporal correlations. This is frequently encountered only in the lifelong setting. (3) *Arbitrary*. There is no assumption on the process generating the tasks, which might be even chosen by an adversary. This is common in a transfer learning scenario between fixed sources and target. (4) *Controlled*. The agent itself can control the generation process, e.g., by choosing the next task to face. This is the common assumption in the curriculum learning domain.

Knowledge transferred. The kind of reused knowledge is perhaps the feature that mostly characterize a transfer algorithm. In principle, any component involved in the learning process can be transferred, no matter whether it belongs to the environment or to the agent (i.e., to the learning algorithm). The following are among the most frequent. (1) *Experience samples*. The agent reuses the random rewards and next states collected in the source tasks. These can be, for instance, tuples $(S_t, A_t, S_{t+1}, R_{t+1})$ in a batch setting or entire trajectories in an online episodic one. (2) *Policies*. The agent reuses behavior. For instance, the agent can transfer policies that were found to be near-optimal for the source tasks, hoping that they will lead to good performance in the target task. Alternatively, *options* can be transferred, i.e., abstract policies that are executed upon entering a certain start condition and until some stopping criterion is verified. (3) *Value functions*. Transferring the values of some policy makes it possible to share what state/actions are good or bad. For instance, it can be used to facilitate the estimation of an optimal value function or to drive exploration in the target task. (4) *Representations*. After facing the source tasks, the agent might find good representations that (globally or locally) describe the task family and thus can be reused. These representations might be, for instance, features extracted

by a neural network, state abstractions, reward features used for shaping, and so on. (5) *Parameters*. Every reinforcement learning algorithm has some tunable parameters that can be transferred, such as learning rates, batch sizes, exploration factors, and so on. For instance, based on the learning experience on the source tasks, it is possible to find parameters that are likely to make the algorithm more efficient on target tasks from the same distribution. This is what most meta-reinforcement learning methods do. (6) *Priors*. In a Bayesian setting, priors naturally embed knowledge about a given problem. These priors can be learned on the source tasks and transferred to the target. For instance, it possible to learn the distribution of possible environments (e.g., rewards or transition dynamics) so that a Bayesian algorithm using it as prior would be more efficient in learning a target task (with respect to using a non-informative prior).

Other dimensions. While we use the above-mentioned dimensions, together with the problem setting, to characterize all existing approaches, it is worth noting that others are considered in the literature. For instance, Taylor and Stone (2009) consider three additional dimensions. (1) The *allowed learners*, i.e., what base reinforcement learning algorithm can be combined with the transfer method. (2) The *inter-task mappings* for cross-domain settings, i.e., whether the method requires mapping to be provided by an expert or learned. (3) The *source task selection*, i.e., whether the algorithm automatically selects what sources to transfer from or let an expert do the work.

Evaluation

The evaluation of a transfer method is a key aspect since it requires understanding when and how knowledge transfer is actually beneficial.

Performance Measures

Informally, we mentioned that transfer methods aim at improving the learning process of a target task. Since this improvement might be reflected along multiple dimensions, different performance measures have been proposed. Besides for evaluation, these measures are crucial for algorithm design since knowledge transfer is typically driven by the desired objectives. We describe the four most common performance measures, though others can be defined (Taylor and Stone, 2009). A visual overview is offered in Figure 3.1.

Jumpstart. This measure refers to the initial performance of the agent in terms of expected return. More specifically, we say that the algorithm achieves a "jumpstart" when its initial policy, computed before interacting with the target task, performs better than the initial policy of an algorithm that does not transfer knowledge at all. The latter algorithm is typically randomly initialized and thus the natural baseline is the uniform policy. By definition, this measure completely ignores the learning process as it only focuses on the initial performance. It is often the primary concern of multi-task learning approaches.

Asymptotic performance. In this case, the focus is on the final performance in the target task, i.e., after the learning process has been carried out. While theoretically this means evaluating the converge after an infinite number of learning steps, in practice the learning

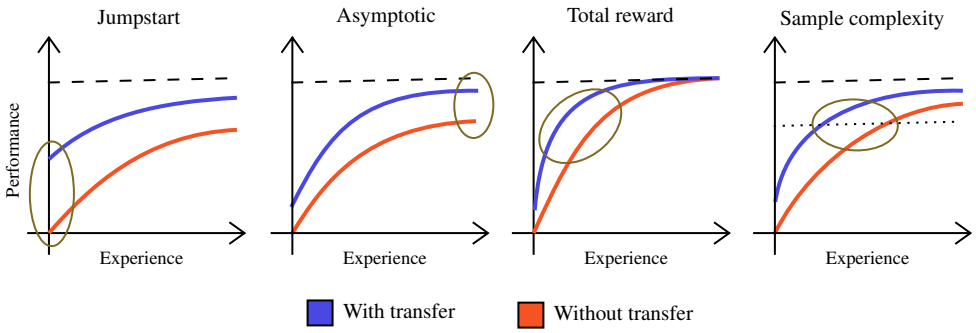

Figure 3.1: *The four performance measures. (1) Jumpstart: only the improvement in initial performance matters. (2) Asymptotic: the algorithms are evaluated based on the solution they converge to. (3) Total reward: the improvement is given by the area between the two learning curves. (4) Sample complexity: total number of samples (i.e., time) necessary to reach a certain threshold. Inspired by Figure 3 of Lazaric (2012).*

process is stopped after a pre-specified number of iterations and the performance are evaluated. The intuition is that, through prior knowledge, a transfer method might be able to converge to better solutions. Similarly to jumpstart, the behavior during learning does not matter.

Total reward or regret. This is perhaps the most natural performance measure to account for the whole learning process. The total reward collected by the agent in all learning steps of the target task is evaluated. Equivalently, one might look at the regret, the difference between the total reward of an oracle that executes the optimal policy at each step and that of the agent. This is where the exploration-exploitation dilemma is most important, and a transfer method that is able to improve this measure also improves the trade-off between these two objectives with respect to learning from scratch.

Sample complexity. The last measure involves the total number of experience samples collected by the agent while interacting with the target environment and before reaching a certain performance threshold. For instance, it is common to seek ϵ-optimal policies, in which case the performance threshold is triggered whenever the agent obtains expected return that is ϵ-close to the optimal one. Assuming that collecting each sample takes a fixed amount of time, this measure is equivalent to the *time-to-threshold* described by Taylor and Stone (2009). Differently from regret, here the actual rewards obtained during learning do not matter. This is closer to the setting of best arm/policy identification (e.g., Audibert and Bubeck, 2010).

Positive and Negative Transfer

When transferring knowledge to a target task, two important effects might occur. (1) *Positive transfer*. In this case, knowledge transfer goes well and there is a strictly positive improvement over learning from scratch. (2) *Negative transfer*. In this case, the reused

knowledge damages the learning process of the target task, which results worse than learning from scratch. We note that both negative and positive transfer refer to one or more of the performance measures introduced before. In some sense, it is even possible that an algorithm achieves both positive and negative transfer for the same target task. For instance, one might obtain positive transfer in jumpstart and negative transfer in regret or sample complexity since good initial performance does not necessarily imply a good learning process.

Avoiding negative transfer is one of the primary concerns in the literature. An algorithm that achieves this property is called *robust* to negative transfer. The definition of robustness depends on the chosen performance measure and potentially on other factors. As such, multiple definitions have been considered (Brunskill and Li, 2013; Mann and Choe, 2013; Zhan et al., 2016; Wang et al., 2019). Informally, we can say that an algorithm is robust to negative transfer if its performance is never, i.e., for any realizable choice of source and target tasks, significantly worse than learning the target task from scratch.

Survey of Transfer Methods

We briefly review the literature on transfer in reinforcement learning and related problems. The discussion is organized by problem setting.

Transfer Learning

Existing algorithms are mostly characterized by the kind of knowledge transferred. We shall thus group them by this feature according to the main alternatives introduced in Section 3.2.

Experience samples. Lazaric et al. (2008b) consider transferring samples, in the form of tuples $(S_t, A_t, S_{t+1}, R_{t+1})$, in batch model-free reinforcement learning. Their method selects the most promising samples to transfer by computing two similarity measures: compliance, a measure of global similarity between the samples of source and target task, and relevance, a measure of local similarity between a single source/target sample. Experimentally, they show good performance using FQI as the base learner. The same batch setting is considered by Laroche and Barlier (2017) under the additional assumption that tasks have the same state-transition dynamics. Their method directly reuses, i.e., without any selection, all reward samples to augment the dataset of FQI. Taylor et al. (2008) propose a method to transfer samples into a model-based algorithm – specifically, Fitted RMAX (Jong and Stone, 2007), though any other can in principle be used – when the tasks have different state-action spaces. The idea is to use an intertask mapping to map the source state-action space into the target one and then directly reuse the transformed samples to improve the model estimation of the target. Lazaric and Restelli (2011) analyze the transfer of samples in batch RL from a theoretical perspective. They demonstrate an interesting trade-off between the total number of samples and the number of tasks from which to transfer, a result that also appeared in the supervised learning literature (Crammer et al., 2008). While all these approaches assume to know which task generated each sample, Zhang and Zavlanos (2020) study the problem of transferring unlabeled instances, i.e., with unknown task identity.

Policies and options. Fernández and Veloso (2006) propose one of the earliest approaches for policy transfer under the assumption that only the reward function changes between tasks. Given a source policy and any reinforcement learning algorithm, the idea is to randomly choose, at each step, either the exploration mechanism of the base learner or the action prescribed by source policy itself. Taylor et al. (2007) consider a cross-domain policy search setting and use intertask mappings to directly transfer neural-network policies. Mahmud et al. (2013) cluster solved tasks in a few groups to ease policy transfer. Their approach is mostly useful in problems where the number of source tasks is very large, e.g., in sequential/lifelong settings. Once tasks are clustered so that only few source policies survive, transfer is carried out by a bandit algorithm (EXP3) that iteratively chooses one of the source policies and or a base learning agent. Azar et al. (2013c) consider the problem of finding the best policy (i.e., the one with highest return) among a set of source policies. They propose a UCB-based algorithm for which they derive sub-linear regret. A Bayesian approach for the same problem is designed by Rosman et al. (2016). Approaches to transfer options instead of policies are proposed by Konidaris and Barto (2007); Croonenborghs et al. (2007); Asadi and Huber (2007); Mehta et al. (2008). Among these, Konidaris and Barto (2007) introduce the concept of agent-space, a feature space that is invariant between task. Their method learns options in this space, thus allowing transfer between tasks with changing state spaces.

Value functions. Singh (1992) consider a composite target task that can be broken down into a sequence of simpler source tasks. They propose a methodology to reuse the action-value functions produced by Q-learning while solving the sub-tasks. Tanaka and Yamamura (2003) are among the first to focus on knowledge transfer in a multi-task/lifelong scenario. Their idea is to keep a running mean and standard deviation of the optimal action-value functions learned in previous tasks. Given a new task, the mean value function is used to initialize the learning process, while the standard deviation expresses the local reliability of this statistics. Taylor and Stone (2005) and Taylor et al. (2007) design a method to transfer an action-value function from a source domain to a target one with different state-action space. They use intertask mapping, human-provided transfer functionals that transform the source action-value function so that is applicable in the target domain. These works are extended by Liu and Stone (2006), who show how to build the intertask mappings automatically using qualitative dynamic Bayes networks. Abel et al. (2018) and Mann and Choe (2013) use value transfer with the aim of improving jumpstart. The idea is to build an optimistic initialization for the value function of the target task by leveraging the source value functions. By combining this initialization with any PAC algorithm (Strehl et al., 2009), they prove robustness to negative transfer in terms of sample complexity. While the method of Mann and Choe (2013) works in cross-domain settings using human-provided intertask mappings, the one of Abel et al. (2018) is discussed in the broader context of lifelong learning. Liu et al. (2019b) study how to reuse single-agent knowledge in a deep multi-agent setting. They define a novel measure of MDP similarity using n-step returns and leverage on it to build a method for value-function transfer.

Representations. The concept of "representation" is quite general and might refer to several components. Taylor and Stone (2007) define the agent's representation as either the learning algorithm, the function approximator, or the parameterization of the latter.

The authors design different methods to transfer each of these components. Perhaps the most immediate thing one associate to the concept of representation are state features and the related state abstractions (Li et al., 2006). These are considered by Walsh et al. (2006) in a transfer scenario where tasks share an underlying state representation. The proposed method learns this state abstraction from a set of source MDPs and directly transfers it to a target task. A similar approach is derived by Banerjee and Stone (2007) with a focus on value-function features and in the specific context of game playing. Ferrante et al. (2008) consider a problem where either the rewards or the transition probabilities change between tasks, but not both. They learn and transfer task representations by using a technique based on proto-value functions. Barreto et al. (2017) assume that tasks have linear reward functions in given features, and that both the transition dynamics and these features are fixed and shared, so that only the reward weights differ. They show that this shared feature representation allows to decompose value functions into reward weights and the so-called successor features, which are related to the transition dynamics and thus transferable. This enables efficient policy evaluation and improvement in new tasks. In a follow-up work (Barreto et al., 2018), the authors show that these features can be learned by deep neural networks without the need of being manually specified. A further extension is done by Lehnert and Littman (2018) for the case where also transition dynamics might change between tasks. A different perspective is considered by (Konidaris and Barto, 2006; Konidaris et al., 2012), who focus on reward shaping. The authors consider two representations for the transfer setting: the problem space, which is specific of each task and not transferable, and the agent space, which is specific of the agent and thus shared across tasks. The proposed method learns a shaping function in agent space, i.e., a function that maps agent-space variables to values and that is used for guiding the learning process in any new task. In a similar work, Snel and Whiteson (2011) empirically study different shaping functions learned from a set of source tasks.

Priors. The transfer of priors typically refers to a Bayesian reinforcement learning setting. The common underlying idea is use the source tasks to learn an informative prior on either an MDP, one of its components, or any higher-level information (such as an optimal policy or value function). Wilson et al. (2007, 2012) propose a hierarchical Bayesian model for the process generating tasks and experience samples. They effectively learn the distribution generating MDPs and use it as prior for driving exploration in the target task in a Thompson sampling fashion. Their framework is applicable in lifelong/multi-task settings. A similar approach is proposed by Liu et al. (2012) with a focus on linearly-parameterized value functions.

Multi-task Learning

While some of the works mentioned before characterize themselves as multi-task (e.g., Tanaka and Yamamura, 2003; Wilson et al., 2007), the focus is on how to perform knowledge transfer. Here we concentrate on approaches to either learn tasks jointly or to train single models (e.g., neural networks) that generalize well across tasks from the same distribution.

Learning tasks jointly. In this case, the agent is concurrently trained and evaluated on the same set of tasks, i.e., the source and target tasks coincide. The goal is to improve over learning each task separately, which can be achieved if the similarities allow to share knowledge among the learning processes. Lazaric and Ghavamzadeh (2010) consider a multi-task policy evaluation problem where the goal is to accurately evaluate a policy simultaneously on all tasks in a given set and assuming limited interactions. Using linear value functions with shared features, they propose hierarchical Bayesian models for the case where tasks share the same value-function prior and for the case where they belong to a small number of clusters. In both cases, they show that evaluating a policy jointly on similar tasks yields better performance, and requires fewer samples, than evaluation on each task separately. Calandriello et al. (2014) assume that the weight vectors of the (linear) action-value functions of the given tasks are jointly sparse, in the sense that they have the only a small number of non-zero components. They design a multi-task variant of FQI that leverages this property to simultaneously compute the solutions of all tasks while sharing this sparse representation.

Training multi-task models. Training single models that generalize across tasks has become increasingly popular with the successes of deep learning, especially thanks to the expressiveness power of neural networks. A common technique is to train a single deep neural network that outputs predictions (e.g., action values or probabilities) for multiple tasks in parallel while sharing lower-level features (Liu et al., 2016a; Yang et al., 2017; Hessel et al., 2019; D'Eramo et al., 2019). D'Eramo et al. (2019) theoretically study the benefits of learning and sharing common representations (i.e., features) while solving similar tasks concurrently. Empirical results, where feature sharing in deep neural networks is combined with different algorithms (including FQI and DQN), support this evidence. The problem is investigated under a computational perspective by Espeholt et al. (2018). The authors propose IMPALA, a distributed actor-critic framework that can optimize concurrently several high-dimensional (e.g., vision-based) tasks. The usage of hierarchical policies for multi-task learning is explored by Wulfmeier et al. (2019). The authors design a two-level policy, with the low level component capturing task-independent reusable skills and the high level one switching between these low-level controllers in a task-aware manner. The method is shown successful in several tasks, notably in a block stacking problem learned on a real robot. For the specific case of goal-based reinforcement learning (i.e., where tasks change only in the goal location), Schaul et al. (2015a) design universal value function approximators (UVFA). A UVFA is a value function $V_\omega(s, g)$ that generalizes over state-goal pairs. This means that, given a UVFA, one can evaluate a policy in any task (i.e., any goal state $g \in \mathcal{S}$). Similarly, the corresponding concept of goal-based policy $\pi_\theta(a|s, g)$ allows to capture (optimal) behavior across all tasks/goals simultaneously. An efficient way to train such a goal-based policy/value-function is hindsight experience replay (Andrychowicz et al., 2017), which uses the intuition that each transition in an unknown environment, despite not moving to the actual goal, leads to a state that could be the goal of some task and hence yields some information about how to solve the latter.

Some recent works have pointed out that training a single model for very different tasks is difficult, mostly because of conflicting objectives which might lead to oscillatory learning trends or to one task dominating all the others. Solutions to these problems are investigated by Hessel et al. (2019), who show how to make sure that all tasks have roughly

the same contribution to the learning process, by Yu et al. (2019), who show how to mitigate the effects of conflicting gradients, and by Bräm et al. (2019), who target the same problem using attention-based techniques. A different technique involves the *distillation* of policies for single tasks into one shared multi-task policy capturing common behavior (Rusu et al., 2015; Teh et al., 2017; Yin and Pan, 2017; Czarnecki et al., 2019). This policy distillation technique is mostly concerned of combining given policies, while training of the latter can be performed separately and thus does not suffer the above-mentioned problems. Once computed, the distilled policy can be used to guide or regularize the learning process of the single-task policies.

Meta-Reinforcement Learning

Although the first ideas date back to Schmidhuber (1987), meta-reinforcement learning has become very popular only recently with the successes of deep neural networks. We classify the recent literature in three main areas: recurrent methods, gradient-based methods, and task inference methods.

Recurrent methods. Recurrent methods represent "learning algorithms" by recurrent neural networks (e.g., LSTMs, Hochreiter and Schmidhuber, 1997) and directly meta-learn their parameters by backpropagation on tasks drawn from a common distribution. The resulting network is then evaluated on tasks from the same distribution. Though its parameters are fixed, the network still exhibits adaptive behavior at test time through its changing activations (Cotter and Conwell, 1990; Prokhorov et al., 2002). In some sense, the meta-learned network can be seen as a learning algorithm that is highly customized for the specific task distribution. While this approach was popularized earlier in supervised learning (Hochreiter et al., 2001; Santoro et al., 2016), two recent works evaluates it for reinforcement learning domains (Wang et al., 2016a; Duan et al., 2016). Here meta-training can be seen as learning a POMDP in which the current task is the unobserved latent variable. This justifies the usage of recurrent networks to approximate its non-Markovian optimal policy. Experiments in these works confirm that the meta-learned recurrent networks show learning behavior at test time, e.g., by trading off exploration and exploitation in target tasks.

Gradient-based methods. While recurrent approaches can ideally represent any learning behavior in a black-box manner, gradient-based methods focus on quickly adapting to new tasks by policy gradients. MAML (Finn et al., 2017) is perhaps the most popular algorithm in this class. The idea is to meta-train policy parameters that, when used as an initialization point for gradient ascent at test time, lead to quick convergence to an optimal policy in one or few steps. Formally, MAML's meta-training aims at optimizing the following objective:

$$\max_{\theta \in \Theta} \mathbb{E}_{\mathcal{M} \sim \mathfrak{D}} \left[J\left(\theta + \alpha \nabla_\theta J(\theta, \mathcal{M}), \mathcal{M}\right) \right]. \tag{3.2}$$

That is, the optimal policy parameters θ should be such that the expected return after taking one gradient step for a specific task is maximized in expectation across tasks. Of course all expectations involved cannot be computed and are approximated by sampling MDPs

from $\mathfrak{D}$ and interacting with them. A similar objective can be derived for the case where multiple gradient steps are performed.

The generality and flexibility of MAML allowed several extensions. Al-Shedivat et al. (2018) extend MAML to the non-stationary setting where tasks are generated by an underlying Markov chain (as opposed to the i.i.d. process of the original paper). The connection between MAML and inference in hierarchical Bayesian models is explored by Grant et al. (2018), while Yoon et al. (2018) propose an alternative Bayesian approach based on non-parametric variational inference. A similar probabilistic variant is presented by Finn et al. (2018), who meta-learn a distribution over parameters that is used to inject noise into the gradient updates at test time. Gupta et al. (2018a) propose a variant of MAML that meta-learns structured exploration strategies. The idea is to augment the policy with latent exploration variables that are meta-learned to inject structured stochasticity into the learning process.

One of the drawbacks of the objective in Equation 3.2 is that its sample-based approximation is often very noisy. Liu et al. (2019a) study different variance reduction techniques that involve the introduction of control variates into the MAML objective. Another limitation is that meta-learning a single vector of parameters for fast adaptation works well when the task distribution is mostly concentrated around certain tasks (e.g., unimodal), while it tends to fail when the distribution allows groups of very different tasks (e.g., multimodal). A multimodal extension of MAML which overcomes this problem is proposed by Vuorio et al. (2018, 2019). Finally, the convergence of gradient-based meta-learners is investigated by Fallah et al. (2020).

Task inference methods. The methods presented so far optimize a given loss function in a black box manner without fully exploiting the structure of reinforcement learning problems. Task inference methods, on the other hand, effectively model and infer uncertainties about the tasks, i.e., the MDP components or their latent representation. Rakelly et al. (2019) meta-learn latent context variables z, on which a multi-task policy $\pi_\theta(a|s,z)$ is conditioned, and prior/posterior distributions over them using amortized variational inference. At test time, these distributions enable efficient posterior sampling for exploring any new task. A similar approach is proposed by Humplik et al. (2019), who treat the problem as a POMDP with unobserved task variables and, as opposed to black-box recurrent methods, meta-learn a belief network able to infer these latent variables. Task inference at training time is conducted in a supervised way with different levels of privileged information, e.g., by knowing the true task parameters (which are however unknown at test time). Lan et al. (2019) train a task encoder to quickly produce a latent representation in new tasks. They combine it with a multi-task policy that directly produces optimal actions given the latent task parameterization. Similarly, Yang et al. (2019); Zhou et al. (2019) meta-learn a probing policy that explores new environments with the purpose of quickly identifying the latent parameters to be plugged into the multi-task optimal policy. While these approaches model the distributions of latent task variables as simple Gaussian, the usage of more complex multi-modal distributions involving Dirichlet priors is explored by Ren et al. (2020). For the case of tasks varying only in their transition dynamics, Sæmundsson et al. (2018) explore the usage of Gaussian processes (GPs). They model the global dynamics as a function of latent task variables over which they put a GP prior and show how to perform efficient inference of the latent task variables.

Part I

Transfer of Samples via Importance Sampling

CHAPTER 4

The Sample Transfer Problem

Introduction

In this part of the thesis, we focus on the problem of transferring experience samples. We recall from Chapter 3 that this problem involves reusing instances, the basic information that the agent collects while interacting with an environment, across different MDPs. More specifically, we call *experience sample* any collection of outcomes observed when taking one or more actions in some states. This could be either a single-step transition $(s_t, a_t, s_{t+1}, r_{t+1})$ or an entire trajectory $(s_0, a_0, s_1, \ldots, s_T)$. We shall consider the same setting and assumptions as in prior works (Lazaric et al., 2008b; Lazaric and Restelli, 2011). Formally, we are given a set of $m + 1$ MDPs $\{\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m\} \subseteq \mathfrak{M}$ where $\mathcal{M}_0$ is the target task and the remaining ones are the source tasks. Each task $\mathcal{M}_j$ is an MDP $\mathcal{M}_j = (\mathcal{S}, \mathcal{A}, P_j, U_j, \rho_j, \gamma)$ with shared state-action space and possibly different state-transition kernel P_j, reward kernel U_j, and initial state distribution ρ_j.[1] While we assume the same state-action space for all tasks, we note that all the techniques presented in this part extend easily to cross-domain settings by using inter-task mappings as done by Taylor et al. (2008). Besides this property, we do not make any additional assumption on the task generation process $\mathfrak{D}$. For each task $\mathcal{M}_j$, we suppose to have a dataset $\mathcal{D}_j$ of n_j experience samples (in either of the forms presented above). We require $n_j > 0$ for the source tasks ($j = 1, \ldots, m$), while it could be the case that $n_0 = 0$, i.e., we have no data from the target at our disposal. The goal is to reuse as many as possible of the source

[1] For ease of exposition, we explicitly work with the marginal distributions P_j and U_j, although we could straightforwardly consider the joint kernel as defined in Section 2.1.

samples to effectively increase the amount of data available for learning the target task $\mathcal{M}_0$, hence reducing the overall sample complexity.

Of course the problem poses many challenges. Unless the source tasks are equivalent to the target task, directly reusing their samples into estimators of target-related variables introduces *bias*. At the same time, increasing the amount of data for these estimators reduces their *variance*. A sample-reuse method is required to trade-off between these two quantities so as to maximize the quality (i.e., minimize error) of the resulting estimators. In some sense, we might say that transferring samples is all about managing the bias-variance trade-off (Crammer et al., 2008). We shall elaborate more on this topic later on with a concrete example.

Unfortunately, the amount of bias introduced by transferring from a certain source task is proportional to how the state-transition and reward kernels of the same task differ from those of the target. Since the MDP models are unknown for all tasks, whether some source samples might be useful or dangerous for the target has to be understood entirely from data. Some previous works ignore this problem, either because they perform direct transfer (Taylor et al., 2008) or because they make strong assumptions on the similarities between tasks (e.g., Laroche and Barlier (2017) assume shared transition models). Some more general approaches (Lazaric et al., 2008b; Lazaric and Restelli, 2011) focus on explicit *selection* of the source samples whose distribution appears closer to the target one. The idea is to compute different similarity measures between MDPs, mostly based on non-parametric estimators of their models, so as to understand what could be reused. Once the samples are selected, they are directly used in the learning process of the target task as if they were generated by the latter. There are at least two limitations. (1) Even if the distribution of the transferred source samples is similar to the one in the target task, the direct reuse introduces some bias that is not accounted for. (2) The similarity metrics proposed by Lazaric et al. (2008b) are mostly heuristics and do not facilitate the analysis. In particular, it is hard to establish whether the algorithm is robust to negative transfer. Lazaric and Restelli (2011) derive theoretical results and well-funded sample selection strategies but for a restricted setting where it is possible to actively generate an arbitrary amount of samples from the source tasks.

In this part of the thesis, we propose novel approaches for the sample-reuse problem that, differently from previous methods, do not require any sample selection. We take a different perspective and transfer *all* the given samples, while re-weighting their contribution to the learning process of the target task based on the similarity between domains (i.e., based on how likely their are to be generated from the target itself). For this purpose, we employ *importance sampling* (Owen, 2013), a standard technique for dealing with distribution mismatches. More specifically, we propose sample-reuse algorithms for two different settings: batch reinforcement learning (Chapter 5), which is considered in all prior works, and online policy search (Chapter 6), which has never been studied before in this context. In both cases, we propose importance-weighted variants of standard reinforcement learning algorithms that only require as input the set of source and target samples together with a corresponding importance weight. Since the whole idea is built on top of a theoretically-funded concept, this enables the derivation of theoretical results for both settings. More precisely, we derive error bounds on the resulting estimators and, under certain conditions, we establish robustness to negative transfer. Since, as we shall see, computing the ideal importance weights requires evaluating density ratios of the form

$\frac{P_0(S_{t+1}|S_t,A_t)}{P_j(S_{t+1}|S_t,A_t)}$ or $\frac{U_0(S_{t+1}|S_t,A_t)}{U_j(S_{t+1}|S_t,A_t)}$, we propose different methods to estimate these weights from samples. Finally, we provide several numerical simulations that show good performance on different continuous domains and the superiority over state-of-the-art baselines.[2]

The remaining part of this chapter is organized as follows. Section 4.2 provides some motivations on why the transfer of experience samples is important in practice. Section 4.3 provides a concrete example on the relation between sample-reuse and the bias-variance trade-off. Finally, Section 4.4 gives an overview of the importance sampling techniques that we adopt.

Why Transferring Samples?

There are at least three reasons why the transfer of experience samples is an appealing problem to study.

(1) *Samples are decoupled from the specific learning algorithms*. In fact, any agent interacting with an environment collects, and possibly stores, state-transitions and rewards. The process generating these, conditioned on the agent's actions, is indeed independent of the learning algorithm and only related to the environment dynamics. More generally, samples can be generated from any distribution. This is a particularly relevant property since it implies high flexibility in practice. For instance, samples could be reused across very different learning algorithms or even from human to artificial agents.

(2) *Transferring samples does not require the source tasks to be solved*. In fact, regardless of the fact that the actions performed in the source instances are optimal or not, the corresponding transition and reward instances yield some information about the source environments. When such environments are related to the target one, transferring these instances could effectively augment the dataset used to perform any estimation task, such as policy evaluation or improvement. This is a quite important property since the majority of algorithms for knowledge transfer do assume the source tasks to be solved or that nearly-optimal solutions are available.

(3) *A large amount of experience from related environments is often available in practice*. Consider, for instance, training an industrial robot to perform a certain task, such as moving objects or assessing the quality of given products. In this setting, the same robot might be required to perform different tasks over time, or the same task might be performed by different robots, e.g., because an outdated component is replaced by a new one. While different robots have different parameters and, thus, dynamics, one often stores their historical operations (i.e., our experience samples), which can be used by human operators or artificial learning agents to tune new controllers. The same situation arises in most real-world control problems where either the control system, the surrounding environment, or the desired objectives change over time, including robotics, self-driving vehicles, power plants, water reservoirs, and so on. The only requirement is that the system stores its operations over time, regardless of how these operations where performed.

[2] It is worth noting that a recent MS thesis (Paterniti, 2020) obtained promising results by applying our sample reuse methods to real driving data to transfer knowledge across cars with different configurations.

Sample Reuse vs Bias-Variance Trade-off

Consider the following simplified problem with respect to the one introduced at the beginning of this chapter. We have only one source task $\mathcal{M}_1$ with a dataset of n_1 reward samples $\mathcal{D}_1 = \{(S_{i,1}, A_{i,1}, R_{i,1}) | i = 1, \dots, n_1\}$. Similarly, we have a dataset $\mathcal{D}_0 = \{(S_{i,0}, A_{i,0}, R_{i,0}) | i = 1, \dots, n_0\}$ of n_0 samples from the target task, potentially with $n_0 \ll n_1$. Suppose all samples are independent. Moreover, assume that the MDPs are finite and that the goal is to estimate the mean reward of the target task $r_0(s, a)$. For a fixed state-action pair (s, a), an unbiased estimator using only the target samples is

$$\hat{r}'_0(s,a) = \frac{1}{n_0(s,a)} \sum_{i=1}^{n_0} R_{i,0} \mathbb{1}\left\{S_{i,0} = s, A_{i,0} = a\right\},$$

where we define $n_j(s,a) := \sum_{i=1}^{n_j} \mathbb{1}\left\{S_{i,j} = s, A_{i,j} = A\right\}$ as the number of samples from (s,a) in task $j \in \{0,1\}$. For simplicity, assume that n_j are fixed and deterministic. Recalling that rewards are bounded by $r_{\max}$ and using Hoeffding's inequality (Boucheron et al., 2003), it is easy to prove the following concentration result for $\hat{r}'_0(s,a)$. For any $\delta \in (0,1)$, with probability at least $1 - \delta$,

$$|\hat{r}'_0(s,a) - r_0(s,a)| \leq r_{\max} \sqrt{\frac{8 \log \frac{2}{\delta}}{n_0(s,a)}}.$$

So $\hat{r}'_0(s,a)$ converges to its expected value at rate $1/\sqrt{n_0(s,a)}$. An alternative estimator that directly reuses all the source samples is

$$\hat{r}''_0(s,a) = \frac{1}{n_0(s,a) + n_1(s,a)} \sum_{j \in \{0,1\}} \sum_{i=1}^{n_j} R_{i,j} \mathbb{1}\left\{S_{i,j} = s, A_{i,j} = A\right\}.$$

Let us analyze its properties. Using the independence of the samples, it is easy to see that its expectation is

$$\mathbb{E}\left[\hat{r}''_0(s,a)\right] = \frac{n_0(s,a) r_0(s,a) + n_1 r_1(s,a)}{n_0(s,a) + n_1(s,a)}.$$

Note that, unless $r_1(s,a) = r_0(s,a)$, $\mathbb{E}\left[\hat{r}''_0(s,a)\right] \neq r_0(s,a)$, hence the estimator is biased for $r_0(s,a)$. Let us now study its error as above. We have

$$\begin{aligned}
|\hat{r}''_0(s,a) - r_0(s,a)| &\leq |\mathbb{E}\left[\hat{r}''_0(s,a)\right] - r_0(s,a)| + |\hat{r}''_0(s,a) - \mathbb{E}\left[\hat{r}''_0(s,a)\right]| \\
&\leq \underbrace{\frac{n_1(s,a)}{n_0(s,a) + n_1(s,a)} |r_0(s,a) - r_1(s,a)|}_{\text{bias}} + \underbrace{r_{\max} \sqrt{\frac{8 \log \frac{2}{\delta}}{n_0(s,a) + n_1(s,a)}}}_{\text{variance}},
\end{aligned}$$

where we applied the triangle inequality and Hoeffding's bound. This in fact reveals a bias-variance trade-off. The bias vanishes when either $r_1(s,a) = r_0(s,a)$ or $n_0(s,a) \to \infty$. The variance decreases at a rate of $1/\sqrt{n_0(s,a) + n_1(s,a)}$. So which estimator is better, $\hat{r}'_0(s,a)$ or $\hat{r}''_0(s,a)$? The answer is non-trivial. $\hat{r}''_0(s,a)$ achieves smaller variance since it effectively uses more samples than $\hat{r}'_0(s,a)$, but it also introduces bias. Intuitively, when the bias is small because $r_0(s,a)$ is similar to $r_1(s,a)$ and $n_0(s,a) \ll n_1(s,a)$, reusing the source samples significantly increases the quality of the resulting estimator.

Importance sampling

In many applications, with reinforcement learning being among the most popular, we are interested in computing the expected value $\mu = \mathbb{E}[f(X)]$ of a function f applied to some random variable X when observing samples from a distribution that is different from the one that generates X. A common example is off-policy evaluation (e.g., Precup, 2000), where the goal is to compute the value function V^π or the expected return $J(\pi)$ of some target policy π provided only with samples from a different behavioral policy π_b. Formally, consider a measurable space $(\mathcal{X}, \mathcal{F})$, a function $f : \mathcal{X} \to \mathbb{R}$, and two probability measures p and q absolutely continuous with respect to the Lebesgue measure. With some abuse of notation, let p and q denote their respective densities. Suppose that p is absolutely continuous with respect to q ($p \ll q$), i.e., $p(x) = 0$ whenever $q(x) = 0$. Importance sampling (IS, Owen and Zhou, 2000) relies on the following simple rewriting of the target expectation:

$$\mathbb{E}_p[f(X)] = \int_{\mathcal{X}} f(x)p(x)\mathrm{d}x = \int_{\mathcal{X}} \frac{p(x)}{q(x)} f(x)q(x)\mathrm{d}x = \mathbb{E}_q\left[\frac{p(X)}{q(X)} f(X)\right],$$

where we write $\mathbb{E}_p[\cdot]$ to denote the expectation operator when the underlying probability measure is p. Therefore, the expectation of $f(X)$ under p can be rewritten as the expectation of $w(X)f(X)$ under q, where $w(X) = \frac{p(X)}{q(X)}$ is called *density ratio* or likelihood ratio or *importance weight*. In statistics, p is often called nominal distribution, while q is called *proposal* or importance distribution. In our transfer setting, we shall often refer to p and q as *target* and *source* distributions, respectively. Assuming that the density ratio $w(X)$ can be evaluated, the importance sampling estimator for $\mu = \mathbb{E}_p[f(X)]$ given n i.i.d. samples from q is

$$\hat{\mu}^{\text{IS}} := \frac{1}{n} \sum_{i=1}^{n} \frac{p(X_i)}{q(X_i)} f(X_i), \quad X_i \sim q. \tag{4.1}$$

The following result characterized the mean and variance of this estimator.

Theorem 4.4.1 (Theorem 9.1 of Owen (2013))**.** *Let $p \ll q$ and $\hat{\mu}^{IS}$ be the importance sampling estimator for $\mu = \mathbb{E}_p[f(X)]$ using n i.i.d. samples from q. Then, $\mathbb{E}_q[\hat{\mu}^{IS}] = \mu$ and*

$$\mathbb{V}\mathrm{ar}_q[\hat{\mu}^{IS}] = \frac{1}{n}\left(\int_{\mathcal{X}} \frac{f(x)^2 p(x)^2}{q(x)}\mathrm{d}x - \mu^2\right).$$

Hence, the IS estimator is *unbiased*, while its variance is low whenever q is roughly proportional to fp. Recall that, if $\hat{\mu}$ is the corresponding estimator which uses n i.i.d. samples directly from p, its variance is

$$\mathbb{V}\mathrm{ar}_p[\hat{\mu}] = \frac{1}{n}\left(\int_{\mathcal{X}} f(x)^2 p(x)\mathrm{d}x - \mu^2\right).$$

Hence, the two estimators are difficult to compare in general and their variance highly depends on the chosen distributions. Unfortunately, while the variance of $\hat{\mu}$ is always bounded above if f is bounded, the same property might not hold for $\hat{\mu}^{\text{IS}}$, whose variance

can even become infinite (see, e.g., Example 9.1 of Owen (2013)). This happens when p and q are quite different, so that p gives high probability mass to regions where q has low mass and f is large. This is a fundamental problem in our transfer settings where we have no control (or only partial control) on the distributions involved and we want to prevent negative transfer. For this reason, we now discuss two more robust estimators than plain importance sampling.

Self-Normalized Importance Sampling

The self-normalized importance sampling (SNIS) estimator is defined as

$$\hat{\mu}^{\text{SNIS}} := \frac{\sum_{i=1}^{n} w(X_i) f(X_i)}{\sum_{i=1}^{n} w(X_i)}, \quad X_i \sim q. \tag{4.2}$$

That is, instead of dividing everything by the number of samples n, we use the total sum of importance weights. This technique has two main benefits. First, it can be used even when the distributions p, q can be evaluated only up to a normalization constant. Second, it has typically much lower variance than the plain IS estimator. Unfortunately, the self-normalized estimator is consistent but biased (see, e.g., Theorem 9.2 of Owen (2013)). Although there is no general theoretical dominance, in practice the self-normalize estimator often yields superior performance than the IS one.

Multiple Importance Sampling

Multiple importance sampling (MIS) (Veach and Guibas, 1995; Veach, 1997) deals with the case where samples from multiple proposal distributions are available. Consider the same setting as before, now with $m+1$ probability measures $p_0, p_1, \dots, p_m$, where p_0 is the target one. Suppose that, for $j = 1, \dots, m$, we are given $n_j > 0$ i.i.d. samples from distribution p_j, while we have access to $n_0 \geq 0$ (possibly zero) i.i.d. samples from the target. The MIS estimator for $\mu := \mathbb{E}_{p_0}[f(X)]$ is

$$\hat{\mu}^{\text{MIS}} = \sum_{j=0}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} h_j(X_{i,j}) \frac{p_0(X_{i,j})}{p_j(X_{i,j})} f(X_{i,j}), \quad X_{i,j} \sim p_j. \tag{4.3}$$

The function h is often referred to as *heuristics* and must be a partition of unity, i.e., $\sum_{j=0}^{m} h_j(x) = 1$ for all $x \in \mathcal{X}$. It is easy to show (Veach and Guibas, 1995) that the MIS estimator is *unbiased* for *any* choice of h that satisfies this property. For instance, by choosing $h_j(x) = \frac{n_j}{\sum_{l=0}^{m} n_l}$ we recover the standard IS estimator (with multiple proposals). A common and convenient choice for h is the *balance heuristics*, $h_j(x) = \frac{n_j p_j(x)}{\sum_{l=1}^{m} n_l p_l(x)}$, for which the MIS estimator reduces to an IS estimator with a mixture of proposals,

$$\hat{\mu}^{\text{BH}} = \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} \frac{p_0(X_{i,j})}{\bar{p}_\alpha(X_{i,j})} f(X_{i,j}), \quad X_{i,j} \sim p_j, \tag{4.4}$$

where $n = \sum_{j=0}^{m} n_j$ and $\bar{p}_\alpha(x) = \sum_{j=0}^{m} \alpha_j p_j(x)$, with $\alpha_j = \frac{n_j}{n}$. This estimator is also known in the literature as deterministic mixture sampling or stratification (Hesterberg,

1995; Owen and Zhou, 2000). It can be shown equivalent to mixture importance sampling (see Section 9.11 of Owen (2013)), with the difference that samples are not really generated by the mixture distribution $\bar{p}_\alpha$. A key property is that, when $n_0 > 0$, the resulting weights are *bounded* by $\frac{n}{n_0}$ since, for all $x \in \mathcal{X}$,

$$\frac{p_0(x)}{\bar{p}_\alpha(x)} = \frac{p_0(x)}{\frac{n_0}{n}p_0(x) + \underbrace{\sum_{j=1}^{m} \frac{n_j}{n} p_j(x)}_{\geq 0}} \leq \frac{n}{n_0}.$$

This fact also implies that the variance of these weights is bounded, a property that, as we have seen before, rarely holds for plain IS. For these reasons, samples from p_0 are often referred to as *defensive*.

Effective Sample Size

The *effective sample size* (ESS) is an important measure of the goodness of an IS estimator. Informally, the ESS is the number n_{eff} such that the variance of the Monte Carlo estimator using n_{eff} independent samples from p matches that of the IS scheme. A common approximation is $\text{ESS} := \frac{n}{1+\mathbb{V}\text{ar}_q[p(x)/q(x)]}$ (Liu, 1996). Equivalently, since the variance at the denominator is $\mathbb{V}\text{ar}_q[p(x)/q(x)] = \int_{\mathcal{X}} \frac{p(x)^2}{q(x)} \mathrm{d}x - 1$, we can write $\text{ESS} = \frac{n}{d_2(p\|q)}$, where

$$d_2(p\|q) := \int \frac{p(x)^2}{q(x)} \mathrm{d}x \tag{4.5}$$

is the exponentiated second-order Renyi divergence. A connection between moments of the importance weights and Renyi divergences has been shown by Cortes et al. (2010). One way to estimate this quantity from samples, though many exist (see, e.g., Martino et al. (2017)), is

$$\widehat{\text{ESS}} = \frac{n}{1 + \frac{1}{n}\sum_{i=1}^{n}(w(X_i) - 1)^2}, \quad X_i \sim q. \tag{4.6}$$

Control Variates

Control variates are a widely applied variance reduction technique for general Monte Carlo estimators (Hammersley and Handscomb, 1964). The underlying idea is that a random variable with known expectation could be used to reduce the variance of a mean estimator for another random variable. One example are the baselines adopted in policy gradient approaches (see Section 2.5.2), which rely on the fact that the log-policy gradients have null expected value. Owen and Zhou (2000) have popularized the usage of control variates for IS and MIS. Let ψ be any vector of functions such that $\mathbb{E}_p[\psi(X)] = 0$. Then, the IS estimator with control variates is

$$\hat{\mu}^{\text{ISCV}} := \frac{1}{n}\sum_{i=1}^{n} \frac{p(X_i)f(X_i) - \beta^T\psi(X_i)}{q(X_i)}, \quad X_i \sim q, \tag{4.7}$$

where β is any vector of parameters. Similarly, the MIS estimator with balance heuristic and control variates is

$$\hat{\mu}^{\text{BHCV}} = \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}\frac{p_0(X_{i,j})f(X_{i,j}) - \beta^T\psi(X_{i,j})}{\bar{p}_\alpha(X_{i,j})}, \quad X_{i,j} \sim p_j. \tag{4.8}$$

It is easy to show that these estimators are unbiased for any choice of β, so one can optimize for the latter variable in order to minimize variance. In the specific case of MIS with balance heuristic, Owen and Zhou (2000) show that the proposal distributions composing the mixture, whose integral is known to be 1, can be used as very effective control variates. In this case, we have that $\psi_j(x) = \frac{p_j(x)}{\bar{p}_\alpha(x)} - 1$ for $j = 0, \ldots, m$. The following theorem is one of the most important results in providing the robustness of the MIS estimator with optimal control variates.

Theorem 4.4.2 (Theorem 2 of Owen and Zhou (2000))**.** *Let $\hat{\mu}^{BHCV}$ be the MIS estimator with balance heuristic and control variates given by $\psi_j(x) = \frac{p_j(x)}{\bar{p}_\alpha(x)} - 1$ for $j = 0, \ldots, m$. Suppose that the estimator uses the control-variate parameters that minimize $\mathbb{V}\text{ar}[\hat{\mu}^{BHCV}]$. Then,*

$$\mathbb{V}\text{ar}[\hat{\mu}^{BHCV}] \leq \min_{j\in\{0,1,\ldots,m\}} \frac{\mathbb{V}\text{ar}_{p_j}[f(X)p_0(X)/p_j(X)]}{n\alpha_j}.$$

In the right-hand side, it is possible to recognize the variances of the single IS estimator that uses only samples from p_j, divided by α_j. This implies that the MIS estimator with optimal control variates is never much worse than the IS scheme using the single *best* source distribution available. Notably, when $n_0 > 0$ defensive samples are available, this theorem implies that the variance of the MIS estimator is never worse than the one of not using importance corrections at all.

Corollary 4.4.1. *Assume the same conditions as in Theorem 4.4.2 and that $n_0 > 0$. Then,*

$$\mathbb{V}\text{ar}[\hat{\mu}^{BHCV}] \leq \frac{\mathbb{V}\text{ar}_{p_0}[f(X)]}{n_0}.$$

We shall rely on this result later on to guarantee robustness to negative transfer for our algorithms.

CHAPTER 5

Importance Weighted Fitted Q-Iteration

This chapter is based on the paper "Importance Weighted Transfer of Samples in Reinforcement Learning" co-authored with Andrea Sessa, Matteo Pirotta, and Marcello Restelli and published at ICML 2018.

Transfer in Batch Reinforcement Learning

In this chapter, we focus on the transfer of experience samples in *batch* reinforcement learning. This is a particularly relevant setting that is often encountered in practice, where one has access to historical data from a set of different environments, without the possibility to collect more, and aims at learning the optimal controller for one of them. The formal setting is the one introduced in Chapter 4, with a set of $m+1$ unknown tasks $\{\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_m\} \subseteq \mathfrak{M}$ where $\mathcal{M}_0$ is the target one. Each task $\mathcal{M}_j$ is an MDP $\mathcal{M}_j = (\mathcal{S}, \mathcal{A}, P_j, U_j, \gamma)$, where we ignore the initial state distribution since it will not be used by the value-based algorithms we consider. From each MDP, we have a dataset $\mathcal{D}_j = \{(S_{i,j}, A_{i,j}, S'_{i,j}, R_{i,j}) | i = 1, \dots, n_j\}$ of n_j experience samples. We have that $S'_{i,j} \sim P_j(\cdot|S_{i,j}, A_{i,j})$ and $R_{i,j} \sim U_j(\cdot|S_{i,j}, A_{i,j})$ for each $j = 0, 1, \dots, n_j$ and $i = 1, \dots, n_j$. For simplicity, we assume that the couples $(S_{i,j}, A_{i,j})$ are i.i.d. from a common fixed distribution $\nu \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$, regardless of the task index. This assumption is only for simplifying the exposition and analysis, but it can be relaxed. We require $n_j > 0$ for all tasks, including the target one, possibly with $n_0 \ll n_j$ for all $j \geq 1$. The goal is to reuse all these data to augment the dataset used to learn the target MDP $\mathcal{M}_0$, so as to obtain better solutions than learning $\mathcal{M}_0$ using only the n_0 target samples (i.e., without

transfer).

Example. Consider the following real-world example, which shall be one of our concrete experiments later on. The problem is *water reservoir management*, where the goal is to decide the amount of water stored into, and released from, the reservoir (e.g., a lake) by controlling its dam. The control is driven by different, contrasting, objectives, including satisfying water demand from external entities (such as farmers), ensuring a minimum storage level, and preventing the capacity from exceeding a flooding threshold. According to our model of the system, which follows a real-world study of Lake Como (Castelletti et al., 2010), interacting with the environment is extremely time expensive; an action, in fact, consist of choosing the amount of water to release in a day and its effects are observed only after 24 hours. This makes learning by online interactions prohibitive. However, we often have access to historical operational data from the dam, i.e., the system logs actions, typically taken by a human expert or a manually tuned controller, together with their effects. Even better, often data from multiple reservoirs can be obtained. These reservoirs might be located in different geographical regions than the one we intend to control, and might be controlled to purse different objectives. Climate changes (which impact the water inflow due to natural causes, such as rains or ice melting) and different objectives imply that the available datasets follow different distributions, i.e., the underlying MDP models differ. Anyway, we can still hope to exploit possible similarities to effectively augment the data available for controlling the target reservoir which, given the prohibitive sample complexity, might be fundamental for even learning something useful.

Our approach. As is common in batch reinforcement learning (Lange et al., 2012), we shall design a value-based algorithm whose goal is to approximate the optimal action-value function $Q^\star(s, a)$ of $\mathcal{M}_0$. Our proposed solution is an importance-weighted variant of Fitted Q-Iteration (FQI), our no-transfer baseline. Unlike previous works (Taylor et al., 2008; Lazaric et al., 2008b; Laroche and Barlier, 2017), our method (1) does not require any source sample selection, and (2) it decouples rewards and state transitions and transfers them separately. This chapter is mostly devoted to the theoretical analysis of our algorithm when provided with any set of weights, not necessarily the true unknown importance weights, and to its empirical evaluation. In order to carry out the latter, we also present a simple method to estimate the true weights from samples, though much better estimators are presented in the next chapter.

Outline. Our detailed contributions are as follows.

1. We propose Importance-Weighted Fitted Q-Iteration (IWFQI), a novel algorithm for the transfer of samples in batch reinforcement learning (Section 5.2). IWFQI transfers all the given samples, while re-weighting their contribution to the learning process using importance sampling;

2. Building on top of existing results for approximate value iteration, we derive a finite-sample analysis of IWFQI that is agnostic to the specific weights used by the algorithm (Section 5.3);

3. Using Gaussian process (GP) regression, we design a simple model-based technique to estimate the importance weights from samples (Section 5.4);

4. We report numerical simulations in different domains (Section 5.5). The results showcase the benefits of IWFQI over state-of-the-art baselines (Lazaric et al., 2008b; Laroche and Barlier, 2017) and its robustness to negative transfer. Our experiments include a simulated variant of the water-reservoir management problem described above.

Importance-Weighted Fitted Q-Iteration

Let us begin by recalling how FQI (Ernst et al., 2005) works when applied to solve the target MDP $\mathcal{M}_0$ (refer to Section 2.5.1 for the pseudo-code). Given a hypothesis space $\mathcal{Q} \subset \mathcal{B}(\mathcal{S} \times \mathcal{A}, q_{\max})$[1] of limited capacity, FQI starts from an initial guess $Q_0 \in \mathcal{Q}$ for the optimal action-value function $Q^{\star}$[2]. Let $\mathcal{D}_0 = \{(S_{i,0}, A_{i,0}, S'_{i,0}, R_{i,0}) | i = 1, \dots, n_0\}$ be the target dataset. At each iteration $k \geq 0$, FQI approximates the application of the optimal Bellman operator in $\mathcal{Q}$ using $\mathcal{D}_0$, such that $Q_{k+1} \approx T^{\star} Q_k$. Formally, it computes

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{Q}} \left\| Q - \widehat{T}^{\star} Q_k \right\|_{\mathcal{D}_0}^2, \tag{5.1}$$

where $\widehat{T}^{\star}$ is the empirical Bellman operator induced by $\mathcal{D}_0$ (see Definition 2.5.1), such that $(\widehat{T}^{\star} Q)(S_i, A_i) := R_i + \gamma \max_{a \in \mathcal{A}} Q(S'_i, a)$. Problem (5.1) is an instance of empirical risk minimization where the couples $(S_i, A_i) \sim \nu$ are the covariates, $Y_i = (\widehat{T}^{\star} Q_k)(S_i, A_i)$ are the targets, and a squared loss function is used. Since $\mathcal{D}_0$ contains only target samples, then $\left\| Q - \widehat{T}^{\star} Q_k \right\|_{\mathcal{D}_0}^2$ is an unbiased estimator of the "true" loss function, i.e.,

$$\begin{aligned}
\mathbb{E}\left[\left\| Q - \widehat{T}^{\star} Q_k \right\|_{\mathcal{D}_0}^2 \right] &= \mathbb{E}_{\substack{(S_i, A_i) \sim \nu, \\ S'_i \sim P_0(\cdot | S_i, A_i), \\ R_i \sim U_0(\cdot | S_i, A_i)}} \left[\frac{1}{n_0} \sum_{i=1}^{n_0} \left| Q(S_i, A_i) - \widehat{T}^{\star} Q_k(S_i, A_i) \right|^2 \right] \\
&= \mathbb{E}_{(S,A) \sim \nu} \left[\mathbb{E}_{\substack{S' \sim P_0(\cdot | S, A), \\ R \sim U_0(\cdot | S, A)}} \left[\left| Q(S, A) - R + \gamma \max_{a \in \mathcal{A}} Q_k(S', a) \right|^2 \right] \right].
\end{aligned}$$

Now suppose that we adopt a naive transfer approach where we form a large dataset $\mathcal{D} = \bigcup_{j=0}^{m} \mathcal{D}_j$ comprising the samples from all given MDPs and directly feed it into FQI to solve (5.1). This approach suffers from sample selection bias (Cortes et al., 2008) and $\left\| Q - \widehat{T}^{\star} Q_k \right\|_{\mathcal{D}}^2$ is no longer an unbiased estimator. In fact, using the decomposition as above, it is easy to see that its expected value is

$$\frac{1}{n} \mathbb{E}_{(S,A) \sim \nu} \left[\sum_{j=0}^{m} n_j \mathbb{E}_{\substack{S' \sim P_j(\cdot | S, A), \\ R \sim U_j(\cdot | S, A)}} \left[\left| Q(S, A) - R + \gamma \max_{a \in \mathcal{A}} Q_k(S', a) \right|^2 \right] \right],$$

[1] Differently from other works (e.g., Farahmand and Precup, 2012; Tosatto et al., 2017), we suppose, for the sake of simplicity, the hypothesis space to be bounded by $q_{\max} \leq r_{\max}/(1-\gamma)$. This assumption can be relaxed by considering truncated functions. We refer the reader to Györfi et al. (2006) for the theoretical consequences of such relaxation.

[2] Throughout this chapter, task-dependent quantities, such as optimal value functions or Bellman operators, for which we do not specify the task implicitly refer to the target $\mathcal{M}_0$

which differs from the one derived before that involves only expectations under $\mathcal{M}_0$. However, it is easy to obtain an unbiased estimator using any of the importance sampling techniques presented in the previous section. For instance, using plain importance sampling, it is enough to multiply the squared loss term associated to each instance $(S_{i,j}, A_{i,j}, S'_{i,j}, R_{i,j})$ by the density ratio

$$w_{i,j} := \frac{P_0(S'_{i,j}|S_{i,j}, A_{i,j})U_0(R_{i,j}|S_{i,j}, A_{i,j})}{P_j(S'_{i,j}|S_{i,j}, A_{i,j})U_j(R_{i,j}|S_{i,j}, A_{i,j})}. \tag{5.2}$$

Imagine for the moment that these weights could be computed exactly for each sample. This is clearly not possible in practice since we do not know the MDP models and later on we shall see how these weights can be approximated. Then, by feeding FQI on the full dataset $\mathcal{D}$ with samples weighted by $w_{i,j}$, we get an algorithm that automatically selects which samples to transfer, i.e., those that, based on the importance weights, are more likely to be generated from the target MDP. This approach looks appealing but presents one important limitation. Consider a simple case where we have a source MDP with the same state-transition dynamics as the target, but with entirely different reward. Then, the importance weights defined above are likely to be very close to zero for any source sample, thus making transfer useless. This because transition and reward samples are transferred jointly, so it is enough that the distribution of one changes significantly to discard the entire couple. However, we would like a method able to leverage the fact that transition dynamics do not change, thus transferring only that part of the sample.

To overcome this limitation, we use the intuition that FQI initialized with $Q_0(s, a) = 0$ fits the target reward function at the first iteration. This leads us to the following variation of FQI, which effectively *decouples* reward and transition samples. At the first iteration, we use all the samples to fit a model $\widehat{r}_0 \approx r_0$ of the target reward function. More specifically, we solve the following weighted regression problem:

$$\widehat{r}_0 = \operatorname*{argmin}_{r \in \mathcal{Q}} \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} w^r_{i,j} \left| r(S_{i,j}, A_{i,j}) - R_{i,j} \right|^2, \tag{5.3}$$

where $\mathcal{Q}$ is the same hypothesis space we consider to represent action-value functions. Since no state-transition samples are involved in this step, it is enough to consider importance weights over the reward distributions. For instance, using plain IS,

$$w^r_{i,j} := \frac{U_0(R_{i,j}|S_{i,j}, A_{i,j})}{U_j(R_{i,j}|S_{i,j}, A_{i,j})}, \tag{5.4}$$

and similarly for MIS or other weighting schemes. Then, we set $Q_0 = \widehat{r}_0$ and, at each iteration $k \geq 0$, we update the current action-value function as

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{Q}} \frac{1}{n} \sum_{j=0}^{m} \sum_{i=0}^{n_j} w^p_{i,j} \left| Q(S_{i,j}, A_{i,j}) - \widetilde{T}^\star Q_k(S_{i,j}, A_{i,j}) \right|^2, \tag{5.5}$$

where $\widetilde{T}$ such that $\widetilde{T}^\star Q_k(S_{i,j}, A_{i,j}) := \widehat{r}_0(S_{i,j}, A_{i,j}) + \gamma \max_a Q_k(S'_{i,j}, a)$ is a *modified* empirical Bellman operator. Intuitively, instead of considering the random rewards $R_{i,j}$, we use the initial estimate $\widehat{r}_0$ of r_0. Similarly as before, since the stochasticity due to the

Algorithm 2 Importance Weighted Fitted Q-Iteration

Require: Number of iterations K, hypothesis space $\mathcal{Q}$, augmented dataset $\widetilde{\mathcal{D}} = \bigcup_{j=0}^{m} \bigcup_{i=1}^{n_j} \left\{ (S_{i,j}, A_{i,j}, S'_{i,j}, R_{i,j}, \widetilde{w}^r_{i,j}, \widetilde{w}^p_{i,j}) \right\}$
Ensure: Action-value function Q_K, greedy policy π_K
  Fit target reward: $\widehat{r}_0 \leftarrow \operatorname{argmin}_{r \in \mathcal{Q}} \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} w^r_{i,j} \left| r(S_{i,j}, A_{i,j}) - R_{i,j} \right|^2$
  Set initial action-value function: $Q_0 \leftarrow \widehat{r}_0$
  for $k = 0, \ldots, K-1$ **do**
    Compute regression targets:
      $Y_{i,j} \leftarrow \widetilde{T}^{\star} Q_k(S_{i,j}, A_{i,j}) := \widehat{r}_0(S_{i,j}, A_{i,j}) + \gamma \max_a Q_k(S'_{i,j}, a)$
    Update action-value function:
      $Q_{k+1} = \operatorname{argmin}_{Q \in \mathcal{Q}} \frac{1}{n} \sum_{j=0}^{m} \sum_{i=0}^{n_j} w^p_{i,j} \left| Q(S_{i,j}, A_{i,j}) - Y_{i,j} \right|^2,$
  end for
  Compute greedy policy: $\pi_K(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a) \;\; \forall s \in \mathcal{S}$
  return Q_K and π_K

reward samples is now removed, only the state-transition kernel needs to be included in the importance weights,

$$w^p_{i,j} := \frac{P_0(S'_{i,j} | S_{i,j}, A_{i,j})}{P_j(S'_{i,j} | S_{i,j}, A_{i,j})}. \tag{5.6}$$

The resulting method (see Algorithm 2), named Importance-Weighted Fitted Q-Iteration (IWFQI), uses this decomposition to enable separate transfer of rewards and state transitions. Of course, in practice the MDP models are unknown and thus the exact importance weights given above (or any variant involving the true distributions) cannot be computed. Therefore, IWFQI is defined in terms of user-provided weights $\widetilde{w}^r_{i,j}$ and $\widetilde{w}^p_{i,j}$. This could be obtained, for instance, by first estimating the MDP models or by directly fitting the density ratios (Sugiyama et al., 2012). We postpone a discussion of estimation techniques to Section 5.4. On the other hand, the following section is devoted to a theoretical analysis of IWFQI as a function of the chosen weights, in a way that is independent of how they are estimated. Intuitively, we study what happens when combining IWFQI with arbitrary weights instead of the true ones (which cannot be computed in practice) so as to better understand the consequences of their estimation from samples.

Theoretical Analysis

Since the main purpose of our analysis is to show what happens when combining an approximate value iteration algorithm with data from different environments, we consider the simplified setting where we have $n = n_1$ samples from a single source task ($m = 1$), but no samples from the target task ($n_0 = 0$). A generalization to the case where target samples or samples from more sources are available is straightforward, and it only complicates our derivation. Some hints on what our results would look like in such a more general case can be found in Section 4.3. Furthermore, we recall that the results provided in this section are *independent from the way the importance weights are estimated*. We note that, differently from existing analyses of the transfer of samples (Crammer et al.,

2008; Lazaric and Restelli, 2011), whose focus is on the trade-off between transfer bias and variance, here we concentrate on the implications of correcting distribution shift as in other works from the supervised learning literature (Cortes et al., 2010; Garcke and Vanck, 2014).

Following prior analyses of approximate value iteration methods (Munos and Szepesvári, 2008; Farahmand, 2011), our measure of interest is $\|Q^\star - Q^{\pi_K}\|_{1,\mu}$, i.e., the expected performance loss, under some distribution $\mu \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$, of the policy π_K returned by IWFQI (which is greedy with respect to Q_K) with respect to the optimal policy $\pi^\star(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q^\star(s, a)$. Here μ is an arbitrary evaluation distribution over $\mathcal{S} \times \mathcal{A}$ that can be freely chosen. In practice, it might coincide with the sampling distribution ν.

Consider the sequence of action-value functions $Q_0, Q_1, \ldots, Q_K$ computed by IWFQI. At each iteration $k \geq 0$, we incur in an error

$$\epsilon_k := T^\star Q_k - Q_{k+1} \tag{5.7}$$

in approximating the optimal Bellman operator. Since IWFQI belongs to the family of approximate value iteration algorithms, we can resort to Theorem 3.4 of Farahmand (2011), which conveniently decomposes $\|Q^\star - Q^{\pi_K}\|_{1,\mu}$ in terms of the per-iteration errors ϵ_k.

Theorem 5.3.1 (Theorem 3.4 of Farahmand (2011))**.** *Let K be a positive integer and $q_{\max} \leq \frac{r_{\max}}{1-\gamma}$. Then, for any sequence $\{Q_k\}_{k=0}^K \subset \mathcal{B}(\mathcal{S} \times \mathcal{A}, q_{\max})$ and the corresponding sequence $\{\epsilon_k\}_{k=0}^K$, we have*

$$\|Q^\star - Q^{\pi_K}\|_{1,\mu} \leq \frac{2\gamma}{(1-\gamma)^2} \left[2\gamma^K q_{\max} + \inf_{b \in [0,1]} \sqrt{C_{\mu,\nu}(K; b) \sum_{k=0}^{K-1} \alpha_k^{2b} \|\epsilon_k\|_\nu^2} \right].$$

The coefficients $C_{\mu,\nu}$ involve the Radon-Nikodym derivative between the evaluation distribution μ and the sampling distribution ν. Intuitively, we have state-action pairs drawn from ν but we evaluate the error according to μ, so the more different are these two distributions, the larger is the bound. The coefficients α_k depend only on the discount factor and are of minor relevance. We refer the reader to Chapter 3 of Farahmand (2011) for the full expressions of these two quantities.

The bound given in Theorem 5.3.1 holds for any approximate value iteration algorithm. The errors $\|\epsilon_k\|_\nu^2$ are the only components depending on the specific learning algorithm. Thus, our problem reduces to bounding such errors for the specific case of IWFQI. Note that our algorithm performs importance-weighted regression at each iteration, hence these terms can be interpreted as the corresponding regression errors. Cortes et al. (2010) already provided a theoretical analysis of importance-weighted regression. However, their results are not immediately applicable to our case since they consider a regression problem where the shift is on the covariate distribution and the target variable is a deterministic function of the input. On the other hand, we have a regression estimation problem where the target is a random variable whose distribution changes across MDPs, and we want to learn its conditional expectation given the input. Therefore, before bounding $\|\epsilon_k\|_\nu^2$, we generalize the results of Cortes et al. (2010) to our setting.

Error Bound for Importance-Weighted Regression

We consider the following general setting, which characterizes all regression problems performed by IWFQI. Let $\mathcal{H} \subset \mathcal{B}(\mathcal{X}, f_{\max})$ be the hypothesis space. Suppose we have a dataset $\mathcal{D} = \{(X_i, Y_i) | i = 1, \dots, n\}$ of n i.i.d. samples, where $X_i \sim \nu$ and $Y_i \sim p_1(\cdot|X_i)$. With some abuse of notation, we denote by $p_1\nu$ their joint distribution. The goal is to approximate the regression function with respect to the target distribution p_0,

$$h^\star(x) := \mathbb{E}_{Y \sim p_0(\cdot|x)}[Y]. \tag{5.8}$$

To this purpose, we solve the following weighted regression problem:

$$\widehat{h}(x) := \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \widetilde{w}(X_i, Y_i) |h(X_i) - Y_i|^2 \right\}, \tag{5.9}$$

where $\widetilde{w}(X_i, Y_i)$ is an arbitrary positive weighting function. Let $w(x, y) = \frac{p_0(y|x)}{p_1(y|x)}$ be the "ideal" weight function. The following result bounds the regression error $\|\widehat{h} - h^\star\|_\nu$. It is the equivalent of Theorem 4 of Cortes et al. (2010) in our setting and we believe it is of independent interest.

Theorem 5.3.2. *Consider the importance-weighted regression problem outlined above. Suppose that* $|Y| \leq f_{\max}$ *almost surely,* $d = \operatorname{pdim}(\{|h(x) - y|^2 \ : h \in \mathcal{H}\}) < \infty$[3]*, and* $\mathbb{E}_{X \sim \nu, Y \sim p_1(\cdot|X)}[\widetilde{w}(X, Y)^2] < \infty$*. Then, for any* $\delta > 0$*, the following holds with probability at least* $1 - 2\delta$*:*

$$\begin{aligned} \|\widehat{h} - h^\star\|_\nu \leq \inf_{h \in \mathcal{H}} \ & \|h - h^\star\|_\nu + f_{\max} \sqrt{\|\varphi\|_{1,\nu}} + 2 f_{\max} \|\widetilde{w} - w\|_{p_1\nu} \\ & + 2^{13/8} f_{\max} \sqrt{\upsilon(\widetilde{w})} \left(\frac{d \log \frac{2ne}{d} + \log \frac{4}{\delta}}{n} \right)^{\frac{3}{16}}, \end{aligned}$$

where $\varphi(x) := \mathbb{E}_{Y \sim p_1(\cdot|x)}[\widetilde{w}(x, Y)] - 1$ *and* $\upsilon(\widetilde{w}) := \sqrt{\mathbb{E}_{\nu, p_1}[\widetilde{w}(X, Y)^2]} + \sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}(X, Y)^2]}$*, with* $\widehat{\mathbb{E}}_{\mathcal{D}}$ *denoting the empirical expectation on* $\mathcal{D}$*.*

Proof. Applying Hölder's inequality, for all $h \in \mathcal{H}$,

$$\mathbb{E}_{p_1\nu} \left[\left(\widetilde{w}(X, Y) |f(X) - Y|^2 \right)^2 \right] \leq 16 f_{\max}^4 \mathbb{E}_{p_1\nu}[\widetilde{w}(X, Y)^2] < \infty.$$

Thanks to the fact that the loss has bounded second moment, we can apply Corollary 1 of Cortes et al. (2010) to $L_h(x, y) := \widetilde{w}(x, y) |h(x) - y|^2$. We have that, with probability at least $1 - \delta$,

$$\begin{aligned} \mathbb{E}_{p_1\nu}[\widetilde{w}(X, Y) |\widehat{h}(X) - Y|^2] \leq \frac{1}{n} \sum_{i=1}^{n} \widetilde{w}(X_i, Y_i) |\widehat{h}(X_i) - Y_i|^2 + \\ 2^{13/4} f_{\max}^2 \sqrt{\mathbb{E}_{p_1\nu}[\widetilde{w}(X, Y)^2]} \left(\frac{d \log \frac{2ne}{d} + \log \frac{4}{\delta}}{n} \right)^{3/8}. \end{aligned} \tag{5.10}$$

[3] We denote by $\operatorname{pdim}(\cdot)$ the pseudo-dimension of a real-valued function class.

Let us now expand the left-hand side of (5.10):

$$\begin{aligned}
\mathbb{E}_{p_1\nu}\left[\widetilde{w}(X,Y)|\widehat{h}(X)-Y|^2\right] &= \mathbb{E}_{\nu}\left[\mathbb{E}_{p_1}\left[\widetilde{w}(X,Y)\left(\widehat{h}^2(X)+Y^2-2\widehat{h}(X)Y\right)\mid X\right]\right] \\
&= \mathbb{E}_{\nu}\Big[\widehat{h}^2(X)\mathbb{E}_{p_1}[\widetilde{w}(X,Y)\mid X]+\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y^2\mid X] \\
&\quad -2\widehat{h}(X)\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]\pm\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]^2\pm\widehat{h}^2(X)\Big] \\
&= \mathbb{E}_{\nu}\Bigg[\left(\widehat{h}(X)-\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]\right)^2+\widehat{h}^2(X)\mathbb{E}_{p_1}[\widetilde{w}(X,Y)\mid X] \\
&\quad +\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y^2\mid X]-\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]^2-\widehat{h}^2(X)\Bigg] \\
&= \|\widehat{h}-\widetilde{h}\|_{\nu}^2+\mathbb{E}_{\nu}[\widehat{h}^2(X)\left(\mathbb{E}_{p_1}[\widetilde{w}(X,Y)-1\mid X]\right)]+z,
\end{aligned}$$

where $z=\mathbb{E}_{\nu}[\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y^2\mid X]-\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]^2]$ is a constant term (independent of $\widehat{h}$) and $\widetilde{h}(x)=\mathbb{E}_{p_1}[\widetilde{w}(x,Y)Y\mid x]$ is the regression function weighted by $\widetilde{w}$. Plugging this into (5.10), we obtain

$$\begin{aligned}
\|\widehat{h}-\widetilde{h}\|_{\nu}^2+\mathbb{E}_{\nu}[\widehat{h}^2(X)\left(\mathbb{E}_{p_1}[\widetilde{w}(X,Y)-1\mid X]\right)]+z \le \frac{1}{n}\sum_{i=1}^{n}\widetilde{w}(X_i,Y_i)|\widehat{h}(X_i)-Y_i|^2 \\
+2^{13/4}f_{\max}^2\sqrt{\mathbb{E}_{p_1\nu}[\widetilde{w}(X,Y)^2]}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{3/8}.
\end{aligned} \tag{5.11}$$

Consider now the hypothesis $h_0\in\mathcal{H}$ such that $h_0=\operatorname{argmin}_{h\in\mathcal{H}}\ \|h-\widetilde{h}\|_{\nu}^2$. Since h_0 is in $\mathcal{H}$ and $\widehat{h}$ was defined as the hypothesis minimizing the empirical weighted loss, we have

$$\frac{1}{n}\sum_{i=1}^{n}\widetilde{w}(X_i,Y_i)|\widehat{h}(X_i)-Y_i|^2\le\frac{1}{n}\sum_{i=1}^{n}\widetilde{w}(X_i,Y_i)|h_0(X_i)-Y_i|^2. \tag{5.12}$$

Similarly to what we did for $\widehat{h}$, we can bound the empirical error of h_0. According to Corollary 1 of (Cortes et al., 2010), we have that for any $\delta>0$, with probability at least $1-\delta$

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\widetilde{w}(X_i,Y_i)|h_0(X_i)-Y_i|^2 \le \mathbb{E}_{p_1\nu}[\widetilde{w}(X,Y)|h_0(X)-Y|^2] \\
+2^{13/4}f_{max}^2\sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}(X,Y)]}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{3/8}.
\end{aligned}$$

Then, using exactly the same argument as before,

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\widetilde{w}(X_i,Y_i)|h_0(X_i)-Y_i|^2 \le \inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_{\nu}^2+\mathbb{E}_{\nu}[h_0^2(X)\left(\mathbb{E}_{p_1}[\widetilde{w}(X,Y)-1\mid X]\right)]+z \\
+2^{13/4}f_{max}^2\sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}(X,Y)]}\left(\frac{d\log\frac{2Ne}{d}+\log\frac{4}{\delta}}{N}\right)^{3/8}.
\end{aligned} \tag{5.13}$$

If we now put (5.11) and (5.13) together by means of (5.12), we get that, with probability at least

$1-2\delta$

$$\begin{aligned}\|\widehat{h}-\widetilde{h}\|_\nu^2 \leq & \inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_\nu^2+\mathbb{E}_\nu\left[\left(h_0^2(X)-\widehat{h}^2(X)\right)\left(\mathbb{E}_{p_1}[\widetilde{w}(X,Y)-1\mid X]\right)\right]\\ &+2^{13/4}f_{max}^2\left(\sqrt{\mathbb{E}_{p_1\nu}[\widetilde{w}(X,Y)^2]}+\sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}(X,Y)^2]}\right)\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{3/8}\\ &\leq \inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_\nu^2+f_{max}^2\|\varphi\|_{1,\nu}+2^{13/4}f_{max}^2\upsilon(\widetilde{w})\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{\frac{3}{8}}.\end{aligned} \tag{5.14}$$

By taking the square root of both sides of (5.14) and using $\sqrt{\sum_i a_i}\leq\sum_i\sqrt{a_i}$ for $a_i\geq 0$, we obtain

$$\|\widehat{h}-\widetilde{h}\|_\nu\leq\inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_\nu+f_{max}\sqrt{\|\varphi\|_{1,\nu}}+2^{13/8}f_{max}\sqrt{\upsilon(\widetilde{w})}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}. \tag{5.15}$$

Furthermore,

$$\inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_\nu\leq\inf_{h\in\mathcal{H}}\|h-h^\star\|_\nu+\|h^\star-\widetilde{h}\|_\nu. \tag{5.16}$$

We can now bound the expected error of $\widehat{h}$ with respect to $h^\star$ by

$$\|\widehat{h}-h^\star\|_\nu\leq\|\widehat{h}-\widetilde{h}\|_\nu+\|\widetilde{h}-h^\star\|_\nu. \tag{5.17}$$

We already provided a bound on the first term, so let us analyze the second one. We have

$$\begin{aligned}\|\widetilde{h}-h^\star\|_\nu^2&=\mathbb{E}_\nu[|\widetilde{h}(X)-h^\star(X)|^2]=\mathbb{E}_\nu[|\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]-\mathbb{E}_{p_0}[Y\mid X]|^2]\\ &=\mathbb{E}_\nu[|\mathbb{E}_{p_1}[\widetilde{w}(X,Y)Y\mid X]-\mathbb{E}_{p_1}[w(X,Y)Y\mid X]|^2]\\ &=\mathbb{E}_\nu[|\mathbb{E}_{p_1}[Y\left(\widetilde{w}(X,Y)-w(X,Y)\right)\mid X]|^2]\\ &\leq\mathbb{E}_\nu[\mathbb{E}_{p_1}[|Y|^2|\widetilde{w}(X,Y)-w(X,Y)|^2\mid X]]\\ &\leq f_{max}^2\mathbb{E}_{p_1\nu}\left[|\widetilde{w}(X,Y)-w(X,Y)|^2\right]\\ &=f_{max}^2\|\widetilde{w}-w\|_{p_1\nu}^2.\end{aligned} \tag{5.18}$$

where the first inequality is due to Jensen's inequality. Thus, $\|\widetilde{h}-h^\star\|_\nu\leq f_{max}\|\widetilde{w}-w\|_{p_1\nu}$. By combining (5.15), (5.16), (5.17), and (5.18), we have

$$\begin{aligned}&\|\widehat{h}-h^\star\|_\nu\leq\|\widehat{h}-\widetilde{h}\|_\nu+\|\widetilde{h}-h^\star\|_\nu\\ &\leq\inf_{h\in\mathcal{H}}\|h-\widetilde{h}\|_\nu+f_{\max}\sqrt{\|g\|_{1,\nu}}+2^{13/8}f_{\max}\sqrt{\upsilon(\widetilde{w})}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}+\|\widetilde{h}-h^\star\|_\nu\\ &\leq\inf_{h\in\mathcal{H}}\|h-h^\star\|_\nu+f_{\max}\sqrt{\|g\|_{1,\nu}}+2^{13/8}f_{\max}\sqrt{\upsilon(\widetilde{w})}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}+2\|\widetilde{h}-h^\star\|_\nu\\ &\leq\inf_{h\in\mathcal{H}}\|h-h^\star\|_\nu+f_{\max}\left(\sqrt{\|g\|_{1,\nu}}+2^{\frac{13}{8}}\sqrt{\upsilon(\widetilde{w})}\left(\frac{d\log\frac{2ne}{d}+\log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}+2\|\widetilde{w}-w\|_{p_1\nu}\right)\end{aligned}$$

which concludes the proof. □

Error Bound for IWFQI

We now use Theorem 5.3.2 to bound the errors $\|\epsilon_k\|_\nu$. Let us first introduce some useful definitions. Let $\widetilde{w}_r(\cdot|s,a)$ and $\widetilde{w}_p(\cdot|s,a)$ be the chosen weight functions. The "pipe" notation is used to indicate that the weights are on the conditional distributions given a state-action pair. Similarly, let w_r and w_p be the "true" density rations. We only require the following assumption, common in importance-weighted regression (Cortes et al., 2010).

Assumption 5.3.1 (Bounded second moment)**.** *The second moments of the chosen weight functions are bounded:* $\mathbb{E}_{R\sim U_1(\cdot|s,a)}[\widetilde{w}_r(R|s,a)^2] < \infty$ *and* $\mathbb{E}_{S'\sim P_1(\cdot|s,a)}[\widetilde{w}_p(S'|s,a)^2] < \infty$ *for all* $(s,a) \in \mathcal{S}\times\mathcal{A}$.

Similarly to Theorem 5.3.2, let $\varphi_r(s,a) = \mathbb{E}_{R\sim U_1(\cdot|s,a)}[\widetilde{w}_r(R|s,a)] - 1$ and $\varphi_p(s,a) = \mathbb{E}_{S'\sim P_1(\cdot|s,a)}[\widetilde{w}_p(S'|s,a)] - 1$. Furthermore, define the following two functions

$$\upsilon_r(\widetilde{w}_r) := \sqrt{\mathbb{E}_{U_1\nu}[\widetilde{w}_r(R|S,A)^2]} + \sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}_r(R|S,A)^2]}, \tag{5.19}$$

$$\upsilon_p(\widetilde{w}_p) := \sqrt{\mathbb{E}_{P_1\nu}[\widetilde{w}_p(S'|S,A)^2]} + \sqrt{\widehat{\mathbb{E}}_{\mathcal{D}}[\widetilde{w}_p(S'|S,A)^2]}. \tag{5.20}$$

As before, we use the notation $U_1\nu$ and $P_1\nu$ to denote the joint distributions of state-action pairs and rewards and next states, respectively. The following corollary is immediate from Theorem 5.3.2.

Corollary 5.3.1. *Suppose that the pseudo-dimension* d *of* $\mathcal{Q}$ *is finite and that Assumption 5.3.1 holds. Let* $\widehat{r}_0$ *be as defined in* (5.3) *with* $n_0 = 0$ *and* $m = 1$*. Then, for any* $\delta > 0$*, with probability at least* $1 - 2\delta$*,*

$$\begin{aligned} ||r_0 - \widehat{r}_0||_\nu \le \inf_{r\in\mathcal{Q}} \; & ||r - r_0||_\nu + q_{\max}\sqrt{||\varphi_r||_{1,\nu}} + 2q_{\max}||\widetilde{w}_r - w_r||_{U_1\nu} \\ & + 2^{13/8} q_{\max}\sqrt{\upsilon_r(\widetilde{w}_r)}\left(\frac{d\log\frac{2ne}{d} + \log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}. \end{aligned} \tag{5.21}$$

Moreover, for any $k \ge 0$*, let* Q_{k+1} *be as defined in* (5.5) *with* $n_0 = 0$*,* $m = 1$*, and* $Q_k \in \mathcal{Q}$*. Then, for any* $\delta > 0$*, with probability at least* $1 - 2\delta$*,*

$$\begin{aligned} ||\bar{T}^\star Q_k - Q_{k+1}||_\nu \le \inf_{Q\in\mathcal{Q}} & ||Q - \bar{T}^\star Q_k||_\nu + q_{\max}\sqrt{||\varphi_p||_{1,\nu}} \\ & + 2q_{\max}||\widetilde{w}_p - w_p||_{P_1\nu} + 2^{13/8} q_{\max}\sqrt{\upsilon_p(\widetilde{w}_p)}\left(\frac{d\log\frac{2ne}{d} + \log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}}, \end{aligned} \tag{5.22}$$

where $\bar{T}^\star Q(s,a) := \widehat{r}_0(s,a) + \int_{\mathcal{S}} P_0(\mathrm{d}s'|s,a)\max_{a'} Q(s',a)$.

Equipped with this result, we are now ready to state the main error bound for IWFQI.

Theorem 5.3.3. *Let* $\{Q_l\}_{l=0}^{k+1}$ *be the sequence of action-value functions produced by IWFQI using a single source dataset of* $n = n_1$ *samples and no target sample. Then,*

for any $\delta > 0$, with probability at least $1 - 4\delta$:

$$\begin{aligned}\|T^*Q_k - Q_{k+1}\|_\nu \leq\ & q_{\max}\sqrt{\|\varphi_p\|_{1,\nu}} + 2q_{\max}\sqrt{\|\varphi_r\|_{1,\nu}} \\ & + 2q_{\max}\|\widetilde{w}_p - w_p\|_{P_1\nu} + 4q_{\max}\|\widetilde{w}_r - w_r\|_{U_1\nu} \\ & + \inf_{Q\in\mathcal{Q}}\|Q - (T^*)^{k+1}Q_0\|_\nu + 2\inf_{r\in\mathcal{H}}\|r - r_0\|_\nu \\ & + 2^{\frac{13}{8}}q_{\max}\left(\sqrt{\upsilon_p(\widetilde{w}_p)} + 2\sqrt{\upsilon_r(\widetilde{w}_r)}\right)\left(\frac{d\log\frac{2ne}{d} + \log\frac{4}{\delta}}{n}\right)^{\frac{3}{16}} \\ & + \sum_{i=0}^{k-1}(\gamma C_{\mathrm{AE}}(\nu))^{i+1}\|\epsilon_{k-i-1}\|_\nu,\end{aligned}$$

where C_{AE} is the concentrability coefficient of one-step transitions as defined in (Farahmand, 2011, Definition 5.2).

Proof. We can decompose the error at iteration k into:

$$\begin{aligned}\|\epsilon_k\|_\nu = \|T^\star Q_k - Q_{k+1}\|_\nu &\leq \|T^\star Q_k - \bar{T}^\star Q_k\|_\nu + \|\bar{T}^\star Q_k - Q_{k+1}\|_\nu \\ &= \|r_0 - \widehat{r}_0\|_\nu + \|\bar{T}^\star Q_k - Q_{k+1}\|_\nu,\end{aligned} \tag{5.23}$$

where $\bar{T}^\star$ is defined in Corollary 5.3.1 as the optimal Bellman operator of the target task using the approximated reward function defined in (5.3). The two terms in (5.23) can be bounded straightforwardly by applying Corollary 5.3.1. In the second term, this gives rise to $\inf_{Q\in\mathcal{Q}}\|Q - \bar{T}^\star Q_k\|_\nu$, which can be further bounded by noticing that

$$\inf_{Q\in\mathcal{Q}}\|Q - \bar{T}^\star Q_k\|_\nu \leq \inf_{Q\in\mathcal{Q}}\|Q - T^\star Q_k\|_\nu + \|T^\star Q_k - \bar{T}^\star Q_k\|_\nu. \tag{5.24}$$

The second term in (5.24) is again $\|r_0 - \widehat{r}_0\|_\nu$, while the first term has already been bounded in Theorem 5.3 of Farahmand (2011) as

$$\inf_{Q\in\mathcal{Q}}\|Q - T^\star Q_k\|_\nu \leq \inf_{Q\in\mathcal{Q}}\|Q - (T^\star)^{k+1}Q_0\|_\nu + \sum_{i=0}^{k-1}(\gamma C_{AE}(\nu))^{i+1}\|\epsilon_{k-i-1}\|_\nu. \tag{5.25}$$

The proof follows by combining the bounds from Corollary 5.3.1 with (5.24) and (5.25) and rearranging. □

Discussion. As expected, four primary sources of error contribute to our bound: (i) the *bias* due to estimated weights (first four terms), (ii) the *approximation error* (fifth and sixth term), (iii) the *estimation error* (seventh term), (iv) the *propagation error* (eighth term). Notice that, assuming to have a consistent estimator for the importance weights (an example is given in Section 5.4), the bias term vanishes as the number of samples n tends to infinity. Furthermore, the estimation error decreases with n, thus vanishing as the number of samples increases. Thus, in the asymptotic case our bound shows that the only source of error is due to the limited capacity of the functional space $\mathcal{Q}$ under consideration, as in all approximate value iteration algorithms. Furthermore, we notice that fitting the reward function and using it instead of the available samples propagates an error term through iterations, i.e., the approximation error $\inf_{r\in\mathcal{Q}}\|r - r_0\|_\nu$. If we were able to estimate the importance weights for the typical case where both reward and

transition samples are used, we could get rid of such error. However, since the resulting weights somehow depend on the joint reward-transition densities, we expect their variance, as measured by $\epsilon(\widetilde{w})$, to be much bigger, thus making the resulting bound even larger. Furthermore, we argue that, when the reward function is simple enough and only a limited number of samples is available, a separate fit might be beneficial even for plain FQI. In fact, the variance of the empirical optimal Bellman operator can be reduced by removing the source of stochasticity due to the reward samples at the cost of propagating a small approximation error through iterations. The bounds for approximate value iteration (e.g., Munos and Szepesvári, 2008; Farahmand, 2011; Farahmand and Precup, 2012) can be straightforwardly extended to such case by adopting a procedure similar to the one described in the proof of Theorem 5.3.3. In many practical applications the reward function is actually known and, thus, does not need to be fitted. In such cases, it is possible to get rid of the corresponding terms in Theorem 5.3.3, allowing transfer to occur without errors even when rewards are completely different between tasks.

Estimating the Importance Weights

We now propose a very simple method to estimate the importance weights. In general, this goal can be efficiently achieved by directly estimating density ratios (Sugiyama et al., 2012). Unfortunately, in our settings, this is not an appealing approach for one main reason: density ratios are not transferable, i.e., they must be recomputed for each new target task. Therefore, we decide to take a more indirect approach and estimate only the missing components, namely the reward and transition models. To this purpose, Gaussian processes (GPs) have been successfully adopted to model stochastic dynamical systems with high-dimensional and continuous state-action spaces (e.g., Kuss and Rasmussen, 2004; Deisenroth and Rasmussen, 2011; Doshi-Velez and Konidaris, 2016; Berkenkamp et al., 2017). Here we show how they can be adopted to estimate the importance weights.

For simplicity, let us focus on the reward model.[4] We assume that the mean-reward function r_j of each task is distributed according to a Gaussian process $r_j \sim \mathcal{GP}(0, \mathcal{K}_j)$ with a zero-mean prior and covariance function $\mathcal{K}_j$. Furthermore, we consider a Gaussian reward generation process, i.e., $U_j(\cdot|s,a) = \mathcal{N}(r_j(s,a), \sigma_j^2)$. When fitting the GP using the dataset $\mathcal{D}_j$ for the j-th task, we obtain a predictive distribution of the form

$$R|s,a \sim \mathcal{N}(\widehat{r}_j(s,a), \sigma_j^2 + \widehat{\sigma}_j^2(s,a)), \tag{5.26}$$

where $\widehat{r}_j$ and $\widehat{\sigma}_j^2$ are the predictive mean and variance, respectively. By directly using this distribution to estimate the importance weights, we obtain

$$\widetilde{w}_{i,j}^r = \frac{\mathcal{N}(R_{i,j}; \widehat{r}_0(S_{i,j}, A_{i,j}), \sigma_0^2 + \widehat{\sigma}_0^2(S_{i,j}, A_{i,j}))}{\mathcal{N}(R_{i,j}; \widehat{r}_j(S_{i,j}, A_{i,j}), \sigma_j^2 + \widehat{\sigma}_j^2(S_{i,j}, A_{i,j}))}. \tag{5.27}$$

A few remarks are due. First, the estimated weights converge to the true ones when the GP predictions are perfect, i.e., when $\widehat{r}_j(s,a) \to r_j(s,a)$ and $\widehat{\sigma}_j^2(s,a) \to 0$ for all $j = 0, \dots, m$. This is an advantage over density-ratio estimation approaches (Sugiyama et al., 2012), where the limited hypothesis space to represent weight functions introduces an

[4] The transition model can be treated analogously to the reward by considering a separate GP for each state dimension (Deisenroth and Rasmussen, 2011).

irreducible approximation error. Second, we note that using the predictive distribution instead of directly plugging the estimated means $\widehat{r}_k$ in place of the true ones adds the predictive variance to the process variance in the weights. This has a sort of regularization effect. In fact, the variance of the source densities, which is often the main cause behind large importance weights, is increased whenever the source GP is inaccurate, thus avoiding the weight from exploding. Finally, we point out that in the original paper (Tirinzoni et al., 2018b), we actually show that the estimator of Equation 5.27 is somehow related to the expected importance weights induced by the GP predictions. Here we neglect such result since it is of minor importance.

Numerical Simulations

We evaluate IWFQI on three different domains with increasing level of complexity. The first one, puddle world, is a synthetic grid-navigation task where both the transition and reward models are Gaussian, thus representing a perfect starting point for evaluating our method. The second one, acrobot, is a control problem in which, more realistically, such assumption does not hold anymore. However, we show this to have a negligible impact on our method. The third one is a simulated variant of the water-reservoir management problem of Section 5.1. We show the intrinsic difficulty of such problem and the importance of transferring samples from previous experience. In all experiments, we compare our method to two existing algorithms for transferring samples into FQI: the relevance-based transfer (RBT) algorithm of Lazaric et al. (2008b) and the shared-dynamics transfer (SDT) algorithm of Laroche and Barlier (2017). All results are averaged over 20 runs and are reported with 95% Student's t confidence intervals. We refer the reader to the original paper (Tirinzoni et al., 2018b) for additional details on the experiments.

Puddle World

Our first experimental domain is a modified version of the puddle world environment presented in Sutton (1996). Puddle world is a two-dimensional continuous grid with a goal area and some elliptical "puddles". The goal is to drive the agent from a starting position to the goal area while avoiding the puddles. The state-space is $[0, 10]^2$, while the action-space is discrete and allows the agent to move in the four cardinal directions. At each time-step, the agent receives a reward of -1 plus a penalization proportional to the distance from all puddles: $R_{t+1} = -1 - 100 \sum_{u \in \mathcal{U}} W_u(S_{t+1})$, where $\mathcal{U}$ is the set of puddles and $W_u(s)$ is the weight of puddle u for state s. In the goal the reward is zero. In our experiments, we modeled $W_u(s)$ as a bivariate Gaussian. Each action moves the agent by α in the corresponding direction. In particular, we consider two versions of the environment: I) *shared dynamics* where $\alpha = 1$ and II) *puddle-based dynamics* where puddles also slow-down the agent by $\alpha = (1 + 5 \sum_{u \in \mathcal{U}} W_u(S_t))^{-1}$. Finally, a white Gaussian noise of $\sigma_r^2 = 0.01$ and $\sigma_p^2 = 0.04$ is added to the reward and the transition model, respectively. In our experiments we set $\gamma = 0.99$ and a maximum horizon of 50 time-steps. We consider three source tasks and one target task, as depicted in Figure 5.1. Notice that the optimal paths to solve each task have at least a small overlapping, thus allowing some knowledge transfer. However, the optimal policy for one task is likely to cross a puddle if carelessly used in another domain. This makes the transfer problem more challenging since the algo-

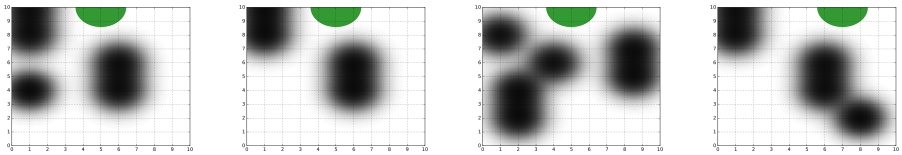

Figure 5.1: *From left to right: the target task and the three source tasks. The agent always starts in the bottom-left corner and must reach the goal area (shown in green). Puddles are shown in black.*

rithm has to figure out which samples should be retained and which should be discarded. For each source task, we generate a dataset of 20 episodes from a nearly-optimal policy. We run IWFQI with weights computed as in Section 5.4, where we set the model noise to be ten times the true value. For evaluating our weight estimation procedure, we also run IWFQI with ideal importance weights (computed as the ratio of the true distributions). In each algorithm, FQI is run for 50 iterations with Extra-Trees (Ernst et al., 2005). An ϵ-greedy policy ($\epsilon = 0.3$) is used to collect data in the target task.

Shared dynamics. We start by showing the results for $\alpha = 1$ in Figure 5.2*(left)*. As expected, FQI alone is not able to learn the target task in such a small number of episodes. On the other hand, IWFQI has a good jump-start and converges to an optimal policy in only 20 episodes. Interestingly, IWFQI with ideal weights has almost the same performance, thus showing the robustness of our weight estimation procedure. RBT also learns the optimal policy rather quickly. However, the limited number of target and source samples available in this experiment makes it perform significantly worse in the first episodes. Since in this version of the puddle world the dynamics do not change between tasks, SDT also achieves good performance, converging to a nearly-optimal policy.

Puddle-based dynamics. We also show the results for the more challenging version of the environment were puddles both penalize and slow-down the agent (Figure 5.2*(right)*). Notice that, in this case, transition dynamics change between tasks, thus making the transfer more challenging. Similarly, as before, our approach quickly learns the optimal policy and is not affected by the estimated weights. Furthermore, the benefits of over-estimating the model noise can be observed from the small improvement over IWFQI-ID. RBT is also able to learn the optimal policy. However, the consequences of inaccurately computing compliance and relevance are more evident in this case, where the algorithm negatively transfers samples in the first episodes. Finally, SDT still shows an improvement over plain FQI, but it is not able to learn the optimal policy due to the bias introduced by the different dynamics.

Acrobot

Acrobot (Sutton and Barto, 2018) is a classic control problem where the goal is to swing-up a two-link pendulum by applying positive or negative torque to the joint between the two links. Due to its non-linear and complex dynamics, Acrobot represents a very challenging problem, requiring a considerable amount of samples to be solved. The state-space

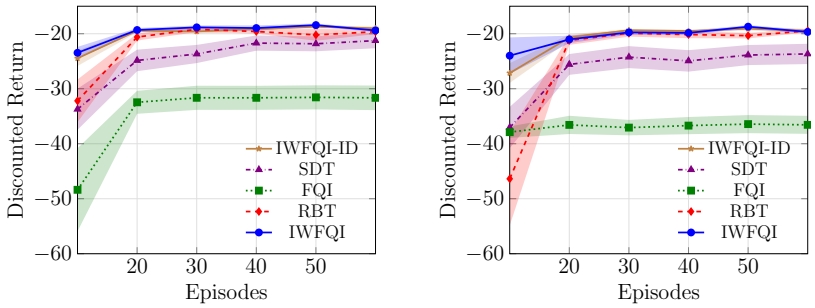

Figure 5.2: *Puddle world with* 20×3 *episodes transferred from* 3 *source tasks in the case of shared dynamics (left) and puddle-based dynamics (right).*

is composed of the two link angles (θ_1, θ_2) and their velocities $(\dot{\theta}_1, \dot{\theta}_2)$. The agent can only apply a torque of $+2$ or -2 to the joint between the two links. The initial state is $(\theta_1, 0, 0, 0)$, where $\theta_1 \sim \mathcal{U}(-2, 2)$. Performance is evaluated starting from multiple states $(\theta_1, 0, 0, 0)$, with θ_1 evenly spaced in $[-2, 2]$. In this experiment, we consider a multi-task scenario where robots might have different link lengths (l_1, l_2) and masses (m_1, m_2). Our target task is the classic Acrobot *swing-up* problem, where the robot has lengths $(1.0, 1.0)$ and masses $(1.0, 1.0)$. The swing-up task has reward

$$R_{sw}(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = -\cos(\theta_1) - \cos(\theta_1 + \theta_2) - 2, \tag{5.28}$$

and terminates whenever $-cos(\theta_1) - cos(\theta_1 + \theta_2) > 1$ or 100 time-steps are reached. We consider two source tasks. The first is another swing-up task where the robot has lengths $(1.1, 0.7)$ and masses $(0.9, 0.6)$. The second is a *constant-spin* task, where the goal is to make the first joint rotate at a fixed constant speed, with lengths $(0.95, 0.95)$ and masses $(0.95, 1.0)$. The constant-spin task has reward

$$R_{cs}(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = -|\dot{\theta}_1 - \pi|, \tag{5.29}$$

and terminates whenever 100 time-steps are reached. Notice the intrinsic difficulty of transfer: the first source task has the same reward as the target but very different dynamics, and conversely for the second source task. Using nearly-optimal policies, we generate 100 episodes from the first source and 50 episodes from the second. We run all algorithms (except SDT since the problem violates the shared-dynamics assumption) for 200 episodes and average over 20 runs.

Results. Results are shown in Figure 5.3*(left)*. We notice that both our approach and RBT achieve a good jump-start and learn faster than plain FQI. However, to better investigate how samples are transferred, we show the transfer ratio from each source task in Figure 5.3*(right)*. Since RBT transfers rewards and transitions jointly, it decides to compensate the highly biased reward samples from the constant-spin task by over-sampling the first source task. However, it inevitably introduces bias from the different dynamics. Our approach, on the other hand, correctly transfers almost all reward samples from the swing-up task, while discarding those from the constant-spin task. Due to transition

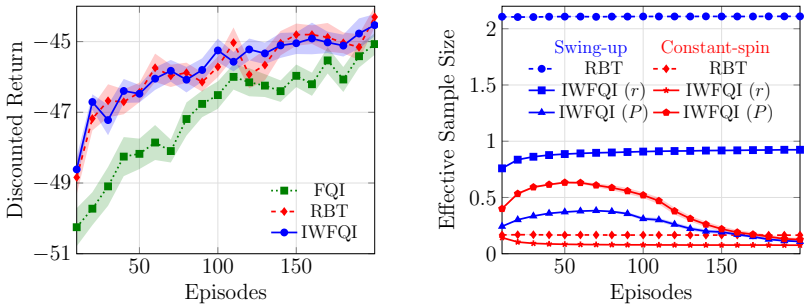

Figure 5.3: *Acrobot swing-up with* $(100 + 50)$ *episodes transferred from* 2 *source tasks. (left) learning performance. (right) relative number of samples transferred from each source task.*

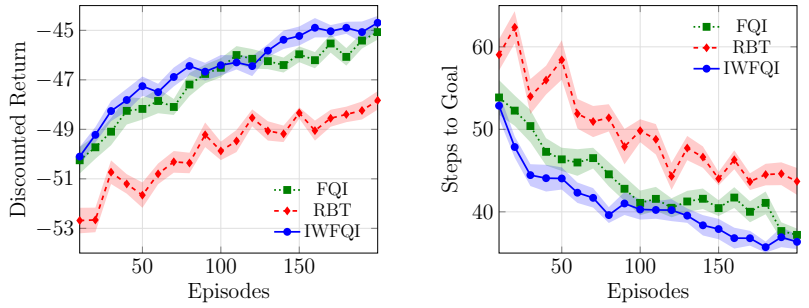

Figure 5.4: *Transfer of samples from the constant-spin task to the swing-up task. (left) discounted expected reward and (right) number of steps before reaching a goal state.*

noise over-estimation, IWFQI achieves an interesting adaptive behaviour: during the initial episodes, when few target samples are available, and the GPs are inaccurate, more samples are transferred. This causes a reduction of the variance in the first phases of learning that is much greater than the increase of bias. However, as more target samples are available, the transfer becomes useless, and our approach correctly decides to discard most transition samples, thus minimizing both bias and variance.

We now show what happens when only the constant-spin source task is available. Clearly, most of the reward samples should be discarded, and conversely for the transition samples. As we can see from Figure 5.4, RBT now performs significantly worse than FQI. This is due to the fact that, by transferring samples jointly, it cannot avoid introducing bias. Our approach, on the other hand, is able to discard the reward samples, thus being *robust to negative transfer*. Furthermore, it achieves a little improvement over FQI due to the few samples transferred.

Water Reservoir Control

In this experiment, we consider a real-world problem where the goal is to learn how to optimally control a water reservoir system. More specifically, the objective is to learn

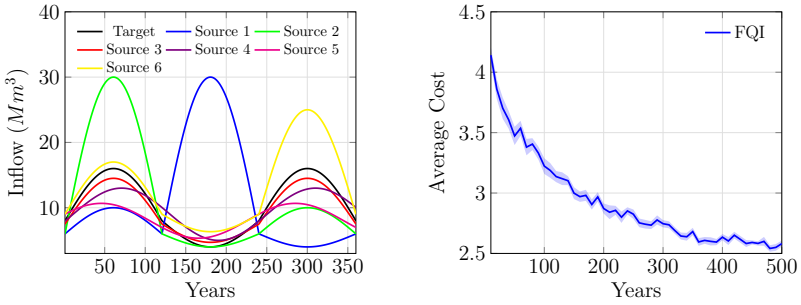

Figure 5.5: *Water reservoir control. (left) Inflow profiles for all tasks. (right) Learning without transfer for* 500 *years.*

a per-day water release policy that meets a given demand while keeping the water level below a flooding threshold. Castelletti et al. (2010) successfully addressed such problem by adopting batch RL techniques. However, the authors proved that, due to the highly non-linear and noisy environment, an enormous amount of historical data is needed to achieve good performance. Consider now the case where a new water reservoir, for which no historical data is available, needs to be controlled. Since each sample corresponds to one day of release, learning by direct interaction with the environment is not practical and leads to poor control policies during the initial years, when only a little experience has been collected. Although we do not know the new environment, it is reasonable to assume that we have access to operational data from existing reservoirs. Then, our solution is to transfer samples to immediately achieve good performance. However, such reservoirs might be located in very different environments and weight objectives differently, thus making transfer very challenging.

We adopt a system model similar to the one proposed in Castelletti et al. (2010). The state variables are the current water storage s_t and day $t \in [1, 365]$, while there are 8 discrete actions, each corresponding to a particular release decision. The system evolves according to the simple mass balance equation $s_{t+1} = s_t + i_t - a_t$, where i_t is the net inflow at day t and is modeled as periodic function, with period of one year, plus Gaussian noise. Given the demand d and the flooding threshold f, the reward function is a convex combination of the two objectives, $r(s_t, a_t) = -\alpha \max\{0, s_t - f\} - \beta(\max\{0, d - a_t\})^2$, where $\alpha, \beta \geq 0$. Due to the different geographic locations, each task has different inflow function $i_j(t) = \bar{i}_j(t) + \mathcal{N}(0, \sigma_p^2)$, where $\sigma_p^2 = 2.0$ is the fixed noise variance. The different mean-inflow functions are shown in Figure 5.5(left). Furthermore, each water reservoir weighs the flooding and demand objectives differently. This is modeled by changing the

	Target	Source 1	Source 2	Source 3	Source 4	Source 5	Source 6
α	0.3	0.8	0.35	0.7	0.4	0.6	0.45
β	0.7	0.2	0.65	0.3	0.6	0.4	0.55

Table 5.1: *Reward parameters for the different water reservoirs.*

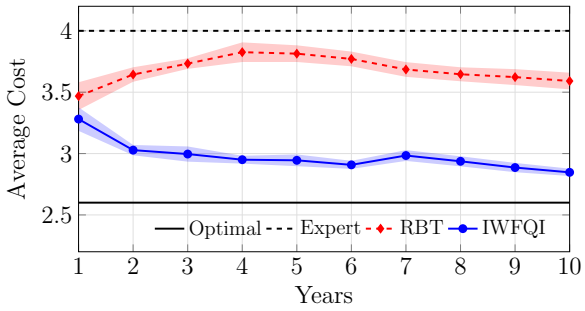

Figure 5.6: *Water reservoir control. Average* cost *per day during the first* 10 *years of learning. IWFQI outperforms the expert and quickly achieves near-optimal performance.*

respective weights α and β. The values for all tasks are reported in Table 5.1. Notice that there is no source task that is globally similar to the target: either some reward structure is shared or some transition structure is, never both. This makes transfer very challenging since samples have to be accurately selected to prevent detrimental consequences.

Transfer experiment. We collected 10800 samples, corresponding to 30 years of historical data, from each of 6 source water reservoirs under a hand-coded expert policy. We compared our approach to FQI and RBT over the first 10 years of learning. An ϵ-greedy policy ($\epsilon = 0.3$) was used to collect batches of 1 year of samples, except for the first batch, for which an expert's policy was used. Results, averaged over 20 runs, are shown in Figure 5.6. We notice that IWFQI immediately outperforms the expert's policy and quickly achieves near-optimal performance. RBT, on the other hand, has a good jump-start but then seems to worsen its performance. Once again, this is because each source task has at least few samples that can be transferred. However, selecting such samples is very complicated and leads to negative transfer in case of failure. Finally, FQI performs significantly worse than all alternatives and is, thus, not reported.

Learning from scratch. To better demonstrate the difficulty of this control problem, we run FQI for 500 episodes (equivalent to 500 years of interaction). Furthermore, to make the problem simpler, we allow the agent to sample the state-action space arbitrarily, so as to have a better exploration. The result is shown in Figure 5.5. Although we significantly simplified the problem and we allowed FQI to gather an enormous amount of data, the algorithm still needs almost 500 years to achieve optimal performance. This demonstrates that solving this task by directly interacting with the real environment is clearly impractical. Thus, transfer of previous knowledge is, in this case, mandatory to achieve good performance.

CHAPTER 6

Sample Reuse in Policy Gradients

This chapter is based on the paper "Transfer of Samples in Policy Search via Multiple Importance Sampling" co-authored with Mattia Salvini and Marcello Restelli and published at ICML 2019.

Introduction

All the approaches we have discussed so far for the transfer of experience samples, including our IWFQI, are specifically designed for batch value-based reinforcement learning. As a consequence, they do not easily generalize to the more common setting where the agent interacts with the environment *online*, while collecting and directly using long sequences of states and actions. This is, for instance, the setting considered by policy search methods (Peters and Schaal, 2008b; Sutton et al., 2000; Deisenroth et al., 2013), where the agent optimizes parametrized policies by iteratively collecting batches of trajectories from the environment. Despite their recent successes, these algorithms typically suffer a large sample complexity. Motivated by this limitation, the goal of this chapter is to extend our importance-sampling techniques for sample reuse in online policy search. More precisely, we shall focus on policy gradient methods. While the most immediate extension would be to design a variant of IWFQI for fitting a critic in an actor-critic algorithm, here we focus on actor-only settings. This requires transferring entire trajectories of states and actions rather than single-step transitions as in IWFQI. We note that a similar problem is the one of reusing samples collected from different policies under the same environment (Zhao et al., 2013; Hachiya et al., 2011), which is thus a special case of our setting.

There are two major complications that prevent a simple extension of the ideas presented in Chapter 5 to this setting. (1) The importance weights would be on entire trajectories, i.e., high-dimensional random variables which would make the resulting estimators' variance extremely large. (2) The trivial procedure to estimate the weights of Section 5.4 requires to have a dataset of samples from the target task. While this is the case for batch reinforcement learning, in online settings the agent initially has no experience from the target and, thus, this method is not applicable. In this chapter, we propose a method for trajectory reuse in policy gradients that uses importance sampling to correct distribution mismatch, exactly as in IWFQI, while addressing the complications above. To address (1), differently from IWFQI, where we showed good empirical performance with plain IS, here we necessitate of more robust estimators. We achieve this by leveraging different techniques, including multiple importance sampling, per-decision importance sampling, and control variates. To address (2), we propose a novel technique for estimating the importance weights by solving a well-chosen optimization problem that takes task uncertainty into account. Interestingly, this technique can be applied even in absence of target samples.

Outline. Our detailed contributions are as follows.

1. We propose an Importance-Weighted Policy Gradient (IWPG) method (Section 6.3), a simple gradient-based algorithm that uses importance sampling for transferring trajectories under different policies and environments. IWPG adaptively determines the batch size for interacting with the target based on the available knowledge from the sources;

2. For the ideal case where the weights can be computed, we propose different importance sampling estimators to reuse trajectories generated by different policies and transition dynamics (Section 6.4). The best combination of these estimators is provably robust to negative transfer and could be of interest outside our specific settings;

3. For the more realistic case where the weights cannot be computed, we propose a method to estimate them while taking task uncertainty into account (Section 6.5);

4. We provide numerical simulations to evaluate our approach in different settings (Section 6.6).

Formal Setting

As standard in policy search, we consider differentiable policies parameterized by $\theta \in \Theta$. Refer to Section 2.5.2 for a recap of policy search methods. Let $\{\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_{m'}\} \subseteq \mathfrak{M}$ be a set of $m' + 1$ unknown MDPs[1], where $\mathcal{M}_j = (\mathcal{S}, \mathcal{A}, P_j, U_j, \rho, \gamma)$. Without loss of generality, we assume the initial state distribution ρ to be shared but possibly unknown. In order to simplify the exposition, we impose the additional assumption that the target mean-reward function $r_0(s, a) = r(s, a)$ is known. This implies that we do not need to transfer reward samples and, in fact, the reward distributions U_j of the different tasks are never used. The extension to the case where the target reward is unknown can be

[1]We use m' instead of m to denote the number of source MDPs since m will be used for other purposes.

done analogously as in Chapter 5. Define a trajectory as a fixed-length sequence $\tau := (s_0, a_0, s_1, \dots, s_T)$ of T states and actions, without reward samples. If τ is generated by executing a policy with parameters θ in MDP $\mathcal{M}$, its distribution is

$$p(\tau|\theta, \mathcal{M}) = \rho(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t). \tag{6.1}$$

Since transfer occurs at trajectory level and the distribution of these trajectories is fully characterized given both an MDP $\mathcal{M}$ and some policy parameters θ, in this chapter only we define a "*source distribution*" (or source proposal) as a couple $(\theta, \mathcal{M})$. From each source distribution $(\theta_j, \mathcal{M}_j)$, for $j = 1, \dots, m'$, we have a dataset $\mathcal{D}_j = \{\tau_{i,j}\}_{i=1}^{n_j}$ of $n_j > 0$ trajectories that are i.i.d. from $p(\cdot|\theta_j, \mathcal{M}_j)$. We assume that the policy parameters θ_j are known for all j. Note that we might have data from multiple policies for each source MDP. Let m_0 be the total number of source distributions initially available to the agent (i.e., m' times the number of policies for each source MDP). While learning the target task $\mathcal{M}_0$, the agent produces a sequence of policies whose trajectories are added to the source dataset online, so that the number of source distributions from which to transfer grows over time.

Additional notation. For a trajectory $\tau = (s_0, a_0, s_1, \dots, s_T)$, we define $g_\theta(\tau) := \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$ as the log-policy gradient evaluated at τ. Similarly, with some abuse of notation, we define $r(\tau) := \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)$ as the total discounted reward along τ. We recall that $\mathbb{E}_{\tau \sim p(\cdot|\theta,\mathcal{M})}[g_\theta(\tau)] = 0$ and that $J(\theta, \mathcal{M}) = \mathbb{E}_{\tau \sim p(\cdot|\theta,\mathcal{M})}[r(\tau)]$ is the expected return of policy π_θ in MDP $\mathcal{M}$. Moreover, the corresponding gradient (see Section 2.5.2) is $\nabla_\theta J(\theta, \mathcal{M}) = \mathbb{E}_{\tau \sim p(\cdot|\theta,\mathcal{M})}[g_\theta(\tau) r(\tau)]$.

Importance-Weighted Policy Gradient

Our idea is quite simple: we learn the target task $\mathcal{M}_0$ using standard gradient-based techniques, while using all data at our disposal (source and target trajectories) for estimating the gradients with the purpose of reducing variance and, thus, achieve faster convergence. As for IWFQI, we apply importance sampling techniques to compensate the distribution shift induced by trajectories generated under different policies and/or transition models. Since the transition models are generally not known, we estimate them from samples in order to be able to compute the importance weights. We now introduce our main methodology, while deferring the specific gradient estimators to Section 6.4 and the model estimation procedure to Section 6.5.

Our importance-weighted policy gradient (IWPG) method is outlined in Algorithm 3. IWPG takes as input the dataset of source trajectories $\widetilde{\mathcal{D}} = \left\{ (\{\tau_{i,j}\}_{i=1}^{n_j}, \theta_j, \widetilde{P}_j) \right\}_{j=1}^{m_0}$ augmented with the policy parameters $\{\theta_j\}_{j=1}^{m_0}$ that generated them and the corresponding (estimated) transition models $\{\widetilde{P}_j\}_{j=1}^{m_0}$. While we assume the policies to be known, the transition models could be estimates of the true ones (hence the "tildes"), as we discuss in Section 6.5. Moreover, IWPG receives an importance-weighted estimator $\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}})$ of the gradients under the target MDP $\mathcal{M}_0$. More precisely, $\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}})$ is an estimator of $\nabla_\theta J(\theta_0, \mathcal{M}_0)$, where θ_0 are the policy parameters used to collect the last batch of

Algorithm 3 Importance-Weighted Policy Gradient (IWPG)

Require: Source dataset $\widetilde{\mathcal{D}} = \left\{\left(\{\tau_{i,j}\}_{i=1}^{n_j}, \theta_j, \widetilde{P}_j\right)\right\}_{j=1}^{m_0}$ where $\tau_{i,j} \sim p(\cdot|\theta_j, \mathcal{M}_j)$, gradient estimator $\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}})$, effective sample size estimator $\widehat{\mathrm{ESS}}(n_0; \widetilde{\mathcal{D}})$, minimum effective sample size $\mathrm{ESS}_{\min}$, minimum batch size $n_{\min}$, step-size sequence η_k, number of iterations K

1: Initialize policy θ^0 and target model $\widetilde{P}_0$ from $\widetilde{\mathcal{D}}$
2: **for** $k = 0, \dots, K$ **do**
3: Find the minimum $n_0 \in \{n_{\min}, \dots, \mathrm{ESS}_{\min}\}$ such that $\widehat{\mathrm{ESS}}(n_0; \widetilde{\mathcal{D}}) \geq \mathrm{ESS}_{\min}$
4: Sample n_0 trajectories $\{\tau_i\}_{i=1}^{n_0}$ from the target MDP $\mathcal{M}_0$ under policy π_{θ^k}
5: Store samples: $\widetilde{\mathcal{D}} \leftarrow \widetilde{\mathcal{D}} \cup \left\{\left(\{\tau_i\}_{i=1}^{n_0}, \theta^k, \widetilde{P}_0\right)\right\}$
6: Update parameters: $\theta^{k+1} \leftarrow \theta^k + \eta_k \widehat{\nabla}_\theta J(\widetilde{\mathcal{D}})$
7: Update target transition model estimate $\widetilde{P}_0$ using $\widetilde{\mathcal{D}}$ (Section 6.5)
8: **end for**

target trajectories (i.e., the zero-th component of $\widetilde{\mathcal{D}}$, which is added by IWPG as learning proceeds). The algorithm also receives an estimator $\widehat{\mathrm{ESS}}(n_0; \widetilde{\mathcal{D}})$ of the effective sample size that the gradient estimator would have when combining all source data with a batch of n_0 trajectories from the current policy. This is used to adaptively choose the target batch size, as we shall discuss shortly.

Let us now concentrate on how the algorithm works. First, IWPG initializes the target policy parameters and transition model (line 1). Both can be done either randomly or by leveraging the source data, e.g., with the purpose of achieving a good jumpstart (Abel et al., 2018). Since our primary concern is the optimization rather than the initialization, we leave this step unspecified. At each iteration, IWPG adaptively computes the target batch size, i.e., the number of trajectories to collect from $\mathcal{M}_0$ with the current policy. This is done by seeking the value n_0 which would guarantee a minimum ESS for the resulting importance-weighted estimator (line 3). The rationale is that, if the gradient of the current policy can be reliably estimated using the source samples, there is no need to collect new trajectories at all. In order to carry out this step, we derive a lower bound on the increase rate of the approximate ESS as a function of n_0. Since this is specific for a class of MIS estimators that we propose, we defer the details to the next section. In practice, we impose a minimum batch size of $n_{\min}$ to avoid degenerate cases (e.g., where we overestimate the ESS due to not collecting target samples). After collecting the new batch (line 4), IWPG adds it to the current dataset (line 5) together with the policy θ^k that generated it. Note that this implies that the number of policy-model pairs in $\widetilde{\mathcal{D}}$ grows with the number of iterations as trajectories from the target task (but policies potentially different from the current one) are stored. In other words, the number of source distributions grows over time. Then, IWPG updates the policy parameters using the chosen weighted estimator on the current dataset (line 6). Similarly, the algorithm updates the current estimate of the target model. Note that, even if estimated, the source models are never updated during learning of the target task. We discuss this fact in more detail in Section 6.5.

Gradient Estimators with Known Models

This section is devoted to the design of robust importance-weighted gradient estimators that are tailored for policy search under the assumption of known transition models. We note that the techniques and results that we develop in this section go beyond our specific setting of transfer across different environments. In particular, our estimators are related to, and might be of interest for, different RL settings such as off-policy evaluation (Precup, 2000; Hachiya et al., 2009; Thomas and Brunskill, 2016; Guo et al., 2017; Liu et al., 2018), off-policy learning (Precup et al., 2001; Mahmood et al., 2014; Geist and Scherrer, 2014; Munos et al., 2016), and sample reuse from different policies (Zhao et al., 2013; Hachiya et al., 2011). Notably, IWPG is directly applicable to these settings, which involve only "intra-environment transfer", i.e., reusing samples generated by different policies.

Freeze IWPG at some iteration and take the dataset $\widetilde{\mathcal{D}} = \left\{(\{\tau_{i,j}\}_{i=1}^{n_j}, \theta_j, P_j)\right\}_{j=0}^{m}$ updated after collecting the batch n_0 of target trajectories from the current policy θ_0. Our goal is to design $\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}})$, i.e., an estimator of $\nabla_\theta J(\theta_0, \mathcal{M}_0)$ that uses *all* data in $\widetilde{\mathcal{D}}$. Note that, with respect to the previous section, we replaced $\widetilde{P}_j$ with P_j in $\widetilde{\mathcal{D}}$ to emphasize that models are known.

Multiple Importance Sampling Estimators

We note that it is easy to derive an importance-weighted variant of the classic REINFORCE gradient estimator (see Section 2.5.2). We have that

$$\widehat{\nabla}_\theta^{\text{IS}} J(\widetilde{\mathcal{D}}) := \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} w_j^{\text{IS}}(\tau_{i,j}) g_{\theta_0}(\tau_{i,j}) r(\tau_{i,j}) \tag{6.2}$$

is an unbiased estimator for $\nabla_\theta J(\theta_0, \mathcal{M}_0)$, where the *importance weight* $w_j^{\text{IS}}(\tau) := \frac{p(\tau|\theta_0, \mathcal{M}_0)}{p(\tau|\theta_j, \mathcal{M}_j)}$ can be computed in closed-form as

$$w_j^{\text{IS}}(\tau) = \prod_{t=0}^{T-1} \frac{\pi_{\theta_0}(a_t|s_t)}{\pi_{\theta_j}(a_t|s_t)} \frac{P_0(s_{t+1}|s_t, a_t)}{P_j(s_{t+1}|s_t, a_t)}. \tag{6.3}$$

Unfortunately, this IS scheme is likely to fail in most cases of practical interest. It is well known, especially from the literature on off-policy estimation (Precup, 2000; Hachiya et al., 2009; Thomas and Brunskill, 2016; Guo et al., 2017; Liu et al., 2018), that importance sampling on long trajectories is likely to give almost zero or huge weights, thus leading to estimators with very high (sometimes infinite) variance (Li et al., 2015a; Jiang and Li, 2016). This drawback is even amplified in our transfer settings, where there is a model mismatch in addition to the one between the policies. Several variance reduction techniques, typically paying a small amount of bias, have been proposed (e.g., self-normalized estimators (Kong, 1992), truncation (Ionides, 2008), flattening (Hachiya et al., 2009)). Fortunately, MIS comes to the rescue in our settings, allowing us to get a low-variance estimator without introducing any bias. Using Equation 4.3, the MIS gradient estimator is

$$\widehat{\nabla}_\theta^{\text{MIS}} J(\widetilde{\mathcal{D}}) = \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} w_j^{\text{MIS}}(\tau_{i,j}) g_{\theta_0}(\tau_{i,j}) r(\tau_{i,j}), \tag{6.4}$$

where $w_j^{\text{MIS}}(\tau) := \frac{n}{n_j} h_j(\tau) w_j^{\text{IS}}(\tau)$ for a general heuristic function h, and $w_j^{\text{MIS}}(\tau) := \frac{p(\tau|\theta_0,\mathcal{M}_0)}{\sum_{j=0}^{m} \alpha_j p(\tau|\theta_j,\mathcal{M}_j)}$ with $\alpha_j = n_j/n$ for the balance heuristic. For brevity, define $q_\alpha(\tau) := \sum_{j=0}^{m} \alpha_j p(\tau|\theta_j,\mathcal{M}_j)$ as the mixture of all distributions. As we already discussed in Section 4.4.2, MIS provides a more robust way to combine different source distributions than plain IS. In particular, when we force a minimum amount of samples from the target distribution, so that $n_0 > 0$ and $\alpha_0 > 0$, the defensive component in the denominator makes the weights, and thus their variance, *bounded*. This resolves one of the main limitations of plain IS when applied to trajectory reuse. However, there is more we can do by exploiting the structure of trajectories and policy gradients, as we see in the next two sections.

Per-decision Estimators

Per-decision IS (Precup, 2000) is a common variance reduction technique from off-policy evaluation. It relies on the intuition that future actions cannot influence past rewards, i.e., each reward $r(s_t, a_t)$ should be weighted only by the probability of a trajectory up to that time. This technique can be easily combined with MIS, leading to the per-decision MIS (PD for short) estimator,

$$\widehat{\nabla}_\theta^{\text{PD}} J(\widetilde{\mathcal{D}}) = \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} \sum_{t=0}^{T-1} \gamma^t w_{j,t}^{\text{PD}}(\tau_{i,j}) g_{\theta_0,t}(\tau_{i,j}) r(s_{i,j}^t, a_{i,j}^t), \tag{6.5}$$

where $g_{\theta,t}(\tau) := g_\theta(\tau_{0:t})$, with $\tau_{0:t}$ being a trajectory up to time t, and $w_{j,t}^{\text{PD}}(\tau) := \frac{n}{n_j} h_{j,t}(\tau) w_j^{\text{IS}}(\tau_{0:t})$. Notice that the heuristics is now a function of time. We show that, if this function is uniformly normalized over time, the resulting estimator remains unbiased.

Theorem 6.4.1 (Unbiasedness of PD estimator)**.** *Let $h_{j,t}(\tau)$ be a function such that, for all $t \in \{0, \dots, T-1\}$ and τ, $\sum_{j=0}^{m} h_{j,t}(\tau) = 1$. Then, the per-decision MIS estimator in* (6.5) *is unbiased.*

Proof. The proof simply starts from the definition of the per-decision MIS estimator and shows that, by leveraging the assumption that the heuristic function is a partition of unity, its expected value is

the policy gradient:

$$\begin{aligned}
\mathbb{E}\left[\widehat{\nabla}^{\mathrm{PD}}_{\theta} J(\widetilde{\mathcal{D}})\right] &= \sum_{j=0}^{m} \mathbb{E}_{p(\tau|\theta_j,\mathcal{M}_j)}\left[\sum_{t=0}^{T-1} h_{j,t}(\tau)\frac{p(\tau_{0:t}|\theta_0,\mathcal{M}_0)}{p(\tau_{0:t}|\theta_j,\mathcal{M}_j)}\gamma^t r(s_t,a_t) g_{\theta_0,t}(\tau)\right] \\
&= \sum_{j=0}^{m}\int \sum_{t=0}^{T-1} h_{j,t}(\tau) p(\tau_{0:t}|\theta_0,\mathcal{M}_0)\gamma^t r(s_t,a_t) g_{\theta_0,t}(\tau)\mathrm{d}\tau \\
&= \int \sum_{t=0}^{T-1} p(\tau_{0:t}|\theta_0,\mathcal{M}_0)\gamma^t r(s_t,a_t) g_{\theta_0,t}(\tau) \underbrace{\sum_{j=0}^{m} h_{j,t}(\tau)}_{=1}\mathrm{d}\tau \\
&= \int \sum_{t=0}^{T-1} p(\tau_{0:t}|\theta_0,\mathcal{M}_0)\gamma^t r(s_t,a_t) g_{\theta_0,t}(\tau)\mathrm{d}\tau \\
&= \mathbb{E}_{p(\tau|\theta_0,\mathcal{M}_0)}\left[\sum_{t=0}^{T-1}\gamma^t r(s_t,a_t)\sum_{l=0}^{t}\nabla_\theta \log \pi_{\theta_0}(a_l|s_l)\right] \\
&= \nabla_\theta J(\theta_0,\mathcal{M}_0),
\end{aligned}$$

where the last equality follows from the unbiasedness of the G(PO)MDP estimator (see Section 2.5.2). □

We shall adopt the balance heuristic, for which $w^{\mathrm{PD}}_{j,t}(\tau) = w^{\mathrm{MIS}}_{j}(\tau_{0:t})$.

Regression-based Control Variates

While the per-decision estimator effectively reduces the sensitivity to the trajectory length T, control variates (a.k.a. baselines) are another popular variance-reduction technique in the policy gradient literature. Here we show how to combine the classic baseline used in REINFORCE (Williams, 1992) with the control variates for the balance heuristic discussed in Section 4.4.4. For the d-th dimension of the gradient, consider a vector of functions $\Psi_d(\tau) := [\psi_{0,d}(\tau),\dots,\psi_{m+1,d}(\tau)]$ such that $\mathbb{E}_{\tau\sim q_\alpha}[\psi_{j,d}(\tau)] = 0$ for every j. In our specific case, we set the first $m+1$ control variates as

$$\psi_{j,d}(\tau) = \frac{p(\tau|\theta_j,\mathcal{M}_j)}{q_\alpha(\tau)} - 1, \quad j = 0,\dots,m. \tag{6.6}$$

These are shown by Owen and Zhou (2000) to be very effective for the balance heuristic, as discussed in Chapter 4. Moreover, we set the remaining control variate to a weighted variant of the standard baseline used by REINFORCE,

$$\psi_{m+1,d}(\tau) = \frac{p(\tau|\theta_0,\mathcal{M}_0)[g_{\theta_0}(\tau)]_d}{q_\alpha(\tau)}. \tag{6.7}$$

It is easy to see that all these functions have zero expectation under the mixture distribution q_α. Therefore, our MIS estimator (6.4) using Ψ_d as control variates is

$$\widehat{\nabla}^{\mathrm{CV}}_{\theta_d} J(\widetilde{\mathcal{D}}) = \widehat{\nabla}^{\mathrm{IS}}_{\theta_d} J(\widetilde{\mathcal{D}}) - \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}\beta_d^T \Psi_d(\tau_{i,j}), \tag{6.8}$$

where $\beta_d \in \mathbb{R}^{m+1}$ is the vector of control-variate coefficients, which can be approximated by solving the regression problem reported in Section 4.4.4. In practice, we can fit β_d and estimate the gradient using different partitions of the current dataset to keep an unbiased estimator.

Theorem 6.4.2. *The estimator* (6.8) *is unbiased for any* β_d*. Furthermore, under the optimal coefficients* β_d^* *minimizing the variance of* (6.8)*, we have* $\mathbb{V}\mathrm{ar}[\widehat{\nabla}^{CV}_{\theta_d} J(\widetilde{\mathcal{D}})] \leq \mathbb{V}\mathrm{ar}[\widehat{\nabla}^{MIS}_{\theta_d} J(\widetilde{\mathcal{D}})]$.

Proof. To prove the unbiasedness, recall that

$$\mathbb{E}_{\tau_{i,j}\sim p(\cdot|\theta_j,\mathcal{M}_j)}\left[\widehat{\nabla}^{\text{MIS}}_{\theta_d} J(\widetilde{\mathcal{D}})\right] = \nabla_{\theta_d} J(\theta_0, \mathcal{M}_0).$$

Thus, we only need to prove that the second term has expected value equal to zero. Hence,

$$\begin{aligned}\mathbb{E}_{\tau_{i,j}\sim p(\cdot|\theta_j,\mathcal{M}_j)}\left[\frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}\beta_d^T\Psi_d(\tau_{i,j})\right] &= \frac{1}{n}\sum_{j=0}^{m} n_j \beta_d^T \mathbb{E}_{\tau\sim p(\cdot|\theta_j,\mathcal{M}_j)}\left[\Psi_d(\tau)\right] \\ &= \frac{1}{n}\sum_{j=0}^{m} n_j \sum_{l=0}^{m+1}\beta_{l,d}\mathbb{E}_{\tau\sim p(\cdot|\theta_j,\mathcal{M}_j)}\left[\psi_{l,d}(\tau)\right].\end{aligned}$$

Control variates $\psi_{0,d},\dots,\psi_{m,d}$ are well-known to have zero expectation (Owen and Zhou, 2000), so let us concentrate on the $(m+1)$-th term, which is similar to the well-known baseline commonly adopted in policy gradient methods. Its expectation can be easily verified to be zero:

$$\begin{aligned}\beta_{m+1,d}\frac{1}{n}\sum_{j=0}^{m} n_j \mathbb{E}_{\tau\sim p(\cdot|\theta_j,\mathcal{M}_j)}\left[\frac{p(\tau|\theta_0,\mathcal{M}_0)[g_{\theta_0}(\tau)]_d}{q_\alpha(\tau)}\right] &= \beta_{m+1,d}\int p(\tau|\theta_0,\mathcal{M}_0)[g_{\theta_0}(\tau)]_d \mathrm{d}\tau \\ &= \beta_{m+1,d}\int p(\tau|\theta_0,\mathcal{M}_0)\sum_{t=0}^{T-1}\nabla_{\theta_d}\log\pi_{\theta_0}(a_t|s_t)\mathrm{d}\tau.\end{aligned}$$

The last integral can be rewritten as

$$\sum_{t=0}^{T-1}\int\rho(s_0)\int\pi_{\theta_0}(a_0|s_0)\cdots\int\pi_{\theta_0}(a_t|s_t)\nabla_{\theta_d}\log\pi_{\theta_0}(a_t|s_t)\mathrm{d}s_0\dots\mathrm{d}a_t,$$

which is equal to zero since $\int\nabla_{\theta_d}\pi_{\theta_0}(a|s)\mathrm{d}a = 0$ for any state s. This concludes the proof of the first statement.

In order to prove the second statement, let $z = [0,0,\dots,0]$ be the $(m+1)$-th dimensional vector of zeros. Then, under the coefficients β_d^* minimizing the variance of the CV estimator (6.8),

$$\begin{aligned}\mathbb{V}\mathrm{ar}[\widehat{\nabla}^{\text{CV}}_{\theta_d} J(\widetilde{\mathcal{D}})] &= \mathbb{V}\mathrm{ar}\left[\widehat{\nabla}^{\text{MIS}}_{\theta_d} J(\widetilde{\mathcal{D}}) - \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}\beta_d^T\Psi_d(\tau_{i,j})\right] \\ &\leq \mathbb{V}\mathrm{ar}\left[\widehat{\nabla}^{\text{MIS}}_{\theta_d} J(\widetilde{\mathcal{D}}) - \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j} z^T\Psi_d(\tau_{i,j})\right] = \mathbb{V}\mathrm{ar}[\widehat{\nabla}^{\text{MIS}}_{\theta_d} J(\widetilde{\mathcal{D}})].\end{aligned}$$

□

The interesting property is that the estimator with optimal control variates has variance never larger than the one of the corresponding MIS estimator. In practice, however, one

often observes dramatic variance reduction even without the optimal parameters. Note that, when the number of dimensions d of the parameter space is large, it is common to fit a unique β for all d. In this case, simply taking $\psi_{m+1}(\tau) = \frac{p(\tau|\theta_0,\mathcal{M}_0)\sum_d [g_{\theta_0}(\tau)]_d}{q_\alpha(\tau)}$ and solving the regression problem is therefore equivalent to (approximately) minimizing $\mathrm{Tr}(\mathbb{C}\mathrm{ov}[\widehat{\nabla}_\theta^{\mathrm{CV}} J(\widetilde{\mathcal{D}})])$. Finally, we note that the control variates can be combined straightforwardly with the PD estimator of Section 6.4.1, which is what we do in our experiments.

Robustness to Negative Transfer

We now show that the MIS estimator with control variates enjoys safety guarantees against negative transfer. We first propose a definition of negative transfer for policy gradient algorithms in terms of convergence to ϵ-optimal stationary points [2]. Informally, we say that an algorithm $\mathscr{A}$ negatively transfers against some baseline $\mathscr{B}$ if, when running under the same conditions, there exists an iteration in which $\mathscr{B}$ *provably* converges to an ϵ-optimal stationary point, while $\mathscr{A}$ does not.

Definition 6.4.1 (Negative transfer)**.** *Let $\mathscr{A}$ and $\mathscr{B}$ be two policy gradient algorithms. Fix an initial parameter θ^0, a learning rate η, a batch size n, and an accuracy $\epsilon > 0$. Then, $\mathscr{A}$ negatively transfers w.r.t. $\mathscr{B}$ ($\mathscr{A} \prec \mathscr{B}$) if there exists an iteration number $k \geq 1$ such that we can guarantee that $\frac{1}{k}\sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{B}}[\|\nabla_\theta J(\theta^l)\|_2^2] \leq \epsilon$ but $\frac{1}{k}\sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}[\|\nabla_\theta J(\theta^l)\|_2^2] > \epsilon$.*

Let $\mathscr{B}_{\mathrm{R}}$ be the REINFORCE algorithm (Williams, 1992) and $\mathscr{B}_{\mathrm{G}}$ be the G(PO)MDP algorithm (Baxter and Bartlett, 2001), both with optimal baselines. The next result shows that IWPG using optimal control variates and the MIS estimator ($\mathscr{A}_{\mathrm{CV}}$) or the PD estimator ($\mathscr{A}_{\mathrm{PDCV}}$) cannot be worse than its no-transfer counterparts.

Theorem 6.4.3. *Assume that the expected return $J(\cdot, \mathcal{M}_0)$ is L-smooth (i.e., its gradient is L-Lipschitz). Let $n_{\min} > 0$ be the minimum batch size for $\mathscr{A}_{CV}$ ($\mathscr{A}_{PDCV}$) and the fixed batch size for $\mathscr{B}_R$ ($\mathscr{B}_G$). Assume all algorithms start from the same parameter θ^0, use a learning rate $0 < \eta \leq \frac{2}{L}$, and that $\mathscr{A}_{CV}$ ($\mathscr{A}_{PDCV}$) uses the optimal CV coefficients β_d^*. Then, for all $\epsilon > 0$:*

$$\mathscr{A}_{CV} \not\prec \mathscr{B}_R, \quad \mathscr{A}_{PDCV} \not\prec \mathscr{B}_G.$$

Theorem 6.4.3 is quite remarkable given that even importance sampling techniques for off-policy corrections, with no model mismatch, tend to fail on long trajectories due to the variance blowing up. Here, on the other hand, the properties of MIS combined with optimal control variates make it possible to prove that the resulting estimators are never worse than their no-transfer counterparts, despite the model mismatch. This in turns, implies that IWPG cannot converge too slowly with respect to plain policy gradients, as we prove shortly. We remark that, according to our definition, the fact that $\mathscr{A}$ is robust against $\mathscr{B}$ does not necessarily imply that $\mathscr{A}$ converges faster than $\mathscr{B}$ but only that, whenever we can prove that $\mathscr{B}$ converged, we can also prove that $\mathscr{A}$ converged. Hence, we might say that $\mathscr{A}$ cannot be much worse than $\mathscr{B}$, the standard (weaker) notion of negative transfer that is often considered in the literature (Taylor and Stone, 2009).

[2] As a standard in non-convex optimization, convergence to stationary points could be replaced with convergence to local maxima at the cost of a more complicated analysis.

Proof. Let $\tilde{J}(\theta) := -J(\theta, \mathcal{M}_0)$ and consider the problem $\operatorname{argmin}_\theta \tilde{J}(\theta)$, equivalent to the original one. Following standard proofs of convergence for non-convex stochastic optimization (see, e.g., Appendix B of (Allen-Zhu, 2017)), we can show that, for a fixed parameter θ^k and algorithm $\mathscr{A}$,

$$\tilde{J}(\theta^k) - \mathbb{E}_{\mathscr{A}}\left[\tilde{J}(\theta^{k+1})\right] \geq \left(\eta - \frac{\eta^2 L}{2}\right) \|\nabla_\theta \tilde{J}(\theta^k)\|_2^2 - \frac{\eta^2 L}{2} \mathbb{V}\mathrm{ar}_{\mathscr{A}}\left[\widehat{\nabla}_\theta^{\mathscr{A}}(\tilde{J}(\theta^k))\right], \tag{6.9}$$

where $\theta^{k+1} = \theta^k - \eta \widehat{\nabla}_\theta^{\mathscr{A}} \tilde{J}(\theta^k)$ and the expectations are taken w.r.t. the stochasticity in the estimation of the gradient. Rearranging (6.9), we obtain

$$\left(\eta - \frac{\eta^2 L}{2}\right) \|\nabla_\theta \tilde{J}(\theta^k)\|_2^2 \leq \tilde{J}(\theta^k) - \mathbb{E}_{\mathscr{A}}\left[\tilde{J}(\theta^{k+1})\right] + \frac{\eta^2 L}{2} \mathbb{V}\mathrm{ar}_{\mathscr{A}}\left[\widehat{\nabla}_\theta^{\mathscr{A}} \tilde{J}(\theta^k)\right]. \tag{6.10}$$

Let us now take the expectation under the whole stochastic process $\theta^{0:k}$, with θ^0 being deterministic and fixed, and sum over iterations the first k iterations. Then,

$$\begin{aligned}
&\left(\eta - \frac{\eta^2 L}{2}\right) \sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}\left[\|\nabla_\theta \tilde{J}(\theta^l)\|_2^2\right] \\
&\leq \tilde{J}(\theta^0) - \mathbb{E}_{\mathscr{A}}\left[\tilde{J}(\theta^{k+1})\right] + \frac{\eta^2 L}{2} \sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}\left[\mathbb{V}\mathrm{ar}_{\mathscr{A}}\left[\widehat{\nabla}_\theta^{\mathscr{A}} \tilde{J}(\theta^l) \mid \theta^l\right]\right] \\
&\leq \tilde{J}(\theta^0) - \tilde{J}(\theta^*) + \frac{\eta^2 L}{2} \sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}\left[\mathbb{V}\mathrm{ar}_{\mathscr{A}}\left[\widehat{\nabla}_\theta^{\mathscr{A}} \tilde{J}(\theta^l) \mid \theta^l\right]\right],
\end{aligned}$$

where $\theta^* = \operatorname{argmin}_\theta \tilde{J}(\theta)$. Rearranging,

$$\frac{1}{k} \sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}\left[\|\nabla_\theta \tilde{J}(\theta^l)\|_2^2\right] \leq \frac{\tilde{J}(\theta^0) - \tilde{J}(\theta^*)}{k\left(\eta - \frac{\eta^2 L}{2}\right)} + \frac{\eta L}{2 - \eta L} \frac{1}{k} \sum_{l=0}^{k-1} \mathbb{E}_{\mathscr{A}}\left[\mathbb{V}\mathrm{ar}_{\mathscr{A}}\left[\widehat{\nabla}_\theta^{\mathscr{A}} \tilde{J}(\theta^l) \mid \theta^l\right]\right]. \tag{6.11}$$

Let us now compare $\mathscr{B}_{\mathrm{R}}$ and $\mathscr{A}_{\mathrm{CV}}$. From Theorem 2 of Owen and Zhou (2000) (restated in Theorem 4.4.2), we know that, for every $j = 0, \dots, m$, a mixture IS estimator with proportions α using the optimal CV parameter β^* has a variance that is upper bounded by that of an IS estimator using only the j-th proposal divided by the proportion α_j of samples from such proposal. Furthermore, this property holds for a MIS estimator as well since its variance is always smaller than the one of the corresponding mixture estimator. In our context, the algorithm $\mathscr{A}_{\mathrm{CV}}$ uses the MIS estimator (6.8) with the optimal CV coefficients. Thus, for any dataset $\widetilde{\mathcal{D}}$ and dimension d,

$$\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_{\theta_d}^{\mathscr{A}_{\mathrm{CV}}} \tilde{J}(\widetilde{\mathcal{D}})\right] \leq \min_{j=0,\dots,m} \frac{\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_{\theta_d}^{\text{IS-}j} \tilde{J}(\widetilde{\mathcal{D}})\right]}{\alpha_j}, \tag{6.12}$$

where $\widehat{\nabla}_{\theta_d}^{\text{IS-}j} \tilde{J}(\widetilde{\mathcal{D}})$ denotes an IS estimator using only the samples in $\widetilde{\mathcal{D}}$ from the j-th proposal $p(\cdot|\theta_j, \mathcal{M}_j)$. Recalling that the 0-th proposal corresponds to the current target distribution, $p(\tau|\theta_0, \mathcal{M}_0)$, and that, by assumption, the minimum number of trajectories that $\mathscr{A}_{\mathrm{CV}}$ collects at each step is $n_{\min}$, we obtain

$$\begin{aligned}
\min_{j=0,\dots,m} \frac{\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_{\theta_d}^{\text{IS-}j} \tilde{J}(\widetilde{\mathcal{D}})\right]}{\alpha_j} &\leq \frac{\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_{\theta_d}^{\text{IS-0}} \tilde{J}(\widetilde{\mathcal{D}})\right]}{\alpha_0} = \frac{\mathbb{V}\mathrm{ar}\left[\frac{1}{n}\sum_{i=1}^{n_0} w_0^{\mathrm{IS}}(\tau_i) g(\tau_i) r(\tau_i)\right]}{\alpha_0} \\
&= \frac{\frac{1}{n}\mathbb{V}\mathrm{ar}\left[g(\tau) r(\tau)\right]}{\alpha_0} \leq \frac{1}{n_{\min}} \mathbb{V}\mathrm{ar}\left[g(\tau) r(\tau)\right] = \mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_{\theta_d}^{\mathscr{B}_{\mathrm{R}}} \tilde{J}(\widetilde{\mathcal{D}})\right].
\end{aligned}$$

The second equality follows from the fact that $w_0^{\text{IS}}(\tau) := \frac{p(\tau|\theta_0,\mathcal{M}_0)}{p(\tau|\theta_0,\mathcal{M}_0)} = 1$. Since this holds for all the policy dimensions d and $\mathbb{V}\text{ar}\left[\widehat{\nabla}_\theta^{\mathscr{A}_{\text{CV}}}\tilde{J}(\widetilde{\mathcal{D}})\right] = \text{Tr}(\mathbb{C}\text{ov}[\widehat{\nabla}_\theta^{\mathscr{A}_{\text{CV}}}\tilde{J}(\widetilde{\mathcal{D}})])$, we obtain

$$\mathbb{V}\text{ar}\left[\widehat{\nabla}_\theta^{\mathscr{A}_{\text{CV}}}\tilde{J}(\widetilde{\mathcal{D}})\right] \leq \mathbb{V}\text{ar}\left[\widehat{\nabla}_\theta^{\mathscr{B}_{\text{R}}}\tilde{J}(\widetilde{\mathcal{D}})\right], \tag{6.13}$$

i.e., the variance of the transfer algorithm in estimating the gradients is always smaller than the one of the no-transfer baseline. Furthermore, by assumption, the variance of the no-transfer baseline is bounded. Let $C_{\mathscr{B}_{\text{R}}}$ be its bound. Then,

$$\frac{1}{k}\sum_{l=0}^{k-1}\mathbb{E}_{\mathscr{B}_{\text{R}}}\left[\|\nabla_\theta\tilde{J}(\theta^l)\|_2^2\right] \leq \frac{1}{k\left(\eta - \frac{\eta^2 L}{2}\right)}\left(\tilde{J}(\theta^0) - \tilde{J}(\theta^*)\right) + \frac{\eta L}{2-\eta L}C_{\mathscr{B}_{\text{R}}}. \tag{6.14}$$

Suppose now that the upper bound (6.14) is less or equal then ϵ, which implies that $\mathscr{B}_{\text{R}}$ converged. Since we showed that $\mathbb{V}\text{ar}\left[\widehat{\nabla}_\theta^{\mathscr{A}_{\text{CV}}}\tilde{J}(\theta)\right] \leq C_{\mathscr{B}_{\text{R}}}$, it must be that, using (6.14),

$$\frac{1}{k}\sum_{l=0}^{k-1}\mathbb{E}_{\mathscr{A}_{\text{CV}}}\left[\|\nabla_\theta\tilde{J}(\theta^l)\|_2^2\right] \leq \epsilon.$$

Hence, whenever we are able to prove that $\mathscr{B}_{\text{R}}$ converged, we are also able to prove that $\mathscr{A}_{\text{CV}}$ converged, which is exactly our definition of robustness against negative transfer.

The proof for $\mathscr{A}_{\text{PDCV}}$ and $\mathscr{B}_{\text{G}}$ proceeds analogously by noticing that the variance of the former is always less or equal than the variance of the latter. □

Adapting the Batch Size of IWPG

Algorithm 3 requires a measure of ESS in order to evaluate the quality of a gradient estimate and, consequently, to adapt the batch size. Here we propose one such measure that is suitable for our estimators and, in particular, for the MIS estimator with balance heuristic. Although several ESS measures for IS have been studied (see, e.g., Martino et al. (2017)), to the best of our knowledge no measure specifically designed for MIS estimators has been proposed. Motivated by the recent work of Elvira et al. (2018), who analyzed the classical ESS (see Section 4.4.3) and empirically demonstrated its effectiveness in MIS, we consider a variant of this measure.

Recall that the ESS involves computing the variance of the importance weights under the given proposals. Since for our application we are satisfied with a lower bound on the ESS, we compute the variance with respect to the mixture of these. This is motivated by the following proposition, which follows directly from the fact that the former variance is always smaller than the latter (see Owen and Zhou (2000)).

Proposition 6.4.1. *Under the balance heuristics, we have that* $\text{ESS} := \frac{n}{1+\mathbb{V}\text{ar}[w^{MIS}(\tau)]} \geq \frac{n}{d_2(p(\cdot|\theta_0,\mathcal{M}_0)\|q_\alpha(\tau))}$.

Thus, all we need to do is to estimate the Renyi divergence between the target distribution and the current mixture of source distributions. To do this, we use the fact that the expected value of importance weights under trajectory distribution q_α is equal to one, so that: $d_2(p(\cdot|\theta_0,\mathcal{M}_0)\|q_\alpha(\tau)) = 1 + \mathbb{V}\text{ar}_{q_\alpha}[\frac{p(\cdot|\theta_0,\mathcal{M}_0)}{q_\alpha}] \simeq 1 + \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}(w_{i,j}-1)^2$, where the sum is over the trajectories from all the proposals and $w_{i,j}$ are their importance weights. This is in practice much better than using a naïve estimate of the second

moment, $\frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j} w_{i,j}^2$, which would lead to an infinite ESS when the target distribution p gets too far from q_α, and better than taking the sample variance of the weights, $1+\frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}(w_{i,j}-\bar{w})^2$, which would result in an ESS of n in such case. Since these degenerate cases are not uncommon in our context (recall the changes in target distribution during learning), it is extremely important to have a guard against them.

Finally, computing the number n_0 of defensive samples to be collected to guarantee a minimum ESS of $\mathrm{ESS}_{\min}$ requires the analysis of the increase rate of the function of Proposition 6.4.1 when adding target trajectories. Although, in the asymptotic case, this rate is 1 (i.e., every new target sample increases the ESS by 1), this may not hold when a finite sample is considered. However, we show that, for a given weight vector w, this rate cannot be worse than $c := \frac{\bar{w}_3+3(1-\bar{w})}{(1+\widehat{\mathbb{V}\mathrm{ar}}[w])^2}$, where $\bar{w}_3 = \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j} w_{i,j}^3$, $\bar{w} = \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j} w_{i,j}$, and $\widehat{\mathbb{V}\mathrm{ar}}[w] = \frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}(w_{i,j}-1)^2$. This leads to the following proposition, whose proof can be found in Appendix A.

Proposition 6.4.2. *The number n_0 of defensive samples to guarantee an ESS greater than or equal to $\mathrm{ESS}_{\min}$ can be computed as*

$$n_0 = \max\left\{n_{\min}, \min\left\{\mathrm{ESS}_{\min}, \left\lceil \frac{\mathrm{ESS}_{\min} - \frac{n}{1+\widehat{\mathbb{V}\mathrm{ar}}[w]}}{\min\{1,c\}} \right\rceil\right\}\right\}. \tag{6.15}$$

This rule is quite intuitive: we look at the difference between the minimum threshold $\mathrm{ESS}_{\min}$ and the current weights' ESS (estimated as $\frac{n}{1+\widehat{\mathbb{V}\mathrm{ar}}[w]}$). The result, rescaled by $1/c$, is the number of target samples we need to compensate. Finally, we clip the value in $[n_{\min}, \mathrm{ESS}_{\min}]$. In practice, we use this rule to compute n_0 at each step of IWPG.

The Case of Unknown Models

While we designed the importance-weighted gradient estimators and derived their theoretical results under the assumption of known transition models, we now discuss how the importance weights can be estimated from data, so as to make IWPG applicable in practice. In general, this goal can be efficiently achieved by directly estimating density ratios (Sugiyama et al., 2012). Unfortunately, in our settings, this is not an appealing approach. Besides the fact that, as already mentioned for IWFQI, density ratios are not transferable, i.e., they must be recomputed for each new policy and/or task, here we have a second complication: our weights are defined over entire trajectories, high-dimensional random variables whose distributions have some structure that would not be exploited by a direct estimator. Therefore, we decide to take a more indirect approach and estimate only the missing components, namely the transition models. Differently from IWFQI, where we reported good empirical result with the naive estimator that directly plugs in the estimated models to compute the importance weights, here we take a more principled approach. In particular, we use the intuition that any chosen set of models induces a bias-variance trade-off into the importance weights and, thus, into the resulting gradient estimator. Therefore, instead of naively plugging in any density estimator or probabilistic model of the uncertain models, we propose an estimator that is aware of the MIS scheme in which these models will be adopted by explicitly trading off bias and variance. As briefly outlined in Section 6.3, we suppose that the source transition models are estimated before the learning process

of the target starts and held fixed. This is reasonable since no additional source data can be collected by IWPG. Therefore, this section is devoted to the estimation of the target transition model P_0, which is updated in each iteration of IWPG with more data being collected. We consider the following assumption.

Assumption 6.5.1 (Uncertain Gaussian Target Model)**.** *Each task $\mathcal{M}_j$ has a Gaussian transition model $P_j(\cdot|s,a) = \mathcal{N}(f_j(s,a), \sigma^2 I)$ with known σ^2 and f_j uncertain according to a distribution $\varphi_j \in \mathcal{P}(\mathfrak{F})$, where $\mathfrak{F}$ is the set of all possible transition functions in the family of tasks $\mathfrak{M}$.*

Assumption 6.5.1 is standard in the model-based literature (e.g., Deisenroth and Rasmussen, 2011), where it is commonly assumed that the system dynamics are characterized by a non-linear function $f(s,a)$ plus some white Gaussian noise. The distributions φ_j describe our uncertainty over the true models given the dataset $\widetilde{\mathcal{D}}$.

As in the previous section, we freeze IWPG at some arbitrary iteration and consider the current dataset $\widetilde{\mathcal{D}}$ and target policy θ_0. The goal is once again to estimate $\nabla_\theta J(\theta_0, \mathcal{M}_0)$. For this purpose, we consider the MIS estimator (6.4) combined with the balance heuristic and seek an estimate of the target transition function $\widetilde{f}_0$ that provably reduces its mean-square error (i.e., that trades off bias and variance). To make this objective and Assumption 6.5.1 more clear, we rewrite the MIS estimator as

$$\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}}, \widetilde{f}_0) := \frac{1}{n} \sum_{j=0}^{m} \sum_{i=1}^{n_j} \frac{p(\tau_{i,j}|\theta_0, \widetilde{f}_0)}{\sum_{l=0}^{m} \alpha_l p(\tau_{i,j}|\theta_l, \widetilde{f}_l)} g_{\theta_0}(\tau_{i,j}) r(\tau_{i,j}), \tag{6.16}$$

where we made explicit the dependence on the "target variable" $\widetilde{f}_0$ and we replaced $\mathcal{M}_j$ by f_j in the trajectory distributions. Similarly, we make this dependence explicit in the mixture of proposal distributions, $q_\alpha(\tau|\widetilde{f}_0) = \sum_{j=0}^{m} \alpha_j p(\tau|\theta_j, \widetilde{f}_j)$. We start by deriving an upper bound to the mean-square error of the estimator (6.16) that better highlights the impact of choosing a wrong model $\widetilde{f}_0$.

Theorem 6.5.1. *Let $\widetilde{f}_0, \dots, \widetilde{f}_m : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be arbitrary transition functions and d be the number of policy parameters. Suppose that $\|g(\tau)r(\tau)\|_\infty \leq b$ almost surely. Then,*

$$\begin{aligned}
\mathbb{E}\left[\|\widehat{\nabla}_\theta J(\widetilde{\mathcal{D}}, \widetilde{f}_0) - \nabla_\theta J(\theta_0, \mathcal{M}_0)\|_2^2\right] &\leq \frac{b^2 d}{n} d_2\left(p(\cdot|\theta_0, \widetilde{f}_0) \| q_\alpha(\cdot|\widetilde{f}_0)\right) \\
&+ c_1 b^2 d \sum_{l=0}^{m} \alpha_l \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_l, \widetilde{f}_l)}\left[\|\bar{f}_l(S_t, A_t) - \widetilde{f}_l(S_t, A_t)\|_2^2\right] \\
&+ c_1 b^2 d \sum_{l=0}^{m} \alpha_l \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_l, \widetilde{f}_l)}\left[\mathrm{Tr}\left(\Sigma_l(S_t, A_t)\right)\right] + c_2,
\end{aligned} \tag{6.17}$$

where the expectation is with respect to $\tau_{i,j} \sim p(\tau|\theta_j, f_j)$ and $f_j \sim \varphi_j$. Here $\bar{f}_l(s,a) := \mathbb{E}_{f_l \sim \varphi_l}[f_l(s,a)]$, $\Sigma_l(s,a) = \mathbb{C}\mathrm{ov}_{f_l \sim \varphi_l}[f_l(s,a)]$, and c_1, c_2 are universal constants.

Proof. In order to simplify notation, let us define $\widehat{\nabla} J(\widetilde{f}_0) := \widehat{\nabla}_\theta J(\widetilde{\mathcal{D}}, \widetilde{f}_0)$ and $\nabla J := \nabla_\theta J(\theta_0, \mathcal{M}_0)$. We have

$$\mathbb{E}\left[\|\widehat{\nabla} J(\widetilde{f}_0) - \nabla J\|_2^2\right] = \mathbb{E}_{f_j \sim \varphi_j}\left[\mathbb{E}_{\tau_{i,j} \sim p(\cdot|\theta_j, f_j)}\left[\|\widehat{\nabla} J(\widetilde{f}_0) - \nabla J\|_2^2 | \{f_j\}_{j=0}^m\right]\right].$$

Let us fix $f_0, \dots, f_m$ and focus on the inner expectation. Using a bias-variance decomposition,

$$\begin{aligned}\mathbb{E}\left[\|\widehat{\nabla} J(\widetilde{f}_0) - \nabla J\|_2^2\right] &= \sum_d \mathbb{E}\left[\left(\widehat{\nabla}_d J(\widetilde{f}_0) - \nabla_d J\right)^2\right] \\ &= \sum_d \underbrace{\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_d J(\widetilde{f}_0)\right]}_{(a)} + \sum_d \underbrace{\left(\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f}_0)\right] - \nabla_d J\right)^2}_{(b)}.\end{aligned}$$

Term (a). Regarding the variance, we have

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_d J(\widetilde{f}_0)\right] &= \mathbb{V}\mathrm{ar}\left[\frac{1}{n}\sum_{j=0}^{m}\sum_{i=1}^{n_j}\frac{p(\tau_{i,j}|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau_{i,j}|\widetilde{f}_0)} g(\tau_{i,j}) r(\tau_{i,j})\right] \\ &= \frac{1}{n^2}\sum_{j=0}^{m} n_j \mathbb{V}\mathrm{ar}_{\tau\sim p(\cdot|\theta_j,f_j)}\left[\frac{p(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau|\widetilde{f}_0)} g(\tau) r(\tau)\right] \\ &\le \frac{1}{n}\mathbb{V}\mathrm{ar}_{\tau\sim q_\alpha(\tau|f_0)}\left[\frac{p(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau|\widetilde{f}_0)} g(\tau) r(\tau)\right] \\ &\le \frac{1}{n}\mathbb{E}_{\tau\sim q_\alpha(\tau|f_0)}\left[\frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha^2(\tau|\widetilde{f}_0)} g^2(\tau) r^2(\tau)\right] \\ &\le \frac{b^2}{n}\mathbb{E}_{\tau\sim q_\alpha(\tau|f_0)}\left[\frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha^2(\tau|\widetilde{f}_0)}\right],\end{aligned}$$

where the first equality leverages trajectory independence and the first inequality follows from Lemma A.4.1 in Appendix A. The last expectation can be further decomposed as

$$\begin{aligned}\mathbb{E}_{\tau\sim q_\alpha(\tau|f_0)}\left[\frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha^2(\tau|\widetilde{f}_0)}\right] &= \int \left(q_\alpha(\tau|f_0) \pm q_\alpha(\tau|\widetilde{f}_0)\right)\frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha^2(\tau|\widetilde{f}_0)}\mathrm{d}\tau \\ &= \int \frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau|\widetilde{f}_0)}\mathrm{d}\tau + \int\sum_{j=0}^{m}\alpha_j\left(p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right)\frac{p^2(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha^2(\tau|\widetilde{f}_0)}\mathrm{d}\tau \\ &\le d_2\left(p(\tau|\theta_0,\widetilde{f}_0), q_\alpha(\tau|\widetilde{f}_0)\right) + \frac{1}{\alpha_0^2}\int\sum_{j=0}^{m}\alpha_j\left|p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right|\mathrm{d}\tau \\ &= d_2\left(p(\tau|\theta_0,\widetilde{f}_0), q_\alpha(\tau|\widetilde{f}_0)\right) + \frac{2}{\alpha_0^2}\sum_{j=0}^{m}\alpha_j D_{\mathrm{TV}}\left(p(\cdot|\theta_j,f_j), p(\cdot|\theta_j,\widetilde{f}_j)\right),\end{aligned}$$

where D_{TV} is the total variation divergence. Note that the last inequality is valid since $\frac{p(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau|\widetilde{f}_0)} \le \frac{1}{\alpha_0}$ thanks to the defensive component in $q_\alpha(\tau|\widetilde{f}_0)$. Thus,

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\widehat{\nabla}_d J(\widetilde{f})\right] \le{}& \frac{b^2}{n} d_2\left(p(\tau|\theta_0,\widetilde{f}_0), q_\alpha(\tau|\widetilde{f}_0)\right) \\ &+ \underbrace{\frac{2b^2}{\alpha_0^2 n}\sum_{j=0}^{m}\alpha_j D_{\mathrm{TV}}\left(p(\cdot|\theta_j,f_j), p(\cdot|\theta_j,\widetilde{f}_j)\right)}_{(c)}.\end{aligned} \tag{6.18}$$

Term (b). First note that $\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f})\right]$ can be written as

$$\begin{aligned}\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f})\right] &= \frac{1}{n}\sum_{j=0}^{m} n_j \mathbb{E}_{\tau\sim p(\cdot|\theta_j,f_j)}\left[\frac{p(\tau|\theta_0,\widetilde{f}_0)}{q_\alpha(\tau|\widetilde{f}_0)} g_d(\tau) r(\tau)\right] \\ &= \int \frac{q_\alpha(\tau|f_0)}{q_\alpha(\tau|\widetilde{f}_0)} p(\tau|\theta_0,\widetilde{f}_0) g_d(\tau) r(\tau) \mathrm{d}\tau,\end{aligned}$$

while $\nabla_d J = \int p(\tau|\theta_0, f_0) g_d(\tau) r(\tau) \mathrm{d}\tau$. Then,

$$\begin{aligned}&\left|\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f})\right] - \nabla_d J\right| = \left|\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f})\right] \pm \int p(\tau|\theta_0,\widetilde{f}_0) g_d(\tau) r(\tau)\mathrm{d}\tau - \nabla_d J\right| \\ &\leq \left|\int \left(\frac{q_\alpha(\tau|f_0)}{q_\alpha(\tau|\widetilde{f}_0)} - 1\right) p(\tau|\theta_0,\widetilde{f}_0) g_d(\tau) r(\tau)\mathrm{d}\tau\right| + \left|\int \left(p(\tau|\theta_0,\widetilde{f}_0) - p(\tau|\theta_0,f_0)\right) g_d(\tau) r(\tau)\mathrm{d}\tau\right| \\ &\leq b\int \left|\frac{q_\alpha(\tau|f_0)}{q_\alpha(\tau|\widetilde{f}_0)} - 1\right| p(\tau|\theta_0,\widetilde{f}_0)\mathrm{d}\tau + b\int \left|p(\tau|\theta_0,\widetilde{f}_0) - p(\tau|\theta_0,f_0)\right|\mathrm{d}\tau \\ &\leq \frac{b}{\alpha_0}\int \left|q_\alpha(\tau|f_0) - q_\alpha(\tau|\widetilde{f}_0)\right|\mathrm{d}\tau + b\int \left|p(\tau|\theta_0,\widetilde{f}_0) - p(\tau|\theta_0,f_0)\right|\mathrm{d}\tau \\ &= \frac{b}{\alpha_0}\sum_{j=0}^{m}\alpha_j \int \left|p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right|\mathrm{d}\tau + b\int \left|p(\tau|\theta_0,\widetilde{f}_0) - p(\tau|\theta_0,f_0)\right|\mathrm{d}\tau.\end{aligned}$$

Since the first addendum contains the second one (for $j=0$), this equation can be upper bounded by $\frac{2b}{\alpha_0}\sum_{j=0}^{m}\alpha_j \int \left|p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right|\mathrm{d}\tau$. Thus,

$$\begin{aligned}\left(\mathbb{E}\left[\widehat{\nabla}_d J(\widetilde{f})\right] - \nabla_d J\right)^2 &\leq \frac{4b^2}{\alpha_0^2}\left(\sum_{j=0}^{m}\alpha_j \int \left|p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right|\mathrm{d}\tau\right)^2 \\ &\leq \frac{4b^2}{\alpha_0^2}\sum_{j=0}^{m}\alpha_j\left(\int \left|p(\tau|\theta_j,f_j) - p(\tau|\theta_j,\widetilde{f}_j)\right|\mathrm{d}\tau\right)^2 \\ &= \underbrace{\frac{8b^2}{\alpha_0^2}\sum_{j=0}^{m}\alpha_j D_{\mathrm{TV}}\left(p(\cdot|\theta_j,f_j)\|p(\cdot|\theta_j,\widetilde{f}_j)\right)^2}_{(d)}.\end{aligned}$$

Then, combining (c) and (d), we obtain

$$\frac{8b^2}{\alpha_0^2}\sum_{j=0}^{m}\alpha_j\left(\frac{D_{\mathrm{TV}}\left(p(\cdot|\theta_j,f_j)\|p(\cdot|\theta_j,\widetilde{f}_j)\right)}{4n} + D_{\mathrm{TV}}\left(p(\cdot|\theta_j,f_j)\|p(\cdot|\theta_j,\widetilde{f}_j)\right)^2\right).$$

Since $kx \leq x^2 + \frac{k^2}{2}$, this equation can be upper bounded by

$$(c)+(d) \leq \underbrace{\frac{16b^2}{\alpha_0^2}\sum_{l=0}^{m}\alpha_l D_{\mathrm{TV}}\left(p(\cdot|\theta_l,f_l)\|p(\cdot|\theta_l,\widetilde{f}_l)\right)^2}_{(e)} + \underbrace{\frac{b^2}{4\alpha_0^2 n^2}}_{(f)}.$$

Here (f) is the constant c_2 in the final bound, while (e) can be upper bounded using Pinsker's inequality as

$$
\begin{aligned}
(e) &\leq \frac{8b^2}{\alpha_0^2} \sum_{l=0}^{m} \alpha_l D_{\mathrm{KL}} \left(p(\cdot|\theta_l, \widetilde{f}_l), p(\cdot|\theta_l, f_l) \right) \\
&\leq \frac{8b^2}{\alpha_0^2} \sum_{l=0}^{m} \alpha_l \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_l, \widetilde{f}_l)} \left[D_{\mathrm{KL}} \left(\widetilde{P}_l(\cdot|S_t, A_t), P_l(\cdot|S_t, A_t) \right) \right],
\end{aligned}
$$

where we applied the recursive property of KL divergences over trajectory distributions. Putting these terms together, we obtain

$$
\begin{aligned}
&\mathbb{E}\left[\|\widehat{\nabla} J(\widetilde{f}_0) - \nabla J\|_2^2 | \{f_j\}_{j=0}^m \right] \leq \frac{b^2}{n} d_2 \left(p(\tau|\theta_0, \widetilde{f}_0), q_\alpha(\tau|\widetilde{f}_0) \right) \\
&+ \frac{8b^2}{\alpha_0^2} \sum_{l=0}^{m} \alpha_l \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_l, \widetilde{f}_l)} \left[D_{\mathrm{KL}} \left(\widetilde{P}_l(\cdot|S_t, A_t), P_l(\cdot|S_t, A_t) \right) \right] + c_2.
\end{aligned}
$$

If we now consider the outer expectation over $f_j \sim \varphi_j$, we note that only the bias term depends on f_j. Since $D_{\mathrm{KL}} \left(\widetilde{P}_j(\cdot|s_t, a_t), P_j(\cdot|s_t, a_t) \right) = \frac{1}{2\sigma^2} \|f_j(s_t, a_t) - \widetilde{f}_j(s_t, a_t)\|_2^2$, the expected KL divergence is

$$
\begin{aligned}
&\mathbb{E}_{f_j \sim \varphi_j} \left[D_{\mathrm{KL}} \left(\widetilde{P}_j(\cdot|S_t, A_t), P_j(\cdot|S_t, A_t) \right) \right] \\
&= \frac{1}{2\sigma^2} \|\bar{f}_j(s_t, a_t) - \widetilde{f}_j(s_t, a_t)\|_2^2 + \frac{1}{2\sigma^2} \mathrm{Tr} \left(\mathbb{C}\mathrm{ov}_{f_j \sim \varphi_j}[f_j(s, a)] \right).
\end{aligned}
$$

Plugging this into the previous display and summing up over all gradient dimensions concludes the proof. □

Let $\mathcal{L}(\widetilde{f}_0)$ denote the value of this bound as a function of $\widetilde{f}_0$. Then, we seek $\widetilde{f}_0$ that minimizes $\mathcal{L}$. Intuitively, we seek for a model that trades off between three different objectives: (1) when few trajectories are available and the target model is highly uncertain, it should stay close to the mixture of source distributions in order to reduce the variance of the resulting estimator (first term); (2) as the number of samples grows, it should move towards $\bar{f}_0$, our best guess for the true model (second term); (3) finally, it should give priority to the regions of the state-action space where the target model is more accurate (third term). The source models $\{\widetilde{f}_j\}_{j=1}^m$, on the other hand, are computed before learning starts. For instance, taking $\widetilde{f}_j = \bar{f}_j$ for all $j = 1, \ldots, m$ (which, for some uncertainty models, like GPs, corresponds to the maximum-a-posteriori) is an appealing choice since it makes the bias term vanish. Note that, although we consider only $\widetilde{f}_0$ as target model, more source proposals might be generated under the target MDP. The bound above can be straightforwardly modified to account this fact, though the result would be almost equivalent. Our MSE-aware approach to model estimation is summarized in Algorithm 4. Although appealing, optimizing the bound for the optimal transition function is non-trivial. We show that, by considering additional assumptions on the underlying structured domain, this can be done efficiently.

Discrete Task Family

We start by considering the simple setting in which $\mathfrak{F} = \{f^1, f^2, \ldots, f^{|\mathfrak{F}|}\}$ is a finite set of possible transition functions. Our uncertainty model for the target transition function f_0

Algorithm 4 MSE-aware Model Estimation

Require: Model space $\mathfrak{F}$, uncertainty model φ_0, dataset $\widetilde{\mathcal{D}} = \left\{(\{\tau_{i,j}\}_{i=1}^{n_j}, \theta_j, \widetilde{P}_j)\right\}_{j=0}^{m}$

1: Update φ_0 using the target trajectories $\{\tau_{i,0}\}_{i=1}^{n_0}$ in $\widetilde{D}$
2: Compute $\widetilde{f}_0 \in \mathfrak{F}$ minimizing the bound of Theorem 6.5.1
3: **return** $\widetilde{f}_0$

is therefore a discrete distribution over this set. Assuming a uniform prior $\varphi_0^0(f) = \frac{1}{|\mathfrak{F}|}$, this distribution can be updated iteratively for every $f \in \mathfrak{F}$ given a batch of n_0 target trajectories $\{\tau_i\}_{i=1}^{n_0}$ under policy π_{θ^k} as

$$\varphi_0^{k+1}(f) \propto \varphi_0^k(f) \prod_{i=1}^{n_0} \prod_{t=0}^{T-1} \pi_{\theta^k}(A_t|S_t)\mathcal{N}(S_{t+1}; f(S_t, A_t), \sigma^2 I).$$

Given φ_0^k, the bound of Theorem 6.5.1 can be easily approximated for every $f \in \mathfrak{F}$. Notice that all expectations in (6.17) are under distributions induced by the chosen models $\{\widetilde{f}_j\}_{j=0}^{m}$ and, therefore, they can be approximated by simulating trajectories without interacting with the true environment.

Reproducing Kernel Hilbert Spaces

We now consider a more general functional space for our transition models. Let $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ and consider a positive semi-definite kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We suppose that $\mathfrak{F}$ is the unique reproducing kernel Hilbert space (RKHS) induced by $\mathcal{K}$. In an RKHS, the reproducing property implies that every function f can be written as $f(x) = \langle f, \mathcal{K}(x, \cdot)\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the dot product on $\mathcal{F}$. For simplicity, we consider each dimension of our transition models separately. We refer the reader to Micchelli and Pontil (2005) for the extension to vector-valued RKHS.

We represent the uncertainty over the target model as a Gaussian process (GP) (Williams and Rasmussen, 2006), $f \sim \mathcal{GP}(0, \mathcal{K})$, with a zero-mean prior and $\mathcal{K}$ as covariance function. As usual in model-based RL (e.g., (Deisenroth and Rasmussen, 2011)), we train conditionally independent GPs for each dimension of the state space. Suppose that we get a set of l training inputs $X = [x_1, \ldots, x_l]^T$ and l training targets $Y = [y_1, \ldots, y_l]^T$ from a batch of trajectories from the target task, where $x_i = (s_i, a_i)$ and $y_i = f_0(x_i) + \mathcal{N}(0, \sigma^2 I)$. Then, the posterior mean function can be evaluated at any point x as $\bar{f}_0(x) = k(x)^T(K + \sigma^2 I)^{-1}Y$ (Williams and Rasmussen, 2006), where $k(x)$ is the vector with entries $k_i(x) = \mathcal{K}(x_i, x)$ and K is the Gram matrix, $K_{ij} = \mathcal{K}(x_i, x_j)$.

Now that we have an uncertainty model for our target transition function, let us move to optimize our bound $\mathcal{L}$ on the MSE. Unfortunately, minimizing $\mathcal{L}(\widetilde{f}_0)$ with respect to $\widetilde{f}_0 \in \mathfrak{F}$ is not as simple as in the finite-model case since (1) $\widetilde{f}_0$ controls the distributions under which expectations are taken, and (2) it appears as a product over several time steps in the Renyi divergence term. For these reasons, we now introduce some simplifications that will lead to a convenient closed-form solution. First, we approximate the two expectations by drawing a small number of trajectories from our last hypothesized model, so that their dependence on the function to be computed is removed. Secondly, we further bound our

objective in a more convenient way. In order to carry out this last step, we derive an upper bound on the exponentiated Renyi divergence with respect to the Kullback-Leibler (KL) divergence, which could be of independent interest. The proof can be found in Appendix A.

Theorem 6.5.2. *Let $(\mathcal{X}, \mathscr{F})$ be a measurable space, P and Q be two probability measures on $\mathcal{X}$ such that $P \ll Q$, and $Q_\alpha = \alpha P + (1-\alpha)Q$ denotes their convex combination with coefficient $\alpha \in (0,1)$. Suppose there exists a finite constant $C > 0$ such that* $\operatorname{ess\,sup} \frac{dP}{dQ} \leq C$. *Then,*

$$d_2(P||Q_\alpha) \leq 1 + u(\alpha) D_{KL}(P||Q), \tag{6.19}$$

where

$$u(\alpha) = \begin{cases} \frac{2C(1-\alpha)^2}{(\alpha C + 1 - \alpha)^3} & \text{if} \quad C \leq \frac{1-\alpha}{2\alpha} \\ \frac{8}{27\alpha} & \text{otherwise.} \end{cases}$$

Using Theorem 6.5.2, our objective $\mathcal{L}(\widetilde{\mathcal{M}})$ can be bounded in a very convenient way.

Proposition 6.5.1. *There exist constants k_1, k_2, k_3 (independent of $\widetilde{f}_0$) such that objective $\mathcal{L}(\widetilde{f}_0)$ given in* (6.17) *can be bounded by*

$$\mathcal{L}(\widetilde{f}_0) \leq k_1 \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)} \left[\sum_{j=1}^{m} \alpha_j \|\widetilde{f}_0(x_t) - \widetilde{f}_j(x_t)\|_2^2 \right]$$
$$+ k_2 \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)} \left[\|\widetilde{f}_0(x_t) - \bar{f}_0(x_t)\|_2^2 \right] + k_3.$$

Our new bound is quite appealing. While the bias term remains unchanged, the variance is now a mixture of expected l_2 distances between $\widetilde{f}_0$ and the (approximate) source models $\widetilde{f}_j$. In practice, we optimize a regularized version of this objective so that the representer theorem of RKHS applies. Furthermore, as mentioned above, we approximate the two expectations by drawing R trajectories from $p(\cdot|\theta_0, \widetilde{f}_0)$ using our last hypothesized model. The resulting objective reduces to a regularized least-squares problem,

$$\operatorname*{argmin}_{\widetilde{f}_0 \in \mathfrak{F}} \frac{1}{R} \sum_{r=1}^{R} \sum_{t=0}^{T-1} \Bigg(k_1 \sum_{j=0}^{m} \alpha_j \|\widetilde{f}_0(x_{r,t}) - \widetilde{f}_j(x_{r,t})\|_2^2$$
$$+ k_2 \|\widetilde{f}_0(x_{r,t}) - \bar{f}_0(x_{r,t})\|_2^2 \Bigg) + \lambda \|\widetilde{f}_0\|_{\mathcal{K}}^2, \tag{6.20}$$

where $\lambda > 0$ is the regularization parameter. Most importantly, its solution is available in closed form.

Proposition 6.5.2. *The function $f^* \in \mathfrak{F}$ minimizing* (6.20) *is*

$$f^*(x) = A^T k(x),$$

where $k(x)$ is the RT-dimensional vector with entries $\mathcal{K}(x_{r,t}, x)$ and

$$A = (k_1 K + k_2 K + \lambda R I)^{-1} (k_1 F_{src} + k_2 \bar{F}),$$

with K being the Gram matrix, $\bar{F} = [\bar{f}_0(x_{1,0}), \dots, \bar{f}_0(x_{R,T-1})]^T$, $F_{src} = \sum_{j=1}^{m} \alpha_j F_j$, and $F_j = [\widetilde{f}_j(x_{1,0}), \dots, \widetilde{f}_j(x_{R,T-1})]^T$.

Proof. Using the representer theorem of RKHS, we have that, for the i-th dimension of the state space we can write $f_i^*(x) = a_i^T k(x)$ for some vector of parameters a_i. The proof follows by taking the gradient of the objective with respect to a_i, equating to zero, and solving for a_i. □

Discussion. One might be wondering why the estimated transition models are not directly used for planning in a standard model-based RL algorithm instead of estimating the importance weights for a model-free approach. It is well-known that even small errors can lead to disastrous performances when the optimal policy is computed under the estimated models. Since in our case the learned models are only used to re-weight samples from the true environment, we argue that the impact of such errors is much more contained. In fact, as far as the weights keep reasonable values, the learning process could potentially be carried out effectively. Furthermore, note that our estimators are consistent as the number of target samples goes to infinity for *any* hypothesized model, not necessarily the true one. Finally, note the differences between the model-estimation approach that we propose here and the one we adopted for IWFQI. The latter is almost equivalent to choosing $\widetilde{f}_j = \bar{f}_j$ for all tasks, i.e., the functions predicted by the GPs. While these are indeed the "best fit" for the true functions given limited data, they are not necessarily the best candidates for reducing the mean-square error of an importance-weighted estimator. The technique developed in this section, on the other hand, explicitly takes this objective into account and, as we shall confirm in our experiments, is indeed more robust to limited data and misspecified models.

Experiments

We evaluate the performance of IWPG on different control tasks. We start by analyzing the different estimators with known models in Section 6.6.1. Then, we consider the full approach with model estimation in Sections 6.6.2 and 6.6.3. We provide the high-level description of each experiment. For the specific hyper-parameters, we refer the reader to Tirinzoni et al. (2019).

Linear-Quadratic Regulator

Our first test domain is the one-dimensional linear-quadratic regulator (LQR) (Dorato et al., 1995; Peters and Schaal, 2008b), a well-known benchmark from the control literature. The system has linear dynamics, $s_{t+1} = As_t + Ba_t + \epsilon$, with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_{\mathbb{P}}^2)$, and quadratic rewards.

We begin by evaluating the MIS estimators proposed in Section 6.4. Besides our proposed estimators, we compare to per-decision IS (PD-IS), which is widely adopted in the literature and can be straightforwardly adapted to our case, and to G(PO)MDP (Baxter and Bartlett, 2001) as our no-transfer baseline.

Transfer experiment. We used Gaussian policies with a linearly parameterized mean and fixed variance.. We set the maximum horizon to $T = 20$. For each run, we randomly

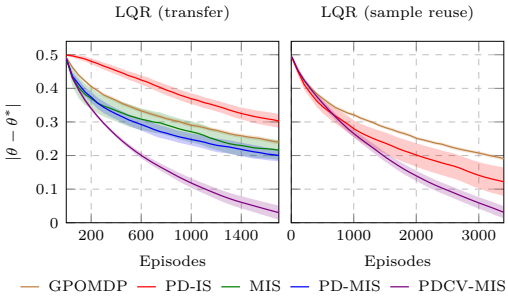

Figure 6.1: *Comparison of the proposed gradient estimators in the LQR domain under known models. (left) transfer experiment, and (right) sample reuse experiment. Each curve is the average of* 40 *independent runs, each re-sampling the source tasks, with Student's t* 95% *confidence intervals*

generated 5 source tasks by uniformly sampling A in $[0.6, 1.4]$ and B in $[0.8, 1.2]$, while the target task was fixed with $A = 1$ and $B = 1$. We considered 8 policies with parameters $\{-0.1, -0.2, \ldots, -0.8\}$ and generated 20 episodes from each model-policy couple to built our initial source dataset. To have a fair comparison, we used the same learning rate of 1e-5 and the same initialization $\theta_0 = -0.1$ for all algorithms. We set the batch size of G(PO)MDP to 10 and used $n_{\min} = 5$ and $\mathrm{ESS}_{\min} = 20$. We learned the target task using standard SGD. Figure 6.1*(left)* shows the distance to the optimal target parameter as a function of the number of episodes. As expected, PD-IS shows a significant amount of negative transfer with respect to G(PO)MDP. This is due to the fact that the huge variance of the importance weights forces the algorithm to collect large batches to guarantee the required ESS. MIS and PD-MIS achieve an improvement over the no-transfer baseline, with the latter having smaller variance. When introducing CVs, the algorithm enjoys much better gradient estimates and significantly outperforms all alternatives. Figure 6.2*(left)* shows the expected return achieved by all alternatives as a function of the number of episodes. The results are coherent with Figure6.1*(left)*, although the differences between the algorithms' performances are harder to appreciate. Figure 6.2(center) shows how the ESS changes at each iteration. The ESS of PD-IS remains almost constant, which is due to the fact that general IS estimators highly depend on the chosen proposal distributions. The MIS estimators do not suffer this problem and their ESS linearly increases with the number of iterations. Finally, Figure 6.2 shows the number of samples collected by each algorithm at each iteration. Coherently with the plot of the ESS, PD-IS needs to collect a high number of samples to meet the $\mathrm{ESS}_{\min}$ requirement. On the other hand, all transfer algorithms manage to learn while sampling the minimum number of trajectories allowed.

Intra-environment transfer. Our estimators can be successfully adopted to reuse samples generated by previous policies in standard (no-transfer) policy gradients. Figure 6.1*(right)* shows the result of learning the same target task as before from scratch (i.e., without any source sample), where each algorithm uses the same (fixed) batch size, learning rate, and initialization. We can appreciate that both the per-decision and our estimator enjoy a speedup over G(PO)MDP, but the former suffers a much higher variance.

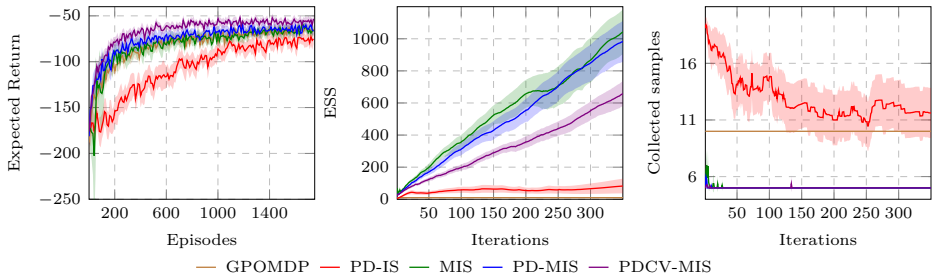

Figure 6.2: *Additional statistics for the LQR experiment of Figure 6.1(left). Expected return (left), effective sample size (center), and the number of samples collected at each iteration (right).*

Cart-pole Balancing

Our second domain is the well-known Cartpole problem (Sutton and Barto, 2018), where the goal is to balance a pole on a moving cart. We generate source tasks by uniformly sampling the mass of the cart in the interval $[0.8, 1.2]$ and the length of the pole in the interval $[0.3, 0.7]$. The target task is the standard Cartpole with cart mass 1.0 and pole length 0.5. For each of 5 source tasks, we consider a sequence of 10 linear policies generated by G(PO)MDP during its learning process and collect 10 episodes from each. While the LQR domain was relatively short-horizon ($T = 20$), here we allow trajectories up to $T = 200$ time steps, which allows us to verify the transferability of long state-action sequences.

We now test our model estimation approaches. Besides G(PO)MDP, we report the performance of the sample reuse (SR) variant of our algorithm where we ignore the source tasks and transfer only past target trajectories. All transfer algorithms use the PD-MIS estimator. We report three different model estimation alternatives combined with IWPG: MSE (discrete) and MSE (RKHS) use the MSE-aware technique for estimating the target model with known sources and a discrete family or a RKHS, respectively; "unknown sources" report the performance of the MSE-aware estimator when the source models are directly estimated by fitting GPs. Figure 6.3*(left)* shows the results. Interestingly, all transfer approaches outperform both G(PO)MDP and SR, which confirms that reusing trajectories from different environments can significantly improve the quality of the learning process. Both our model estimation approaches perform comparably to the ideal estimators. While this should be expected for the discrete estimator, the continuous one works well since an accurate GP model of the Cartpole dynamics can be obtained with a relatively small amount of samples. This fact can be verified from the GP curve, where the weights have been estimated by directly plugging in the GP predictions instead of optimizing our bound. Not surprisingly, the algorithm that estimates the source models performs comparably to the best alternatives. However, the fact that the fitted GP leads to accurate estimators does not imply that it is an accurate model of the system dynamics which can be used for planning. The gray curve in Figure 6.3*(left)*, which shows the performance achieved by optimal policies for the estimated dynamics, confirms this statement.

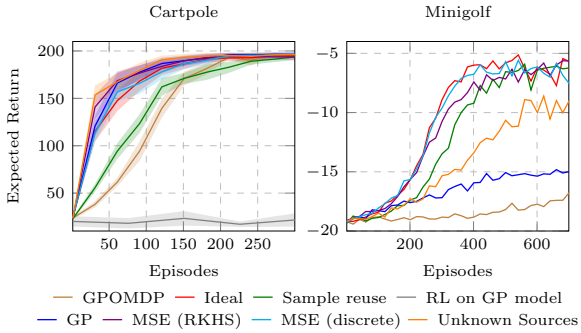

Figure 6.3: *Comparison of our model estimation approach in the Cartpole (left) and minigolf (right) domains. Each curve is the average of* 40 *independent runs, each re-sampling the source tasks, with Student's t* 95% *confidence intervals.*

Minigolf

In this last domain, we want to study how much the proposed transfer algorithms can speed up the learning process of an agent playing a minigolf game by reusing the experience made on different minigolf courses. The various tasks may differ in the length of the putter (between $70cm$ and $100cm$), in the hole size (between $10cm$ and $15cm$), and in the dynamic friction coefficient (whose range was measured empirically Penner (2002) between 0.065 and 0.196). The minigolf domain was originally introduced in the RL field by Lazaric et al. (2008a). We changed the dynamics following the modeling developed by Penner (2002) in order to make the problem more realistic (see Appendix D.3 of Tirinzoni et al. (2019)).

In this experiment, we adopt Gaussian policies with a fourth-order polynomial basis function. We generated 5 source tasks by randomly sampling dynamic friction coefficient, hole size, and putter length from the realistic ranges defined above. Furthermore, we considered 10 source policies of increasing quality and generated 40 episodes from each model-policy pair. The target task is fixed with a friction of 0.131, a putter of $100cm$, and a hole of diameter $10cm$. All transfer algorithms use the PDCV estimator. The results are shown in Figure 6.3*(right)*. Unlike the simpler Cartpole domain, G(PO)MDP is not able to learn the task in such a small number of episodes. Interestingly, due to the high level of noise present in this environment, direct estimation of weights using the GP predictions leads to unsatisfactory results. On the other hand, both our model estimation approaches solve the task with performance comparable to the ideal estimator. We note that the speed-up over the SR algorithm is not as remarkable as in the previous experiment due to the little amount of knowledge transfer that can be achieved in this more complicated setting. We finally point out that most of the simplifications introduced in the previous sections do not hold in this domain. In fact, the transition models are not Gaussian, while the noise is heteroscedastic and changes between tasks. Despite these relaxations, our approach can be applied without suffering any considerable performance degradation.

Part II

Variational Transfer Methods

CHAPTER 7

Exploration via Variational Value Transfer

This chapter is based on the paper "Transfer of Value Functions via Variational Methods" co-authored with Rafael Rodriguez Sanchez and Marcello Restelli and published at NeurIPS 2018.

Introduction

In the previous chapters, we have seen how to design methods that efficiently transfer experience samples across tasks with different transition dynamics and reward functions. While sample transfer can be potentially combined with any base algorithm and has a number of real-world applications, its study neglects another fundamental problem in the transfer literature: how an agent should *act*, i.e., how it should *explore*, in a new target environment when provided with prior knowledge from a set of related source tasks. In some sense, the two problems are almost orthogonal: sample transfer focuses on how to reuse *old* source data in order to augment the agent's experience in the target task, while the exploration problem focuses on how to collect *new* data in the same task, both with the goal of improving the learning process. While the exploration-exploitation dilemma is well studied in classic reinforcement learning without prior knowledge (e.g., Auer et al., 2009; Strehl et al., 2009; Azar et al., 2017), when the agent acts in a structured domain the problem becomes highly non-trivial. Here a good agent is expected to exploit the source knowledge to improve its exploration policy. Consider, for instance, a family of tasks that are equivalent in some region of the state-action space. Then, when provided with sufficient knowledge from the source tasks, the agent can avoid exploring such region at

all in the target task. This is in contrast to what the agent would do when learning from scratch, where a good exploration policy should visit the whole (reachable) state-action space. In this chapter, we start by studying this problem on the *practical* side. That is, our goal is to design algorithms that scale to complex problems with continuous state spaces and that transfer source knowledge to obtain an informed exploration behavior in new tasks. We defer a theoretical treatment of this problem in simpler tabular MDP and bandit settings to Part III.

Under the usual assumption that tasks in the given family are drawn from an unknown distribution, an intuitive choice to design a transfer algorithm is to characterize the uncertainty over the target task or its solution. Then, an ideal algorithm would leverage prior knowledge from the source tasks to explore the target task so as to reduce this uncertainty. This simple intuition makes Bayesian methods appealing approaches for transfer in RL, and many previous works have been proposed in this direction. These include the hierarchical Bayesian models of Wilson et al. (2007); Lazaric and Ghavamzadeh (2010) and the hidden-parameter models of Doshi-Velez and Konidaris (2016); Killian et al. (2017). However, most of these algorithms require specific, and sometimes restrictive, assumptions (e.g., on the distributions involved or the function approximators adopted), which might limit their practical applicability. The importance of having transfer algorithms that alleviate the need for strong assumptions and easily adapt to different contexts motivates us to take a more general approach.

Similarly to Lazaric and Ghavamzadeh (2010), we assume that the tasks share similarities in their value functions and use the given source tasks to learn a distribution over such functions. Then, we use this distribution as a prior for learning the target task and we propose a variational approximation of the corresponding posterior which is computationally efficient[1]. Leveraging recent ideas from randomized value functions Osband et al. (2014); Azizzadenesheli et al. (2018); Osband et al. (2019), we design a Thompson Sampling-based algorithm that efficiently explores the target task by sampling repeatedly from the posterior and acting greedily with respect to the sampled value function. We show that our approach is very general, in the sense that it can work with any parametric function approximator and any prior/posterior distribution models, while we specifically combine it with Gaussian and mixture-of-Gaussian distributions. For these distributions, we derive a finite-sample analysis of our approach and numerically evaluate it on different continuous domains with increasing level of difficulty.

Preliminaries

We consider a structured domain $\mathfrak{E} = (\mathfrak{M}, \mathfrak{D})$, where $\mathfrak{M}$ is a family of MDPs with shared state-action space $\mathcal{S} \times \mathcal{A}$ and the task-generation process $\mathfrak{D}$ is a probability distribution over $\mathfrak{M}$ from which i.i.d. tasks are sampled. We suppose that, after having solved a set of m source tasks $\{\mathcal{M}_1, \dots, \mathcal{M}_m\}$, the agent is required to solve a new target task $\mathcal{M}_0$. Each task is an MDP[2] $\mathcal{M}_j = (\mathcal{S}, \mathcal{A}, P_j, r_j, \gamma)$ drawn i.i.d. from $\mathfrak{D}$, i.e., $\mathcal{M}_j \sim \mathfrak{D}$. In this chapter, we focus on value-based methods with function approximation. We consider a

[1] Hence the the name "variational transfer method" refers to algorithms that use variational approximations of distributions induced by the different tasks for knowledge transfer.

[2] For simplicity, we assume the reward function of each task to be deterministic, though stochastic rewards can be treated with no additional efforts.

parametric family of action-value functions (in the remaining called simply Q-functions), $\mathcal{Q} = \{Q_\omega : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \mid \omega \in \mathbb{R}^d\}$, and we assume each function in $\mathcal{Q}$ to be uniformly bounded by $\frac{r_{\max}}{1-\gamma}$. We only require that any function $Q_\omega \in \mathcal{Q}$ is differentiable with respect to ω, while we do not pose any other restriction on the chosen function approximator (which can be, e.g., a linear model or a neural network). We suppose each task to be solved using the same parametrized functional space, so that the agent is given a set of source parameters $\mathcal{W}_{\text{src}} = \{\omega_1, \omega_2, \dots, \omega_m\}$ such that $Q_{\omega_j} \simeq Q_j^\star$ for all $j = 1, \dots, m$, where $Q_j^\star$ is the optimal Q-function for task $\mathcal{M}_j$. The goal is to exploit this prior knowledge to quickly solve the target task, i.e., to find ω_0 such that $Q_{\omega_0} \simeq Q_0^\star$. No additional prior knowledge is required besides the source parameters $\mathcal{W}_{\text{src}}$.

Remark 7.2.1. *Since the approaches we derive in this chapter use only the source parameters $\mathcal{W}_{\text{src}}$ as prior knowledge and we never refer to the source MDP models, in the remaining we shall drop the subscript "0" from MDP components denoting the target task whenever clear from the context.*

Finally, we need to specify the measures we adopt to assess the quality of an approximate Q-function Q_ω. A possible measure is its *Bellman error* (or Bellman residual), defined by $B_\omega := T^\star Q_\omega - Q_\omega$. If we assume the existence of a distribution ν over $\mathcal{S} \times \mathcal{A}$, a sound objective is to directly minimize the squared Bellman error of Q_ω under ν, denoted by $\|B_\omega\|_\nu^2$. Unfortunately, it is well known that an unbiased estimator of this quantity requires two independent samples of the next state s' for each s, a (e.g., Maillard et al. (2010)). In practice, the Bellman error is typically replaced by the TD error $b(\omega)$, i.e., the random quantity $b(\omega) = R + \gamma \max_{a'} Q_\omega(S', a') - Q_\omega(S, A)$ which approximates the former using a single transition sample. Finally, given a dataset $\mathcal{D} = \{(S_i, A_i, S_i', R_i)\}_{i=1}^n$ of n samples, the squared TD error is computed as $\|B_\omega\|_{\mathcal{D}}^2 = \frac{1}{n}\sum_{i=1}^n (R_i + \gamma \max_{a'} Q_\omega(S_i', a') - Q_\omega(S_i, A_i))^2 = \frac{1}{n}\sum_{i=1}^n b_i(\omega)^2$.

Variational Value Transfer

Let us observe that the distribution $\mathfrak{D}$ over MDPs induces a distribution over their optimal Q-functions. In other words, if known, this distribution effectively acts as a prior over the solution (i.e., the optimal Q-function) of tasks drawn from $\mathfrak{D}$. Since in our setting we represent action-value functions by a space $\mathcal{Q}$ of limited capacity, we can replace this distribution by one over $\mathcal{Q}$ or, more simply, by one over parameters ω. Call $p \in \mathcal{P}(\mathbb{R}^d)$ this distribution, such that p puts high probability mass to parameters ω that are likely to induce near-optimal Q-functions on tasks drawn from $\mathfrak{D}$.

Assume, for the moment, that we know the distribution $p(\omega)$ and consider a dataset $\mathcal{D} = \{(S_i, A_i, S_i', R_i)\}_{i=1}^n$ of samples from the target task $\mathcal{M}_0 \sim \mathfrak{D}$ that we want to solve. Then, we can compute the posterior distribution over parameters given such dataset by applying Bayes theorem as $p(\omega|\mathcal{D}) \propto p(\mathcal{D}|\omega)p(\omega)$. Unfortunately, this cannot be directly used in practice since we do not have a model of the likelihood $p(\mathcal{D}|\omega)$. In such case, it is very common to make strong assumptions on the MDPs or the Q-functions to get tractable posteriors. However, in our transfer settings, all distributions involved depend on the family of tasks under consideration and making such assumptions is likely to limit the applicability to specific problems. Thus, we take a different approach to derive a more general, but still well-grounded, solution. We use the intuition that, given a dataset $\mathcal{D}$ of

experience samples from the target task, the quality of a Q-function $Q_\omega \in \mathcal{Q}$ (i.e., how well it approximates $Q_0^\star$) can be measured by its TD error $\|B_\omega\|_\mathcal{D}^2$. In order to derive a meaningful posterior distribution, we take inspiration from PAC-Bayesian theory (Guedj, 2019). Note that, given the prior $p(\omega)$, our final goal is to move the total probability mass over the parameters minimizing this empirical loss measure, i.e., the TD error. Then, from PAC-Bayesian theory we have that the optimal Gibbs posterior q, which minimizes an oracle upper bound on the expected loss, takes the form (e.g., Catoni (2007)):

$$q(\omega) = \frac{e^{-\Lambda\|B_\omega\|_\mathcal{D}^2} p(\omega)}{\int e^{-\Lambda\|B_{\omega'}\|_\mathcal{D}^2} p(d\omega')}, \tag{7.1}$$

for some parameter $\Lambda > 0$. Since Λ is typically chosen to increase with the number of samples n, in the remaining, we set it to $\lambda^{-1}n$, for some constant $\lambda > 0$. Note that, whenever the term $e^{-\Lambda\|B_\omega\|_\mathcal{D}^2}$ can be interpreted as the actual likelihood of $\mathcal{D}$, q becomes a classic Bayesian posterior. Anyway, this specific form has a quite intuitive meaning: a dataset $\mathcal{D}$ is more likely given that ω is approximately optimal if it induces low TD error on Q_ω. Although we now have an appealing distribution, the integral at the denominator of (7.1) is intractable to compute even for simple Q-function models. Thus, we propose a variational approximation q_ξ by considering a simpler family of differentiable distributions parameterized by $\xi \in \Xi$. Then, our problem reduces to finding the variational parameters ξ such that q_ξ minimizes the KL divergence with respect to the Gibbs posterior q. From the theory of variational inference (e.g., Blei et al. (2017)), this can be shown to be equivalent to minimizing the (negative) *evidence lower bound* (ELBO):

$$\min_{\xi\in\Xi} \mathcal{L}(\xi) := \mathbb{E}_{\omega\sim q_\xi}\left[\|B_\omega\|_\mathcal{D}^2\right] + \frac{\lambda}{n} D_{\mathrm{KL}}\left(q_\xi(\omega)\|p(\omega)\right). \tag{7.2}$$

Intuitively, the approximate posterior balances between placing probability mass over those weights ω that have low expected TD error (first term), and staying close to the prior distribution (second term). Assuming that we can compute the gradients of (7.2) with respect to the variational parameters ξ, our objective can be optimized using any stochastic optimization algorithm.

Algorithm. The overall idea behind our variational value transfer algorithm is quite simple: we estimate the prior distribution $p(\omega)$ from the source parameters $\mathcal{W}_{\mathrm{src}}$ and use it to guide the whole learning process of the target task. More precisely, we interact with the target task online while maintaining and updating the approximate posterior distribution $q_\xi(\omega)$ from the observed samples. At each time step, we use posterior sampling (Thompson, 1933; Osband et al., 2014) to drive exploration; assuming that the agent is in state S_t, we first sample ω from the current posterior q_ξ and then act greedily with respect to Q_ω, i.e., we choose $A_t = \mathrm{argmax}_{a\in\mathcal{A}} Q_\omega(S_t, a)$. After collecting new experience, we update the current posterior by stochastic gradient descent on $\mathcal{L}$ and repeat the whole process. The detailed pseudo-code is reported in Algorithm 5. Note that the resulting procedure resembles approximate Q-learning, e.g., a DQN (Mnih et al., 2015), where, instead of directly minimizing the TD error, we minimize its expectation under weights drawn from the learned distribution, which is regularized to stay close to the prior, hence exploiting source knowledge. Similarly to a DQN, we use *experience replay* to optimize $\mathcal{L}$, i.e., at each step we draw a mini-batch from $\mathcal{D}$ in order to compute $\nabla_\xi \mathcal{L}$.

Algorithm 5 Variational Transfer

Require: Unknown target task $\mathcal{M}_0$, source parameters $\mathcal{W}_{\text{src}}$
1: Estimate prior $p(\omega)$ from $\mathcal{W}_{\text{src}}$
2: Initialize variational parameters: $\xi \leftarrow \operatorname{argmin}_\xi D_{\text{KL}}(q_\xi \| p)$
3: Initialize dataset: $\mathcal{D} = \emptyset$
4: **repeat**
5: Sample initial state: $S_0 \sim \rho_0$
6: **while** S_t is not terminal **do**
7: Sample weights: $\omega \sim q_\xi(\omega)$
8: Take greedy action: $A_t = \operatorname{argmax}_a Q_\omega(S_t, a)$
9: Observe transition $S_{t+1} \sim \mathcal{P}_0(\cdot|S_t, A_t)$ and reward $R_{t+1} = r_0(S_t, A_t)$
10: Store data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S_t, A_t, S_{t+1}, R_{t+1})\}$
11: Estimate gradient $\nabla_\xi \mathcal{L}(\xi)$ using a mini-batch $\mathcal{D}' \subseteq \mathcal{D}$
12: Update ξ from $\nabla_\xi \mathcal{L}(\xi)$ using any optimizer, e.g., ADAM (Kingma and Ba, 2014)
13: **end while**
14: **until** forever

Discussion. The key property of our approach is the weight resampling at line 7, which resembles *Thompson sampling* (Thompson, 1933) and closely relates to the recent value function randomization (Osband et al., 2014; Azizzadenesheli et al., 2018; Osband et al., 2019). At each step we guess what is the task we are trying to solve based on our current belief and we act as if such guess were true. This mechanism allows an efficient adaptive exploration of the target task. Intuitively, during the first steps of interaction, the agent is very uncertain about the current task, and such uncertainty induces stochasticity in the chosen actions, allowing a rather informed exploration to take place. Consider, for instance, that actions that are bad on average for all tasks are improbable to be sampled, while this cannot happen in uninformed exploration strategies, like ϵ-greedy, before learning takes place. As the learning process goes on, the agent quickly figures out which task it is solving, thus moving all the probability mass over the weights minimizing the TD error. From that point, sampling from the posterior is approximately equivalent to deterministically taking such weights, and no more exploration is performed. Finally, notice the *generality* of the proposed approach: as far as the objective $\mathcal{L}$ is differentiable in the variational parameters ξ, and its gradients can be efficiently computed, any approximator for the Q-function and any prior/posterior distributions can be adopted. For the latter, we describe two practical choices in the next two sections.

We note that our approach relates to recent algorithms for meta-learning or fast adaptation of weights in neural networks (Finn et al., 2017; Grant et al., 2018; Amit and Meir, 2018). Such approaches typically assume to have full access to the task distribution $\mathcal{D}$ (i.e., samples from $\mathcal{D}$ can be obtained on-demand) and build meta-models that quickly adapt to new tasks drawn from the same distribution. On the other hand, we assume only a fixed and limited set of source tasks, together with their approximate solutions, is available. Then, our goal is to speed-up the learning process of a new target task from $\mathcal{D}$ by transferring only these data, without requiring additional source tasks or experience samples from them.

Gaussian Variational Transfer

The most immediate, and widely-adopted, choice for the prior and posterior families is to consider Gaussian distributions. As we shall see, this choice makes our algorithm very efficient and easy to implement. We model the prior as a multivariate Gaussian $p(\omega) = \mathcal{N}(\mu_p, \Sigma_p)$ and we learn its parameters from the set of source weights using maximum likelihood estimation (with a small regularization to make sure the covariance is positive definite). Then, our variational family is the set of all well-defined Gaussian distributions, i.e., the variational parameters are $\Xi = \big\{(\mu, \Sigma) \mid \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d\times d}, \Sigma \succ 0\big\}$. To prevent the covariance from becoming not positive definite, we consider its Cholesky decomposition $\Sigma = LL^T$ and we learn the lower-triangular Cholesky factor L instead. In this case, deriving the gradient of the variational objective $\mathcal{L}$ is very simple. The KL divergence between the prior and approximate posterior can be computed in closed-form as

$$D_{\mathrm{KL}}\left(q_\xi \| p\right) = \frac{1}{2}\left(\log\frac{|\Sigma_p|}{|\Sigma|} + \mathrm{Tr}\left(\Sigma_p^{-1}\Sigma\right) + (\mu-\mu_p)^T\Sigma_p^{-1}(\mu-\mu_p) - d\right),$$

for $\xi = (\mu, L)$ and $\Sigma = LL^T$. Its gradients with respect to the variational parameters are

$$\nabla_\mu D_{\mathrm{KL}}\left(q_\xi \| p\right) = \Sigma_p^{-1}(\mu-\mu_p), \quad \nabla_L D_{\mathrm{KL}}\left(q_\xi \| p\right) = \Sigma_p^{-1}L - (L^{-1})^T$$

Finally, the gradients w.r.t. the expected likelihood term of the variational objective (7.2) can be computed using the reparameterization trick (e.g., Hoffman et al. (2013); Rezende et al. (2014)) as

$$\nabla_\mu \mathbb{E}_{\omega\sim\mathcal{N}(\mu,LL^T)}\left[||B_\omega||^2_{\mathcal{D}}\right] = \mathbb{E}_{v\sim\mathcal{N}(0,I)}\left[\nabla_\omega||B_\omega||^2_{\mathcal{D}}\right] \text{ for } \omega = Lv + \mu$$

$$\nabla_L \mathbb{E}_{\omega\sim\mathcal{N}(\mu,LL^T)}\left[||B_\omega||^2_{\mathcal{D}}\right] = \mathbb{E}_{v\sim\mathcal{N}(0,I)}\left[\nabla_\omega||B_\omega||^2_{\mathcal{D}}\cdot v^T\right] \text{ for } \omega = Lv + \mu$$

Mixture of Gaussian Variational Transfer

Although the Gaussian model is very appealing as it allows for a simple and efficient way of computing the variational objective and its gradients, in practice it rarely allows us to describe the prior distribution accurately. In fact, even for families of tasks in which the reward and transition models are Gaussian, the optimal Q-values might be far from being normally distributed. Depending on the family of tasks under consideration and, since we are learning a distribution over weights, on the chosen function approximator, the prior might have arbitrarily complex shapes. When the information loss due to the Gaussian approximation becomes too severe, the algorithm is likely to fail at capturing any similarities between the tasks. We now propose a variant to successfully solve this problem, while keeping the algorithm efficient and simple enough to be applied in practice.

Given the source parameters $\mathcal{W}_{\mathrm{src}} = \{\omega_1, \ldots, \omega_m\}$, we model our estimated prior as a mixture with equally-weighted isotropic Gaussians centered at each weight, $p(\omega) = \frac{1}{m}\sum_{j=1}^m \mathcal{N}(\omega|\omega_j, \sigma_p^2 I)$. This model resembles a kernel density estimator Scott (2015) with bandwidth σ_p^2 and, due to its nonparametric nature, it allows capturing arbitrarily complex distributions. Consistently with the prior, we model our approximate posterior as a mixture of Gaussians. Using c mixture components, our posterior is $q_\xi(\omega) =$

$\frac{1}{c}\sum_{i=1}^{c}\mathcal{N}(\omega|\mu_i,\Sigma_i)$, with variational parameters $\xi = (\mu_1,\dots,\mu_c,\Sigma_1,\dots,\Sigma_c)$. Once again, we learn Cholesky factors instead of full covariances. Finally, since the KL divergence between two mixtures of Gaussians has no closed-form expression, we rely on an upper bound to such quantity, so that the negative ELBO still upper bounds the KL between the approximate and the exact posterior. Among the many upper bounds available, we adopt the one derived by Hershey and Olsen (2007). For the sake of completeness, we state this result, instantiated for our specific setting, in the next theorem.

Theorem 7.3.1 (Hershey and Olsen (2007))**.** *Let* $p = \frac{1}{m}\sum_{j=1}^{m}\mathcal{N}(\omega_j,\sigma_p^2 I)$ *and* $q_\xi = \frac{1}{c}\sum_{i=1}^{c}\mathcal{N}(\mu_i,\Sigma_i)$ *be the prior and posterior distributions, respectively. Then, for any* $\alpha\in\mathbb{R}^{c\times m}$ *and* $\beta\in\mathbb{R}^{m\times c}$ *such that* $\sum_{j=1}^{m}\beta_{j,i}=\frac{1}{c}$ *and* $\sum_{i=1}^{c}\alpha_{i,j}=\frac{1}{m}$*,*

$$D_{\mathrm{KL}}\left(q_\xi\|p\right) \le D_{\mathrm{KL}}(\beta\|\alpha) + \sum_{i=1}^{c}\sum_{j=1}^{m}\beta_{j,i}D_{\mathrm{KL}}(\mathcal{N}(\mu_i,\Sigma_i)\|\mathcal{N}(\omega_j,\sigma_p^2 I)).$$

As shown by Hershey and Olsen (2007), a simple fixed-point procedure can be adopted to find α,β that minimize this upper bound:

$$\alpha_{i,j} = \frac{\beta_{j,i}}{m\sum_{l=1}^{c}\beta_{j,l}}, \quad \beta_{j,i} = \frac{\alpha_{i,j}e^{-D_{\mathrm{KL}}(\mathcal{N}(\mu_i,\Sigma_i)\|\mathcal{N}(\omega_j,\sigma_p^2 I))}}{c\sum_{l=1}^{m}\alpha_{i,l}e^{-D_{\mathrm{KL}}(\mathcal{N}(\mu_i,\Sigma_i)\|\mathcal{N}(\omega_l,\sigma_p^2 I))}}.$$

Using Theorem 7.3.1, we can directly derive an upper bound to the negative ELBO of Equation 7.2,

$$\begin{aligned}\mathcal{L}(\xi) \le \widetilde{\mathcal{L}}(\xi) := \frac{1}{c}\sum_{i=1}^{c}\mathbb{E}_{\omega\sim\mathcal{N}(\omega|\mu_i,\Sigma_i)}\left[\|B_\omega\|_{\mathcal{D}}^2\right] + \frac{\lambda}{n}D_{\mathrm{KL}}(\beta\|\alpha) \\ + \frac{\lambda}{n}\sum_{i=1}^{c}\sum_{j=1}^{m}\beta_{j,i}D_{\mathrm{KL}}(\mathcal{N}(\mu_i,\Sigma_i)\|\mathcal{N}(\omega_j,\sigma_p^2 I)).\end{aligned} \tag{7.3}$$

In practice, we optimize the upper bound $\widetilde{\mathcal{L}}(\xi)$ instead of $\mathcal{L}$. Note that $\widetilde{\mathcal{L}}(\xi)$ can be easily differentiated with respect to $\xi = (\mu_1,...,\mu_C,\Sigma_1,...,\Sigma_C)$ as we have seen for the Gaussian case.

Minimizing the TD Error

From Sections 7.3.1 and 7.3.2, we know that differentiating the negative ELBO $\mathcal{L}$ with respect to ξ requires differentiating $\|B_\omega\|_{\mathcal{D}}^2$ with respect to ω. Unfortunately, the TD error is well-known to be non-differentiable due to the presence of the max operator. This issue is rarely a problem since typical value-based algorithms are semi-gradient methods, i.e., they do not differentiate the targets (see, e.g., Chapter 11 of Sutton and Barto (2018)). However, our transfer settings are quite different from common RL. In fact, our algorithm is likely to start from Q-functions that are very close to an optimum and aims only to adapt the weights in some direction of lower error so as to quickly converge to the solution of the target task. Unfortunately, this property does not hold for most semi-gradient algorithms. Even worse, many online RL algorithms combined with complex function approximators (e.g., DQNs) are well-known to be unstable, especially when approaching an optimum,

and require many tricks and tuning to work well (Schaul et al., 2015b; Van Hasselt et al., 2016). This property is clearly undesirable in our case, as we only aim at adapting already good solutions. Thus, we consider using a residual gradient algorithm Baird (1995). To differentiate the targets, we replace the optimal Bellman operator with the mellow Bellman operator introduced in Asadi and Littman (2017), which adopts a softened version of max called *mellowmax*:

$$\operatorname*{mm}_a Q_\omega(s,a) := \frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_a e^{\kappa Q_\omega(s,a)} \tag{7.4}$$

where κ is a hyperparameter and $|\mathcal{A}|$ is the number of actions. The mellow Bellman operator, which we denote as $\widetilde{T}$, has several appealing properties: (i) it converges to the maximum as $\kappa \to \infty$, (ii) it has a unique fixed-point, and (iii) it is *differentiable*. Denoting by $\widetilde{B}_\omega = \widetilde{T}Q_\omega - Q_\omega$ the Bellman residual with respect to the mellow Bellman operator $\widetilde{T}$, we have that the corresponding TD error, $||\widetilde{B}_\omega||^2_{\mathcal{D}}$, is now differentiable with respect to ω.

Although residual algorithms have guaranteed convergence, they are typically much slower than their semi-gradient counterpart. Baird (1995) proposed to project the gradient in a direction that achieves higher learning speed, while preserving convergence. This projection is obtained by including a parameter $\psi \in [0,1]$ in the TD error gradient such that

$$\nabla_\omega \left\|\widetilde{B}_\omega\right\|^2_{\mathcal{D}} = \frac{2}{n}\sum_{i=1}^{n}\widetilde{b}_i(\omega)\Big(\gamma\psi\nabla_\omega \operatorname*{mm}_{a'} Q_\omega(S_i',a') - \nabla_\omega Q_\omega(S_i,A_i)\Big),$$

where $\widetilde{b}_i(\omega) = R_i + \gamma\,\mathrm{mm}_{a'}\,Q_\omega(S_i',a') - Q_\omega(S_i,A_i)$. Notice that ψ trades-off between the semi-gradient ($\psi = 0$) and the full residual gradient ($\psi = 1$). A good criterion for choosing such parameter is to start with values close to zero (to have faster learning) and move to higher values when approaching the optimum (to guarantee convergence). In our experiments, we shall use the gradient expression above to update parameters, while treating ψ as a hyper-parameter.

Theoretical Analysis

This section is devoted to the theoretical analysis of our variational value transfer method. We first analyze the properties of the mellow Bellman operator by providing a result of independent interest. We then derive a finite-sample analysis of our algorithm combined with Gaussian and mixture-of-Gaussian distributions. Our results confirm the theoretical superiority of the latter choice.

Analysis of the Mellowmax Operator

A first important question that we need to answer is whether replacing max with mellow-max in the Bellman operator constitutes a strong approximation or not. It has been proven (Asadi and Littman, 2017) that the mellow Bellman operator is a non-expansion under the L_∞-norm and, thus, has a unique fixed-point. However, how such fixed-point differs from the one of the optimal Bellman operator remains an open question. Since mellow-max monotonically converges to max as $\kappa \to \infty$, it would be desirable if the fixed point of the

corresponding operator also monotonically converged to the fixed point of the optimal one. We confirm that this property actually holds in the following theorem, which we believe is of independent interest.

Theorem 7.4.1. *Let $Q^\star$ be the fixed-point of the optimal Bellman operator T. Define the action-gap function $g(s)$ as the difference between the value of the best action and the second best action at each state s. Let $\widetilde{Q}$ be the fixed-point of the mellow Bellman operator $\widetilde{T}$ with parameter $\kappa > 0$ and denote by $\beta_\kappa > 0$ the inverse temperature of the induced Boltzmann distribution (as in Asadi and Littman (2017)). Then:*

$$\left\|Q^\star - \widetilde{Q}\right\|_\infty \leq \frac{2\gamma r_{\max}}{(1-\gamma)^2} \left\| \frac{1}{1 + \frac{1}{|\mathcal{A}|} e^{\beta_\kappa g}} \right\|_\infty . \tag{7.5}$$

Notice that $\widetilde{Q}$ converges to $Q^\star$ exponentially fast as κ (equivalently, β_κ) increases and the action gaps are all larger than zero.

Proof. We begin by noticing that:

$$\begin{aligned}\left\|Q^* - \widetilde{Q}\right\|_\infty &= \left\|TQ^* - \widetilde{T}\widetilde{Q}\right\|_\infty = \left\|TQ^* - \widetilde{T}Q^* + \widetilde{T}Q^* - \widetilde{T}\widetilde{Q}\right\|_\infty \\ &\leq \left\|TQ^* - \widetilde{T}Q^*\right\|_\infty + \left\|\widetilde{T}Q^* - \widetilde{T}\widetilde{Q}\right\|_\infty \leq \left\|TQ^* - \widetilde{T}Q^*\right\|_\infty + \gamma \left\|Q^* - \widetilde{Q}\right\|_\infty ,\end{aligned}$$

where the first inequality follows from Minkowsky's inequality and the second one from the contraction property of the mellow Bellman operator. This implies that:

$$\left\|Q^* - \widetilde{Q}\right\|_\infty \leq \frac{1}{1-\gamma} \left\|TQ^* - \widetilde{T}Q^*\right\|_\infty . \tag{7.6}$$

Let us bound the norm on the right-hand side separately. In order to do that, we will bound the function $\left|TQ^*(s,a) - \widetilde{T}Q^*(s,a)\right|$ point-wisely for any pair (s,a). By applying the definition of the optimal and mellow Bellman operators, we obtain:

$$\begin{aligned}\left|TQ^*(s,a) - \widetilde{T}Q^*(s,a)\right| &= \left|r(s,a) + \gamma \mathbb{E}\left[\max_{a'} Q^*(s',a')\right] - r(s,a) - \gamma \mathbb{E}\left[\operatorname*{mm}_{a'} Q^*(s',a')\right]\right| \\ = \gamma \left|\mathbb{E}\left[\max_{a'} Q^*(s',a')\right] - \mathbb{E}\left[\operatorname*{mm}_{a'} Q^*(s',a')\right]\right| &\leq \gamma \mathbb{E}\left[\left|\max_{a'} Q^*(s',a') - \operatorname*{mm}_{a'} Q^*(s',a')\right|\right] .\end{aligned} \tag{7.7}$$

Bounding this quantity reduces to bounding $|\max_a Q^*(s,a) - \operatorname{mm}_a Q^*(s,a)|$ point-wisely for any s. Recall that applying the mellow Bellman operator is equivalent to computing an expectation under a Boltzmann distribution with inverse temperature β_κ induced by κ (Asadi and Littman, 2017). Thus, we can write:

$$\begin{aligned}\left|\max_a Q^*(s,a) - \operatorname*{mm}_a Q^*(s,a)\right| &= \left|\sum_a Q^*(s,a)\left(\pi^*(a|s) - \pi_{\beta_\kappa}(a|s)\right)\right| \\ &\leq \sum_a |Q^*(s,a)|\,|\pi^*(a|s) - \pi_{\beta_\kappa}(a|s)| \\ &\leq \frac{r_{\max}}{1-\gamma} \sum_a |\pi^*(a|s) - \pi_{\beta_\kappa}(a|s)| ,\end{aligned} \tag{7.8}$$

where π^* is the optimal (deterministic) policy w.r.t. Q^* and π_{β_κ} is the Boltzmann distribution induced by Q^* with inverse temperature β_κ:

$$\pi_{\beta_\kappa}(a|s) = \frac{e^{\beta_\kappa Q^*(s,a)}}{\sum_{a'} e^{\beta_\kappa Q^*(s,a')}}.$$

Denote by $a_1(s)$ the optimal action for state s under Q^*. We can then write:

$$\begin{aligned}\sum_a |\pi^*(a|s) - \pi_{\beta_\kappa}(a|s)| &= |\pi^*(a_1(s)|s) - \pi_{\beta_\kappa}(a_1(s)|s)| + \sum_{a \neq a_1(s)} |\pi^*(a|s) - \pi_{\beta_\kappa}(a|s)| \\ &= |1 - \pi_{\beta_\kappa}(a_1(s)|s)| + \sum_{a \neq a_1(s)} |\pi_{\beta_\kappa}(a|s)| = 2\,|1 - \pi_{\beta_\kappa}(a_1(s)|s)|\,. \end{aligned} \tag{7.9}$$

Finally, denoting with $a_2(s)$ the second-best action in state s, let us bound this last term:

$$\begin{aligned}|1 - \pi_{\beta_\kappa}(a_1(s)|s)| &= \left|1 - \frac{e^{\beta_\kappa Q^*(s,a_1(s))}}{\sum_{a'} e^{\beta_\kappa Q^*(s,a')}}\right| = \left|1 - \frac{e^{\beta_\kappa \left(Q^*(s,a_1(s)) - Q^*(s,a_2(s))\right)}}{\sum_{a'} e^{\beta_\kappa (Q^*(s,a') - Q^*(s,a_2(s)))}}\right| \\ &= \left|1 - \frac{e^{\beta_\kappa g(s)}}{\sum_{a'} e^{\beta_\kappa (Q^*(s,a') - Q^*(s,a_2(s)))}}\right| \\ &= \left|1 - \frac{e^{\beta_\kappa g(s)}}{e^{\beta_\kappa g(s)} + \sum_{a' \neq a_1(s)} e^{\beta_\kappa (Q^*(s,a') - Q^*(s,a_2(s)))}}\right| \\ &\leq \left|1 - \frac{e^{\beta_\kappa g(s)}}{e^{\beta_\kappa g(s)} + |\mathcal{A}|}\right| = \left|\frac{1}{1 + \frac{1}{|\mathcal{A}|} e^{\beta_\kappa g(s)}}\right|. \end{aligned} \tag{7.10}$$

Combining Eq. (7.8), (7.9), and (7.10), we obtain:

$$\left|\max_a Q^*(s,a) - \operatorname*{mm}_a Q^*(s,a)\right| \leq \frac{2 r_{\max}}{1-\gamma} \left|\frac{1}{1 + \frac{1}{|\mathcal{A}|} e^{\beta_\kappa g(s)}}\right|.$$

Finally, using Eq. (7.7) we get:

$$\left|TQ^*(s,a) - \widetilde{T}Q^*(s,a)\right| \leq \frac{2\gamma r_{\max}}{1-\gamma} \mathbb{E}\left[\left|\frac{1}{1 + \frac{1}{|\mathcal{A}|} e^{\beta_\kappa g(s')}}\right|\right].$$

Taking the norm and plugging this into Eq. (7.6) concludes the proof. □

Finite-Sample Analysis

The second question that we need to answer is whether we can provide any guarantee on our algorithm's performance when given limited data. To address this point, we consider the two variants of Algorithm 5 from Section 7.3.1 and 7.3.2 with linear approximators. Specifically, we consider the family of linearly parameterized value functions $Q_\omega(s,a) = \omega^T \phi(s,a)$ with bounded weights $\|\omega\|_2 \leq \omega_{\max}$ and uniformly bounded feature functions $\|\phi(s,a)\|_2 \leq \phi_{\max}$. We assume only a finite dataset $\mathcal{D} = \{(S_i, A_i, S_i', R_i)\}_{i=1}^n$ is available and provide a finite-sample analysis bounding the expected (mellow) Bellman error under the variational distribution minimizing the objective (7.2) for any fixed target task $\mathcal{M}_0$. To simplify the analysis, we assume that state-action pairs (S_i, A_i) in $\mathcal{D}$ are i.i.d. from some distribution ν. While this is not really the case for the online setting where Algorithm 5 is applied, it does not contradict the main focus of the analysis, i.e., to show that

the learned posterior distribution concentrate on Q-functions with low Bellman error as more samples are observed. Unfortunately, an analysis of the online exploration behavior of our algorithm (e.g., regret or sample complexity bounds) is considerably more difficult in this context with continuous state spaces and complex function approximators.

Theorem 7.4.2. *Let $\widehat{\xi}$ be the variational parameters minimizing the objective* (7.2) *on a dataset $\mathcal{D}$ of n i.i.d. samples distributed according to $\mathcal{M}_0$ and ν. Let $\omega^\star = \operatorname{argmin}_\omega ||\widetilde{B}_\omega||_\nu^2$ and define $\upsilon(\omega^*) \triangleq \mathbb{E}_{\mathcal{N}(\omega^*, \frac{1}{n}I)}[v(\omega)]$, with $v(\omega) := \mathbb{E}_\nu\left[\mathbb{V}\mathrm{ar}_{P_0}\left[\widetilde{b}(\omega)\right]\right]$. Then, there exist constants c_1, c_2, c_3 such that, with probability at least $1-\delta$ over the choice of the dataset $\mathcal{D}$,*

$$\mathbb{E}_{q_{\widehat{\xi}}}\left[\left\|\widetilde{B}_\omega\right\|_\nu^2\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + \upsilon(\omega^*) + c_1\sqrt{\frac{\log\frac{2}{\delta}}{n}} + \frac{c_2 + \lambda d\log n + \lambda\varphi\left(\mathcal{W}_{\mathrm{src}}\right)}{n} + \frac{c_3}{n^2},$$

where $\varphi(\mathcal{W}_{\mathrm{src}}) = \|\omega^* - \mu_p\|_{\Sigma_p^{-1}}^2$ when the Gaussian version of Algorithm 5 is used with prior $p(\omega) = \mathcal{N}(\mu_p, \Sigma_p)$ estimated from $\mathcal{W}_{\mathrm{src}}$, while

$$\varphi(\mathcal{W}_{\mathrm{src}}) = \frac{1}{\sigma_p^2}\sum_{\omega\in\mathcal{W}_{\mathrm{src}}} \frac{e^{-\frac{1}{2\sigma_p^2}\|\omega^*-\omega\|_2^2}}{\sum_{\omega'\in\mathcal{W}_{\mathrm{src}}} e^{-\frac{1}{2\sigma_p^2}\|\omega^*-\omega'\|_2^2}}\|\omega^* - \omega\|_2^2 \tag{7.11}$$

when the mixture version of Algorithm 5 is used with c components and bandwidth σ_p^2 for the prior.

Let us comment on this result before carrying out the proof. Four main terms constitute our bound: the approximation error due to the limited hypothesis space (first term), the variance (second and third terms), the distance to the prior (fourth term), and a constant term decaying as $\mathcal{O}(n^2)$. As we might have expected, the only difference between the bounds for the two versions of Algorithm 5 is in the term $\varphi(\mathcal{W}_{\mathrm{src}})$, i.e., the distance between the optimal weights ω^* and the source weights $\mathcal{W}_{\mathrm{src}}$. Specifically, for the mixture version we have the *softmin* (i.e., a smoothened minimum) distance to the source tasks' weights (Equation (7.11)), while for the Gaussian one we have the distance to the mean of such weights. This property shows a clear advantage of using the mixture version rather than the Gaussian one: in order to tighten the bound, it is enough to have at least one source task that is close to the optimal solution of the target task. In fact, the Gaussian version requires the source tasks to be, on average, similar to the target task in order to perform well, while the mixture version only requires this property for one of them. In both cases, when the term $\varphi(\mathcal{W}_{\mathrm{src}})$ is reduced, the dominating error is due to the variance of the estimates, and, thus, the algorithm is expected to achieve good performance rather quickly, as new data is collected. Furthermore, as $n \to \infty$ the only error terms remaining are the irreducible approximation error due to the limited functional space and the variance term $\upsilon(\omega^*)$. The latter is due to the fact that we minimize a biased estimate of the Bellman error and can be removed in cases where double sampling of the next state is possible (e.g., in simulation).

Before proving Theorem 7.4.2, we state some important properties (proved in Appendix B) of the variational approximation introduced in Section 7.3. Our results generalize those of existing works that consider variational approximations of intractable Gibbs posteriors (Alquier et al., 2016).

Lemma 7.4.1. *Let p and q be arbitrary distributions over weights ω, and ν be a probability measure over $\mathcal{S} \times \mathcal{A}$. Consider a dataset $\mathcal{D}$ of n i.i.d. samples where state-action couples are distributed according to ν and define $v(\omega) := \mathbb{E}_\nu \left[\mathbb{V}\mathrm{ar}_{P_0} \left[\widetilde{b}(\omega) \right] \right]$. Then, for any $\lambda > 0$ and $\delta > 0$, with probability at least $1 - \delta$, the following two inequalities hold simultaneously:*

$$\mathbb{E}_q \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] \leq \mathbb{E}_q \left[\left\| \widetilde{B}_\omega \right\|_\mathcal{D}^2 \right] - \mathbb{E}_q \left[v(\omega) \right] + \frac{\lambda}{n} D_{\mathrm{KL}}(q \| p) + \frac{4 r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}},$$

$$\mathbb{E}_q \left[\left\| \widetilde{B}_\omega \right\|_\mathcal{D}^2 \right] \leq \mathbb{E}_q \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] + \mathbb{E}_q \left[v(\omega) \right] + \frac{\lambda}{n} D_{\mathrm{KL}}(q \| p) + \frac{4 r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

From Lemma 7.4.1 we can straightforwardly prove the following result which will be of fundamental importance in the remaining.

Lemma 7.4.2. *Consider the same assumptions as in Lemma 7.4.1 and denote by $\widehat{\xi}$ the minimizer of* (7.2) *using dataset $\mathcal{D}$. Then,*

$$\mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] \leq \inf_{\xi \in \Xi} \left\{ \mathbb{E}_{q_\xi} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] + \mathbb{E}_{q_\xi} \left[v(\omega) \right] + 2 \frac{\lambda}{n} D_{\mathrm{KL}}(q_\xi \| p) \right\} + \frac{8 r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

It is worth noting the generality of Lemma 7.4.2: in bounding the expected Bellman error we do not need to assume any particular distribution, nor do we have to assume any particular function approximator. We are now ready to prove our main result.

Proof of Theorem 7.4.2. The proof is divided in two parts; the first one deals with the Gaussian case (Section 7.3.1), while the second one deals with mixture-of-Gaussian distributions (Section 7.3.2).

Gaussian distributions. Using Lemma 7.4.2 with variational parameters $\widehat{\xi} = (\widehat{\mu}, \widehat{\Sigma})$, we have

$$\begin{aligned} \mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] &\leq \inf_{\xi \in \Xi} \left\{ \mathbb{E}_{q_\xi} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] + \mathbb{E}_{q_\xi} \left[v(\omega) \right] + 2 \frac{\lambda}{n} KL(q_\xi || p) \right\} + 8 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \\ &\leq \mathbb{E}_{\mathcal{N}(\omega^*, kI)} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] + \mathbb{E}_{\mathcal{N}(\omega^*, kI)} \left[v(\omega) \right] + 2 \frac{\lambda}{n} D_{\mathrm{KL}} \left(\mathcal{N}(\omega^*, kI) \| p \right) + 8 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}, \end{aligned} \tag{7.12}$$

where the second inequality is due to the fact that, since Lemma 7.4.2 contains an infimum over the variational parameters, we can upper bound its right-hand side by choosing any specific ξ from Ξ. Here, we choose $\mu = \omega^*$ and $\Sigma = kI$, for some positive constant $k > 0$. Let us now bound these terms separately.

(1) *Bounding the expected Bellman error.* We have

$$\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\left\|\widetilde{B}_\omega\right\|_\nu^2\right] = \mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\mathbb{E}_\nu\left[(\widetilde{T}Q_\omega - Q_\omega)^2\right]\right] = \mathbb{E}_\nu\left[\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[(\widetilde{T}Q_\omega - Q_\omega)^2\right]\right]$$
$$= \mathbb{E}_\nu\left[\mathbb{E}^2_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right]\right] + \mathbb{E}_\nu\left[\mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right]\right]. \tag{7.13}$$

Let us bound these two terms point-wisely for each pair $\langle s, a\rangle$. For the first expectation, we have

$$\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right] = \mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[r_0(s,a) + \gamma\mathbb{E}_{s'}\left[\min_{a'}\omega^T\phi(s',a')\right] - \omega^T\phi(s,a)\right]$$
$$= r_0(s,a) + \gamma\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\mathbb{E}_{s'}\left[\min_{a'}\omega^T\phi(s',a')\right]\right] - \omega^{*T}\phi(s,a). \tag{7.14}$$

To bound the second term, we adopt Jensen's inequality:

$$\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\mathbb{E}_{s'}\left[\min_{a'}\omega^T\phi(s',a')\right]\right] = \mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}e^{\kappa\omega^T\phi(s',a')}\right]\right]$$
$$\leq \mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[e^{\kappa\omega^T\phi(s',a')}\right]\right]. \tag{7.15}$$

Now, since we know that $\omega^T\phi(s',a') \sim \mathcal{N}(\omega^{*T}\phi(s',a'), k\,\phi(s',a')^T\phi(s',a'))$, $e^{\kappa\omega^T\phi(s',a')}$ follows a log-normal law with mean $e^{\kappa\omega^{*T}\phi(s',a') + \frac{1}{2}\kappa^2 k\phi(s',a')^T\phi(s',a')}$. Thus,

$$\mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[e^{\kappa\omega^T\phi(s',a')}\right]\right]$$
$$= \mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}e^{\kappa\omega^{*T}\phi(s',a') + \frac{1}{2}\kappa^2 k\phi(s',a')^T\phi(s',a')}\right]$$
$$= \mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}e^{\kappa\omega^{*T}\phi(s',a')}e^{\frac{1}{2}\kappa^2 k\phi^2_{\max}}\right]$$
$$= \mathbb{E}_{s'}\left[\frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_{a'}e^{\kappa\omega^{*T}\phi(s',a')}\right] + \frac{1}{2}\kappa k\phi^2_{\max} = \mathbb{E}_{s'}\left[\min_{a'}\omega^{*T}\phi(s',a')\right] + \frac{1}{2}\kappa k\phi^2_{\max}.$$

Plugging this into (7.15) and then into (7.14), we obtain

$$\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right] \leq r_0(s,a) + \gamma\mathbb{E}_{s'}\left[\min_{a'}\omega^{*T}\phi(s',a')\right] + \frac{1}{2}\gamma\kappa k\phi^2_{\max} - \omega^{*T}\phi(s,a)$$
$$= \widetilde{B}_{\omega^*} + \frac{1}{2}\gamma\kappa k\phi^2_{\max}.$$

This implies that

$$\mathbb{E}^2_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right] \leq \left(\widetilde{B}_{\omega^*} + \frac{1}{2}\gamma\kappa k\phi^2_{\max}\right)^2 \leq 2\widetilde{B}^2_{\omega^*} + \frac{1}{2}\gamma^2\kappa^2k^2\phi^4_{\max},$$

where the second inequality follows from Cauchy-Schwarz inequality. Going back to (7.13), the first term can now be upper bounded by:

$$\mathbb{E}_\nu\left[\mathbb{E}^2_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right]\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + \frac{1}{2}\gamma^2\kappa^2k^2\phi^4_{\max}.$$

Let us now consider the variance term of (7.13) and derive a bound that holds point-wisely for any s, a. We have:

$$\begin{aligned}
\mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right] &= \mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[r_0(s,a) + \gamma\mathbb{E}_{s'}\left[\min_{a'}\omega^T\phi(s',a')\right] - \omega^T\phi(s,a)\right] \\
&= \mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[\gamma\mathbb{E}_{s'}\left[\min_{a'}\omega^T\phi(s',a') - \frac{1}{\gamma}\omega^T\phi(s,a)\right]\right] \\
&= \mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[\gamma\mathbb{E}_{s'}\left[\min_{a'}\omega^T\left(\phi(s',a') - \frac{1}{\gamma}\phi(s,a)\right)\right]\right] \\
&= \gamma^2\mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,I)}\left[\mathbb{E}_{s'}\left[\min_{a'}\sqrt{k}\,\omega^T\left(\phi(s',a') - \frac{1}{\gamma}\phi(s,a)\right)\right]\right].
\end{aligned}$$

From Cauchy-Schwarz inequality:

$$\sqrt{k}\left|\omega^T\left(\phi(s',a') - \frac{1}{\gamma}\phi(s,a)\right)\right| \leq \sqrt{k}\,\|\omega\|\left\|\phi(s',a') - \frac{1}{\gamma}\phi(s,a)\right\| \leq \sqrt{k}\,\omega_{\max}\phi_{\max}\frac{1+\gamma}{\gamma}.$$

Then, the variance can be straightforwardly bounded using Popoviciu's inequality as

$$\mathbb{V}\mathrm{ar}_{\mathcal{N}(\omega^*,kI)}\left[\widetilde{T}Q_\omega - Q_\omega\right] \leq \gamma^2\frac{1}{4}\left(2\sqrt{k}\,\omega_{\max}\phi_{\max}\frac{1+\gamma}{\gamma}\right)^2 = k\left(\omega_{\max}\phi_{\max}(1+\gamma)\right)^2.$$

We can finally plug everything into (7.13), thus obtaining

$$\mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + \frac{1}{2}\gamma^2\kappa^2k^2\phi_{\max}^4 + k\left(\omega_{\max}\phi_{\max}(1+\gamma)\right)^2.$$

(2) *Bounding the KL divergence.* We have

$$\begin{aligned}
D_{\mathrm{KL}}\left(\mathcal{N}(\omega^*,kI)\|p\right) &= D_{\mathrm{KL}}\left(\mathcal{N}(\omega^*,kI)\|\mathcal{N}(\mu_p,\Sigma_p)\right) \\
&= \frac{1}{2}\left(\log\frac{|\Sigma_p|}{k^d} + k\mathrm{Tr}\left(\Sigma_p^{-1}\right) + \|\omega^* - \mu_p\|_{\Sigma_p^{-1}}^2 - d\right) \\
&\leq \frac{1}{2}d\log\frac{\sigma_{\max}}{k} + \frac{1}{2}d\frac{k}{\sigma_{\min}} + \frac{1}{2}\|\omega^* - \mu_p\|_{\Sigma_p^{-1}}^2.
\end{aligned}$$

Now, putting all together into (7.12):

$$\begin{aligned}
\mathbb{E}_{q_{\widehat{\xi}}}\left[\left\|\widetilde{B}_\omega\right\|_\nu^2\right] \leq\ & 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + \frac{1}{2}\gamma^2\kappa^2k^2\phi_{\max}^4 + k\left(\omega_{\max}\phi_{\max}(1+\gamma)\right)^2 \\
&+ \mathbb{E}_{\mathcal{N}(\omega^*,kI)}\left[v(\omega)\right] + \frac{\lambda}{n}d\log\frac{\sigma_{\max}}{k} + \frac{\lambda}{n}d\frac{k}{\sigma_{\min}} \\
&+ \frac{\lambda}{n}\|\omega^* - \mu_p\|_{\Sigma_p^{-1}}^2 + 8\frac{r_{\max}^2}{(1-\gamma)^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}.
\end{aligned}$$

Since the bound holds for any $k > 0$, we can set $k := 1/n$, thus obtaining

$$\begin{aligned}
&\mathbb{E}_{q_{\widehat{\xi}}}\left[\left\|\widetilde{B}_\omega\right\|_\nu^2\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + v(\omega^*) + \frac{1}{n^2}\left(\frac{1}{2}\gamma^2\kappa^2\phi_{\max}^4 + \frac{\lambda d}{\sigma_{\min}}\right) \\
&+ \frac{1}{n}\left(\omega_{\max}^2\phi_{\max}^2(1+\gamma)^2 + \lambda d(\log\sigma_{\max} + \log n) + \lambda\|\omega^* - \mu_p\|_{\Sigma_p^{-1}}^2\right) + 8\frac{r_{\max}^2}{(1-\gamma)^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}
\end{aligned}$$

Finally, defining the constants $c_1 = \frac{8r_{\max}^2}{\sqrt{2}(1-\gamma)^2}$, $c_2 = \omega_{\max}^2\phi_{\max}^2(1+\gamma)^2 + \lambda d \log \sigma_{\max}$, and $c_3 = \frac{1}{2}\gamma^2\kappa^2\phi_{\max}^4 + \frac{\lambda d}{\sigma_{\min}}$, we obtain

$$\mathbb{E}_{q_{\widehat{\xi}}}\left[\left\|\widetilde{B}_{\omega}\right\|_{\nu}^2\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_{\nu}^2 + \upsilon(\omega^*) + c_1\sqrt{\frac{\log\frac{2}{\delta}}{n}} + \frac{c_2 + \lambda d\log n + \lambda\left\|\omega^* - \mu_p\right\|_{\Sigma_p^{-1}}^2}{n} + \frac{c_3}{n^2}.$$

This concludes the proof of the theorem for the Gaussian case.

Mixture-of-Gaussian distributions. Here we apply Lemma 7.4.2 with variational parameters $\widehat{\xi} = (\widehat{\mu}_1, \dots, \widehat{\mu}_c, \widehat{\Sigma}_1, \dots, \widehat{\Sigma}_c)$, while upper bounding the minimum empirical loss (i.e., the ELBO) by choosing $\mu_i = \omega^*$ and $\Sigma_i = kI$ for all $i = 1, \dots, c$. These choices yield exactly the bound (7.12) obtained for the Gaussian case, with the only difference that the prior p in $D_{\mathrm{KL}}(\mathcal{N}(\omega^*, kI)\|p)$ is now a mixture of Gaussians. Thus, we only need to upper bound this term in order to carry out the proof, while the other ones can be treated exactly as in the Gaussian case. Using Theorem 7.3.1,

$$D_{\mathrm{KL}}\left(\mathcal{N}(\omega^*, kI)\|p\right) \leq D_{\mathrm{KL}}(\beta\|\alpha) + \sum_{j=1}^{m}\beta_j D_{\mathrm{KL}}(\mathcal{N}(\omega^*, kI)\|\mathcal{N}(\omega_j, \sigma_p^2 I)), \tag{7.16}$$

where $\alpha, \beta \in \mathbb{R}^m$ are the "optimal" vectors defined in Section 7.3.2 (with the second index neglected since the posterior has now a single component). By instantiating the fixed-point procedure to compute the optimal vectors, we obtain

$$\alpha_j = \frac{1}{m}, \quad \beta_j = \frac{e^{-D_{\mathrm{KL}}(\mathcal{N}(\omega^*, kI)\|\mathcal{N}(\omega_j, \sigma_p^2 I))}}{\sum_{l=1}^{m} e^{-D_{\mathrm{KL}}(\mathcal{N}(\omega^*, kI)\|\mathcal{N}(\omega_l, \sigma_p^2 I))}}.$$

Using the closed-form of the KL divergence between two Gaussian distributions,

$$\beta_j = \frac{e^{-\frac{1}{2\sigma_p^2}\left\|\omega^* - \omega_j\right\|_2^2}}{\sum_{l=1}^{m} e^{-\frac{1}{2\sigma_p^2}\left\|\omega^* - \omega_l\right\|_2^2}} \quad \forall j = 1, \dots, m.$$

Let us bound the two terms of (7.16) separately. For the first one, we have

$$D_{\mathrm{KL}}(\beta\|\alpha) = \sum_{j=1}^{m}\beta_j \log\frac{\beta_j}{\alpha_j} = \sum_{j=1}^{m}\beta_j\log\beta_j - \sum_{j=1}^{m}\beta_j\log\frac{1}{m} \leq \log(m),$$

where the inequality holds since the first term is negative. For the second term of (7.16),

$$\begin{aligned}\sum_{j=1}^{m}\beta_j D_{\mathrm{KL}}(\mathcal{N}(\omega^*, kI)\|\mathcal{N}(\omega_j, \sigma_p^2 I)) &= \frac{1}{2}\sum_{j=1}^{m}\beta_j\left(d\log\frac{\sigma_p^2}{k} + d\frac{k}{\sigma_p^2} + \frac{1}{\sigma_p^2}\left\|\omega^* - \omega_j\right\|_2^2 - d\right) \\ &\leq \frac{1}{2}d\log\frac{\sigma_p^2}{k} + \frac{1}{2}d\frac{k}{\sigma_p^2} + \sum_{j=1}^{m}\beta_j\frac{1}{2\sigma_p^2}\left\|\omega^* - \omega_j\right\|_2^2 = \frac{1}{2}d\log\frac{\sigma_p^2}{k} + \frac{1}{2}d\frac{k}{\sigma_p^2} + \frac{1}{2}\varphi\left(\mathcal{W}_{\mathrm{src}}\right),\end{aligned}$$

where the last equality holds from the definition of β_j and φ in (7.11). we defined the vector Δ whose components are $\Delta_j = \frac{1}{2\sigma_p^2}\left\|\omega^* - \omega_j\right\|$. Putting the two terms together,

$$D_{\mathrm{KL}}\left(\mathcal{N}(\omega^*, kI)\|p\right) \leq \log(m) + \frac{1}{2}d\log\frac{\sigma_p^2}{k} + \frac{1}{2}d\frac{k}{\sigma_p^2} + \frac{1}{2}\varphi\left(\mathcal{W}_{\mathrm{src}}\right).$$

Notice that, from now on, one can simply apply the same steps as in the Gaussian case with $\sigma_{\max} = \sigma_{\min} = \sigma_p^2$ and $\frac{1}{2}\left\|\omega^* - \mu_p\right\|_{\Sigma_p^{-1}}^2$ replaced by $\frac{1}{2}\varphi\left(\mathcal{W}_{\mathrm{src}}\right)$. Thus, by redefining the three constants to

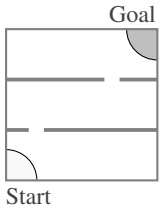

Figure 7.1: *The rooms problem. The agent starts from the bottom left corner and has to reach the top right corner, while passing through two doors whose positions change horizontally across tasks.*

$c_1 = \frac{8r_{\max}^2}{\sqrt{2}(1-\gamma)^2}$, $c_2 = \omega_{\max}^2\phi_{\max}^2(1+\gamma)^2 + \lambda d \log \sigma_p^2 + 2\lambda \log(m)$, and $c_3 = \frac{1}{2}\gamma^2\kappa^2\phi_{\max}^4 + \frac{\lambda d}{\sigma_p^2}$, we can write that, with probability at least $1-\delta$,

$$\mathbb{E}_{q_{\widehat{\xi}}}\left[\left\|\widetilde{B}_\omega\right\|_\nu^2\right] \leq 2\left\|\widetilde{B}_{\omega^*}\right\|_\nu^2 + \upsilon(\omega^*) + c_1\sqrt{\frac{\log\frac{2}{\delta}}{n}} + \frac{c_2 + \lambda d \log n + \lambda\varphi\left(\mathcal{W}_{\text{src}}\right)}{n} + \frac{c_3}{n^2},$$

which concludes the proof. □

Experiments

We experimentally evaluate our approach in four different domains with increasing level of difficulty. In all experiments, we compare our Gaussian variational transfer algorithm (GVT) and the version using a c-component mixture of Gaussians (c-MGVT) to plain no-transfer RL (NT) with ϵ-greedy exploration and to a simple transfer baseline in which we randomly pick one source Q-function and fine-tune from its weights (FT). Each curve in our plots is the result of 20 independent runs, each re-sampling the target and source tasks, with 95% confidence intervals. Additional details on the parameters adopted can be found in Tirinzoni et al. (2018a).

The Rooms Problem

We consider an agent navigating in the environment depicted in Figure 7.1. The agent starts in the bottom-left corner and must move from one room to another to reach the goal position in the top-right corner. The rooms are connected by small doors whose locations are unknown to the agent. The state-space is modeled as a 10×10 continuous grid, while the action-space is the set of 4 movement directions (up, right, down, left). After each action, the agent moves by 1 in the chosen direction and the final position is corrupted by Gaussian noise $\mathcal{N}(0, 0.2)$. In case the agent hits a wall, its position remains unchanged. The reward is 1 when reaching the goal (after which the process terminates) and 0 otherwise, while the discount factor is $\gamma = 0.99$. In this experiment, we consider linearly parameterized Q-functions with 121 equally-spaced radial basis features. We generate a set of 50 source tasks for the three-room environment of Figure 7.1 by sampling both door locations uniformly in the allowed space, and solve all of them by directly minimizing the TD error as presented in Section 7.3.3. Then, we use our algorithms to transfer from 10 source tasks sampled from the previously generated set. Since the agent does not know

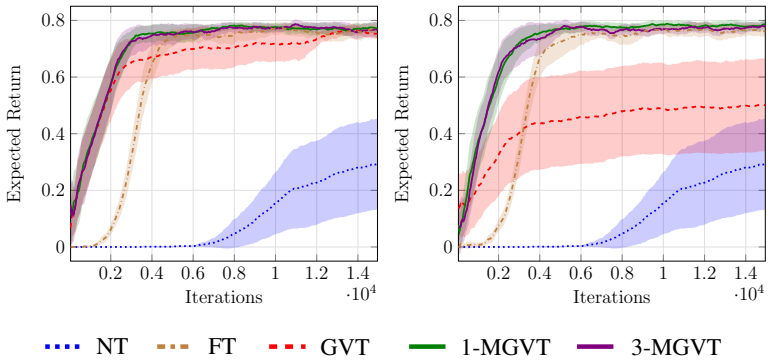

Figure 7.2: *Transfer in the rooms problem. (left) Both doors are uniformly sampled on the horizontal axis. (right) The bottom door is kept fixed in the center in the source tasks and allowed to vary in the targe task.*

the locations of the doors in advance and receives only very sparse feedback, it must efficiently explore the environment to figure out (i) their positions, and (ii) how to reach the goal. While this might be a complicated problem for plain RL, our transfer algorithm should be able to figure out the door positions quickly. In fact, notice that, although different, the optimal Q-functions for all tasks share some similarities. For example, once the agent has passed the last door before the goal, the Q-values are exactly the same in all tasks. The same does not hold for positions nearby the starting state. However, it is clear that there should be a preference over actions *up* and *right*, rather than *down* and *left* (which are worse in all tasks).

Results. The average return over the last 50 learning episodes as a function of the number of iterations is shown in Figure 7.2(left). As expected, the no-transfer (NT) algorithm fails at learning the task in so few iterations due to the limited exploration provided by an ϵ-greedy policy. On the other hand, all our algorithms achieve a significant speed-up and converge to the optimal performance in few iterations, with GVT being slightly slower. FT achieves good performance as well, but it takes more time to adapt a random source Q-function. Interestingly, we notice that there is no advantage in adopting more than 1 component for the posterior in MGVT. This result is intuitive since, as soon as the algorithm figures out which is the target task, all the components move towards the same region.

To better understand the differences between GVT and MGVT, we now consider transferring from a slightly different distribution than the one from which target tasks are drawn. We generate 50 source tasks again but this time with the bottom door fixed at the center and the other one moving. Then, we repeat the previous experiment, allowing both doors to move when sampling target tasks. The results are shown in Figure 7.2(right). Interestingly, MGVT seems almost unaffected by this change, proving that it has sufficient representation power to generalize to slightly different task distributions. The same does not hold for GVT, which now is not able to solve many of the sampled target tasks, as can be noticed from the higher variance. Furthermore, the good performance of FT proves that

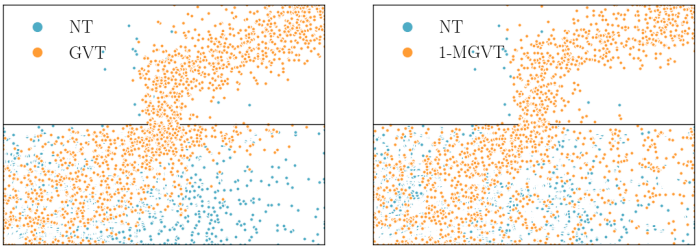

Figure 7.3: *Exploration in the two-room problem: ϵ-greedy vs (a) GVT and (b) 1-MGVT.*

GVT is, indeed, subject to a loss of information due to averaging the source weights. This result proves again that assuming Gaussian distributions can pose severe limitations in our transfer settings.

Finally, we investigate the exploratory behavior induced by our transfer algorithms and compare it to simple ϵ-greedy exploration. In Figure 7.3, we show the positions visited by the agent when running 2000 iterations of NT, GVT, and 1-MGVT. Observing Figure 7.3(left), it is possible to understand the difference between the ϵ-greedy exploration and the resulting behavior from GVT. It is noticeable that NT is not capable to lead the agent to the goal within the given iterations as most of the states visited are sparse within the first room, whereas GVT is able to concentrate more of its effort in looking for the door around the middle of the wall. After finding it, within the second room, the positions concentrate in the path leading to the goal. This is not surprising as the value function should be equal for all tasks after crossing the door. In the other case, Figure 7.3(right) shows a similar situation, but it is quite interesting to notice how sparser the exploration of 1-MGVT is with respect to GVT. Indeed, 1-MGVT is able to actually explore the right part of the first room within these iterations, which might be seen as the result of the prior model being able to capture more information than the Gaussian; hence, the higher speed-up in convergence and robustness to changes in the distribution from which target tasks are drawn. Indeed, as 1-MGVT is able to allow for more flexible exploration, it is capable to discover how to best solve the task much faster than GVT.

Classic Control

We now consider two well-known classic control environments: Cartpole and Mountain Car (Sutton and Barto, 2018). For both, we generate 20 source tasks by uniformly sampling their physical parameters (cart mass, pole mass, pole length for Cartpole and car speed for Mountain Car) and solve them by directly minimizing the TD error as in the previous experiment. We parameterize Q-functions using neural networks with one layer of 32 hidden units for Cartpole and 64 for Mountain Car. In this experiment, we use a Double Deep Q-Network (DDQN) Van Hasselt et al. (2016) to provide a stronger no-transfer baseline for comparison.

Results. The results are shown in Figure 7.4. For Cartpole (left plot), all variational transfer algorithms are almost zero-shot. This result is expected since, although we vary the system parameters in a wide range, the optimal Q-values of states near the balanced

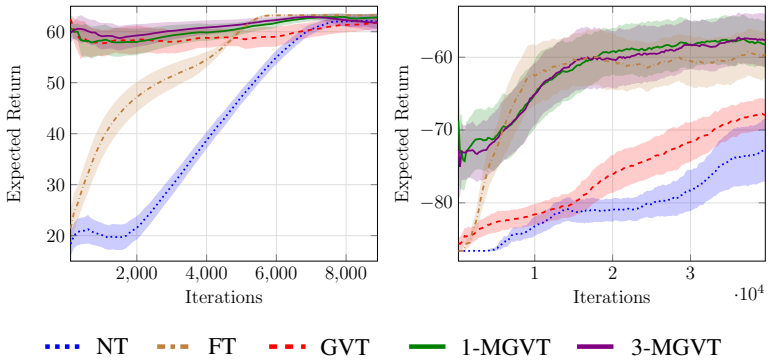

Figure 7.4: *Transfer in Cartpole (left) and Mountain Car (right).*

position are similar for all tasks. On the contrary, in Mountain Car (right plot) the optimal Q-functions become very different when changing the car speed. This phenomenon hinders the learning of GVT in the target task, while MGVT achieves a good jump-start and converges in fewer iterations. Similarly to the rooms domain, the naive weight adaptation of FT makes it slower than MGVT in both domains.

Maze Navigation

Finally, we consider a robotic agent navigating mazes. At the beginning of each episode, the agent is dropped to a random position in a $10m^2$ maze and must reach a goal area in the smallest time possible. The robot is equipped with sensors detecting its absolute position, its orientation, the distance to any obstacle within $2m$ in 9 equally-spaced directions, and whether the goal is present in the same range. The only actions available are *move forward* with speed $0.5m/s$ or *rotate* (in either direction) with speed of $\pi/8\ rad/s$. Each time step corresponds to $1s$ of simulation. The reward is 1 for reaching the goal and 0 otherwise, while the discount factor is $\gamma = 0.99$. For this experiment, we design a set of 20 different mazes (see Figure 7.5) with varying degree of difficulty and that hold few similarities that would be useful for transferring. Moreover, we ensure 4 groups of mazes that are characterized by the same goal position. We solve them using a DDQN with two layers of 32 neurons and ReLU activations. Then, we fix a target maze and transfer from 5 source mazes uniformly sampled from such set (excluding the chosen target). To further assess the robustness of our method, we now consider transferring from the Q-functions learned by DDQNs instead of those obtained by minimizing the TD error as in the previous domains. From our considerations of Sections 7.3.3 and 7.4, the fixed-points of the two algorithms are different, which creates a further challenge for our method.

Results. We show the results for two different target mazes in Figure 7.6. Once again, MGVT achieves a remarkable speed-up over (no-transfer) DDQN. This time, using 3 components achieves slightly better performance than using only 1, which is likely due to the fact that the task distribution is much more complicated than in the previous domains. For the same reason, GVT shows negative transfer and performs even worse than DDQN.

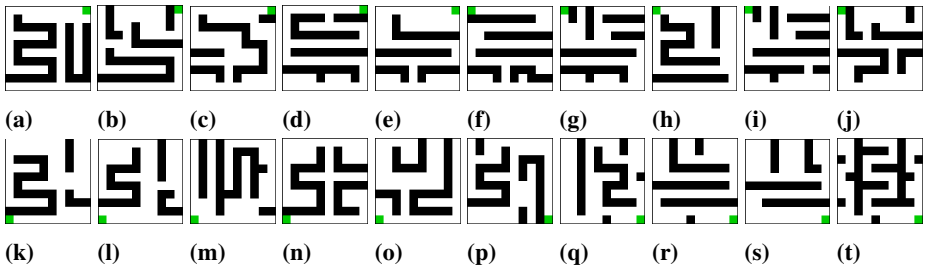

Figure 7.5: *Set of mazes for the Maze Navigation task.*

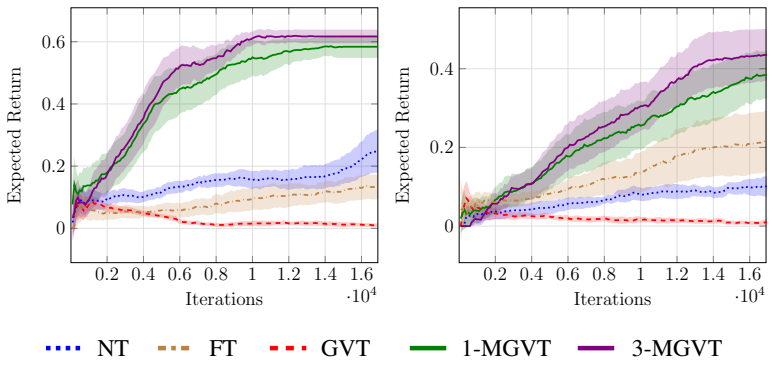

Figure 7.6: *Maze navigation task. (left) The target maze is the one of Figure 7.5a and (right) the target maze is the one of Figure 7.5g.*

Similarly, FT performs much worse than in the previous domains and negatively transfer in the more complicated target maze of Figure 7.6(left).

Comparison to Fast-Adaptation Algorithms

We now show a comparison to the recently proposed meta-learner MAML (Finn et al., 2017). We repeat the previous experiments, focusing on the navigation tasks, using two different versions of MAML. In the first one (MAML-full), we perform meta-training using the full distribution over tasks for a number of iterations that allows the meta-policy to converge. In the second one (MAML-batch), we execute the meta-train only on the same number of fixed source tasks as the one used for our algorithm, allowing the meta-policy to reach convergence again. In both cases, we meta-test on random tasks sampled from the full distribution. The results are shown in Figure 7.7 in comparison to our best algorithm (3-MGVT), where each curve is obtained by averaging 5 meta-testing runs for each of 4 different meta-policies. In both cases, the full version of MAML achieves a much better jumpstart and adapts much faster than our approach. However, this is no longer the case when limiting the number of source tasks. In fact, this situation reduces to the case in which the task distribution at meta-training is a discrete uniform over the fixed source tasks, while at meta-testing the algorithm is required to generalize to a different distribution. This is a case that arises frequently in practice for which MAML was not

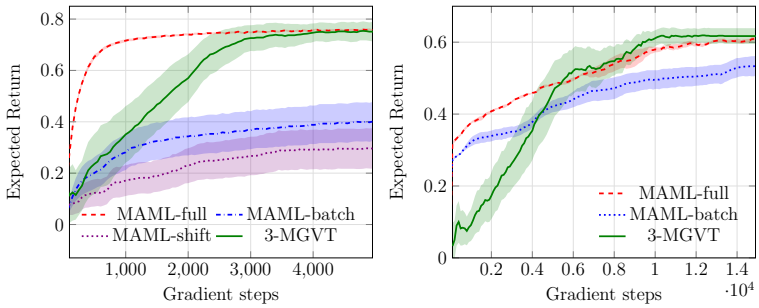

Figure 7.7: *MAML vs 3-MGVT. (left) The rooms problem and (right) maze navigation.*

specifically designed. Things get even worse when we explicitly add a shift to the meta-training distribution as we did in Figure 7.2(right) for the rooms problem (MAML-shift in Figure 7.7(left)). Although we meta-trained on the full distribution, the final performance was even worse than the one using the fixed source tasks. Note that we compare the algorithms with respect to the number of gradient steps, even if our approach collects only one new sample at each iteration while MAML collects a full batch of trajectories.

Part III

Exploration in Structured Domains

CHAPTER 8

Learning and Transfer in Structured Domains

Introduction

So far we focused on building practical transfer methods that only require experience from some unknown source tasks and the ability to interact with an unknown target environment, while in only few cases we explicitly modeled or made use of the underlying structured domain. For instance, in Chapter 6 we considered specific task families for deriving our weight-estimation procedure, while in Chapter 7 we assumed tasks with linear value functions to analyze our method. Besides these assumptions, our algorithms implicitly exploit structural properties of the underlying domain to transfer knowledge, though it is not clear to what extent from a theoretical viewpoint.

This part is devoted to a theoretical treatment of the general problem of learning and transfer in structured domains, with a focus on *exploration*. We recall from Chapter 3 that a *structured domain* is a tuple $\mathfrak{E} = (\mathfrak{M}, \mathfrak{D})$, where $\mathfrak{M}$ is a set of realizable MDPs and $\mathfrak{D}$ is the process from which they are generated (see Definition 3.1.1). An agent acting in a structured domain $\mathfrak{E}$ faces one or more MDPs $\mathcal{M} \in \mathfrak{M}$ generated from $\mathfrak{D}$. As already mentioned, the name "structured" indicates that tasks in $\mathfrak{E}$ share some similarities which provide the agent with prior knowledge and/or enable knowledge transfer. This structure can manifest in three components:

- *The set $\mathfrak{M}$ of realizable tasks*. In some problems there exists only a small subset of possible MDPs that can be faced by the agent (Azar et al., 2013a; Brunskill and

Li, 2013). Knowing or learning this set allows to discard a priori some possibilities when facing a new target task, thus reducing the uncertainty over the latter.

- *A shared representation*. In some cases the MDPs in $\mathfrak{M}$ have a common representation. For instance, $\mathfrak{M}$ could be a set of MDPs with linear reward functions with varying parameters and fixed feature bases (Barreto et al., 2017, 2018), or the set of MDPS with dynamics governed by some hidden parameters (Doshi-Velez and Konidaris, 2016; Killian et al., 2017).
- *The task generation process* $\mathfrak{D}$. Similarly to the first point, understanding the generation process can effectively reduce the uncertainty over new tasks. For instance, some problems present sequential correlations, so that the distribution of each new task depends on those faced before (Choi et al., 2000b,a).

Depending on what the agent knows about $\mathfrak{E}$, we distinguish *learning* and *transfer* problems. Whenever the agent knows $\mathfrak{M}$ (and potentially $\mathfrak{D}$) the problem is that of *learning* in a structured domain. In this case, having solved some tasks from $\mathfrak{E}$ provides little or no information about the target task since the underlying structure (i.e., the set of realizable MDPs and their properties) is already known. On the other hand, when the agent has no knowledge about the structured domain $\mathfrak{E}$, facing some source tasks provides information about $\mathfrak{E}$ itself, which in turns provides information about the target task. Extrapolating knowledge about $\mathfrak{E}$ from the source tasks and using it to speed up the learning process of the target task corresponds to *knowledge transfer*. In particular, a general scheme for a transfer algorithm, that we shall use for our theoretical study, consists in alternating two main steps:

1. *Learn structure from previously-solved tasks*. The agent uses knowledge obtained while facing previous tasks to build/refine its understanding of the underlying structured domain. For instance, it can try to estimate any of the three components mentioned above.
2. *Exploit structure to learn new tasks*. The agent uses its current knowledge about the structured domain to quickly learn a new target task. That is, knowledge from the source tasks is effectively transferred to solve the target through the estimated structured domain.

These two steps are then repeated over and over, with the agent constantly refining its knowledge about the structured domain and using it to quickly learn new tasks. This is an appealing framework for the theoretical study of transfer problems that is indeed adopted in several previous works (Brunskill and Li, 2013; Azar et al., 2013a; Liu et al., 2016b; Sun et al., 2020). In particular, it allows to decouple the structure learning and exploitation steps, so that one can study the two problems (almost) independently.

In this part, we focus on the problem of *exploiting structure* to improve *exploration*, both in the case where the structure is exactly *known* and in the case where it is learned from experience and thus possibly *misspecified*. Among the many open questions (as oulined in Chapter 1), we shall address the following ones. (1) How to build structure-aware algorithms that never perform worse than unstructured baselines? (2) How to build statistically optimal strategies? (3) How to handle misspecified structures? Before diving into our contributions, we better formalize and discuss the problems of learning (Section 8.2) and transfer (Section8.3) in structured domains.

Learning in Structured Domains

In order to keep the problem close to the existing literature, we restrict our definition of structured domain to *parametrized* tasks.

Definition 8.2.1 (Parameterized structured domain)**.** *A parameterized structured domain is a tuple $\mathfrak{E}_\Theta = (\Theta, \mathfrak{M}_\Theta, \mathfrak{D})$, where Θ is a set of parameters, $\mathfrak{M}_\Theta = \{\mathcal{M}_\theta\}_{\theta\in\Theta}$ is a hypothesis space of parametrized task models, and $\mathfrak{D}$ is the task generation process.*

When learning in a parameterized structured domain (from now on simply called structured domain), the agent knows $\mathfrak{E}_\Theta$, though it does not know the identity of the target task $\mathcal{M}_{\theta^\star}$ (or, equivalently, of the parameter $\theta^\star$) that it is required to solve. We focus on *finite* MDP and bandit problems in *frequentist* settings, i.e., we aim at deriving problem-dependent guarantees for learning the unknown target task parameterized by $\theta^\star$. For this reason, we shall never make use of the task generation process $\mathfrak{D}$. The families of parametrized task models $\mathfrak{M}_\Theta = \{\mathcal{M}_\theta\}_{\theta\in\Theta}$ that we study are

- *Multi-armed bandits* (Chapter 9). In this case, $\mathcal{M}_\theta = (\mathcal{A}, \mu_\theta)$ is a bandit problem with a finite set $\mathcal{A}$ of K arms and mean rewards $\mu_\theta = \{\mu_\theta(a)\}_{a\in\mathcal{A}}$. Each arm $a \in \mathcal{A}$ yields rewards bounded in $[0, 1]$.
- *Linear contextual bandits* (Chapter 10). We have $\mathcal{M}_\theta = (\mathcal{X}, \mathcal{A}, \mu_\theta, \rho)$, where $\mathcal{X}$ is a finite set of contexts, $\mathcal{A}$ is a finite set of arms, $\mu_\theta = \{\mu_\theta(x, a)\}_{x\in\mathcal{X}, a\in\mathcal{A}}$ are the mean rewards, ρ is the context distribution. The mean reward of each context-arm pair is $\mu_\theta(x, a) = \phi(x, a)^T\theta$, where ϕ is a known feature map, while the reward distribution is Gaussian with known variance σ^2.
- *Discounted MDPs* (Chapter 11). We have that $\mathcal{M}_\theta = (\mathcal{S}, \mathcal{A}, P_\theta, U_\theta, \gamma)$ is an MDP with finite states $\mathcal{S}$ and actions $\mathcal{A}$, parametrized transition probabilities P_θ and reward distributions U_θ, and discount factor γ.

Since each hypothesis $\mathcal{M}_\theta$ is known given its parameter θ, in the remaining we shall alternatively refer to each $\theta \in \Theta$ as (bandit/MDP) problem/instance/model. When learning in this structured setting, samples from one action/arm might provide information about $\theta^\star$ which, through the knowledge of $\mathfrak{E}_\Theta$, might provide information about the entire task model $\mathcal{M}_{\theta^\star}$. This makes the problem of *exploration* in a structured domain interesting and challenging.

Structured Bandits

The problem of learning in structured domains has been widely studied in the bandit literature. The performance measure for a bandit algorithm π that receives as input the structured domain $\mathfrak{E}_\Theta$ is the expected regret after n steps,

$$R_n^\pi(\theta^\star, \mathfrak{E}_\Theta) := n \max_{a\in\mathcal{A}} \mu_{\theta^\star}(a) - \mathbb{E}_{\pi,\theta^\star}\left[\sum_{t=1}^{n} \mu_{\theta^\star}(A_t)\right].$$

Note that the regret depends on $\mathfrak{E}_\Theta$ through the strategy π, which might use structure to choose its actions. Interestingly, the regret of a structure-aware algorithm on the *same* instance $\theta^\star$ might change when provided with different structures.

The most popular works consider *specific* structures, i.e., they assume a precise model/representation for the realizable bandit problems. The model of Definition 8.2.1 formalizes bandits with *general* structures (Agrawal et al., 1988; Graves and Lai, 1997; Burnetas and Katehakis, 1996; Azar et al., 2013a; Lattimore and Munos, 2014; Combes et al., 2017), i.e., those that generalize the majority of specific settings considered in the literature. For instance, an unstructured (disjoint) bandit class with K arms, mean-rewards in $[0, 1]$, and Gaussian noise with fixed variance σ^2 can be formalized by $\Theta = [0, 1]^K$, $\mu_\theta = \{\theta_a\}_{a\in\mathcal{A}}$ for $\theta \in \Theta$, and reward distributions $\nu_\theta(a) = \mathcal{N}(\theta_a, \sigma^2)$. In a linear structure the mean reward of each arm is assumed to be a linear combination of given d-dimensional feature vectors ϕ and an unknown parameter $\theta^\star$, while the observed rewards are corrupted by zero-mean sub-Gaussian noise. The corresponding structure is thus described by $\Theta \subset \mathbb{R}^d$ (typically compact) and $\mu_\theta = \{\phi(a)^T\theta\}_{a\in\mathcal{A}}$. Several algorithms have been proposed for this setting, such as extensions of UCB (Abbasi-Yadkori et al., 2011) and Thompson sampling (Agrawal and Goyal, 2013; Abeille and Lazaric, 2017). Examples of other specific structures include combinatorial bandits (Cesa-Bianchi and Lugosi, 2012), Lipschitz bandits (Magureanu et al., 2014), ranking bandits (Combes et al., 2015), and unimodal bandits (Yu and Mannor, 2011). Interestingly, for these structured setting, approaches based on the popular optimism in the face of uncertainty (such as UCB and, to some extent, Thompson sampling) are typically not asymptotically optimal in a problem-dependent sense (Lattimore and Szepesvari, 2017), while they are optimal for unstructured (disjoint) problems (see Section 2.6). This makes the design of optimal structure-aware strategies an appealing and challenging direction, as we shall see in Chapter 10.

Recently, there has been a growing interest in designing bandit strategies to exploit general structures (those of Definition 8.2.1) without any additional specific assumption. The structured UCB algorithm, proposed almost-simultaneously by Lattimore and Munos (2014) and Azar et al. (2013a), applies the OFU principle to general structures. Atan et al. (2018) proposed a greedy algorithm for the special case where all arms are informative, while Wang et al. (2018) extended these settings to consider correlations only within certain groups of arms and independence among them. Gupta et al. (2018b) generalized UCB and TS to exploit the structure and quickly identify sub-optimal arms. One of the interesting findings of these works is that, in some structures, constant regret (i.e., independent of n) is possible. We shall call these strategies *confidence-based* since they explicitly maintain the uncertainties about the true bandit and use these to trade-off exploration/exploitation. Although conceptually simple, confidence-based strategies are typically hard to design and analyze in a fully structure-aware manner. In fact, in structured problems, pulling an arm provides not only a sample of its mean, but also information about the bandit problem itself through the knowledge of the overall structure. In turn, information about the problem itself potentially allow to refine the estimates of the means of *all* arms. Combes et al. (2017) made a significant step in exploiting this interplay between arms and bandit problems in the very definition of the algorithm itself. The authors derived a structure-aware lower bound characterizing the optimal pull counts as the solution to an optimization problem. Their algorithm, OSSB, approximates this solution and achieves asymptotic optimality for any general structure. However, since the lower bound depends on the true (unknown) bandit at hand, this approach requires to force some exploration to guarantee a sufficiently accurate solution. For this reason, we shall call this kind of strategy *forced-exploration*. Recently, Degenne et al. (2020b) propose an algorithm (SPL)

that addresses these limitations. SPL replaces forced-exploration with confidence intervals about the true bandit problem and uses an iterative incremental method to solve the optimization problem of the lower bound which makes the approach computationally efficient. Similarly, Jun and Zhang (2020) design a confidence-based algorithm that is near-optimal for general structures, though less computationally efficient than SPL.

Structured MDPs

Structured MDPs are formalized analogously as structured bandits, with the only difference that both reward distributions and transition probabilities are parameterized by θ. For the problem of regret minimization, the first algorithm that is asymptotically optimal for MDPs with general structures was designed by Ok et al. (2018). The authors build on top of the same ideas as OSSB, by iteratively recomputing and tracking the lower bound while using forced-exploration to guarantee a minimum level of estimation accuracy. Besides the general structures, the literature focuses on MDPs with linear rewards and/or transition probabilities (Yang and Wang, 2019; Lattimore and Szepesvari, 2019; Jin et al., 2020; Zanette et al., 2020) and on reinforcement learning in metric spaces, e.g., Lipschitz MDPs (Lakshmanan et al., 2015; Jian et al., 2019; Song and Sun, 2019; Sinclair et al., 2019). Finally, contextual MDPs (Hallak et al., 2015; Modi et al., 2018; Sun et al., 2018) and hidden-parameter MDPs (Doshi-Velez and Konidaris, 2016; Killian et al., 2017), where rewards and/or transition probabilities depend on some latent context/parameter, are another popular model for structured problems.

Transfer in Structured Domains

The problem decomposition described in the introduction is popular for the theoretical study of transfer in reinforcement learning. Brunskill and Li (2013) assume that there exists an unknown finite set of realizable MDPs. They derive a method to learn this set from experienced tasks and propose a variant of E^3 (Kearns and Singh, 2002) that explores a target task so as to quickly identify it in the learned set. They derive bounds on the sample complexity for learning any new task that clearly show the benefits of knowledge transfer. Liu et al. (2016b) extend these ideas to the more challenging problem of continuous MDPs, while still showing sample-complexity bounds. Azar et al. (2013a) consider a sequential multi-armed bandit setting where the agent faces a sequence of bandit problems drawn i.i.d. from a finite set. They propose a spectral learning method to estimate the set of possible problems and combine it with a variant of UCB that exploits the given structure. Their regret guarantees show that the method is robust to negative transfer, i.e., it never performs worse than plain UCB. Recently, Sun et al. (2020) learn structure in the form of transition templates that can be readily reused to speed up the learning process of new tasks. While the considered setting is similar to the one of Brunskill and Li (2013), they derive refined sample complexity bounds. Mann and Choe (2013) and Abel et al. (2018) study how to to achieve a jumpstart. They learn structure in the form of an optimistic initialization to the target action-value function and use it in combination with any PAC algorithm. The resulting methods provably achieve positive transfer. Mahmud et al. (2013) aim at reusing policies. The learned structure is in the form of clusters of solved tasks, which result in a set of policies from which to transfer. To learn the target task, they use

the adversarial-bandit algorithm EXP3 (Auer et al., 2002b) over the set of source policies augmented with a base reinforcement learning algorithm. Notably, they prove robustness to negative transfer, i.e., the method never performs much worse than the base learner.

Learning Structure from Source Tasks

The structure learning method is one of the key components of transfer algorithms that use the decomposition introduced before. While we shall throughly study the problem of exploiting given structure in the next chapters, here we briefly summarize the main methods to estimate the structure from source tasks. These methods can be adapted to learn the structure for our algorithms, hence enabling their application to transfer settings.

Clustering methods. The idea behind the method of Brunskill and Li (2013) is to cluster the tasks faced by the agent in a few groups, so that each group is likely to contain only tasks corresponding to the same underlying MDP. The method assumes that an upper bound to the total number of realizable MDPs (assumed finite) is known and uses it to set the number of clusters. Clustering is then performed by comparing the empirical MDP models of each task computed from the observed samples. In particular, when two empirical models are more distant than a given threshold in a single state-action pair (for either the rewards or transition probabilities), they are put in different clusters. As the authors formally verify, whenever the realizable MDPs have a known minimum positive distance (which is used to decide an appropriate threshold) and the empirical model of each task is accurate, this method groups the faced tasks correctly.

Spectral methods. Consider a structured domain with a finite set of parameters Θ. One can see the problem as learning a latent variable model, where the latent variable is the task parameter θ while the observations are the samples collected by the agent (e.g., states, actions, and rewards). The goal is then to recover the conditional distributions of the observations given task parameters, i.e., the underlying MDP models. Spectral methods (Anandkumar et al., 2012, 2014) have been widely applied for this purpose. In particular, in reinforcement learning, they have been applied for learning POMDPs (Azizzadenesheli et al., 2016; Guo et al., 2016) and for learning structure in a sequential i.i.d. transfer setting (Azar et al., 2013a). The idea is to estimate certain moments of the observed samples whose tensor decomposition (Anandkumar et al., 2014) yields the underlying models. Differently from clustering approaches, these methods do not require assumptions on the minimum gap between models and typically come with strong finite-sample guarantees on the estimation error. Similarly to the sequential i.i.d. setting, spectral methods can be used to recover the underlying models when tasks are temporally-correlated through an hidden Markov chain, as we have shown in Tirinzoni et al. (2020c).

Representation learning. Methods for learning representations that are shared across multiple tasks are popular especially in the supervised learning community (Pan and Yang, 2009). For instance, some of these methods try to learn shared features for a multi-task regression setting Argyriou et al. (2007); Li et al. (2015b); Wang et al. (2016b). In reinforcement learning, similar ideas are used by Ammar et al. (2014) in a lifelong setting to learn the common representation of continually evolving tasks.

CHAPTER 9

Arm Elimination in Structured Bandits

This chapter is based on the paper "A Novel Confidence-Based Algorithm for Structured Bandits" co-authored with Alessandro Lazaric and Marcello Restelli and published at AISTATS 2020.

Introduction

In this chapter, we begin our study of the exploration problem in domains with known structure. We a consider a structured-bandit domain $\mathfrak{E}_\Theta$ (see Definition 8.2.1) with a set of parameters Θ and a hypothesis space $\mathfrak{M}_\Theta = \{\mathcal{M}_\theta\}_{\theta\in\Theta}$ that we fix throughout the whole chapter. Each model $\mathcal{M}_\theta$ is a bandit with a set $\mathcal{A}$ of K arms and rewards bounded in $[0, 1]$ with mean $\mu_\theta(a)$ for each $a \in \mathcal{A}$. We shall never make explicit use of the specific reward distribution as the algorithms use only the boundedness assumption. Many confidence-based strategies have been proposed for this bandit setting with general structures (Azar et al., 2013a; Lattimore and Munos, 2014; Atan et al., 2018; Wang et al., 2018; Gupta et al., 2018b). While these confidence-based strategies are in general not asymptotically optimal, they often enjoy good finite-time performance (as opposed to optimal strategies based on forced exploration). One downside is that the regret bounds derived in these papers do not fully reflect how the algorithm exploits the given structure. The main complication in analyzing these approaches is that arms are correlated through the underlying structure and, thus, should be somehow analyzed jointly. On the other hand, the typical problem-dependent analyses (on top of which the majority of the above-mentioned prior works build) consider each arm independently, as if the problem were unstructured.

In this chapter, we make a step forward in understanding bandit strategies for general structures. We propose a novel confidence-based arm-elimination strategy for which we derive a regret bound that clearly shows how the algorithm benefit the structure. More precisely, we propose an algorithm running through phases. At the beginning of each phase, the set of bandit models compatible with the confidence intervals computed so far is built and the corresponding optimal arms are repeatedly pulled in a round-robin fashion, until the end of the phase. For this strategy, we prove an upper bound on the expected regret that, compared to existing bounds, better shows the potential benefits of exploiting the structure. The key finding is that the number of pulls to a sub-optimal arm a can be significantly reduced by exploiting the information obtained while pulling other arms, and notably the arm that is most informative for this purpose, i.e., the arm for which the mean of the true bandit differs the most from that of any other bandit in which arm a is optimal. This is in contrast to existing methods, which rely exclusively on the samples obtained from arm a to identify its sub-optimality (a property that is true for the unstructured settings). Since our algorithm requires to know the horizon n, we design a practical anytime extension for which, under the same assumptions as in (Lattimore and Munos, 2014), we derive a constant-regret bound with a better scaling in the relevant structure-dependent quantities. Finally, we report numerical simulations in some simple illustrative structures that confirm our theoretical findings.

It is worth noting that, in the original paper (Tirinzoni et al., 2020a), we also try to characterize in what structures our confidence-based strategy is near-optimal. Notably, we derive a matching lower-bound for certain structures where constant regret is possible. While these results provide more insights on the overall problem, they require significantly more complex notation and, thus, they were not reported in this thesis.

Notation. We denote by $\mu_\theta^\star := \max_{a \in \mathcal{A}} \mu_\theta(a)$ the optimal reward for bandit $\theta \in \Theta$ and, for the sake of readability, we assume that the corresponding optimal arm, $a_\theta^\star := \operatorname{argmax}_{a \in \mathcal{A}} \mu_\theta(a)$, is unique for all models. The sub-optimality gap of arm $a \in \mathcal{A}$ is $\Delta_\theta(a) := \mu_\theta^\star - \mu_\theta(a)$, while the model gap with respect to $\theta' \in \Theta$ is $\Gamma_a(\theta, \theta') := |\mu_\theta(a) - \mu_{\theta'}(a)|$. It is known that the gaps Δ characterize the complexity of a bandit problem in the unstructured case. We denote by $\mathcal{A}^\star(\Theta) := \{a \in \mathcal{A} | \exists \theta \in \Theta : a_\theta^\star = a\}$ the set of arms that are optimal for at least one model in Θ. Similarly, we call $\Theta_a^\star := \{\theta \in \Theta | a_\theta^\star = a\}$ the set of models in which arm $a \in \mathcal{A}$ is optimal. In the remaining, whenever θ is dropped from a model-dependent quantity, we implicitly refer to the target $\theta^\star$.

The Structured UCB Algorithm

Before presenting our method, we provide a detailed description of the Structured UCB algorithm, proposed almost simultaneously by Lattimore and Munos (2014) and Azar et al. (2013a). We also derive a slightly refined analysis of this method so as to facilitate comparison with out results.

Structured UCB (SUCB)[1] is a natural application of the optimism in face of uncertainty

[1]The algorithm was originally called UCB-S by Lattimore and Munos (2014) and mUCB by Azar et al. (2013a). Here we use the general acronym SUCB.

principle to general structures. At each step t, the algorithm builds a confidence set

$$\tilde{\Theta}_t = \left\{ \theta \in \Theta \;\middle|\; \forall a \in A : |\mu_\theta(a) - \hat{\mu}_{t-1}(a)| < \sqrt{\frac{\alpha \log t}{2N_{t-1}(a)}} \right\} \tag{9.1}$$

containing the models compatible with *all* the confidence intervals built separately for each arm. We recall from Section 2.6 hat $N_{t-1}(a)$ is the number of times $a \in \mathcal{A}$ was selected prior to time t and $\hat{\mu}_{t-1}(a)$ is the empirical reward mean. Then, SUCB pulls the optimistic arm

$$A_t = \operatorname*{argmax}_{a \in \mathcal{A}} \sup_{\theta \in \tilde{\Theta}_t} \mu_\theta(a). \tag{9.2}$$

In other words, the arm-selection strategy is exactly UCB, while the structure is encoded into the confidence sets. In particular, the latter are tightened by taking the intersection between confidence intervals built independently for each arm and the set of hypotheses $\mathfrak{M}_\Theta$. While taking the optimistic arm ensures that "good" arms are selected, refining the confidence set $\tilde{\Theta}_t$ allows to exploit the structure to possibly discard arms more rapidly. Interestingly, SUCB reduces to UCB whenever the structure Θ encodes an unstructured problem (see Section 8.2).

Lattimore and Munos (2014) derived the same upper bound to the regret as the one of UCB (see their Theorem 2) without making any assumption on Θ. That is, for a suitable choice of α, there exist constants c, c' such that

$$R_n^{\mathrm{SUCB}}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \neq a^\star_{\theta^\star}} \frac{c \log n}{\Delta_{\theta^\star}(a)} + c'. \tag{9.3}$$

This bound, however, does not fully reflect how the algorithm exploits the given structure. Azar et al. (2013a) derived a more structure-aware bound, but only for finite Θ. The next theorem combines the best of these analyses, thus providing a regret bound that scales with the model gaps rather than the sub-optimality gaps and that holds for any structure.

Theorem 9.2.1. *There exist constants $c, c' > 0$ (independent of n) such that for any structure $\mathfrak{E}_\Theta$ and instance $\theta^\star \in \Theta$, for $\alpha > 2$, the expected regret over n steps of the SUCB algorithm is upper-bounded as*

$$R_n^{\mathrm{SUCB}}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \in \mathcal{A}^\star(\Theta) \setminus \{a^\star_{\theta^\star}\}} \frac{c \Delta_{\theta^\star}(a) \log n}{\inf_{\theta \in \Theta_a^+(\theta^\star)} \Gamma_a(\theta, \theta^\star)^2} + c',$$

where we define the following set of optimistic models with respect to $\theta^\star$:

$$\Theta_a^+(\theta^\star) := \{\theta \in \Theta | \mu_\theta^\star > \mu_{\theta^\star}^\star, a_\theta^\star = a\}. \tag{9.4}$$

This result shows that SUCB is able to leverage the knowledge of Θ to improve over UCB, which does not use structure at all. First, the summation is limited to arms that are optimal in at least one model in Θ. Second, the number of pulls of a sub-optimal arm $a \neq a^\star_{\theta^\star}$ depends on the model gap $\Gamma_a(\theta_a^+, \theta^\star)$ with respect to the *hardest model* $\theta_a^+ \in \Theta_a^+(\theta^\star)$ to be distinguished from $\theta^\star$ by pulling a. This gap can be much larger (and provably never smaller) than the sub-optimality gap $\Delta_{\theta^\star}(a)$ which appears in unstructured settings (e.g., UCB), thus significantly reducing the final regret.

Proof. Let $F_t := \mathbb{1}\left\{\theta^\star \in \tilde{\Theta}_t\right\}$. Consider any sub-optimal arm $a \in \mathcal{A}$ and suppose $A_t = a$ and $F_t = 1$. Since a is pulled, there exists some $\bar{\theta} \in \tilde{\Theta}_t$ such that $\bar{\theta} \in \Theta_a^+$. These facts imply

$$\Gamma_a(\bar{\theta}, \theta^\star) = |\mu_{\bar{\theta}}(a) - \mu_{\theta^\star}(a)| \le |\mu_{\bar{\theta}}(a) - \hat{\mu}_{t-1}(a)| + |\hat{\mu}_{t-1}(a) - \mu_{\theta^\star}(a)| \le 2\sqrt{\frac{\alpha \log t}{2N_{t-1}(a)}}.$$

Therefore,

$$N_{t-1}(a) \le \frac{2\alpha \log t}{\Gamma_a^2(\bar{\theta}, \theta^\star)} \le \left\lceil \frac{2\alpha \log n}{\inf_{\theta \in \Theta_a^+} \Gamma_a^2(\theta, \theta^\star)} \right\rceil =: u_a(n).$$

Then,

$$\begin{aligned}
\mathbb{E}[N_n(a)] &= \mathbb{E}\left[\sum_{t=1}^n \mathbb{1}\left\{A_t = a\right\}\right] = \mathbb{E}\left[\sum_{t=1}^n \mathbb{1}\left\{A_t = a \wedge N_{t-1}(a) \le u_a(n)\right\}\right] \\
&\quad + \mathbb{E}\left[\sum_{t=1}^n \mathbb{1}\left\{A_t = a \wedge N_{t-1}(a) > u_a(n)\right\}\right] \\
&\le u_a(n) + \mathbb{E}\left[\sum_{t=u_a(n)+1}^n \mathbb{1}\left\{A_t = a \wedge N_{t-1}(a) > u_a(n)\right\}\right] \\
&\le u_a(n) + \mathbb{E}\left[\sum_{t=u_a(n)+1}^n \mathbb{1}\left\{A_t = a \wedge F_t = 0\right\}\right],
\end{aligned}$$

where the last inequality follows since pulling arm a at time step t implies that either $N_{t-1}(a) \le u_a(n)$ or the true parameter is not in the confidence set (i.e., $F_t = 0$). Then,

$$\begin{aligned}
R_n^{\mathrm{SUCB}}(\theta^\star, \mathfrak{E}_\Theta) &\overset{(a)}{=} \sum_{a \in \mathcal{A}^\star(\Theta)} \Delta_{\theta^\star}(a) \mathbb{E}[N_n(a)] \\
&\overset{(b)}{\le} \sum_{a \in \mathcal{A}^\star(\Theta)} \Delta_{\theta^\star}(a) \left(u_a(n) + \mathbb{E}\left[\sum_{t=u_a(n)+1}^n \mathbb{1}\left\{A_t = a \wedge F_t = 0\right\}\right]\right) \\
&\overset{(c)}{\le} \sum_{a \in \mathcal{A}^\star(\Theta)} \Delta_{\theta^\star}(a) u_a(n) + \Delta_{\max} \sum_{t=1}^n \mathbb{P}\{F_t = 0\},
\end{aligned}$$

where (a) holds since arms that are sub-optimal for all models in Θ are never pulled, (b) follows from the bound on the number of pulls derived above, and (c) follows from the definition of $\Delta_{\max} = \max_{a \in \mathcal{A}^\star(\Theta)} \Delta_{\theta^\star}(a)$ and the fact that at each time only one arm is pulled. The second term can be bounded using Lemma 5 of Lattimore and Munos (2014) (by setting the sub-Gaussian variance factor to $\sigma^2 = \frac{1}{4}$ for bounded rewards and taking the union bound only over $\mathcal{A}^\star(\Theta)$) by

$$\sum_{t=1}^n \mathbb{P}\{F_t = 0\} \le 2|\mathcal{A}^\star(\Theta)| \sum_{t=1}^n t^{1-\alpha} \le \frac{2|\mathcal{A}^\star(\Theta)|(\alpha - 1)}{\alpha - 2}.$$

The theorem follows by combining the last two displays and renaming the constants. □

It is worth noting that this proof is carried out independently for each sub-optimal arm, without considering the information that the algorithm might collected while pulling other arms.

Algorithm 6 Structured Arm Elimnation (SAE)

Require: Set of parameters Θ, hypothesis space $\mathfrak{M}_\Theta$, horizon n, scalars $\alpha > 0, \beta \geq 1$

Initialize confidence set: $\tilde{\Theta}_0 \leftarrow \Theta$

Initialize set of active arms: $\tilde{\mathcal{A}}_0 \leftarrow \mathcal{A}^\star(\Theta)$

Initialize removal threshold: $\tilde{\Gamma}_0 \leftarrow 1$

Foreach phase $h = 0, 1, \dots$ **do**

Play all active arms until $\left\lceil \frac{\alpha \log n}{\tilde{\Gamma}_h^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil$ pulls are reached for all $a \in \tilde{A}_h$

Build confidence set: $\tilde{\Theta}_{h+1} \leftarrow \left\{ \theta \in \Theta \middle| \, \forall a \in \mathcal{A} : |\hat{\mu}_h(a) - \mu_\theta(a)| < \sqrt{\frac{\alpha \log n}{N_h(a)}} \right\}$

Update set of active arms: $\tilde{A}_{h+1} = \mathcal{A}^\star(\tilde{\Theta}_{h+1}) \cap \tilde{A}_h$

Decrease removal threshold: $\tilde{\Gamma}_{h+1} \leftarrow \frac{\tilde{\Gamma}_h}{2}$

End

Structured Arm Elimination

Our structured arm elimination (SAE) strategy (Algorithm 6) is a phased algorithm inspired by Improved UCB (Auer and Ortner, 2010). In each phase h, the algorithm keeps a confidence set $\tilde{\Theta}_h$ of the same form as (9.1). More precisely, the confidence set $\tilde{\Theta}_h$ contains the models such that the mean of each arm $a \in \mathcal{A}$ does not deviate too much from the empirical one $\hat{\mu}_{h-1}(a)$ according to its number of pulls $N_{h-1}(a)$, both computed at the end of the previous phase. Then, all active arms (i.e., those that are optimal for at least one of the models in the confidence set) are played until a well-chosen count is reached. Such count is computed to ensure that all models that are sufficiently distant from the target $\theta^\star$ (according to an exponentially-decaying removal threshold $\tilde{\Gamma}_h$) are discarded from the confidence set. Once all the models in which a certain arm $a \in \mathcal{A}$ is optimal have been eliminated, a is labeled as inactive and no longer pulled. Algorithm 6 can be applied to any set of models (e.g., not only finite ones) as far as we can determine the set of optimal arms at each step. This is an optimization problem that can be solved efficiently for, e.g., linear, piecewise-linear, and convex structures, while it becomes intractable in general.

Note that SAE is not an optimistic algorithm since it might pull arms that are never optimistic with respect to $\theta^\star$. This property is due to the phased nature of the algorithm, such that no *optimistic bias* in selecting the active arms is used, unlike in SUCB. While in unstructured problems SUCB and SAE reduce to UCB and improved UCB, respectively, and have similar regret guarantees (i.e., each arm is pulled roughly the same amount of times in the two algorithms), in structured problems they may behave very differently, as we shall see in the next examples.

Examples. Figure 9.1 presents two simple structures in which SUCB and SAE significantly differ. The model set is divided in different regions. Since all bandits in the same region have, for the purpose of our discussion, the same properties, we call θ_1 any model in the first part, θ_2 any model in the second, and so on. Note that the following comments hold for an ideal realization in which certain high-probability events occur.

In the structure of Figure 9.1*(left)*, arm 2 is never optimistic since its mean is always below the value of the optimal arm $\mu_{\theta_1}(a_1)$. Therefore, SUCB never pulls it and needs

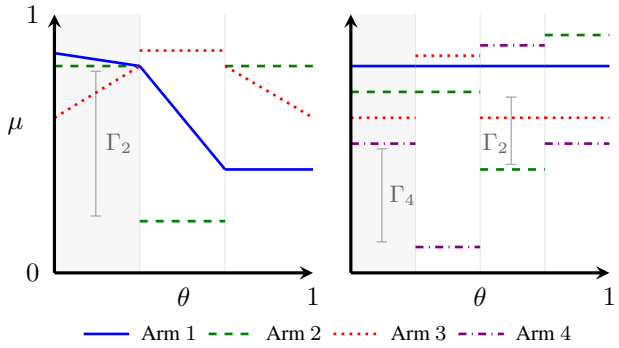

Figure 9.1: *Two structures in which SUCB and SAE significantly differ. The true model is any in the shaded region.* (left) *SUCB never pulls an informative arm.* (right) *SUCB discards an informative arm too early.*

only to discard the optimistic arm 3. This, in turn, takes $\mathcal{O}(1/\Gamma_3^2(\theta_1, \theta_2))$ pulls of such arm, which can be rather large. Since SAE pulls also arm 2, the large gap $\Gamma_2(\theta_1, \theta_2)$ (Γ_2 in the figure) allows to discard arm 3 much sooner. From the definition of the algorithm, SAE also needs to discard arm 2. Once again, this can be done quickly due to the large gap $\Gamma_1(\theta_1, \theta_3)$ and the fact that the optimal arm 1 is always pulled.

In the structure of Figure 9.1*(right)*, the optimistic bias makes SUCB pull the arms starting from the one with the highest value, arm 2, downwards to the optimal one, arm 1. Since the gap $\Gamma_2(\theta_1, \theta_3)$ (Γ_2 in the figure) is larger than $\Gamma_2(\theta_1, \theta_4)$, SUCB implicitly discards θ_3, and so arm 4, before arm 2. Thus, once both these arms have been eliminated, the algorithm takes $\mathcal{O}(1/\Gamma_3^2(\theta_1, \theta_2))$ pulls of arm 3 to discard the arm itself. By simultaneously pulling all four arms, SAE discards arm 3 first using the pulls of arm 4 (the one prematurely discarded by SUCB) due to the large gap $\Gamma_4(\theta_1, \theta_2)$ (Γ_4 in the figure). Finally, the deletion of the remaining two sub-optimal arms occurs with the same number of pulls as SUCB, and it can be verified that the overall regret is much smaller.

Regret Analysis

We derive a bound to the regret of SAE, the main result of this chapter.

Definitions. Let us start by defining the "good" event under which the true model $\theta^\star$ is never discarded from the confidence set. Formally, let $E := \{\forall h = 0, \dots, \lceil \log_2 n \rceil : E_h \text{ holds}\}$, with E_h denoting the following event:

$$E_h := \left\{ \forall a \in \mathcal{A} : |\hat{\mu}_{h-1}(a) - \mu_{\theta^\star}(a)| < \frac{1}{\beta} \sqrt{\frac{\alpha \log n}{N_{h-1}(a)}} \right\}.$$

In order to carry out our analysis, we need to characterize the arms pulled in each phase, which are specified by the sets of active arms $\left\{\tilde{\mathcal{A}}_h\right\}_{h \geq 0}$. Since these sets are random quantities, we cannot study them directly. Instead, we introduce a deterministic sequence of active arm sets $\{\mathcal{A}_h\}_{h \geq 0}$ that effectively works as a proxy for $\left\{\tilde{\mathcal{A}}_h\right\}_{h \geq 0}$ and, under the

good event E, allows us to define how many samples are needed for arms to be discarded. Formally, we define the following three sequences of sets of arms, whose definitions depend on each other. For each $h \geq 0$, we define

$$\mathcal{A}_{h+1} := \mathcal{A}_h \setminus \bar{\mathcal{A}}_h, \tag{9.5}$$

$$\bar{\mathcal{A}}_h := \left\{ a \in \mathcal{A}_h \mid \tilde{\Gamma}_h \leq \inf_{\theta \in \Theta^\star_a} \max_{a' \in \underline{\mathcal{A}}_h \cup \{a\}} \Gamma_{a'}(\theta, \theta^\star) \right\}, \tag{9.6}$$

$$\underline{\mathcal{A}}_{h+1} := \left\{ a \in \mathcal{A}_{h+1} \mid \tilde{\Gamma}_h > k_\beta \inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma_{a'}(\theta, \theta^\star)}{2^{[h - \bar{h}_{a'}]_+}} \right\}. \tag{9.7}$$

with $\mathcal{A}_0 = \underline{\mathcal{A}}_0 = \mathcal{A}^\star(\Theta)$. The constant k_β is $k_\beta := \frac{1}{\beta - 1}\sqrt{(\beta+1)^2 + \frac{1}{\log n}}$ and $\bar{h}_{a'} := \max_{h \in \mathbb{N}^+}\{h \mid a' \in \mathcal{A}_h\}$ is the last phase in which arm a' is active in our deterministic sequence $\{\mathcal{A}_h\}_{h \geq 0}$. These sets play a fundamental role in our analysis. In particular, under the good event E, $\bar{\mathcal{A}}_h$ is such that each arm $a \in \bar{\mathcal{A}}_h$ is *certainly eliminated* at the end of phase h. Similarly, $\underline{\mathcal{A}}_h$ is such that each arm $a \in \underline{\mathcal{A}}_h$ is *certainly active* in phase h ($a \in \tilde{\mathcal{A}}_h$). Finally, $\mathcal{A}_h$ contains all those arms that are *potentially active* in phase h (that is, we cannot guarantee that they have been eliminated before). The reader might be wondering why we need to explicitly maintain and distinguish the set of potentially active and certainly active arms. As we shall see, if we can prove that certain arms are pulled in a certain phase (hence the sets $\underline{\mathcal{A}}_h$), we can also show that the algorithm uses their *information* (i.e., their model gaps) to discard certain other arms/models faster. Hence, one of the key novelties, and complications, in our analysis is that we do not only care about proving that sub-optimal arms are not pulled after certain phases, but also about guaranteeing that some arms are not discarded too early since their pulls might allow to discard other models/arms. The parameter β plays an important role for this purpose. In particular, $\underline{\mathcal{A}}_h$ is essentially the set of arms for which the number of pulls to the active arms at the previous phase is below the removal threshold by a *margin* defined by k_β. For example, for large n, setting $\beta = 3$ yields $k_\beta \simeq 2$, which in turn implies that $\underline{\mathcal{A}}_h$ is the set of arms such that $\tilde{\Gamma}_h > \inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma_{a'}(\theta, \theta^\star)}{2^{[h - \bar{h}_{a'} - 1]_+}}$. This, as we shall prove shortly, is close to saying that all the arms that are not certainly eliminated at the end of phase h are also active in such phase.

Analysis. We begin our analysis by showing that, with high probability, the true model $\theta^\star$ is always contained in the confidence set by a certain margin (which depends on β). Unlike previous works, we need this to guarantee that sub-optimal arms are not eliminated too early. The proof of the following Lemma is quite standard and is thus deferred to Appendix C.

Lemma 9.3.1. *Let $\alpha > 0$, $\beta \geq 1$, and E be the good event defined above. Then, the probability that E does not hold can be upper bounded by*

$$\mathbb{P}\{E^c\} \leq |\mathcal{A}^\star(\Theta)| n^{-2\frac{\alpha}{\beta^2}} (\log_2 n + 2)^2.$$

In the next results, we shall assume that the "good" event E holds. We now show a sufficient condition for eliminating a model from the confidence set.

Lemma 9.3.2. *Suppose there exist an arm $a \in \mathcal{A}$, a model $\theta \in \Theta$, and a phase $h \geq 0$ such that $N_h(a) \geq \left(1 + \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{\Gamma_a^2(\theta, \theta^\star)}$. Then, under event E, $\theta \notin \tilde{\Theta}_{h'}$ for all $h' > h$.*

Proof. Suppose there exists a phase $h' > h$ such that $\theta \in \tilde{\Theta}_{h'}$. Then,

$$\Gamma_a(\theta, \theta^\star) = |\mu_\theta(a) - \mu_{\theta^\star}(a)| \overset{(a)}{\leq} |\mu_\theta(a) - \hat{\mu}_{h'}(a)| + |\hat{\mu}_{h'}(a) - \mu_{\theta^\star}(a)|$$
$$\overset{(b)}{<} \left(1 + \frac{1}{\beta}\right) \sqrt{\frac{\alpha \log n}{N_{h'-1}(a)}} \overset{(c)}{\leq} \left(1 + \frac{1}{\beta}\right) \sqrt{\frac{\alpha \log n}{N_h(a)}},$$

where (a) follows from the triangle inequality, (b) from the fact that θ is in the confidence set and E holds, and (c) from $h' > h$ and the monotonicity of the number of pulls. Therefore, it must be that

$$N_h(a) < \left(1 + \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{\Gamma_a^2(\theta, \theta^\star)},$$

which is a contradiction. Thus, we must have $\theta \notin \tilde{\Theta}_{h'}$. □

Lemma 11.4.2 justifies our choice of the number of pulls in each phase: if $a \in \mathcal{A}$ is active in phase h and $\Gamma_a(\theta, \theta^\star) \geq \tilde{\Gamma}_h$, then θ is eliminated at the end of the phase. The following result is an immediate consequence of Lemma 11.4.2. It shows a condition on the number of pulls such that, under the 'good' event E, an arm is discarded.

Lemma 9.3.3. *Let $h \geq 0$, $a \in \mathcal{A}$, and suppose that, for any model $\theta \in \Theta_a^\star$ there exists an arm $a' \in \mathcal{A}$ such that $N_h(a') \geq \left(1 + \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{\Gamma_{a'}^2(\theta, \theta^\star)}$. Then, under event E, $i \notin \tilde{\mathcal{A}}_{h'}$ for all $h' > h$.*

Proof. All models with a as optimal arm are discarded in phase h by Lemma 11.4.2. Therefore, $\forall \theta \in \Theta_a^\star : \theta \notin \tilde{\Theta}_{h+1}$, which also implies that $a \notin \tilde{\mathcal{A}}_{h'}$ for all $h' > h$. □

The interesting consequence of Lemma 9.3.3 is that, thanks to the structure, it is enough that one active arm $a' \in \mathcal{A}$ is pulled sufficiently often to eliminate a potentially *different* arm $a \in \mathcal{A}$.

While so far we focused on model/arm elimination, we now show a somewhat opposite result: when all arms have not been pulled too much, some models can be guaranteed to lie in the confidence set.

Lemma 9.3.4. *Let $h \geq 0$, $\theta \in \Theta$, and suppose $N_h(a) \leq \left(1 - \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{\Gamma_a^2(\theta, \theta^\star)}$ for all arms $i \in \mathcal{A}$. Then, under event E, $\theta \in \tilde{\Theta}_{h+1}$.*

Proof. Notice that, for all arms $a \in \mathcal{A}$, $\Gamma_a(\theta, \theta^\star) \leq \left(1 - \frac{1}{\beta}\right) \sqrt{\frac{\alpha \log n}{N_h(a)}}$. Therefore,

$$|\hat{\mu}_h(a) - \mu_\theta(a)| \overset{(a)}{\leq} |\hat{\mu}_h(a) - \mu_{\theta^\star}(a)| + |\mu_{\theta^\star}(a) - \mu_\theta(a)| = |\hat{\mu}_h(a) - \mu_{\theta^\star}(a)| + \Gamma_a(\theta, \theta^\star)$$
$$\overset{(b)}{<} \frac{1}{\beta} \sqrt{\frac{\alpha \log n}{N_h(a)}} + \Gamma_a(\theta, \theta^\star) \overset{(c)}{\leq} \sqrt{\frac{\alpha \log n}{N_h(a)}},$$

where (a) follows from the triangle inequality, (b) from the fact that E holds, and (c) from the condition on the number of pulls above. This implies that $\theta \in \tilde{\Theta}_{h+1}$. □

While it is quite strange to ensure that a model is not discarded, we shall make extensive use of this result. In fact, the presence of a model in the confidence set guarantees that its optimal arm is pulled. Consequently, by the previous lemmas, knowing what arms are pulled allows us to characterize what models/arms are discarded. Using this result, w know show a condition on $\tilde{\Gamma}_{h-1}$ under which a model $\theta \neq \theta^\star$ can be guaranteed to belong to $\tilde{\Theta}_h$.

Lemma 9.3.5. *Let $h \geq 1$, $\theta \in \Theta$, and $\alpha \geq \beta^2$. For all $a \in \mathcal{A}^\star(\Theta)$, let $\tilde{h}_a \leq h-1$ be such that either $a \notin \tilde{A}_{\tilde{h}_a+1}$ or $\tilde{h}_a = h-1$. Suppose the following condition holds*

$$\tilde{\Gamma}_{h-1} \geq k_\beta \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma_{a'}(\theta, \theta^\star)}{2^{h-\tilde{h}_{a'}-1}}, \tag{9.8}$$

where $k_\beta := \frac{1}{\beta-1}\sqrt{(\beta+1)^2 + \frac{1}{\log n}}$. Then, under event E, $\theta \in \tilde{\Theta}_h$.

Proof. Fix any arm $a \in \mathcal{A}^\star(\Theta)$. By assumption a is pulled at most in phase $\tilde{h}_a$. Therefore, its number of pulls at the end of phase $h-1$ can be bounded by

$$\begin{aligned} N_{t-1}(a) &= \left\lceil \frac{\alpha \log n}{\tilde{\Gamma}^2_{\tilde{h}_a}} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil = \left\lceil \frac{\alpha \log n}{4^{h-\tilde{h}_a-1}\tilde{\Gamma}^2_{h-1}} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \\ &\leq \frac{\alpha \log n}{4^{h-\tilde{h}_a-1}\tilde{\Gamma}^2_{h-1}} \left(1 + \frac{1}{\beta}\right)^2 + 1, \end{aligned}$$

where the second equality is from $\tilde{\Gamma}_{\tilde{h}_a} = \frac{1}{2^{\tilde{h}_a}} = \frac{2^{h-1}}{2^{\tilde{h}_a} 2^{h-1}} = 2^{h-\tilde{h}_a-1}\tilde{\Gamma}_{h-1}$. The constant term can be upper bounded by

$$1 = \frac{(\beta+1)^2 \log n}{(\beta+1)^2 \log n} \overset{(a)}{\leq} \frac{\alpha(\beta+1)^2 \log n}{\beta^2(\beta+1)^2 \log n} 4^{\tilde{h}_a} \frac{4^{h-1}}{4^{h-1}} \overset{(b)}{=} \frac{1}{(\beta+1)^2 \log n} \frac{\alpha \log n}{4^{h-\tilde{h}_a-1}\tilde{\Gamma}^2_{h-1}} \left(1 + \frac{1}{\beta}\right)^2,$$

where (a) follows from $\alpha \geq \beta^2$ and (b) from the definition of $\tilde{\Gamma}_{h-1}$. Hence,

$$\begin{aligned} N_{t-1}(a) &\overset{(a)}{\leq} \left(1 + \frac{1}{(\beta+1)^2 \log n}\right) \frac{\alpha \log n}{4^{h-\tilde{h}_a-1}\tilde{\Gamma}^2_{h-1}} \left(1 + \frac{1}{\beta}\right)^2 \\ &\overset{(b)}{\leq} \left(1 - \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{4^{h-\tilde{h}_a-1} \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma^2_{a'}(\theta,\theta^\star)}{4^{h-\tilde{h}_{a'}-1}}} \leq \left(1 - \frac{1}{\beta}\right)^2 \frac{\alpha \log n}{\Gamma^2_a(\theta, \theta^\star)}, \end{aligned}$$

where in (a) we applied the two inequalities derived above and in (b) we used the condition (9.8) on $\tilde{\Gamma}_{h-1}$. This argument can be repeated for all other arms in $\mathcal{A}^\star(\Theta)$. Therefore, Lemma 9.3.4 together with the fact that arms not in $\mathcal{A}^\star(\Theta)$ are never pulled, implies $\theta \in \tilde{\Theta}_h$. □

The following theorem is the key result that will be used to prove the final regret bound. It shows that the sets $\bar{\mathcal{A}}_h$ and $\underline{\mathcal{A}}_h$ defined in Section 9.3 have the intended meaning. That is, under the good event E, all arms in $\bar{\mathcal{A}}_h$ are certainly eliminated at the end of phase h, while all arms in $\underline{\mathcal{A}}_h$ are certainly pulled in the same phase.

Theorem 9.3.1. *Let $\beta \geq 1$ and $\alpha = \beta^2$. Then, under event E, the following two statements are true for all $h \geq 0$:*

$$\forall a \in \bar{\mathcal{A}}_h : a \notin \tilde{\mathcal{A}}_{h'} \; \forall h' > h, \tag{9.9}$$

$$\forall a \in \underline{\mathcal{A}}_h : a \in \tilde{\mathcal{A}}_h. \tag{9.10}$$

Proof. We prove the theorem by induction on h.

1) Base case ($h = 0, 1$). We show both $h = 0$ and $h = 1$ as base cases since the recursive definition of the sets $\underline{\mathcal{A}}_h$ starts from $h = 1$ and depends on $\bar{\mathcal{A}}_h$. The recursive definition of the latter, on the other hand, starts from $h = 0$.

1.1) First phase ($h = 0$). Since $\tilde{\mathcal{A}}_0 = \mathcal{A}^\star(\Theta)$ by the initialization step of Algorithm 6, (9.10) trivially holds. If $\bar{\mathcal{A}}_0$ is empty, (9.9) trivially holds as well. Suppose $\bar{\mathcal{A}}_0$ is not empty and fix any arm $a \in \bar{\mathcal{A}}_0$. For all arms $a' \in \mathcal{A}^\star(\Theta)$,

$$N_0(a') \overset{(a)}{=} \left\lceil \frac{\alpha \log n}{\tilde{\Gamma}_0^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \overset{(b)}{\geq} \left\lceil \frac{\alpha \log n}{\inf_{\theta \in \Theta_a^\star} \max_{l \in \mathcal{A}^\star(\Theta)} \Gamma_l^2(\theta, \theta^\star)} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil,$$

where (a) is from the number of pulls in Algorithm 6 and the fact that all arms in $\mathcal{A}^\star(\Theta)$ are active, and (b) follows from the definition of $\bar{\mathcal{A}}_0$. Therefore, for all $\theta \in \Theta_a^\star$ there exists some arm $a' \in \mathcal{A}^\star(\Theta)$ whose number of pulls at the end of phase 0 is at least

$$N_0(a') \geq \left\lceil \frac{\alpha \log n}{\Gamma_{a'}^2(\theta, \theta^\star)} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil.$$

Hence, Lemma 9.3.3 ensures that $a \notin \tilde{\mathcal{A}}_{h'}$ for all $h' > 0$, which in turn implies that (9.9) holds.

1.2) Second phase ($h = 1$). Let us start from (9.10). Take any arm $a \in \mathcal{A}_1 := \mathcal{A}^\star(\Theta) \setminus \bar{\mathcal{A}}_0$ and suppose

$$\tilde{\Gamma}_0 > k_\beta \inf_{\theta \in \Theta_i^\star} \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma_{a'}(\theta, \theta^\star)}{2^{\max\{-\bar{h}_{a'}, 0\}}} \tag{9.11}$$

holds. Since $2^{\max\{-\bar{h}_{a'}, 0\}} = 1$ for all $a' \in \mathcal{A}^\star(\Theta)$, (9.11) implies that there exists some model $\bar{\theta} \in \Theta_a^\star$ such that $\tilde{\Gamma}_0 \geq k_\beta \max_{a' \in \mathcal{A}^\star(\Theta)} \Gamma_{a'}(\bar{\theta}, \theta^\star)$. Thus, we can directly apply Lemma 9.3.5 using $\tilde{h}_{a'} = 0$ for all $a' \in \mathcal{A}^\star(\Theta)$ and obtain $\bar{\theta} \in \tilde{\Theta}_1$. This implies $a \in \tilde{\mathcal{A}}_1$, from which (9.10) holds.

The proof of (9.9) proceeds similarly as for $h = 0$. Take any arm $a \in \bar{\mathcal{A}}_1$ (assuming the set is not empty). We have just proved that all arms $a' \in \underline{\mathcal{A}}_1$ are pulled in phase $h = 1$. If arm a has already been removed, (9.9) trivially holds. Hence, we can safely assume that $a \in \tilde{\mathcal{A}}_1$. Therefore, arms in $\underline{\mathcal{A}}_1 \cup \{a\}$ are active and the number of pulls is sufficient to apply Lemma 9.3.3, which implies (9.9).

2) Inductive step ($h > 1$). Now assume the two statements hold for $h' = 0, 1, \ldots, h-1$. This implies, in particular, that an arm $a \in \bar{\mathcal{A}}_{h'}$, $h' \leq h - 1$, is not pulled after h'. Once again, take any arm $a \in \underline{\mathcal{A}}_h$. The definition of $\underline{\mathcal{A}}_h$ implies

$$\tilde{\Gamma}_{h-1} \geq k_\beta \max_{a' \in \mathcal{A}^\star(\Theta)} \frac{\Gamma_{a'}(\bar{\theta}, \theta^\star)}{2^{\max\{h - \bar{h}_{a'} - 1, 0\}}}$$

for some $\bar{\theta} \in \Theta_a^\star$. Notice that, by the inductive assumption, all arms $a' \in \mathcal{A}^\star(\Theta) \setminus \mathcal{A}_h$ are not pulled after $\bar{h}_{a'} \leq h - 1$. On the other hand, for all arms $a' \in \mathcal{A}_h$, it must be that $\bar{h}_{a'} \geq h$. Thus, we can apply Lemma 9.3.5 by setting $\tilde{h}_{a'} = \bar{h}_{a'}$ for arms $a' \in \mathcal{A}^\star(\Theta) \setminus \mathcal{A}_h$ and $\tilde{h}_{a'} = h - 1$ for arms $a' \in \mathcal{A}_h$. Hence, $\bar{\theta} \in \tilde{\Theta}_h$ and (9.10) holds. Finally, since all arms in $\underline{\mathcal{A}}_h$ are pulled in phase h, we can show that (9.9) holds using exactly the same argument as for the second base case ($h = 1$). □

Main result. We are now ready to state our main result. It shows that the regret incurred by SAE for pulling a sub-optimal arm $a \in \mathcal{A}$ is inversely proportional to the maximum model-gap (taken over the set of arms that are active when arm a is discarded) with respect to the hardest model to be eliminated in $\Theta_a^\star$.

Theorem 9.3.2. *Let $\beta \geq 1$, $\alpha = \beta^2$, $n \geq 64$, and $c_\beta := 4(1+\beta^2)$. Then,*

$$R_n^{SAE}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \in \mathcal{A}^\star(\Theta) \setminus \{a^\star_{\theta^\star}\}} \frac{4(1+\beta^2)\Delta_{\theta^\star}(a)\log n}{\inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star_a} \Gamma_{a'}(\theta, \theta^\star)^2} + 2|\mathcal{A}^\star(\Theta)|,$$

where $\mathcal{A}^\star_a = \underline{\mathcal{A}}_{\bar{h}_a} \cup \{a\}$ is the set of arms that are certainly active in the last phase when a is active.

Proof. The expected regret can be written as

$$R_n \overset{(a)}{\leq} \sum_{t=1}^{n} \mathbb{E}\left[\Delta_{\theta^\star}(A_t)|E\right] + n\mathbb{P}\left\{E^c\right\} \overset{(b)}{=} \sum_{a \in \mathcal{A}} \Delta_{\theta^\star}(a)\mathbb{E}\left[N_n(a)|E\right] + n\mathbb{P}\left\{E^c\right\},$$

where in (a) we upper bounded the gaps by 1 and used $\mathbb{E}\left[\mathbb{1}\left\{E^c\right\}\right] = \mathbb{P}\left\{E^C\right\}$, while in (b) we used the standard rewriting in terms of the number of pulls. We now upper bound the expected number of pulls of each sub-optimal arm a when conditioned on event E. Since $a \in \bar{\mathcal{A}}_{\bar{h}_a}$, Theorem 9.3.1 ensures that arm a is not pulled after phase $\bar{h}_a$. Hence,

$$\begin{aligned} N_n(a) &\overset{(a)}{\leq} \left\lceil \frac{\alpha \log n}{\tilde{\Gamma}^2_{\bar{h}_a}} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \overset{(b)}{=} \left\lceil \frac{4\alpha \log n}{\tilde{\Gamma}^2_{\bar{h}_a - 1}} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \\ &\overset{(c)}{\leq} \left\lceil \frac{4(1+\beta^2)\log n}{\inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star_a} \Gamma^2_{a'}(\theta, \theta^\star)} \right\rceil, \end{aligned}$$

where (a) follows immediately from Theorem 9.3.1 and Algorithm 6, while (b) from $\tilde{\Gamma}_{\bar{h}_a} = \frac{\tilde{\Gamma}_{\bar{h}_a - 1}}{2}$. To show (c), notice that $\tilde{\Gamma}_{\bar{h}_a - 1} > \inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star_a} \Gamma_{a'}(\theta, \theta^\star)$ from the definition of $\bar{\mathcal{A}}_{\bar{h}_a}$ (if this did not hold, arm a would be eliminated in phase $\bar{h}_a - 1$ since $\underline{\mathcal{A}}_{\bar{h}_a} \subseteq \underline{\mathcal{A}}_{\bar{h}_a - 1}$). Therefore, the regret conditioned on event E can be upper bound by

$$\sum_{a \in \mathcal{A}^\star(\Theta)} \frac{4(1+\beta^2)\Delta_{\theta^\star}(a)\log n}{\inf_{\theta \in \Theta^\star_a} \max_{a' \in \mathcal{A}^\star_a} \Gamma^2_{a'}(\theta, \theta^\star)} + |\mathcal{A}^\star(\Theta)|,$$

where we used $\lceil x \rceil \leq x + 1$ and $\sum_{a \in \mathcal{A}^\star(\Theta)} \Delta_{\theta^\star}(a) \leq |\mathcal{A}^\star(\Theta)|$.

Let us now consider the probability of E not holding. Using Lemma 9.3.1 with $\alpha = \beta^2$, together with $(\log_2 n + 2)^2 \leq n$ for $n \geq 64$, we obtain

$$n\mathbb{P}\left\{E^c\right\} \leq |\mathcal{A}^\star(\Theta)| \frac{(\log_2 n + 2)^2}{n} \leq |\mathcal{A}^\star(\Theta)|,$$

which, combined with the previous bound, concludes the proof.

□

Discussion. First, as a sanity check, we verify that the regret bound of Theorem 9.3.2 is never worse than the one of UCB. That is, SAE is never negatively affected by the knowledge of the structure and, whenever applied to unstructured problems, the algorithm is, apart from multiplicative/additive constants, finite-time optimal.

Proposition 9.3.1. *The SAE algorithm is always sub-UCB, in the sense that there exist constants $c, c' > 0$ such that its regret satisfies*

$$R_n^{SAE}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \in \mathcal{A} \setminus \{a^\star_{\theta^\star}\}} \frac{c \log n}{\Delta_{\theta^\star}(a)} + c'.$$

The proof of this proposition can be found in Appendix C. The key property of Theorem 9.3.2 is that the regret suffered for discarding a sub-optimal arm a does not necessarily scale with the model gaps of such arm (i.e., $\Gamma_a(\theta, \theta^\star)$ for $\theta \in \Theta_a^\star$) but with those of the *most effective* arm in $\mathcal{A}_a^\star$. Compared to SUCB, in which the elimination of a model $\theta \in \Theta_a^\star$ requires $\mathcal{O}(1/\Gamma_a^2(\theta, \theta^\star))$ pulls of arm a, SAE needs only $\mathcal{O}(1/\max_{a' \in \mathcal{A}_a^\star} \Gamma_{a'}^2(\theta, \theta^\star))$, which is by definition always smaller. Note that, to be precise, SUCB can potentially eliminate models using the pulls of any arm since the confidence sets are built as in SAE. However, in general, it is not possible to prove the same regret bound since the optimism induces a specific pull order that might prevent the algorithm from choosing the arm with the largest model gap. Obviously, SAE does not know this arm in advance and, therefore, ensures it is pulled by choosing all active arms. However, the additional regret incurred to achieve this property can make the algorithm, in some cases, worse than SUCB. In fact, a key difference is that SUCB stops playing a sub-optimal arm a when all optimistic models in Θ_a^+ are discarded (see Theorem 9.2.1), while SAE needs to eliminate all models in which arm a is optimal (even non-optimistic ones). Therefore, although SAE improves the elimination of all optimistic models, it suffers further regret for discarding non-optimistic ones and, in general, the two algorithms are not comparable. A special case are those structures in which the hardest models to be eliminated for each arm a are in the optimistic set, in which SAE provably improves over SUCB (proof in Appendix C).

Proposition 9.3.2. *If Θ is such that, for each sub-optimal $a \in \mathcal{A}^\star(\Theta)$,*

$$\inf_{\theta \in \Theta_a^\star} \max_{a' \in \mathcal{A}_a^\star} \Gamma_{a'}(\theta, \theta^\star)^2 = \inf_{\theta \in \Theta_a^+} \max_{a' \in \mathcal{A}_a^\star} \Gamma_{a'}(\theta, \theta^\star)^2, \tag{9.12}$$

SAE is sub-SUCB, in the sense that its regret can be upper bounded by the one of Theorem 9.2.1.

Anytime SAE and Constant Regret

Algorithm 6 cannot be applied whenever the horizon n is unknown, as the length of each phase explicitly depends on it. This has the additional drawback of preventing constant regret from being achieved since a $\log n$ term naturally appears in the resulting bound. As shown by Lattimore and Munos (2014), there exist structures in which constant regret can be obtained and it would be desirable for our strategy to exploit this fact. We, therefore, propose an anytime extension (Algorithm 7). The idea is once again similar to the one by Auer and Ortner (2010): we split the horizon into different *periods* with exponentially increasing length. Therefore, in Algorithm 7, and throughout this section, we overload our notation by adding a superscript k to denote the period of each period-dependent quantity. The key property is that our approach does not reset in each period (as Auer and Ortner (2010) do) but retains the last confidence sets. Though this makes the proofs more involved, we shall see that it allows us to guarantee a constant regret. One can see the analogy between our non-resetting phased approach and the standard way of handling unknown horizons in online algorithms. In the latter case, we typically replace $\log n$ with $\log t$ in the confidence sets, while here we do the same with $\log \tilde{n}_k$. Then, after proving that certain high-probability events occur at each time/period, we can carry out the proofs without forcing any reset. Due to the additional complications introduced by the anytime extension (in particular, controlling the sets $\underline{A}_h$), we were able to prove only a weaker

Algorithm 7 Anytime SAE (ASAE)

Require: Set of parameters Θ, hypothesis space $\mathfrak{M}_\Theta$, scalars $\alpha > 0, \beta \geq 1, \eta > 0$

1: **Initialization**: $\tilde{n}_0 \leftarrow 2$, $\tilde{\Theta}^{-1} \leftarrow \Theta$
2: **Foreach period** $k = 0, 1, \dots$ **do**
3: Initialize confidence sets: $\tilde{\Theta}_0^k \leftarrow \tilde{\Theta}^{k-1}$, $\tilde{\mathcal{A}}_0^k \leftarrow \mathcal{A}^\star(\tilde{\Theta}_0^k)$
4: Run Algorithm 6 with $n = \tilde{n}_k$, $\tilde{\Theta}_0 = \tilde{\Theta}_0^k$, and $\tilde{\mathcal{A}}_0 = \tilde{\mathcal{A}}_0^k$
5: Update horizon: $\tilde{n}_{k+1} \leftarrow \tilde{n}_k^{1+\eta}$
6: **End**

bound than the one in Theorem 9.3.2 which, however, retains the same benefits. Since the proof heavily relies on the one of Therem 9.3.2, we report it in Appendix C.

Theorem 9.4.1. *Let $\eta = 1$, $\alpha = 2$, and $\beta = 1$. Then,*

$$R_n^{ASAE}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \in \mathcal{A}^\star \setminus \{a_{\theta^\star}^\star\}} \frac{192 \Delta_{\theta^\star}(a) \log n}{\inf_{\theta \in \Theta_a^\star} \max_{a' \in \{a, a_{\theta^\star}^\star\}} \Gamma_{a'}(\theta, \theta^\star)^2} + 6|\mathcal{A}^\star(\Theta)|.$$

The new bound has the same form as the one of Algorithm 6, except for the fact that the set of active arms for eliminating each $a \in \mathcal{A}$ is reduced to $\{a, a_{\theta^\star}^\star\} \subseteq \mathcal{A}_a^\star$. Note, however, that the presence of these two arms is enough to prove Proposition 9.3.1 and 9.3.2.

Remark 9.4.1. *Algorithm 7 is sub-UCB and, under the same conditions as in Proposition 9.3.2, is also sub-SUCB.*

We now prove a constant-regret bound for Algorithm 7. We need the following assumption from (Lattimore and Munos, 2014), which was proven both necessary and sufficient to achieve constant regret.

Assumption 9.4.1 (Informative optimal arm)**.** *Let $a^\star = a_{\theta^\star}^\star$ be the optimal arm of the true bandit $\theta^\star$. The structure Θ satisfies*

$$\Gamma_\star := \inf_{\theta \in \Theta \setminus \Theta_{a^\star}^\star} \Gamma_{a^\star}(\theta, \theta^\star) > 0.$$

In words, when a model is $\Gamma^\star$-distant (or less) in arm $a^\star$ from $\theta^\star$, its optimal arm is still $a^\star$. Therefore, pulling $a^\star$ eventually discards all sub-optimal arms. This is fundamental to guarantee that, after the algorithm has pulled $a^\star$ a sufficient number of times, no sub-optimal arm can become active again due to the increasing period length (hence we choose $a^\star$ forever).

Theorem 9.4.2. *Let $\eta = 1$, $\alpha = \frac{5}{2}$, $\beta = 1$, $\bar{t} := \frac{20|\mathcal{A}^\star(\Theta)| \log 2}{\Gamma_\star^2} + 2|\mathcal{A}^\star(\Theta)|$, and suppose Assumption 9.4.1 holds. Then,*

$$R_n^{ASAE}(\theta^\star, \mathfrak{E}_\Theta) \leq \sum_{a \in \mathcal{A}^\star \setminus \{a^\star\}} \frac{480 \Delta_{\theta^\star}(a) \log \bar{t}}{\inf_{\theta \in \Theta_a^\star} \max_{a' \in \{a, a_{\theta^\star}^\star\}} \Gamma_{a'}(\theta, \theta^\star)^2} + 9|\mathcal{A}^\star(\Theta)|.$$

Once again, the proof can be found in Appendix C. This bound improves over the one shown by Lattimore and Munos (2014) for SUCB in its dependence on $\bar{t}$, which can be

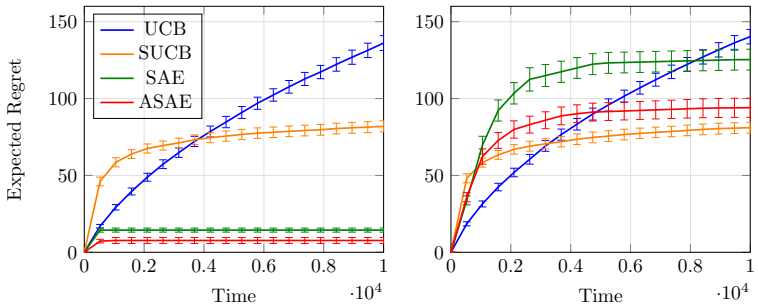

Figure 9.2: *Expected regret in the structured problem of Figure 9.1*(left). *(left) SUCB never pulls an informative arm and is outperformed by SAE. (right) The same structure with non-informative arm* 2*: SAE pulls a useless arm and performs worse than SUCB.*

understood as the time at which the algorithm transitions to the constant regret regime. While Lattimore and Munos (2014) proved $\bar{t} \simeq \mathcal{O}(\max\{1/\Gamma_\star^2, 1/\Delta_{\min}^2\})$, here we show that such time does not depend on the minimum gap $\Delta_{\min} = \min_{a:\Delta_{\theta^\star}(a)>0} \Delta_{\theta^\star}(a)$. This is intuitive since, by Assumption 9.4.1, $\mathcal{O}(1/\Gamma_\star^2)$ pulls of $a^\star$ should be enough to identify the optimal arm. Although the analysis of SUCB can be improved by replacing the minimum sub-optimality gap with the minimum model gap $\Gamma_{\min}^2$, it seems that this dependence is tight. As an example, consider a structure in which the optimal arm is very informative ($\Gamma_\star \gg 0$) but never optimistic. SUCB will never pull it until all optimistic models are discarded, which requires $\mathcal{O}(1/\Gamma_{\min}^2)$ steps in the worst case. Note that, whenever it is applied to structures satisfying Assumption 9.4.1, the bound of Theorem 9.4.1 does not show constant regret since the proof uses an implicit worst-case argument (i.e., Assumption 9.4.1 is assumed false).

Numerical Simulations

We perform two different classes of experiments. In the first one, we consider well-chosen structures that allow us to better understand the behavior of all algorithms. In the second one, we randomize the structures to provide a more general comparison. In all experiments, we run SAE and its anytime version (ASAE), SUCB, and UCB on Bernoulli bandits. We also compared to the WAGP algorithm of Atan et al. (2018), which however incurred linear regret in all our experiments (their assumptions never hold in our structures) and, therefore, is omitted from the plots. We use $\alpha = 2$ for all algorithms and $\beta = 1$ for SAE. Each plotted curve is the average of 100 independent runs with 95% Student's t confidence intervals.

Hand-coded Structures We first consider the structure of Figure 9.1*(left)*. We set $n = 10,000$ and $\eta = 0.1$. The results are shown in Figure 9.2(left). SUCB suffers a large regret for removing models in which arm 3 is optimal. On the other hand, SAE quickly discards these models by pulling arm 2, which, in turn, is eliminated by pulling arm 1. Hence the much lower regret, with the anytime version that performs slightly better. Notice also that Assumption 9.4.1 is verified and SAE obtains constant regret. SUCB eventually transitions

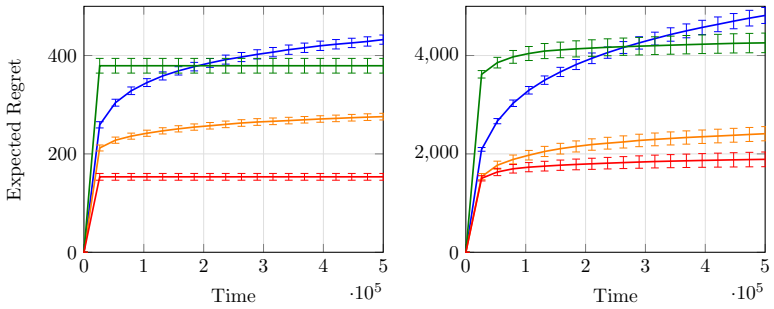

Figure 9.3: *(left) Expected regret on the structure of Figure 9.1*(right)*: SUCB discards and informative arm too early. (right) Randomly-generated structures with hard instances and informative arms: SAE, on average, leverages information to discard the hard instances.*

to constant regret too but needs a longer horizon. Alternatively, we can show an example where SUCB is expected to perform better. We modify the structure of Figure 9.1*(left)* to make arm 2 non-informative (i.e., we set its mean to the highest value in the figure for all models) and run the experiment under the same setting. Figure 9.2(right) shows that, as expected, SAE suffers from some additional regret for discarding the useless arm and performs worse than SUCB. However, it remains sub-UCB as proved in Section 9.3.

We now consider the structure of Figure 9.1*(right)*. We set $n = 500,000$, $\eta = 0.01$, and report the results in Figure 9.3(left). The arm ordering induced by SUCB (from the most optimistic to the optimal one) leads the algorithm to discard arm 4 before even pulling it once. Such arm, however, could be used to quickly discard arm 3, which is what SAE does. Notice that the larger regret of SAE with respect to its anytime counterpart is mainly due to the fact that phased procedures update the confidence sets much less than online approaches. This drawback is alleviated in the anytime version, which reduces the duration of some of these phases and retains good empirical performance.

Randomized Structures We now consider random structures. In each run, we first randomize a set of 100 models with 50 arms by drawing their means from the uniform distribution and we randomly choose the true model among them. Then, we build 50 additional 'hard' models by perturbing a random arm of the true model to become optimal and optimistic, and another random arm to become informative. In particular, the mean of the first random arm is set to $\mu^\star(\theta^\star) + 0.2\epsilon$, with $\epsilon \sim \mathcal{U}([0,1])$, while the second to $1/10$ of the original mean (so that we potentially get a larger model gap). The results are shown in Figure 9.3(right). Most of the regret suffered by SUCB is due to the hard instances we introduced. Some of them are likely to be eliminated by informative arms, but this is not always guaranteed by the SUCB strategy. Both versions of SAE, on the other hand, implicitly exploit these informative arms, with the anytime version outperforming all alternatives. Once again, the original version suffers a high initial regret due to the phases.

CHAPTER 10

Asymptotically Optimal Exploration in Linear Bandits

This chapter is based on the paper "An Asymptotically Optimal Primal-Dual Incremental Algorithm for Linear Contextual Bandits" co-authored with Matteo Pirotta, Marcello Restelli, and Alessandro Lazaric and published at NeurIPS 2020.

Introduction

In the last chapter, we focused on designing simple confidence-based strategies for *general* structures. These strategies enjoy provably good finite-time performance (notably, bounded regret whenever possible), they are easy to implement, and they are computationally efficient for most structures of interest. Unfortunately, they are in general not asymptotically optimal (except for very specific cases). While finite-time behavior is ultimately the performance measure of interest, the study of asymptotic optimality often yields intuition on how to build strategies that fully exploit the given structure to reduce regret. An increasingly popular technique to obtain structure-optimal strategies for regret minimization (Lattimore and Szepesvari, 2017; Combes et al., 2017; Hao et al., 2019; Degenne et al., 2020b) or pure exploration (Degenne et al., 2019, 2020a; Jedra and Proutiere, 2020; Zaki et al., 2020) is to derive algorithms from asymptotic problem-dependent lower bounds.

In this chapter, we follow this line of works for the specific case of *contextual* linear bandit setting (e.g., Lattimore and Szepesvári, 2020), where at each time step t the

learner observes a context X_t drawn i.i.d. from a context distribution ρ, pulls an arm A_t, and receives a reward Y_t drawn from a distribution whose expected value is a linear combination between the features $\phi(X_t, A_t)$ describing context and arm, and the unknown parameter $\theta^\star$. That is, differently from the standard MAB setting introduced in Section 2.6, here the learner observes an additional side information (the context) on which it can condition its decisions. This setting formalizes a wide range of problems such as online recommendation systems, clinical trials, dialogue systems, and many others (Bouneffouf and Rish, 2019). Popular algorithmic principles, such as optimism-in-face-of-uncertainty and Thompson sampling (Thompson, 1933), have been applied to this setting leading to algorithms such as OFUL Abbasi-Yadkori et al. (2011) and LINTS Agrawal and Goyal (2013); Abeille and Lazaric (2017) with strong finite-time worst-case regret guarantees. Unfortunately, as we already discussed in Chapter 8, these algorithms are not asymptotically optimal (in a problem-dependent sense) as they fail to adapt to the structure of the problem at hand (Lattimore and Szepesvari, 2017). In fact, in the CLB setting, the values of different arms are tightly connected through the linear assumption and a possibly suboptimal arm may provide a large amount of information about $\theta^\star$ and thus the optimal arm. Optimistic algorithms naturally discard suboptimal arms and thus may miss the chance to acquire information about $\theta^\star$ and significantly reduce the regret.

To the best of our knowledge, the recent work of Hao et al. (2019) is the only one to study asymptotic optimality in contextual linear bandits. The authors introduced OAM, an asymptotically optimal algorithm for this setting that takes inspiration from OSSB (Combes et al., 2017). While OAM effectively exploits the linear structure and outperforms other bandit algorithms, it suffers from major limitations. From an algorithmic point of view, similarly to OSSB (Combes et al., 2017), at each exploration step, OAM requires solving the optimization problem defined in the regret lower bound, which can hardly scale beyond problems with a handful of contexts and arms. Furthermore, OAM implements a forcing exploration strategy that often leads to long periods of linear regret and introduces a linear dependence on the number of arms. Finally, the regret analysis reveals a critical dependence on the inverse of the smallest probability of a context (i.e., $\min_x \rho(x)$), thus suggesting that OAM may suffer from poor finite-time performance in problems with unbalanced context distributions.[1] While it is not applicable in contextual settings, the SPL algorithm recently introduced by Degenne et al. (2020b) resolves some of the limitations of OAM while being asymptotically optimal for general structures. Inspired by efficient algorithms for best-arm identification (Degenne et al., 2019), Degenne et al. (2020b) reformulate the optimization problem in the lower bound as a saddle-point problem and show how to leverage online learning algorithms to avoid recomputing the exploration strategy from scratch at each step. Furthermore, SPL removes any form of forced exploration by introducing optimism into the estimated optimization problem. As a result, SPL is computationally efficient and achieves better empirical performance in problems with general structures. Unfortunately, the price SPL has to pay for handling general structures is a regret that scales linearly with the number of arms even when applied to linear structures (where simple approaches like LINUCB scale only with the feature dimensions d).

[1] Interestingly, Hao et al. (2019) explicitly mention in their conclusions the importance of properly managing the context distribution to achieve satisfactory finite-time performance.

Outline. In this chapter, we follow similar steps as in Degenne et al. (2020b) and introduce SOLID, a novel asymptotically-optimal algorithm for contextual linear bandits. Our detailed contributions are as follows.

1. We reformulate the optimization problem of the lower bound for contextual linear bandits (Combes et al., 2017; Ok et al., 2018; Hao et al., 2019) by introducing an additional constraint to guarantee bounded solutions and by explicitly decoupling the context distribution and the exploration policy (Section 10.3). While we bound the bias introduced by the constraint, we also illustrate how the resulting exploration policy is better adapted to unbalanced context distributions;

2. Leveraging the Lagrangian dual formulation associated with the constrained lower-bound optimization problem, we derive SOLID, an efficient primal-dual learning algorithm that incrementally updates the exploration strategy at each time step (Section 10.4). Furthermore, we replace forced exploration with an optimistic version of the optimization problem by specifically leveraging the linear structure of the problem. Finally, SOLID does not require any explicit tracking step and it samples directly from the current exploration strategy;

3. We derive regret guarantees showing that SOLID is asymptotically optimal and in finite time it removes any dependence on the context distribution, in particular $\min_x \rho(x)$ (Section 10.5). Moreover, we remove, for the first time, any linear dependence on the number of arms and our bound only scales with $\log |\mathcal{A}|$. Finally, we derive a $\tilde{\mathcal{O}}(|\mathcal{X}|\sqrt{dn})$ finite-time worst-case regret bound (with $|\mathcal{X}|$ contexts, d features, and horizon n) which shows that SOLID is the first algorithm to be both *minimax optimal* and *asymptotically optimal* for linear (non-contextual) bandits;

4. We empirically compare to a number of state-of-the-art methods for contextual linear bandits and show how SOLID is more computationally efficient, often has the smallest regret, and it is robust to unbalanced context distributions (Section 10.7).

Contextual Linear Bandits

We first need to better formalize (and adapt our notation to) the contextual linear bandit setting. The new notation is exactly the same as before, with the context made explicit in all context-dependent quantities. Let $\mathcal{X}$ be the set of contexts and $\mathcal{A}$ be the set of arms with cardinality $|\mathcal{X}| < \infty$ and $|\mathcal{A}| < \infty$, respectively. Each context-arm pair is embedded into $\mathbb{R}^d$ through a feature map $\phi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^d$. For any reward model $\theta \in \mathbb{R}^d$, we denote by $\mu_\theta(x, a) = \phi(x, a)^T\theta$ the expected reward for each context-arm pair. Let $a^\star_\theta(x) := \operatorname{argmax}_{a\in\mathcal{A}} \mu_\theta(x, a)$ and $\mu^\star_\theta(x) := \max_{a\in\mathcal{A}} \mu_\theta(x, a)$ denote the optimal arm and its value for context x and parameter θ. We define the sub-optimality gap of arm a for context x in model θ as $\Delta_\theta(x, a) := \mu^\star_\theta(x) - \mu_\theta(x, a)$. We make the following assumption for the given structure.

Assumption 10.2.1. *The realizable parameters belong to a compact subset Θ of $\mathbb{R}^d$ such that $\|\theta\|_2 \leq B$ for all $\theta \in \Theta$. The features are bounded, i.e., $\|\phi(x, a)\|_2 \leq L$ for all $x \in \mathcal{X}, a \in \mathcal{A}$. The context distribution is supported over the whole context set, i.e., $\rho(x) \geq \rho_{\min} > 0$ for all $x \in \mathcal{X}$. Finally, we assume $\theta^\star$ has a unique optimal arm in each context (see e.g., Combes et al., 2017; Hao et al., 2019).*

That is, the underlying structured domain $\mathfrak{E}_\Theta$ (see Definition 8.2.1) is described by $\Theta = \{\theta \in \mathbb{R}^d : \|\theta\|_2 \leq B\}$ and $\mathfrak{M}_\Theta = \{\{\theta^T \phi(x,a)\}_{x \in \mathcal{X}, a \in \mathcal{A}} : \theta \in \Theta\}$. The learner knows the fixed feature map ϕ, the bounds B, L, while as usual it does not know the true parameter $\theta^\star$ and the context distribution ρ. Here we also assume that each context-arm pair yields random Gaussian rewards with known variance σ^2. Given two parameters $\theta, \theta' \in \mathbb{R}^d$, we define

$$d_{x,a}(\theta, \theta') := \frac{1}{2\sigma^2}(\mu_\theta(x,a) - \mu_{\theta'}(x,a))^2, \tag{10.1}$$

which corresponds to the Kullback-Leibler divergence between the Gaussian reward distributions of the two models in context x and arm a. Finally, we redefine a bandit strategy $\pi := \{\pi_t\}_{t \geq 1}$ as a sequence of measurable functions $\pi_t(H_{t-1}, X_t)$ of the current context X_t and of the past history $H_{t-1} := (X_1, A_1, Y_1, \ldots, X_{t-1}, A_{t-1}, Y_{t-1})$. Its expected regret compares the performance of π with those of an oracle strategy that pulls the optimal arm of each observed context,

$$R_n^\pi(\theta^\star, \mathfrak{E}_\Theta) := \mathbb{E}_{\rho,\pi}\left[\sum_{t=1}^n (\mu_\theta^\star(X_t) - \mu_\theta(X_t, A_t))\right]. \tag{10.2}$$

Regularized least-squares estimator. The regularized least-square estimate of $\theta^\star$ using t samples is $\widehat{\theta}_t := \overline{V}_t^{-1} U_t$, where

$$\overline{V}_t := \sum_{s=1}^t \phi(X_s, A_s)\phi(X_s, A_s)^T + \nu I \tag{10.3}$$

is the regularized design matrix, with $\nu \geq 1$, I the $d \times d$ identity matrix, and $U_t := \sum_{s=1}^t \phi(X_s, A_s) Y_s$. The estimator $\widehat{\theta}_t$ satisfies the following concentration inequality.

Theorem 10.2.1. *Let $\delta \in (0,1)$, $n \geq 3$, and $\widehat{\theta}_t$ be a regularized least-square estimator obtained using $t \in [n]$ samples collected using an arbitrary bandit strategy $\pi := \{\pi_t\}_{t \geq 1}$. Then,*

$$\mathbb{P}\left\{\exists t \in [n] : \|\widehat{\theta}_t - \theta^\star\|_{\overline{V}_t} \geq \sqrt{c_{n,\delta}}\right\} \leq \delta,$$

where $c_{n,\delta}$ is of order $\mathcal{O}(\log(1/\delta) + d \log\log n)$.

We derived Theorem 10.2.1 in the original paper (Tirinzoni et al., 2020b) as a result of independent interest. We refer the reader to such paper for the proof. For the usual choice $\delta = 1/n$, $c_{n,1/n}$ is of order $\mathcal{O}(\log n + d \log\log n)$, which illustrates how the dependency on d is on a lower-order term w.r.t. n (as opposed to the well-known concentration bound derived by Abbasi-Yadkori et al. (2011)). This result is the counterpart of Theorem 8 of Lattimore and Szepesvari (2017) for the concentration on the reward parameter estimation error instead of the prediction error.

Lower Bound

We recall the asymptotic lower bound for multi-armed bandit problems with structure from Lai and Robbins (1985); Combes et al. (2017); Ok et al. (2018). We recall from

Definition 2.6.1 that a bandit strategy π is *uniformly consistent* on structure $\mathfrak{E}_\Theta$ (satisfying Assumption 10.2.1) if $R_n^\pi(\theta, \mathfrak{E}_\Theta) = o(n^\alpha)$ for any $\alpha > 0$ and any $\theta \in \Theta$.

Proposition 10.3.1. *Let $\pi := \{\pi_t\}_{t\geq 1}$ by a uniformly consistent bandit strategy on structure $\mathfrak{E}_\Theta$. Then,*

$$\liminf_{n\to\infty} \frac{R_n(\theta^\star, \mathfrak{E}_\Theta)}{\log(n)} \geq v^\star(\theta^\star), \tag{10.4}$$

where $v^\star(\theta^\star)$ is the value of the optimization problem

$$\begin{aligned} \inf_{\eta(x,a)\geq 0} \quad & \sum_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} \eta(x,a)\Delta_{\theta^\star}(x,a) \\ \text{s.t.} \quad & \inf_{\theta'\in\Theta_{\text{alt}}} \sum_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} \eta(x,a) d_{x,a}(\theta^\star, \theta') \geq 1, \end{aligned} \tag{P}$$

where $\Theta_{\text{alt}} := \{\theta' \in \Theta \mid \exists x \in \mathcal{X},\ a^\star_{\theta^\star}(x) \neq a^\star_{\theta'}(x)\}$ is the set of alternative parameters such that the optimal arm changes for at least one context.

The variables $\eta(x,a)$ can be interpreted as the number of pulls allocated to each context-arm pair so that enough information is obtained to correctly identify the optimal arm in each context while minimizing the regret. As such, we often refer to a solution $\eta^\star$ of (P) as an optimal exploration strategy.

In the specific case of linear bandit and Gaussian noise, the infimum over alternative models in the constraint of (P) can be computed in closed form when $\Theta = \mathbb{R}^d$ as[2]

$$2\sigma^2 \inf_{\theta'\in\Theta_{\text{alt}}} \sum_{x,a} \eta(x,a) d_{x,a}(\theta^\star, \theta') = \min_{\substack{x\in\mathcal{X},\\ a\neq a^\star_{\theta^\star}(x)}} \frac{\Delta_{\theta^\star}(x,a)^2}{\|\phi(x,a) - \phi^\star_{\theta^\star}(x))\|^2_{V_\eta^{-1}}}, \tag{10.5}$$

where $V_\eta = \sum_{x,a} \eta(x,a)\phi(x,a)\phi(x,a)^T$ and $\phi^\star_{\theta^\star}(x) = \phi(x, a^\star_{\theta^\star}(x))$.

Formulating the lower bound in terms of the solution of (P) is not desirable for two main reasons. First, (P) is not a well-posed optimization problem since the inferior may not be attainable, i.e., the optimal solution may allocate an infinite number of pulls to some optimal arms. Second, (P) removes any dependency on the context distribution ρ. In fact, the optimal solution $\eta^\star$ of (P) may prescribe to select a context-arm (x,a) pair a large number of times, despite x having low probability of being sampled from ρ. While this has no impact on the asymptotic performance of $\eta^\star$ (as soon as $\rho_{\min} > 0$), building on $\eta^\star$ to design a learning algorithm may lead to poor finite-time performance. In order to mitigate these issues, we propose a variant of the previous lower bound obtained by adding a constraint on the cumulative number of pulls in each context and explicitly decoupling the context distribution ρ and the *exploration policy* $\omega(x,a)$ defining the probability of selecting arm a in context x. Given $z \in \mathbb{R}_{>0}$, we define the optimization problem

$$\begin{aligned} \min_{\omega\in\Omega} \quad & z\mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}} \omega(x,a)\Delta_{\theta^\star}(x,a)\right] \\ \text{s.t.} \quad & \inf_{\theta'\in\Theta_{\text{alt}}} \mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star, \theta')\right] \geq 1/z \end{aligned} \tag{P$_z$}$$

[2]When Θ contains only bounded parameters as we assumed in Assumption 10.2.1, the infimum admits no closed-form expression (see Degenne et al. (2020a)). For simplicity, we shall use this closed-form expression in our experiments, while it is not needed in our theoretical results.

where $\Omega = \{\omega(x,a) \geq 0 \mid \forall x \in \mathcal{X} : \sum_{a\in\mathcal{A}} \omega(x,a) = 1\}$ is the probability simplex. We denote by $\omega^\star_{z,\theta^\star}$ the optimal solution of (P_z) and $u^\star(z,\theta^\star)$ its associated value (if the problem is unfeasible we set $u^\star(z,\theta^\star) = +\infty$). Inspecting ($\mathrm{P}_z$), we notice that z serves a a global constraint on the number of samples. In fact, for any $\omega \in \Omega$, the associated number of samples $\eta(x,a)$ allocated to a context-arm pair (x,a) is now $z\rho(x)\omega(x,a)$. Since ρ is a distribution over $\mathcal{X}$ and $\sum_a \omega(x,a) = 1$ in each context, the total number of samples sums to z. As a result, (P_z) admits a minimum and it is more amenable to designing a learning algorithm based on its Lagrangian relaxation. Furthermore, we notice that z can be interpreted as defining a more "finite-time" formulation of the lower bound. Finally, we remark that the total number of samples that can be assigned to a context x is indeed constrained to $z\rho(x)$. This constraint crucially makes (P_z) more context aware and forces the solution ω to be more adaptive to the context distribution. In Section 10.4, we leverage these features to design an incremental algorithm whose finite-time regret does not depend on $\rho_{\min}$, thus improving over previous algorithms Lattimore and Szepesvari (2017); Hao et al. (2019), as supported by the empirical results in Section 9.5. The following lemma provides a characterization of (P_z) and its relationship with (P).

Lemma 10.3.1. *Let $\underline{z}(\theta^\star) := \min\{z > 0 : (\mathrm{P}_z)$ is feasible$\}$, be the minimum value of z that induces a feasible optimization problem, $\overline{z}(\theta^\star) := \max_{x\in\mathcal{X}} \sum_{a\neq a^\star_{\theta^\star}(x)} \frac{\eta^\star(x,a)}{\rho(x)}$, and $z^\star(\theta^\star) := \sum_{x\in\mathcal{X}} \sum_{a\neq a^\star_{\theta^\star}(x)} \eta^\star(x,a)$. Then*

$$\frac{1}{\underline{z}(\theta^\star)} = \max_{\omega\in\Omega} \inf_{\theta'\in\Theta_{\mathrm{alt}}} \mathbb{E}_\rho \left[\sum_{a\in\mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star,\theta')\right]$$

and there exists a constant $c_\Theta > 0$ such that, for any $z \in (\underline{z}(\theta^\star), +\infty)$,

$$u^\star(z,\theta^\star) \leq v^\star(\theta^\star) + \frac{2zBL\underline{z}(\theta^\star)}{z - \underline{z}(\theta^\star)} \cdot \Gamma(z,\theta^\star),$$

where

$$\Gamma(z,\theta^\star) := \begin{cases} 1 & \text{if } z < \overline{z}(\theta^\star) \\ \min\left\{\max\left\{\frac{c_\Theta\sqrt{2}z^\star(\theta^\star)}{\sigma\sqrt{z}}, \frac{z^\star(\theta^\star)}{z}\right\}, 1\right\} & \text{otherwise} \end{cases}$$

The proof is provided in Appendix D. The first result characterizes the range of z for which (P_z) is feasible. Interestingly, $\underline{z}(\theta^\star) < +\infty$ is the inverse of the sample complexity of the best-arm identification problem and the associated solution is the one that maximizes the amount of information gathered about the reward model $\theta^\star$. As z increases, $\omega^\star_{z,\theta^\star}$ becomes less aggressive in favoring informative context-arm pairs and more sensitive to the regret minimization objective. The second result quantifies the bias with respect to the optimal solution of (P_z). For $z \geq \overline{z}(\theta^\star)$, the error decreases approximately at a rate $1/\sqrt{z}$ showing that the solution of (P_z) can be made arbitrarily close to $v^\star(\theta^\star)$.

Lagrangian Formulation

In designing our learning algorithm, we build on the Lagrangian relaxation of (P_z). As we shall see, it is more convenient to replace the minimization of the action gaps $\Delta_{\theta^\star}$ in (P_z)

by a maximization of the mean rewards $\mu_{\theta^\star}$. Therefore, consider the following variant of (P_z):

$$\begin{aligned} \max_{\omega \in \Omega} \quad & \mathbb{E}_\rho \Big[\sum_{a \in \mathcal{A}} \omega(x,a) \mu_{\theta^\star}(x,a) \Big] \\ \text{s.t.} \quad & \inf_{\theta' \in \Theta_{\mathrm{alt}}} \mathbb{E}_\rho \Big[\sum_{a \in \mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star, \theta') \Big] \geq 1/z \end{aligned} \tag{$\overline{\mathrm{P}}_z$}$$

This problem differs from (P_z) since we replaced the action gaps with the means in the objective function and avoided scaling the latter by z. Let $\overline{\omega}^\star_{z,\theta^\star}$ the optimal solution of ($\overline{\mathrm{P}}_z$) and $\overline{u}^\star(z,\theta^\star)$ be its associated value (if the problem is unfeasible we set $\overline{u}^\star(z,\theta^\star) = +\infty$). Since the feasibility set is equivalent in (P_z) and ($\overline{\mathrm{P}}_z$) as we only changed the objective function, the following proposition is immediate.

Proposition 10.3.2. *The following properties hold:*

1. *Both* (P_z) *and* ($\overline{\mathrm{P}}_z$) *are feasible for* $z \geq \underline{z}(\theta^\star)$*;*
2. $\omega^\star_{z,\theta^\star} = \overline{\omega}^\star_{z,\theta^\star}$*;*
3. $u^\star(z,\theta^\star) = z\,(\mu^\star - \overline{u}^\star(z,\theta^\star))$ *where* $\mu^\star = \mathbb{E}_\rho[\mu^\star_{\theta^\star}(x)]$.

Due to the equivalence demonstrated in Proposition 10.3.2, in the remaining we shall write $\omega^\star_z$ to denote both $\omega^\star_{z,\theta^\star}$ and $\overline{\omega}^\star_{z,\theta^\star}$.

We now introduce the Lagrangian relaxation of ($\overline{\mathrm{P}}_z$). For any $\omega \in \Omega$, let $f(\omega;\theta^\star)$ denote the objective function and $g(\omega;z,\theta^\star)$ denote the KL constraint:

$$\begin{aligned} f(\omega;\theta^\star) &= \mathbb{E}_\rho \Big[\sum_{a \in \mathcal{A}} \omega(x,a) \mu_{\theta^\star}(x,a) \Big], \\ g(\omega;z,\theta^\star) &= \inf_{\theta' \in \Theta_{\mathrm{alt}}} \mathbb{E}_\rho \Big[\sum_{a \in \mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star,\theta') \Big] - \frac{1}{z}. \end{aligned}$$

The Lagrangian relaxation problem of ($\overline{\mathrm{P}}_z$) is

$$\min_{\lambda \geq 0} \max_{\omega \in \Omega} \Big\{ h(\omega,\lambda;z,\theta^\star) := f(\omega;\theta^\star) + \lambda g(\omega;z,\theta^\star) \Big\}, \tag{P_λ}$$

where $\lambda \in \mathbb{R}_{\geq 0}$ is a multiplier. We denote by $\lambda^\star(z,\theta^\star)$ the optimal multiplier for problem (P_λ). We note that f is linear in ω, while g is concave since it is an infimum of affine functions. Hence, the maximization in (P_λ) is a non-smooth concave optimization problem.

Strong duality. We now verify that strong duality holds for the Lagrangian formulation (P_λ) (with respect to ($\overline{\mathrm{P}}_z$)) when $z > \underline{z}(\theta^\star)$. This is immediate from the existence of a Slater point, as shown in the following proposition.

Proposition 10.3.3 (Slater Condition)**.** *For any* $z > \underline{z}(\theta^\star)$*, there exists a* strictly *feasible solution* $\underline{\omega}$*, i.e.,* $g(\underline{\omega};z,\theta^\star) > 0$.

Proof. This is a direct consequence of the fact that (see Lemma 10.3.1)

$$\max_{\omega\in\Omega}\inf_{\theta'\in\Theta_{\mathrm{alt}}}\mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}}\omega(x,a)d_{x,a}(\theta^\star,\theta')\right]=\frac{1}{\underline{z}(\theta^\star)}>\frac{1}{z}.$$

□

Thus, the optimal solution of (P_λ) is $(\lambda^\star(z,\theta^\star),\omega_z^\star)$.

Boundedness of the optimal multipliers. We recall the following basic result.

Lemma 10.3.2 (Lemma 3 of Nedić and Ozdaglar (2009))**.** *For any $z>\underline{z}(\theta^\star)$, if $\overline{\omega}_z$ is a Slater point for ($\overline{P}_z$),*

$$\lambda^\star(z,\theta^\star)\le\frac{f(\omega_z^\star;\theta^\star)-f(\overline{\omega}_z;\theta^\star)}{g(\overline{\omega}_z;z,\theta^\star)}$$

Using Lemma 10.3.2, we can prove the following result which will be very useful for the regret analysis.

Lemma 10.3.3. *For any $z\ge 2\underline{z}(\theta^\star)$,*

$$\lambda^\star(z,\theta^\star)\le 2BL\underline{z}(\theta^\star). \tag{10.6}$$

Proof. From Proposition 10.3.3, $\underline{\omega}$ (the solution of the associated pure-exploration problem) is a Slater point for problem (P_z). Then, by Lemma 10.3.2,

$$\lambda^\star(z,\theta^\star)\le\frac{f(\omega_z^\star;\theta^\star)-f(\underline{\omega};\theta^\star)}{g(\underline{\omega};z,\theta^\star)}.$$

Let $\mathrm{kl}(\omega)$ denote the expected KL of ω, so that $g(\omega;z,\theta^\star)=\mathrm{kl}(\omega)-1/z$. Then,

$$\frac{f(\omega_z^\star;\theta^\star)-f(\underline{\omega};\theta^\star)}{\mathrm{kl}(\underline{\omega})-1/z}\le\frac{f(\omega_z^\star;\theta^\star)}{\mathrm{kl}(\underline{\omega})-1/z}\le\frac{BL}{\mathrm{kl}(\underline{\omega})-1/z}. \tag{10.7}$$

Furthermore, since $\mathrm{kl}(\underline{\omega})=1/\underline{z}(\theta^\star)$,

$$\lambda^\star(z,\theta^\star)\le\frac{BL z\underline{z}(\theta^\star)}{z-\underline{z}(\theta^\star)}\le 2BL\underline{z}(\theta^\star),$$

where the last inequality holds for $z\ge 2\underline{z}(\theta^\star)$. This concludes the proof. □

We notice that building an algorithm on top of the Lagrangian relaxation of ($\overline{P}_z$) seems more straightforward than the technique used by Degenne et al. (2020b) to translate the constrained optimization appearing in the lower bound into a saddle-point problem by taking the ratio between information and regret. Not only it is unclear whether a similar approach can be adapted to the contextual linear case, our formulation naturally leads to designing an efficient incremental learning algorithm built around a primal-dual projected gradient approach to the solution of (P_λ).

Algorithm 8 SOLID

Require: Multiplier λ_1, confidence values $\{\beta_t\}_t$ and $\{\gamma_t\}_t$, maximum multiplier $\lambda_{\max}$, normalization factors $\{z_k\}_{k\geq 0}$, phase lengths $\{p_k\}_{k\geq 0}$, step sizes $\alpha_k^\lambda, \alpha_k^\omega$

Set $\omega_1(x,a) \leftarrow \frac{1}{|\mathcal{A}|}$, $\overline{V}_0 = \nu I$, $U_0 = 0$, $\theta_0 = 0$, $S_0 \leftarrow 0$
Phase index: $K_1 \leftarrow 0$
for $t = 1, 2, \dots, n$ **do**
  Receive context $X_t \sim \rho$
  Set $K_{t+1} \leftarrow K_t$
  if $\inf_{\theta' \in \overline{\Theta}_{t-1}} \|\widehat{\theta}_{t-1} - \theta'\|^2_{\overline{V}_{t-1}} > \beta_{t-1}$ **then**
    // EXPLOITATION STEP
    $A_t \leftarrow \operatorname{argmax}_{a\in\mathcal{A}} \mu_{\widehat{\theta}_{t-1}}(X_t, a)$
    $\lambda_{t+1} \leftarrow \lambda_t$, $\omega_{t+1} \leftarrow \omega_t$
  else
    // EXPLORATION STEP
    Sample arm: $A_t \sim \omega_t(X_t, \cdot)$
    Set $S_t \leftarrow S_{t-1} + 1$
    // UPDATE SOLUTION
    Compute $q_t \in \partial h_t(\omega_t, \lambda_t, z_{K_t})$ (see Equation 10.10)
    Update policy: $\omega_{t+1}(x,a) \leftarrow \frac{\omega_t(x,a) e^{\alpha^\omega_{K_t} q_t(x,a)}}{\sum_{a'\in\mathcal{A}} \omega_t(x,a') e^{\alpha^\omega_{K_t} q_t(x,a')}}$
    Update multiplier: $\lambda_{t+1} \leftarrow \min\{[\lambda_t - \alpha^\lambda_{K_t} g_t(\omega_t, z_{K_t})]_+, \lambda_{\max}\}$
    // PHASE STOPPING TEST
    if $S_t - S_{T_{K_t}-1} = p_k$ **then**
      Change phase: $K_{t+1} \leftarrow K_t + 1$
      Reset solution: $\omega_{t+1} \leftarrow \omega_1$, $\lambda_{t+1} \leftarrow \lambda_1$
    end if
  end if
  Pull A_t and observe outcome Y_t
  Update $\overline{V}_t$, U_t, $\widehat{\theta}_t$, $\widehat{\rho}_t$ using X_t, A_t, Y_t
end for

Asymptotically Optimal Linear Primal Dual Algorithm

We introduce SOLID (aSymptotic Optimal Linear prImal Dual), which combines a primal-dual approach to incrementally compute the solution of an optimistic estimate of the Lagrangian relaxation (P_λ) within a scheme that, depending on the accuracy of the estimate $\widehat{\theta}_t$, separates *exploration* steps, where arms are pulled according to the exploration policy ω_t, and *exploitation* steps, where the greedy arm is selected. The values of the input parameters for which SOLID enjoys regret guarantees are reported in Section 10.5. In the following, we detail the main ingredients composing the algorithm (see Algorithm 8).

Estimation. SOLID stores and updates the regularized least-square estimate $\widehat{\theta}_t$ using all samples observed over time. We denote by $\mu_{\widehat{\theta}_t}(x,a) = \phi(x,a)^T \widehat{\theta}_t$ and by $a^\star_{\widehat{\theta}_t}(x) =$

$\operatorname{argmax}_{a\in\mathcal{A}} \mu_{\widehat{\theta}_t}(x,a)$ the corresponding estimated reward and estimated optimal arm, respectively. SOLID also estimates the context distribution as $\widehat{\rho}_t(x) = \frac{1}{t}\sum_{s=1}^{t} \mathbb{1}\{X_s = x\}$.

Accuracy test and tracking. Similar to previous algorithms leveraging asymptotic lower bounds, we build on the generalized likelihood ratio test (e.g., Degenne et al., 2019) to verify the accuracy of the estimate $\widehat{\theta}_t$. At the beginning of each step t, SOLID first computes $\inf_{\theta'\in\overline{\Theta}_{t-1}} \|\widehat{\theta}_{t-1} - \theta'\|^2_{\overline{V}_{t-1}}$, where $\overline{\Theta}_{t-1} = \{\theta' \in \Theta \mid \exists x \in \mathcal{X},\ a^\star_{\widehat{\theta}_{t-1}}(x) \neq a^\star_{\theta'}(x)\}$ is the set of alternative models.[3] This quantity measures the accuracy of the algorithm, where the infimum over alternative models defines the problem θ' that is closest to $\widehat{\theta}_{t-1}$ and yet different in the optimal arm of at least one context.[4] This serves as a worst-case scenario for the true $\theta^\star$, since if $\theta^* = \theta'$ then selecting arms according to $\widehat{\theta}_{t-1}$ would lead to linear regret. If the accuracy exceeds a threshold β_{t-1}, then SOLID performs an exploitation step, where the estimated optimal arm $a^\star_{\widehat{\theta}_{t-1}}(X_t)$ is selected in the current context. On the other hand, if the test fails, the algorithm moves to an exploration step, where an arm A_t is sampled according to the estimated exploration policy $\omega_t(X_t,\cdot)$. While this approach is considerably simpler than standard tracking strategies (e.g., selecting the arm with the largest gap between the policy ω_t and the number of pulls), in Section 10.5 we show that sampling from ω_t achieves the same level of tracking efficiency.

Optimistic primal-dual subgradient descent. At each step t, we define an estimated optimistic version of the Lagrangian relaxation (P_λ) as

$$f_t(\omega) := \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\left(\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) + \varphi_t(x,a)\right), \tag{10.8}$$

$$g_t(\omega,z) := \inf_{\theta'\in\overline{\Theta}_{t-1}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\left(\bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') + \frac{2BL}{\sigma^2}\varphi_t(x,a)\right) - \frac{1}{z}, \tag{10.9}$$

$$h_t(\omega,\lambda,z) := f_t(\omega) + \lambda g_t(\omega,z), \tag{10.10}$$

where $\varphi_t(x,a) := \sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}^{-1}_{t-1}}$ is a confidence interval for the mean reward at x,a with γ_t a suitable parameter to define its size. $\bar{\mu}_\theta(x,a) := \min\{\max\{\mu_\theta(x,a), -BL\}, BL\}$, $\bar{d}_{x,a}(\theta,\theta') := \frac{1}{2\sigma^2}(\bar{\mu}_\theta(x,a) - \mu_{\theta'}(x,a))^2$ are the clipped rewards and KL divergences, respectively. These are such that $|\bar{\mu}_\theta(x,a)| \le |\mu_\theta(x,a)|$ and $\bar{d}_{x,a}(\theta,\theta') \le d_{x,a}(\theta,\theta')$.

Notice that we do not use optimism on the context distribution, which is simply replaced by its empirical estimate. Therefore, h_t is not necessarily optimistic with respect to the original Lagrangian function h. Nonetheless, we prove in Section 10.5 that this level of optimism is sufficient to induce enough exploration to have accurate estimates of $\theta^\star$. This is in contrast with the popular forced exploration strategy (e.g. Lattimore and Szepesvari, 2017; Combes et al., 2017; Ok et al., 2018; Hao et al., 2019), which prescribes a minimum fraction of pulls ϵ such that at any step t, any of the arms with less than ϵS_t pulls is selected, where S_t is the number of exploration rounds so far. While this strategy

[3] In our implementation we use the closed-form for the infimum as in (10.5).

[4] In practice, it is more efficient to take the infimum only over problems with different optimal arm in the last observed context X_t. This is indeed what we do in our experiments and all our theoretical results follow using this alternative definition with only minor changes.

is sufficient to guarantee a minimum level of accuracy for $\widehat{\theta}_t$ and to obtain asymptotic regret optimality, in practice it is highly inefficient as it requires selecting all arms in each context regardless of their value or amount of information.

At each step t, SOLID updates the estimates of the optimal exploration policy ω_t and the Lagrangian multiplier λ_t. In particular, given the sub-gradient q_t of $h_t(\omega_t, \lambda_t, z_{K_t})$, SOLID updates ω_t and λ_t by performing one step of projected sub-gradient descent with suitable learning rates $\alpha^{\omega}_{K_t}$ and $\alpha^{\lambda}_{K_t}$. In the update of ω_t, we perform the projection onto the simplex Ω using an entropic metric, while the multiplier is clipped in $[0, \lambda_{\max}]$. While this is a rather standard primal-dual approach to solve the Lagrangian relaxation (P_λ), the interplay between estimates $\widehat{\theta}_t$, ρ_t, the optimism used in h_t, and the overall regret performance of the algorithm is at the core of the analysis in Section 10.5. This approach significantly reduces the computational complexity compared to algorithms like OSSB (Combes et al., 2017) and OAM (Hao et al., 2019), which require solving problem P at each exploratory step. In Section 9.5, we show that the incremental nature of SOLID allows it to scale to problems with much larger context-arm spaces. Furthermore, we leverage the convergence rate guarantees of the primal-dual gradient descent to show that the incremental nature of SOLID does not compromise the asymptotic optimality of the algorithm (see Section 10.5).

The z parameter. While the primal-dual algorithm is guaranteed to converge to the solution of (P_z) for any fix z, it may be difficult to properly tune z to control the error with respect to (P). SOLID uses the fact that the error scales as $1/\sqrt{z}$ (Lemma 10.3.1 for z sufficiently large) and it increases z over time. Given as input two non-decreasing sequences $\{p_k\}_k$ and $\{z_k\}_k$, at each phase k, SOLID uses z_k in the computation of the subgradient of h_t and in the definition of f_t and g_t. After p_k explorative steps, it resets the policy ω_t and the multiplier λ_t and transitions to phase $k+1$. Since $p_k = S_{T_{k+1}-1} - S_{T_k-1}$ is the number of *explorative* steps of phase k starting at time T_k, the actual number of steps during k may vary. Note that at the end of each phase only the optimization variables are reset, while the learning variables (i.e., $\widehat{\theta}_t$, $\overline{V}_t$, and $\widehat{\rho}_t$) use all the samples collected through phases.

Main Results

We state and discuss the main theoretical results before carrying out the analysis. We start from a finite-time problem-dependent regret bound, the main contribution of this chapter, where we confirm that SOLID is asymptotically optimal. We then state a worst-case regret bound where we show that SOLID is also minimax optimal for linear (non-contextual) bandits and that its regret does not scale with certain quantities that appear in the problem-dependent bound. The full problem-dependent analysis is reported in the next section. The worst-case analysis, on the other hand, is a simpler variant of the problem-dependent one and is thus deferred to Appendix D.

Problem-dependent regret bound We need the following assumption.

Assumption 10.5.1. *The maximum multiplier used by* SOLID *is such that* $\lambda_{\max} \geq 2BL\underline{z}(\theta^\star)$.

While an assumption on the maximum multiplier is rather standard for the analysis of primal-dual projected subgradient (e.g., Nedić and Ozdaglar, 2009; Efroni et al., 2020), we conjecture that it may be actually relaxed in our case by replacing the fixed $\lambda_{\max}$ by an increasing sequence as done for $\{z_k\}_k$.

Theorem 10.5.1. *Consider a contextual linear bandit with contexts $\mathcal{X}$, arms $\mathcal{A}$, reward parameter $\theta^\star$, features bounded by L, zero-mean Gaussian noise with variance σ^2 and context distribution ρ satisfying Assumption 10.2.1. If* SOLID *is run with confidence values $\beta_{t-1} = c_{n,1/n}$ and $\gamma_t = c_{n,1/S_t^2}$, where $c_{n,\delta}$ is defined as in Theorem 10.2.1, learning rates $\alpha_k^\lambda = \alpha_k^\omega = 1/\sqrt{p_k}$ and increasing sequences $z_k = z_0 e^k$ and $p_k = z_k e^{2k}$, for some $z_0 \geq 1$, then it is* ***asymptotically optimal*** *with the same constant as in the lower bound of Proposition 10.3.1. Furthermore, for any finite n the regret of* SOLID *is bounded as*

$$R_n(\theta^\star, \mathfrak{E}_\Theta) \leq v^\star(\theta^\star)\frac{c_{n,1/n}}{2\sigma^2} + C_{\log}(\log\log n)^{\frac{1}{2}}(\log n)^{\frac{3}{4}} + C_{\text{const}}, \tag{10.11}$$

where the two constants $C_{\log}$ and C_{const} are $C_{\log} = lin_{\geq 0}(v^\star(\theta^\star), |\mathcal{X}|, L^2, B^2, \sqrt{d}, 1/\sigma^2)$ and $C_{\text{const}} = v^\star(\theta^\star)\frac{B^2L^2}{\sigma^2} + lin_{\geq 0}(L, B, z_0(\underline{z}(\theta^\star)/z_0)^3, (\overline{z}(\theta^\star)/z_0)^2)$.[5]

The first result shows that SOLID run with an exponential schedule for z is asymptotic optimal, while the second one provides a bound on the finite-time regret. We can identify three main components in the finite-time regret. **1)** The first term scales with the logarithmic term $c_{n,1/n} = O(\log n + d\log\log n)$ and a leading constant $v^\star(\theta^\star)$, which is optimal as shown in Proposition 10.3.1. In most cases, this is the dominant term of the regret. **2)** Lower-order terms in $o(\log n)$. Notably, a regret of order $\sqrt{\log n}$ is due to the incremental nature of SOLID and it is directly inherited from the convergence rate of the primal-dual algorithm we use to optimize (P_z). The larger term $(\log n)^{3/4}$ that we obtain in the final regret is actually due to the schedule of $\{z_k\}$ and $\{p_k\}$. While it is possible to design a different phase schedule to reduce the exponent towards $1/2$, this would negatively impact the constant regret term. **3)** The constant regret C_{const} is due to the exploitation steps, burn-in phase and the initial value z_0. The regret due to z_0 takes into account the regime when (P_z) is unfeasible ($z_k < \underline{z}(\theta^\star)$) or when z_k is too small to assess the rate at which $u^\star(z_k, \theta^\star)$ approaches $v^\star(\theta^\star)$ ($z < \overline{z}(\theta^\star)$), see Lemma 10.3.1. Notably, the regret due to the initial value z_0 vanishes when $z_0 > \overline{z}(\theta^\star)$. A more aggressive schedule for z_k reaching $\overline{z}(\theta^\star)$ in few phases would reduce the initial regret at the cost of a larger exponent in the sub-logarithmic terms.

The sub-logarithmic terms in the regret have only logarithmic dependency on the number of arms. This is better than existing algorithms based on exploration strategies built from lower bounds. OSSB (Combes et al., 2017) indeed depends on $|\mathcal{A}|$ directly in the main $\mathcal{O}(\log n)$ regret terms. While the regret analysis of OAM is asymptotic, it is possible to identify several lower-order terms depending linearly on $|\mathcal{A}|$. In fact, OAM as well as OSSB require forced exploration on each context-arm pair, which inevitably translates into regret. In this sense, the dependency on $|\mathcal{A}|$ is hard-coded into the algorithm and cannot be improved by a better analysis. SPL depends linearly on $|\mathcal{A}|$ in the explore/exploit threshold (the equivalent of our β_t) and in other lower-order terms due to the analysis of the

[5] *lin*$(\cdot)$ denotes any function with linear or sublinear dependence on the inputs (ignoring logarithmic terms). For example, $lin_{\geq 0}(x, y^2) \in \{a_0 + a_1 x + a_2 y + a_3 y^2 + a_4 xy^2 : a_i \geq 0\}$.

tracking rule. On the other hand, SOLID never requires all arms to be repeatedly pulled and we were able to remove the linear dependence on $|\mathcal{A}|$ through a refined analysis of the sampling procedure that we carry out shortly. This is inline with the experimental results where we did not notice any explicit linear dependence on $|\mathcal{A}|$.

The constant regret term depends on the context distribution is $\overline{z}(\theta^\star)$ (Lemma 10.3.1). Nonetheless, this dependency disappears whenever z_0 is a fraction $\overline{z}(\theta^\star)$. This is in striking contrast with OAM, whose analysis includes several terms depending on the inverse of the context probability $\rho_{\min}$. This confirms that SOLID is able to better adapt to the distribution generating the contexts. While the phase schedule of Theorem 10.5.1 leads to an asymptotically-optimal algorithm and sublinear-regret in finite time, it may be possible to find a different schedule having the same asymptotic performance and better finite-time guarantees, although this may depend on the horizon n.

As shown in (Hao et al., 2019), when the features of the optimal arms span $\mathbb{R}^d$, the asymptotic lower bound vanishes (i.e., $v^\star(\theta^\star) = 0$). In this case, selecting optimal arms is already informative enough to correctly estimate $\theta^\star$ and no explicit exploration is needed and SOLID, like OAM, has sub-logarithmic regret.

Worst-case analysis. The constant terms in Theorem 10.5.1 are due to a naive bound which assumes linear regret in those phases where z_k is small (e.g., when the optimization problem is infeasible). While this simplifies the analysis for asymptotic optimality, we verify that SOLID always suffers sub-linear regret, regardless of the values of z_k. For the following result, we do not require Assumption 10.5.1 to hold.

Theorem 10.5.2 (Worst-case regret bound)**.** *Let z_k be arbitrary, $p_k = e^{rk}$ for some constant $r \geq 1$, and the other parameters be the same as in Theorem 10.5.1. Then, for any n the regret of* SOLID *is bounded as*

$$R_n(\theta^\star, \mathfrak{E}_\Theta) \leq 3BL\pi^2 \left(4 + \frac{\lambda_{\max} BL}{\sigma^2}\right) + \frac{2e^r \lambda_{\max}^2}{r}\sqrt{n} + C_{\mathrm{sqrt}} \left(1 + \frac{\lambda_{\max} BL}{\sigma^2}\right) \log(n)\sqrt{n},$$

where $C_{\mathrm{sqrt}} = lin_{\geq 0}(|\mathcal{X}|, \sqrt{d}, B, L)$.

Notably, this bound removes the dependencies on $\underline{z}(\theta^\star)$ and $\overline{z}(\theta^\star)$, while its derivation is agnostic to the values of z_k. Interestingly, we could set $\lambda_{\max} = 0$ and the algorithm would completely ignore the KL constraint, thus focusing only on the objective function. This is reflected in the worst-case bound since all terms with a dependence on σ^2 or a quadratic dependence on BL disappear. The key result is that the objective function alone, thanks to optimism, is sufficient for proving sub-linear regret but not for proving asymptotic optimality. More precisely, the resulting bound is $\widetilde{\mathcal{O}}(|\mathcal{X}|\sqrt{nd})$, which matches the minimax optimal rate apart from the dependence on $|\mathcal{X}|$. The latter could be reduced to $\sqrt{|\mathcal{X}|}$ by a better analysis, but a polynomial dependence on $|\mathcal{X}|$ is likely to be unavoidable since SOLID makes explicit use of the set of contexts. It remains an open question how to design an asymptotically optimal algorithm for the contextual case whose regret does not scale with $|\mathcal{X}|$.

Problem-Dependent Analysis

Outline

We start by analyzing the action sampling strategy (Section 10.6.2) on top of which we construct some high-probability events (Section 10.6.3). We carry out the fool proof in Section 10.6.4. An outline is as follows.

Step 1. (Section 10.6.4) Using the confidence set of Theorem 10.2.1, we show that the regret suffered when the algorithm enters the exploitation step is finite;

Step 2. (Section 10.6.4) Using the properties of our action sampling strategy, we reduce the regret incurred during exploration rounds to the sum of objective values of the policies computed incrementally by primal-dual gradient ascent;

Step 3. (Section 10.6.4) By combining standard tools from convex optimization with the properties of our confidence intervals, we relate the sum of objective values at each phase to the corresponding optimal value and constraint violations;

Step 4. (Section 10.6.4) We relate the sum of constraints to the exploitation test used by SOLID. In particular, using the fact that the algorithm is not in the exploitation step, we show that the sum of constraints cannot be larger than $\mathcal{O}(\log n)$;

Step 5. (Section 10.6.4) We combine the results obtained in the previous steps to show a first bound on the expected regret suffered during the exploration rounds. Our bound has the optimal dependency on $v^\star(\theta^\star) \log n$ but scales with the expected number $\mathbb{E}[K_n]$ of phases executed by the algorithm;

Step 6. (Section 10.6.4) By relating the upper bound on the sum of constraints computed at Step 3 and a lower bound on the same quantity, we obtain an upper bound on K_n as a function of the chosen sequences p_k, z_k;

Step 7. (Section 10.6.4) We derive the final result by combining the bound on K_n of Step 5 using the exponential schedule for p_k, z_k with the partial regret bound of Step 4.

Additional notation. Since most of our proof involves bounding the regret incurred during the exploration rounds, we introduce some additional notation to better characterize such rounds. Let

$$E_t := \mathbb{1}\left\{ \inf_{\theta' \in \overline{\Theta}_{t-1}} \|\widehat{\theta}_{t-1} - \theta'\|^2_{\overline{V}_{t-1}} \leq \beta_{t-1} \right\} \tag{10.12}$$

be the event that exploration occurs at time t. We denote by $S_t := \sum_{s=1}^{t} \mathbb{1}\{E_t\}$ the number of exploration rounds up to time t and by $N_t^E(x,a) := \sum_{s=1}^{t} \mathbb{1}\{X_t = x, A_t = a, E_t\}$ the number of visits to (x,a) in these exploration rounds. We use $K_t \in \{0,1,\dots\}$ to denote the phase index at time t and T_k to denote the time at which phase k starts. Moreover, $\mathcal{T}_k := \{t \in [n] : K_t = k\}$ is the set of time steps in phase k and $\mathcal{T}_k^E := \{t \in \mathcal{T}_k : E_t\}$ are the exploration rounds in phase k.

Action Sampling

SOLID does not use standard tracking approaches for action selection (e.g., cumulative tracking (Garivier and Kaufmann, 2016; Degenne et al., 2019) or direct tracking (Combes et al., 2017; Hao et al., 2019)) but a sampling strategy. Despite being simpler and more practical than tracking, we show that sampling from ω_t enjoys nice theoretical guarantees. In the following results we define the filtration $\mathcal{F}_t$ as the σ-algebra generated by the t-step history, $H_t = (X_1, A_1, Y_1, \dots, X_t, A_t, Y_t)$.

The following result bounds the deviation between expectations of measurable functions under the sequence of conditional probabilities played by the algorithm and the same functions evaluated at the observed contexts/arms. This result will be very useful in the regret analysis to avoid undesirable linear dependencies on the number of arms.

Lemma 10.6.1. *Let $\{\omega_t\}_{t\geq 1}$ be such that $\omega_t \in \Omega$ and ω_t is $\mathcal{F}_{t-1}$-measurable. Let $\{X_t\}_{t\geq 1}$ be a sequence of i.i.d. contexts distributed according to ρ and $\{A_t\}_{t\geq 1}$ be such that $A_t \sim \omega_t(X_t, \cdot)$. Let $\{\varphi_t^i\}_{t\geq 1, i\in[m]}$ be a sequence of functions $\varphi_t^i : \mathcal{X}\times\mathcal{A} \to [-b, b]$ such that $\varphi_t^i(x, a)$ is $\mathcal{F}_{t-1}$-measurable for all $i \in [m]$. Then,*

$$\sum_{\substack{t\geq 1,\\ i\in[m]}} \mathbb{P}\left\{E_t, \left|\sum_{s\leq t:E_s}\left(\varphi_s^i(X_s, A_s) - \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}}\rho(x)\omega_s(x, a)\varphi_s^i(x, a)\right)\right| > b\sqrt{\frac{S_t}{2}\log(mS_t^2)}\right\} \leq \frac{\pi^2}{3}.$$

Proof. Fix $i \in [m]$. Let $Z_t := \varphi_t^i(X_t, A_t)$ and τ_s be a random variable such that the s-th exploration round occurs at time $\tau_s + 1$. Notice that $\{\tau_s\}_{s\geq 1}$ is a strictly-increasing sequence (i.e., $\tau_{s+1} > \tau_s$) of stopping times w.r.t. $\{\mathcal{F}_t\}_{t\geq 1}$. Furthermore, define

$$W_s := Z_{\tau_s+1} - \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}}\rho(x)\omega_{\tau_s+1}(x, a)\varphi_{\tau_s+1}^i(x, a)$$

and let $\mathcal{G}_s := \mathcal{F}_{\tau_s+1}$. Using Lemma 10 in Jian et al. (2019), we have that $\{W_s, \mathcal{G}_s\}_{s\geq 1}$ is a martingale difference sequence (with differences bounded by b). Therefore, by Azuma's inequality

$$\mathbb{P}\left\{\left|\sum_{i=1}^{s} W_i\right| > b\sqrt{\frac{s}{2}\log\frac{2}{\delta}}\right\} \leq \delta.$$

Let $a_t := b\sqrt{\frac{S_t}{2}\log\left(mS_t^2\right)}$ and fix some $\bar{t} \geq 1$. Then,

$$\begin{aligned}
&\sum_{t=1}^{\bar{t}} \mathbb{1}\left\{E_t, \left|\sum_{s\leq t:E_s}\left(Z_s - \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}}\rho(x)\omega_s(x, a)\varphi_s^i(x, a)\right)\right| > a_t\right\} \\
&\quad\leq \sum_{s\geq 1}\mathbb{1}\left\{\left|\sum_{j=1}^{s}\left(Z_{\tau_j+1} - \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}}\rho(x)\omega_{\tau_j+1}(x, a)\varphi_{\tau_j+1}^i(x, a)\right)\right| > a_{\tau_s+1}, \tau_s + 1 \leq \bar{t}\right\} \\
&\quad\leq \sum_{s\geq 1}\mathbb{1}\left\{\left|\sum_{j=1}^{s} W_j\right| > b\sqrt{\frac{s}{2}\log(ms^2)}\right\}.
\end{aligned}$$

In the last inequality, we used the fact that $a_{\tau_s+1} = b\sqrt{\frac{s}{2}\log(ms^2)}$. Taking expectations and applying Azuma's inequality with $\delta = \frac{2}{ms^2}$,

$$\sum_{t=1}^{\bar{t}}\mathbb{P}\left\{E_t, \left|\sum_{s\leq t:E_s}\left(Z_s - \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}}\rho(x)\omega_s(x, a)\varphi_s^i(x, a)\right)\right| > a_t\right\} \leq \sum_{s\geq 1}\frac{2}{ms^2} = \frac{\pi^2}{3m}.$$

The result holds for all $\bar{t} \geq 1$ and the proof follows by summing over all $i \in [m]$. □

Choosing $m = |\mathcal{X}||\mathcal{A}|$ and $\varphi_t^{x,a}(x', a') = \mathbb{1}\{x' = x, a' = a\}$, so that $b = 1$, directly yields the following corollary.

Corollary 10.6.1. *Let $\{\omega_t\}_{t\geq 1}$ be such that $\omega_t \in \Omega$ and ω_t is $\mathcal{F}_{t-1}$-measurable. Let $\{X_t\}_{t\geq 1}$ be a sequence of i.i.d. contexts distributed according to ρ and $\{A_t\}_{t\geq 1}$ be such that $A_t \sim \omega_t(X_t, \cdot)$. Then,*

$$\sum_{t\geq 1}\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} \mathbb{P}\left\{E_t, \left|N_t^E(x,a) - \rho(x)\sum_{s\leq t:E_s}\omega_s(x,a)\right| > \sqrt{\frac{S_t}{2}\log\left(S_t^2|\mathcal{X}||\mathcal{A}|\right)}\right\} \leq \frac{\pi^2}{3}.$$

Corollary 10.6.1 provides an analogous result to those obtained by tracking strategies, where the empirical pull counts are shown close to the sequence of conditional probabilities computed by the optimizer. Despite being simpler, our sampling rule achieves similar efficiency as existing tracking rules. In particular, our bound scales with $\log|\mathcal{A}|$, a factor that appears in the tightest known analysis of cumulative tracking (Degenne et al., 2020b). The factor $\sqrt{S_t \log S_t}$ is not typically found in tracking strategies for MABs. However, we note that such dependency would naturally appear when generalizing these strategies to the contextual case.

High-Probability Events

We now report the high-probability events used throughput the proof.

Let $\Phi_{x,a} := \phi(x,a)\phi(x,a)^T$. We define the following events:

true regret close to objective values

$$G_t^{\Delta} := \left\{\left|\sum_{s\leq t:E_s}\left(\Delta_{\theta^\star}(X_s,A_s) - \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\Delta_{\theta^\star}(x,a)\right)\right| \leq 2LB\sqrt{S_t\log S_t}\right\}, \tag{10.13}$$

true confidence intervals close to expected confidence intervals

$$G_t^{\phi} := \left\{\left|\sum_{s\leq t:E_s}\left(\|\phi(X_s,A_s)\|_{\bar{V}_{s-1}^{-1}} - \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}}\right)\right| \leq \frac{L}{\nu}\sqrt{S_t\log S_t}\right\}, \tag{10.14}$$

true design matrix close to expected design matrix

$$G_t^{d} := \left\{\left\|\sum_{s\leq t:E_s}\left(\Phi_{X_s,A_s} - \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\Phi_{x,a}\right)\right\|_{\infty} \leq L^2\sqrt{S_t\log(dS_t)}\right\}, \tag{10.15}$$

well-estimated context distribution

$$G_t^{\rho} := \left\{\forall x\in\mathcal{X} : |\widehat{\rho}_{t-1}(x) - \rho(x)| \leq 2\max\left(\sqrt{\frac{\log(|\mathcal{X}|S_t^2)}{2S_t}}, \frac{2}{t}\right)\right\}, \tag{10.16}$$

well-estimated parameters

$$G_t^{\theta} := \left\{\|\widehat{\theta}_{t-1} - \theta^\star\|_{\overline{V}_{t-1}} \leq \sqrt{\gamma_t}\right\}. \tag{10.17}$$

Furthermore, we define $G_t := \{G_t^\Delta, G_t^\phi, G_t^d, , G_t^\rho, G_t^\theta\}$ as the *"good" event* and let $M_t = \sum_{s=1}^t \mathbb{1}\{E_s, \neg G_s\}$ be the number of exploration rounds in which the good event does not hold. This can be bounded in expectation as follows.

Lemma 10.6.2. *Let $M_t = \sum_{s=1}^t \mathbb{1}\{E_s, \neg G_s\}$ be the number of exploration rounds in which the good event does not hold, then*

$$\mathbb{E}[M_t] \leq \frac{3\pi^2}{2}.$$

Proof. Using the definition of G_s together with the union bound,

$$\begin{aligned}\mathbb{E}[M_t] = \sum_{s=1}^t \mathbb{P}\{E_s, \neg G_s\} \leq \sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\Delta\right\} + \sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\phi\right\} \\ + \sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^d\right\} + \sum_{s=1}^t \mathbb{P}\{E_s, \neg G_s^\rho\} + \sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\theta\right\}.\end{aligned}$$

The first and second term can be bounded by Lemma 10.6.1 by noticing that $\Delta_{\theta^\star}(x,a) \leq 2LB$ and that $\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}}$ is $\mathcal{F}_{s-1}$-measurable and upper-bounded by $\frac{L}{\nu}$ at all time steps. Thus,

$$\sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\Delta\right\} + \sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\phi\right\} \leq \frac{2\pi^2}{3}.$$

Similarly, the third term can be bounded by Lemma 10.6.1 by taking a union bound over all elements of $\Phi_{x,a}$ (for a total of d^2 elements) and noting that each term is bounded by L^2. Thus,

$$\sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^d\right\} \leq \frac{\pi^2}{3}.$$

The fourth term is

$$\begin{aligned}&\sum_{s=1}^t \mathbb{P}\{E_s, \neg G_s^\rho\} \leq \sum_{\substack{s\geq 1,\\ x\in\mathcal{X}}} \mathbb{P}\left\{E_s, |\widehat{\rho}_{s-1}(x) - \rho(x)| > 2\max\left(\sqrt{\frac{\log(|\mathcal{X}|S_s^2)}{2S_s}}, \frac{2}{s}\right)\right\} \\ &\leq \sum_{\substack{s\geq 1,\\ x\in\mathcal{X}}} \mathbb{P}\left\{E_s, |\widehat{\rho}_{s-1}(x) - \widehat{\rho}_s(x)| + |\widehat{\rho}_s(x) - \rho(x)| > 2\max\left(\sqrt{\frac{\log(|\mathcal{X}|S_s^2)}{2S_s}}, \frac{2}{s}\right)\right\} \\ &\leq \sum_{\substack{s\geq 1,\\ x\in\mathcal{X}}} \mathbb{P}\left\{E_s, |\widehat{\rho}_{s-1}(x) - \widehat{\rho}_s(x)| > \frac{2}{s}\right\} + \sum_{\substack{s\geq 1,\\ x\in\mathcal{X}}} \mathbb{P}\left\{E_s, |\widehat{\rho}_s(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}|S_s^2)}{2S_s}}\right\} \leq \frac{\pi^2}{3}.\end{aligned}$$

Here we used the fact that the absolute difference between two consecutive empirical means with samples bounded by 1 cannot be larger than $\frac{2}{s}$. We also used Lemma D.2.1 to bound the second term. Finally, the fifth term can be directly bounded by Lemma D.2.2:

$$\sum_{s=1}^t \mathbb{P}\left\{E_s, \neg G_s^\theta\right\} \leq \frac{\pi^2}{6}.$$

Combining the five bounds concludes the proof. □

Regret Proof

We start decomposing the regret based on whether E_t holds or not:

$$R_n = \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\left\{\neg E_t\right\} + \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\left\{E_t\right\} = R_n^{\text{exploit}} + R_n^{\text{explore}}.$$

Throughout the proof, as stated in the main theorem, we use $\beta_{t-1} := c_{n,1/n}$ and $\gamma_t := c_{n,1/S_t^2}$.

Regret during Exploitation

We show that the regret suffered when exploitation occurs is finite. Let $\beta_{t-1} := c_{n,1/n}$, where $c_{n,\delta}$ was defined in Theorem 10.2.1. Then $F_t := \mathbb{1}\left\{\|\widehat{\theta}_{t-1} - \theta^\star\|^2_{\overline{V}_{t-1}} \leq c_{n,1/n}\right\}$ is the event under which the true model belongs to the confidence set, which holds with probability at least $1 - 1/n$ by the same theorem. We leverage this to decompose the regret during exploitation as:

$$R_n^{\text{exploit}} = \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\left\{\neg E_t, F_t\right\} + \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\left\{\neg E_t, \neg F_t\right\}.$$

The expectation of the second term is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{n} \underbrace{\Delta_{\theta^\star}(X_t, A_t)}_{\leq 2LB}\mathbb{1}\left\{\neg E_t, \neg F_t\right\}\right] \leq 2LB \cdot \mathbb{E}\left[\sum_{t=1}^{n}\mathbb{1}\left\{\neg F_t\right\}\right] \leq 2LB \sum_{t=1}^{n} \underbrace{\mathbb{P}\left\{\neg F_t\right\}}_{\leq 1/n} \leq 2LB,$$

where we bounded $\mathbb{P}\left\{\neg F_t\right\} \leq \frac{1}{n}$ by using Theorem 10.2.1 with $\delta = 1/n$. Regarding the first term, we have two possible cases. If $a^\star_{\widehat{\theta}_{t-1}}(X_t) = a^\star_{\theta^\star}(X_t)$, then the algorithm suffers no regret since by definition it pulls the empirically optimal arm (which is the optimal arm in this case). If $a^\star_{\widehat{\theta}_{t-1}}(X_t) \neq a^\star_{\theta^\star}(X_t)$, then it must be that $\theta^\star \in \overline{\Theta}_{t-1}$, that is, the true model is in the set of alternative models for the current context. Under $\neg E_t$, this implies that

$$\|\widehat{\theta}_{t-1} - \theta^\star\|^2_{\overline{V}_{t-1}} \geq \inf_{\theta' \in \overline{\Theta}_{t-1}} \|\widehat{\theta}_{t-1} - \theta'\|^2_{\overline{V}_{t-1}} > \beta_{t-1} = c_{n,1/n},$$

which is a contradiction with respect to F_t. Therefore, $\neg E_t$ and F_t cannot hold at the same time and the algorithm suffers no regret. Combining these results, we conclude

$$\mathbb{E}\left[R_n^{\text{exploit}}\right] \leq 2LB.$$

Regret under Exploration

The key challenge is to bound the regret during the exploration rounds. We proceed by following the steps outlined in Section 10.6.1.

From Regret to Objective Values

We decompose the regret incurred during exploration as

$$R_n^{\text{explore}} := \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\{E_t\} \leq \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\{E_t, G_t\} + 2LB \underbrace{\sum_{t=1}^{n} \mathbb{1}\{E_t, \neg G_t\}}_{:=M_n}.$$

Refer to Section 10.6.3 for the definition of G_t. The second term is M_n, the number of exploration rounds in which the good event does not hold, and can be bounded in expectation by using Lemma 10.6.2. The first one can be bounded by using the good event. Suppose, without loss of generality, that E_n and G_n hold (if they do not, the following reasoning can be repeated for the last time step at which these events hold). Then, using G_t^Δ (see Section 10.6.3),

$$\begin{aligned}\sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\{E_t, G_t\} &= \sum_{t \leq n : E_t} \Delta_{\theta^\star}(X_t, A_t) \\ &\leq \sum_{t \leq n : E_t} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x, a)\Delta_{\theta^\star}(x, a) + 2LB\sqrt{S_n \log S_n}.\end{aligned}$$

Using the definition of phase, we can rewrite the first summation as

$$\sum_{t \leq n : E_t} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x, a)\Delta_{\theta^\star}(x, a) = \sum_{k=0}^{K_n} \sum_{t \in \mathcal{T}_k^E} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x, a)\Delta_{\theta^\star}(x, a).$$

Recall that K_t is the (random) phase index at time t, while $\mathcal{T}_k^E$ is the set of exploration rounds in phase k. Let $\underline{k} := \min\{k \in \mathbb{N} | z_k \geq 2\underline{z}(\theta^\star)\}$. We split the sum into phases before and after $\underline{k}$. For those before, we have

$$\sum_{k < \underline{k}} \sum_{t \in \mathcal{T}_k^E} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x, a)\Delta_{\theta^\star}(x, a) \leq 2LB \sum_{k < \underline{k}} |\mathcal{T}_k^E| \leq 2LB \sum_{k < \underline{k}} p_k,$$

which yields at most finite regret since $\{p_k\}$ is increasing. Let us now fix a phase $k \geq \underline{k}$ and bound the regret during its exploration rounds ($\mathcal{T}_k^E$). Note that the optimization problem in each phase $k \geq \underline{k}$ is feasible. We have

$$\begin{aligned}&\sum_{t \in \mathcal{T}_k^E} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x, a)\Delta_{\theta^\star}(x, a) \\ &\quad= \sum_{t \in \mathcal{T}_k^E : G_t} \sum_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} \rho(x)\omega_t(x, a)\Delta_{\theta^\star}(x, a) + \sum_{t \in \mathcal{T}_k^E : \neg G_t} \sum_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} \rho(x)\omega_t(x, a)(\mu^\star_{\theta^\star}(x) - \mu_{\theta^\star}(x, a)) \\ &\quad\leq \sum_{t \in \mathcal{T}_k^E : G_t} \sum_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} \rho(x)\omega_t(x, a)\Delta_{\theta^\star}(x, a) + M_{n,k}\mu^\star - \sum_{t \in \mathcal{T}_k^E : \neg G_t} \sum_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} \rho(x)\omega_t(x, a)\mu_{\theta^\star}(x, a).\end{aligned}$$

Here we defined $\mu^\star := \sum_{x \in \mathcal{X}} \rho(x)\mu^\star_{\theta^\star}(x)$ and $M_{n,k}$ as the number of exploration rounds during phase k where the good event does not hold. The last term can be bounded by

$M_{n,k}BL$. Regarding the remaining two,

$$\begin{aligned}
&\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\Delta_{\theta^\star}(x,a)+M_{n,k}\mu^\star\\
&\qquad=(p_k-M_{n,k})\mu^\star+M_{n,k}\mu^\star-\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\mu_{\theta^\star}(x,a)\\
&\qquad=p_k\mu^\star+\underbrace{\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}(\hat{\rho}_{t-1}(x)-\rho(x))\sum_{a\in\mathcal{A}}\omega_t(x,a)\mu_{\theta^\star}(x,a)}_{(a)}\\
&\qquad\quad-\underbrace{\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}\hat{\rho}_{t-1}(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\mu_{\theta^\star}(x,a)}_{(b)}.
\end{aligned}$$

Term (a) can be bounded by

$$(a)\le LB\underbrace{\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}|\hat{\rho}_{t-1}(x)-\rho(x)|}_{\zeta_{n,k}}.$$

The second term $\zeta_{n,k}$ will be bounded shortly over all phases by means of Lemma D.2.5. We now provide a lower bound to term (b). The first step is to relate this to the objective function optimized by the algorithm. Using the definition of G_t and Lemma D.2.3,

$$\begin{aligned}
(b)&\ge\sum_{t\in\mathcal{T}_k^E:G_t}\sum_{x\in\mathcal{X}}\hat{\rho}_{t-1}(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\left(\bar{\mu}_{\hat{\theta}_{t-1}}(x,a)-\sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}\right)\\
&\pm\sum_{t\in\mathcal{T}_k^{E:\neg G_t}}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\hat{\rho}_{t-1}(x)\omega_t(x,a)\underbrace{\bar{\mu}_{\hat{\theta}_{t-1}}(x,a)}_{|\cdot|\le LB}\pm\sum_{t\in\mathcal{T}_k^E}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\hat{\rho}_{t-1}(x)\omega_t(x,a)\sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}\\
&\ge\sum_{t\in\mathcal{T}_k^E}f_t(\omega_t)-M_{n,k}BL-2\sum_{t\in\mathcal{T}_k^E}\sum_{x\in\mathcal{X}}\hat{\rho}_{t-1}(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}\\
&\ge\sum_{t\in\mathcal{T}_k^E}f_t(\omega_t)-M_{n,k}BL-2\sqrt{\gamma_n}\Psi_{n,k}.
\end{aligned}\tag{10.18}$$

In the last step, we used $\sqrt{\gamma_t}\le\sqrt{\gamma_n}$ (which is by definition $\mathcal{O}(\log S_n)$) and defined $\Psi_{n,k}:=\sum_{t\in\mathcal{T}_k^E}\sum_{x\in\mathcal{X}}\hat{\rho}_{t-1}(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}$.

To wrap-up the regret bound we have obtained so far, summing over all phases,

$$\begin{aligned}
R_n^{\text{explore}}&\le 2LB\sum_{k<\underline{k}}p_k+\sum_{k\ge\underline{k}}^{K_n}p_k\mu^\star+LB\underbrace{\sum_{k\ge\underline{k}}^{K_n}\zeta_{n,k}}_{\le\zeta_n}-\sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E}f_t(\omega_t)\\
&\quad+2LB\underbrace{\sum_{k\ge\underline{k}}^{K_n}M_{n,k}}_{\le M_n}+2LBM_n+2\sqrt{\gamma_n}\underbrace{\sum_{k\ge\underline{k}}^{K_n}\Psi_{n,k}}_{\le\Psi_n}+2LB\sqrt{S_n\log S_n}.
\end{aligned}$$

Here we defined

$$\zeta_n := \sum_{t\le n: E_t: G_t} \sum_{x\in\mathcal{X}} |\hat{\rho}_{t-1}(x) - \rho(x)|, \quad \Psi_n := \sum_{t\le n: E_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \hat{\rho}_{t-1}(x)\omega_t(x,a)\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}.$$

ζ_n can be bounded by Lemma D.2.5 and Ψ_n by Lemma D.2.6. Both terms are of order $\mathcal{O}(\sqrt{S_n \log S_n})$. In order to simplify notation, we keep the specific bounds implicit in the remaining. Therefore, our partial regret bound is

$$\begin{aligned} R_n^{\text{explore}} &\le 2LB \sum_{k<\underline{k}} p_k + \sum_{k\ge\underline{k}}^{K_n} p_k\mu^\star - \sum_{k\ge\underline{k}}^{K_n} \sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) \\ &\quad + 4LBM_n + 2\sqrt{\gamma_n}\Psi_n + LB\zeta_n + 2LB\sqrt{S_n \log S_n}. \end{aligned} \tag{10.19}$$

Bounding the Sum of Objective Values

Our goal here is to lower bound the sum of objective values. As before, fix some phase index $k \ge \underline{k}$ and let $\lambda \ge 0$ be arbitrary. By recalling that the optimization process is reset at the beginning of each phase and using Corollary D.2.1 with $\alpha_k^\lambda = \alpha_k^\omega = 1/\sqrt{p_k}$ and $\omega = \omega_{z_k}^\star$ (the optimal solution of problem (P_{z_k})),

$$\begin{aligned} \sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) &\ge \sum_{t\in\mathcal{T}_k^E} h_t(\omega_{z_k}^\star, \lambda_t, z_k) - \lambda \sum_{t\in\mathcal{T}_k^E} g_t(\omega_t, z_k) \\ &\quad - \left(\log|\mathcal{A}| + \frac{b_\omega^2 + b_\lambda^2}{2} + \frac{(\lambda-\lambda_1)^2}{2}\right)\sqrt{p_k}. \end{aligned} \tag{10.20}$$

We recall that b_λ and b_ω are the maximum sub-gradients in λ and ω, respectively. We now lower-bound the first term on the right-hand side. Since $h_t(\omega_{z_k}^\star, \lambda_t, z_k) = f_t(\omega_{z_k}^\star) + \lambda_t g_t(\omega_{z_k}^\star, z_k)$, $f_t(\omega_{z_k}^\star) \ge -LB$, $g_t(\omega_{z_k}^\star, z_k) \ge -\frac{1}{z_k}$, and $\lambda_t \le \lambda_{\max}$, this term, evaluated on steps where G_t does not hold, can be lower-bounded by $\sum_{t\in\mathcal{T}_k^{E}:\neg G_t} h_t(\omega_{z_k}^\star, \lambda_t, z_k) \ge -(LB + \lambda_{\max}/z_k)M_{n,k}$. For any step $t \in \mathcal{T}_k^E$ in which G_t holds, the optimism property (Lemma D.2.4) yields

$$\begin{aligned} f_t(\omega_{z_k}^\star) &\ge \sum_{x\in\mathcal{X}} (\hat{\rho}_{t-1}(x) - \rho(x)) \underbrace{\sum_{a\in\mathcal{A}} \omega_{z_k}^\star(x,a)\mu_{\theta^\star}(x,a)}_{|\cdot|\le LB} + f(\omega_{z_k}^\star) \\ &\ge f(\omega_{z_k}^\star) - LB \sum_{x\in\mathcal{X}} |\hat{\rho}_{t-1}(x) - \rho(x)|, \end{aligned}$$

and

$$\begin{aligned} g_t(\omega_{z_k}^\star, z_k) &\ge \inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \hat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_{z_k}^\star(x,a) d_{x,a}(\theta^\star, \theta') - \frac{1}{z_k} \pm g(\omega_{z_k}^\star) \\ &\ge \inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} (\hat{\rho}_{t-1}(x) - \rho(x)) \sum_{a\in\mathcal{A}} \omega_{z_k}^\star(x,a) d_{x,a}(\theta^\star, \theta') + g(\omega_{z_k}^\star) \\ &\ge g(\omega_{z_k}^\star) - \frac{2L^2B^2}{\sigma^2} \sum_{x\in\mathcal{X}} |\hat{\rho}_{t-1}(x) - \rho(x)|. \end{aligned}$$

Combining these two and using $\lambda_t \le \lambda_{\max}$,

$$\sum_{t\in\mathcal{T}_k^E:G_t} h_t(\omega^\star_{z_k},\lambda_t,z_k) \ge \sum_{t\in\mathcal{T}_k^E:G_t} \big(f(\omega^\star_{z_k})+\lambda_t g(\omega^\star_{z_k})\big) - LB\left(1+\frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_{n,k}.$$

Note that $g(\omega^\star_{z_k}) \ge 0$ since by assumption $\omega^\star_{z_k}$ is feasible for the optimization problem (P_{z_k}). Furthermore, $\sum_{t\in\mathcal{T}_k^E:G_t} f(\omega^\star_{z_k}) = \sum_{t\in\mathcal{T}_k^E} f(\omega^\star_{z_k}) - \sum_{t\in\mathcal{T}_k^E:\neg G_t} \underbrace{f(\omega^\star_{z_k})}_{|\cdot|\le LB} \ge$ $p_k f(\omega^\star_{z_k}) - LBM_{n,k}$. Therefore, we obtain the following lower-bound on the sum of optimal objective values:

$$\sum_{t\in\mathcal{T}_k^E} h_t(\omega^\star_{z_k},\lambda_t,z_k) \ge p_k f(\omega^\star_{z_k}) - LB\left(1+\frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_{n,k} - (2LB+\lambda_{\max}/z_k)M_{n,k}.$$

Plugging this back into (10.20),

$$\begin{aligned}\sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) \ge p_k f(\omega^\star_{z_k}) - \lambda\sum_{t\in\mathcal{T}_k^E} g_t(\omega_t,z_k) - a_\lambda\sqrt{p_k}\\ - LB\left(1+\frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_{n,k} - (2LB+\lambda_{\max}/z_k)M_{n,k}, \quad (10.21)\end{aligned}$$

where, for simplicity, we defined $a_\lambda := \left(\log|\mathcal{A}| + \frac{b_\omega^2+b_\lambda^2}{2} + \frac{(\lambda-\lambda_1)^2}{2}\right)$. Summing over all phases,

$$\begin{aligned}\sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) \ge \sum_{k\ge\underline{k}}^{K_n} p_k f(\omega^\star_{z_k}) - \lambda\sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E} g_t(\omega_t,z_k) - a_\lambda\sum_{k\ge\underline{k}}^{K_n}\sqrt{p_k}\\ - LB\left(1+\frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_n - (2LB+\lambda_{\max})M_n, \quad (10.22)\end{aligned}$$

where we used $\sum_{k\ge\underline{k}}^{K_n} M_{n,k} \le M_n$, $\sum_{k\ge\underline{k}}^{K_n}\zeta_{n,k} \le \zeta_n$, and $z_k \ge 1$.

Bounding the sum of constraints

Our next step is to upper bound $\sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E:E_t} g_t(\omega_t,z_k)$, the sum of constraints of the policies played by the algorithm during feasible phases (those with $z_k \ge 2\underline{z}(\theta^\star)$). The intuition is that this term cannot be large (i.e., it cannot be above $\mathcal{O}(\log n)$), otherwise the exploitation test would trigger and we would not be exploring at step n. Using the definition of $g_t(\omega,z_k)$ (Equation 10.9) and splitting the sum based on the good event

$$\begin{aligned}&\sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E} g_t(\omega_t,z_{K_t})\\ &\le \sum_{t\le n:E_t}\inf_{\theta'\in\bar{\Theta}_{t-1}}\sum_{x\in\mathcal{X}}\hat{\rho}_{t-1}(x)\sum_{a\in\mathcal{A}}\omega_t(x,a)\bar{d}_{x,a}(\hat{\theta}_{t-1},\theta') + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n - \sum_{k\ge\underline{k}}^{K_n}\sum_{t\in\mathcal{T}_k^E}\frac{1}{z_k}\\ &\le \underbrace{\sum_{t\le n:E_t,G_t}\inf_{\theta'\in\bar{\Theta}_{t-1}}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\hat{\rho}_{t-1}(x)\omega_t(x,a)\bar{d}_{x,a}(\hat{\theta}_{t-1},\theta')}_{①} + \frac{2L^2B^2}{\sigma^2}M_n + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n - \sum_{k\ge\underline{k}}^{K_n}\frac{p_k}{z_k}.\end{aligned}$$

Note that in the first step above we implicitly upper bounded the sum of KLs on the feasible phases with the sum of KLs over all exploration rounds. We can use the definition of G_t and the optimism (Lemma D.2.4) to upper bound the first sum by

$$\begin{aligned}
① &\leq \sum_{t\leq n:E_t,G_t} \inf_{\theta'\in\bar{\Theta}_{t-1}} \sum_{x\in\mathcal{X}} \hat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star,\theta') + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n \\
&\leq \underbrace{\sum_{t\leq n:E_t,G_t} \inf_{\theta'\in\bar{\Theta}_{t-1}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star,\theta')}_{②} + \frac{2L^2B^2}{\sigma^2} \underbrace{\sum_{t\leq n:E_t,G_t} \sum_{x\in\mathcal{X}} |\rho(x) - \hat{\rho}_{t-1}(x)|}_{=\zeta_n} \\
&\quad + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n.
\end{aligned}$$

Furthermore, the first term can be upper bounded by replacing each set $\bar{\Theta}_{t-1}$ over which the infimum is taken by Θ_{alt} (if the two sets were different, such term would be zero). Therefore,

$$\begin{aligned}
② &\leq \sum_{t\leq n:E_t,G_t} \inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star,\theta') \\
&\leq \underbrace{\inf_{\theta'\in\Theta_{alt}} \sum_{t\leq n:E_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star,\theta')}_{③}, \qquad (10.23)
\end{aligned}$$

where we moved the infimum outside the outer sum and added the remaining steps where G_t does not hold. Let $\Phi_{x,a} := \phi(x,a)\phi(x,a)^T$ and $V_{n,e} := \sum_{t\leq n:E_t} \Phi_{X_t,A_t}$ be the design matrix of the exploration rounds. Using the definition of $d_{x,a}$,

$$\begin{aligned}
③ &= \frac{1}{2\sigma^2} \inf_{\theta'\in\Theta_{alt}} (\theta^\star-\theta')^T \left(\sum_{t\leq n:E_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\Phi_{x,a} \pm V_{n,e} \right) (\theta^\star-\theta') \\
&\leq \inf_{\theta'\in\Theta_{alt}} \left\{ \frac{1}{2\sigma^2}\|\theta^\star-\theta'\|^2_{V_{n,e}} + \frac{1}{2\sigma^2}\|\theta^\star-\theta'\|_2^2 \left\| \sum_{t\leq n:E_t} \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\Phi_{x,a} - V_{n,e} \right\|_2 \right\} \\
&\leq \inf_{\theta'\in\Theta_{alt}} \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}} N_n^E(x,a) d_{x,a}(\theta^\star,\theta') + \frac{2B^2}{\sigma^2} \left\| \sum_{t\leq n:E_t} \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\Phi_{x,a} - V_{n,e} \right\|_2 \\
&\leq \inf_{\theta'\in\Theta_{alt}} \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}} N_n^E(x,a) d_{x,a}(\theta^\star,\theta') + \frac{2B^2\sqrt{d}}{\sigma^2} \left\| \sum_{t\leq n:E_t} \sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\Phi_{x,a} - V_{n,e} \right\|_\infty .
\end{aligned}$$

Recall that G_n holds. Then, by using the definition of G^d to bound the norm,

$$③ \leq \underbrace{\inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} N_{n-1}^E(x,a) d_{x,a}(\theta^\star,\theta')}_{④} + \frac{2B^2L^2}{\sigma^2} + \frac{2B^2L^2}{\sigma^2}\sqrt{dS_n \log(dS_n)}.$$

Here we used $N_n(x,a) = N_{n-1}(x,a) + \mathbb{1}\{X_n = x, A_n = a\}$ and upper bounded the KL at round n by its maximum value. Moreover, similarly to Lemma D.2.4 we can show that

$$④ \leq \inf_{\theta' \in \Theta_{alt}} \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} N^E_{n-1}(x,a) \bar{d}_{x,a}(\hat{\theta}_{n-1}, \theta') + \frac{2LB\sqrt{\gamma_n}}{\sigma^2} \underbrace{\sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} N^E_{n-1}(x,a) \|\phi(x,a)\|_{\bar{V}^{-1}_{n-1}}}_{\leq \Psi_n}.$$

The upper bound on the second term can be extracted from the proof of Lemma D.2.6. The first term can be finally related to the exploitation test:

$$\begin{aligned}
\inf_{\theta' \in \Theta_{alt}} \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} N^E_{n-1}(x,a) \bar{d}_{x,a}(\hat{\theta}_{n-1}, \theta') &\leq \inf_{\theta' \in \bar{\Theta}_{n-1}} \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} N^E_{n-1}(x,a) \bar{d}_{x,a}(\hat{\theta}_{n-1}, \theta') \\
&= \frac{1}{2\sigma^2} \inf_{\theta' \in \bar{\Theta}_{n-1}} \|\hat{\theta}_{n-1} - \theta'\|^2_{V_{n-1}} \\
&\leq \frac{1}{2\sigma^2} \inf_{\theta' \in \bar{\Theta}_{n-1}} \|\hat{\theta}_{n-1} - \theta'\|^2_{\bar{V}_{n-1}} \leq \frac{\beta_{n-1}}{2\sigma^2},
\end{aligned}$$

where the equality follows from (10.5), the second-last inequality holds since $\bar{V}_{n-1} \succeq V_{n-1}$, and the last inequality holds since the algorithm is exploring at step n. By gathering all the results together, we get

$$\begin{aligned}
\sum_{k \geq \underline{k}}^{K_n} \sum_{t \in \mathcal{T}^E_K : E_t} g_t(\omega_t, z_{K_t}) \leq \frac{\beta_{n-1}}{2\sigma^2} - \sum_{k \geq \underline{k}}^{K_n} \frac{p_k}{z_k} + \frac{2L^2B^2}{\sigma^2} M_n + \frac{6LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n + \frac{2B^2L^2}{\sigma^2}\zeta_n \\
+ \frac{2B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right). \qquad (10.24)
\end{aligned}$$

Back to the regret during exploration

So far we have (1) reduced the total regret during exploration to the sum of objective values (Equation 10.19), (2) related this quantity to the optimal values of each phase (Equation 10.22), and (3) derived an upper bound to the total sum of constraints (Equation 10.24). We now combine all these results. If we first plug (10.22) into (10.19),

$$\begin{aligned}
R^{\text{explore}}_n \leq 2LB \sum_{k < \underline{k}} p_k + \sum_{k \geq \underline{k}}^{K_n} p_k \mu^\star - \sum_{k \geq \underline{k}}^{K_n} p_k f(\omega^\star_{z_k}) + \lambda \sum_{k \geq \underline{k}}^{K_n} \sum_{t \in \mathcal{T}^E_k} g_t(\omega_t, z_k) + a_\lambda \sum_{k \geq \underline{k}}^{K_n} \sqrt{p_k} \\
+ (6LB + \lambda_{\max}) M_n + 2\sqrt{\gamma_n}\Psi_n + LB\left(2 + \frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_n + 2LB\sqrt{S_n \log S_n}. \qquad (10.25)
\end{aligned}$$

Then, plugging (10.24) into this inequality,

$$\begin{aligned}
R^{\text{explore}}_n \leq 2LB \sum_{k < \underline{k}} p_k + \sum_{k \geq \underline{k}}^{K_n} p_k \mu^\star - \sum_{k \geq \underline{k}}^{K_n} p_k f(\omega^\star_{z_k}) + \lambda \frac{\beta_{n-1}}{2\sigma^2} - \lambda \sum_{k \geq \underline{k}}^{K_n} \frac{p_k}{z_k} + a_\lambda \sum_{k \geq \underline{k}}^{K_n} \sqrt{p_k} \\
+ \left(\lambda \frac{2L^2B^2}{\sigma^2} + 6LB + \lambda_{\max}\right) M_n + \left(2 + \frac{6LB\lambda}{\sigma^2}\right)\sqrt{\gamma_n}\Psi_n + 2LB\sqrt{S_n \log S_n} \\
+ LB\left(2 + \frac{2LB(\lambda_{\max} + \lambda)}{\sigma^2}\right)\zeta_n + \frac{2\lambda B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right). \qquad (10.26)
\end{aligned}$$

Let us simplify this expression so that it becomes more readable. First, we note that

$$\sum_{k\geq\underline{k}}^{K_n} p_k\mu^\star - \sum_{k\geq\underline{k}}^{K_n} p_k f(\omega^\star_{z_k}) = \sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k}\underbrace{z_k(\mu^\star - f(\omega^\star_{z_k}))}_{=u^\star(z_k,\theta^\star)} = \sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k} u^\star(z_k,\theta^\star).$$

Taking the expectation of both sides, we obtain

$$\begin{aligned}\mathbb{E}\left[R_n^{\text{explore}}\right] \leq 2LB\sum_{k<\underline{k}} p_k + \mathbb{E}\left[\sum_{k\geq\underline{k}}^{K_n}\frac{p_k}{z_k}u^\star(z_k,\theta^\star)\right] + \lambda\frac{\beta_{n-1}}{2\sigma^2} - \lambda\mathbb{E}\left[\sum_{k\geq\underline{k}}^{K_n}\frac{p_k}{z_k}\right] \\ + a_\lambda\mathbb{E}\left[\sum_{k\geq\underline{k}}^{K_n}\sqrt{p_k}\right] + \mathbb{E}\left[\mathcal{O}(\sqrt{S_n\log S_n})\right].\end{aligned}$$

The remaining expectations on the right-hand side are due to the fact that K_n (hence S_n) is still random. Setting $\lambda = v^\star(\theta^\star)$ and combining the second and fourth terms, we get

$$\begin{aligned}\sum_{k\geq\underline{k}}^{K_n}\frac{p_k}{z_k}u^\star(z_k,\theta^\star) - \lambda\sum_{k\geq\underline{k}}^{K_n}\frac{p_k}{z_k} &= \sum_{k\geq\underline{k}}^{K_n}\frac{p_k}{z_k}\left(u^\star(z_k,\theta^\star) - v^\star(\theta^\star)\right) \\ &= \sum_{k\geq\underline{k}:z_k<\bar{z}(\theta^\star)}\frac{p_k}{z_k}\left(u^\star(z_k,\theta^\star) - v^\star(\theta^\star)\right) + \sum_{k:z_k\geq\bar{z}(\theta^\star)}^{K_n}\frac{p_k}{z_k}\left(u^\star(z_k,\theta^\star) - v^\star(\theta^\star)\right),\end{aligned}$$

where $\bar{z}(\theta^\star) := \max_{x\in\mathcal{X}}\sum_{a\neq a^\star_{\theta^\star}(x)}\frac{\eta^\star(x,a)}{\rho(x)}$ was defined in Lemma 10.3.1. For $k\geq\underline{k}$, we can use the perturbation bound (Lemma 10.3.1) on both terms. We obtain,

$$\sum_{k\geq\underline{k}:z_k<\bar{z}(\theta^\star)}\frac{p_k}{z_k}\left(u^\star(z_k,\theta^\star) - v^\star(\theta^\star)\right) \leq BL\underline{z}(\theta^\star)\sum_{k\geq\underline{k}:z_k<\bar{z}(\theta^\star)}\frac{p_k}{z_k - \underline{z}(\theta^\star)}$$

and

$$\sum_{\substack{k\geq\underline{k}:\\ z_k\geq\bar{z}(\theta^\star)}}^{K_n}\frac{p_k}{z_k}\left(u^\star(z_k,\theta^\star) - v^\star(\theta^\star)\right) \leq BL\underline{z}(\theta^\star)z^\star(\theta^\star)\sum_{\substack{k\geq\underline{k}:\\ z_k\geq\bar{z}(\theta^\star)}}^{K_n}\frac{p_k}{z_k - \underline{z}(\theta^\star)}\max\left\{\frac{c_\Theta\sqrt{2}}{\sigma\sqrt{z_k}}, \frac{1}{z_k}\right\}.$$

Partial regret bound Plugging these bounds into the expected regret,

$$\begin{aligned}\mathbb{E}\left[R_n^{\text{explore}}\right] \leq \underbrace{2BL\sum_{k<\underline{k}}p_k}_{\text{I}} + \underbrace{BL\underline{z}(\theta^\star)\sum_{\substack{k\geq\underline{k}:\\ z_k<\bar{z}(\theta^\star)}}\frac{p_k}{z_k - \underline{z}(\theta^\star)}}_{\text{II}} + \underbrace{v^\star(\theta^\star)\frac{\beta_{n-1}}{2\sigma^2}}_{\text{III}} + \underbrace{a_\lambda\mathbb{E}\left[\sum_{k\geq\underline{k}}^{K_n}\sqrt{p_k}\right]}_{\text{IV}} \\ \underbrace{+ BL\underline{z}(\theta)z^\star(\theta^\star)\mathbb{E}\left[\sum_{k:z_k\geq\bar{z}(\theta^\star)}^{K_n}\frac{p_k}{z_k - \underline{z}(\theta^\star)}\max\left\{\frac{c_\Theta\sqrt{2}}{\sigma\sqrt{z_k}}, \frac{1}{z_k}\right\}\right]}_{\text{V}} + \underbrace{\mathbb{E}\left[\mathcal{O}(\sqrt{S_n\log S_n})\right]}_{\text{VI}}.\end{aligned} \tag{10.27}$$

The six terms constituting the bound are (from left to right):

I. finite regret suffered in the phases where the optimization problem is infeasible;

II. finite regret suffered in the phases in which we do not know much about the convergence rate of $u^\star(z, \theta^\star)$ to $v^\star(\theta^\star)$. This term is likely an artefact of the analysis;

III. asymptotically-optimal regret rate;

IV. regret suffered due to the incremental gradient updates and inversely proportional to the step sizes;

V. regret suffered due to the fact that we solve (P_z) instead of (P);

VI. other low-order terms mostly due to the concentration bounds.

Note that, since $\beta_{n-1} = c_{n,1/n}$ and $c_{n,1/n} \to 2\sigma^2 \log n$ as $n \to \infty$,

$$\limsup_{n\to\infty} \frac{v^\star(\theta^\star)\beta_{n-1}}{2\sigma^2 \log n} = v^\star(\theta^\star),$$

which is the asymptotically-optimal regret rate as prescribed by (P).

Bounding the total number of phases

So far we proved an upper bound on the regret incurred during exploration which depends on the (random) number of phases. We now upper bound this random variable as a function of z_k and p_k. In particular, we achieve this by focusing on the constraints only. The intuition is that, if the primal-dual algorithm works, then the sequence of policies played cannot violate the constraints *at each phase* too much. At the same time, these policies cannot satisfy the constraints too much, otherwise the exploitation test would trigger and the algorithm would not be exploring at step n. Relating these two we obtain a bound on K_n.

Recall that, as we assumed before, n is an exploration step in which the good event G_n holds. Using (10.23) and the equations thereafter, we have

$$\begin{aligned}\inf_{\theta' \in \Theta_{alt}} &\sum_{t \le n : E_t} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star, \theta') \\ &\le \frac{\beta_{n-1}}{2\sigma^2} + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n + \frac{2B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right). \end{aligned} \tag{10.28}$$

where the last two terms are $\mathcal{O}(\sqrt{S_n \log S_n})$.

We now provide a lower-bound on the same quantity. Fix a phase index $k \ge \underline{k}$. From (10.21), we have

$$\begin{aligned}\sum_{t \in \mathcal{T}_k^E} (f_t(\omega_t) + \lambda g_t(\omega_t, z_k)) \ge p_k f(\omega_{z_k}^\star) - a_\lambda \sqrt{p_k} - (2LB + \lambda_{\max}) M_{n,k} \\ - LB\left(1 + \frac{2LB\lambda_{\max}}{\sigma^2}\right)\zeta_{n,k}, \end{aligned} \tag{10.29}$$

The left-hand side can be upper-bounded by using the optimism property to obtain the true objective and constraint. Regarding the objective function, we have

$$\begin{aligned}
&\sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) = \sum_{t\in\mathcal{T}_k^E:G_t} f_t(\omega_t) + \sum_{t\in\mathcal{T}_k^E:\neg G_t} f_t(\omega_t) \\
&\leq \sum_{\substack{t\in\mathcal{T}_k^E:\\ G_t}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \hat{\rho}_{t-1}(x)\omega_t(x,a)\bar{\mu}_{\hat{\theta}_{t-1}}(x,a) + \sum_{\substack{t\in\mathcal{T}_k^E:\\ \neg G_t}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \hat{\rho}_{t-1}(x)\omega_t(x,a)\bar{\mu}_{\hat{\theta}_{t-1}}(x,a) + \sqrt{\gamma_n}\Psi_{n,k} \\
&\leq \underbrace{\sum_{t\in\mathcal{T}_k^E:G_t} \sum_{x\in\mathcal{X}} \hat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\bar{\mu}_{\hat{\theta}_{t-1}}(x,a)}_{(a)} + BLM_{n,k} + \sqrt{\gamma_n}\Psi_{n,k}.
\end{aligned}$$

Regarding the sum over the good events, using Lemma D.2.4,

$$\begin{aligned}
(a) &\leq \sum_{t\in\mathcal{T}_k^E:G_t} \sum_{x\in\mathcal{X}} \hat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\mu_{\theta^\star}(x,a) + \sqrt{\gamma_n}\Psi_{n,k} \\
&\leq \sum_{t\in\mathcal{T}_k^E} \underbrace{\sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\mu_{\theta^\star}(x,a)}_{=f(\omega_t)} + BL \underbrace{\sum_{t\in\mathcal{T}_k^E:G_t} \sum_{x\in\mathcal{X}} |\hat{\rho}_t(x) - \rho(x)|}_{\zeta_{n,k}} + \sqrt{\gamma_n}\Psi_{n,k}.
\end{aligned} \tag{10.30}$$

Therefore,

$$\sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) \leq \sum_{t\in\mathcal{T}_k^E} f(\omega_t) + BL\zeta_{n,k} + 2\sqrt{\gamma_n}\Psi_{n,k} + BLM_{n,k}.$$

We can follow the same reasoning to upper bound the sum of constraints. Since the KLs are upper-bounded by $2B^2L^2/\sigma^2$,

$$\sum_{t\in\mathcal{T}_k^E} g_t(\omega_t, z_k) \leq \sum_{t\in\mathcal{T}_k^E} g(\omega_t, z_k) + \frac{2B^2L^2}{\sigma^2}\zeta_{n,k} + \frac{4BL}{\sigma^2}\sqrt{\gamma_n}\Psi_{n,k} + \frac{2B^2L^2}{\sigma^2}M_{n,k}.$$

Combining the bounds on f and g,

$$\begin{aligned}
\sum_{t\in\mathcal{T}_k^E} (f(\omega_t) + \lambda g(\omega_t, z_k)) \geq p_k f(\omega_{z_k}^\star) &- \left(3BL + \lambda_{\max} + \lambda\frac{2B^2L^2}{\sigma^2}\right) M_{n,k} - a_\lambda\sqrt{p_k} \\
&- 2BL\left(1 + \frac{(\lambda_{\max}+\lambda)BL}{\sigma^2}\right)\zeta_{n,k} - \left(2 + \frac{4BL\lambda}{\sigma^2}\right)\sqrt{\gamma_n}\Psi_{n,k}.
\end{aligned}$$

Let $\bar{\omega}_{t,k} := \frac{1}{p_k}\sum_{t\in\mathcal{T}_k^E}\omega_t$ be the average policy played in phase k. Since f is linear and g is concave, $\sum_{t\in\mathcal{T}_k^E}(f(\omega_t) + \lambda g(\omega_t, z_k)) \leq p_k f(\bar{\omega}_{t,k}) + \lambda p_k g(\bar{\omega}_{t,k}, z_k)$. We now set

$$\lambda = \begin{cases} 2\lambda_{\max} & \text{if } [g(\bar{\omega}_{t,k}, z_k)]_- \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $[x]_- = \min\{x, 0\}$. Therefore,

$$\begin{aligned}
p_k\left(f(\bar{\omega}_{t,k}) - f(\omega_{z_k}^\star) + 2\lambda_{\max}[g(\bar{\omega}_{t,k}, z_k)]_-\right) \geq &- \left(3BL + \lambda_{\max} + \lambda_{\max}\frac{4B^2L^2}{\sigma^2}\right) M_{n,k} \\
&- a_{\lambda_{\max}}\sqrt{p_k} - 2BL\left(1 + \frac{3\lambda_{\max}BL}{\sigma^2}\right)\zeta_{n,k} - \left(2 + \frac{8BL\lambda_{\max}}{\sigma^2}\right)\sqrt{\gamma_n}\Psi_{n,k}.
\end{aligned}$$

Lemma 10.3.3 together with Assumption 10.5.1 ensures that, for $k \geq \underline{k}$, $\lambda^\star(z_k, \theta^\star) \leq \lambda_{\max}$. Thus, we can apply Theorem 42 of Efroni et al. (2020) and obtain

$$p_k g(\bar{\omega}_{t,k}, z_k) \geq p_k [g(\bar{\omega}_{t,k}, z_k)]_- \geq -\left(3BL + \lambda_{\max} + \lambda_{\max}\frac{4B^2L^2}{\sigma^2}\right)\frac{M_{n,k}}{2\lambda_{\max}} - \frac{a_{\lambda_{\max}}\sqrt{p_k}}{2\lambda_{\max}} - 2BL\left(1 + \frac{3\lambda_{\max}BL}{\sigma^2}\right)\frac{\zeta_{n,k}}{2\lambda_{\max}} - \left(2 + \frac{8BL\lambda_{\max}}{\sigma^2}\right)\frac{\sqrt{\gamma_n}\Psi_{n,k}}{2\lambda_{\max}}.$$

Summing both sides over all phases,

$$\begin{aligned}\sum_{k\geq\underline{k}}^{K_n} p_k g(\bar{\omega}_{t,k}, z_k) &= \sum_{k\geq\underline{k}}^{K_n} p_k \left(\inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \bar{\omega}_{t,k}(x,a) d_{x,a}(\theta^\star, \theta') - \frac{1}{z_k}\right) \\ &= \sum_{k\geq\underline{k}}^{K_n} \left(\inf_{\theta'\in\Theta_{alt}} \sum_{t\in\mathcal{T}_k^E} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star, \theta') - \frac{p_k}{z_k}\right) \\ &\leq \inf_{\theta'\in\Theta_{alt}} \sum_{t\leq n: E_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a) d_{x,a}(\theta^\star, \theta') - \sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k}.\end{aligned}$$

Therefore,

$$\inf_{\theta'\in\Theta_{alt}} \sum_{t\leq n: E_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a) d_{x,a}(\theta^\star, \theta') \geq \sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k} - \left(3BL + \lambda_{\max} + \lambda_{\max}\frac{4B^2L^2}{\sigma^2}\right)\frac{M_n}{2\lambda_{\max}} - \frac{a_{\lambda_{\max}}\sqrt{p_k}}{2\lambda_{\max}} - 2BL\left(1 + \frac{3\lambda_{\max}BL}{\sigma^2}\right)\frac{\zeta_n}{2\lambda_{\max}} - \left(2 + \frac{8BL\lambda_{\max}}{\sigma^2}\right)\frac{\sqrt{\gamma_n}\Psi_n}{2\lambda_{\max}}.$$

Combining this with (10.28), we obtain the following inequality:

$$\begin{aligned}\sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k} \leq{}& \frac{\beta_{n-1}}{2\sigma^2} + \frac{a_{\lambda_{\max}}\sqrt{p_k}}{2\lambda_{\max}} + \frac{2LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n + \frac{2B^2L^2}{\sigma^2}\sqrt{dS_n \log(dS_n)} \\ &+ \left(3BL + \lambda_{\max} + \lambda_{\max}\frac{4B^2L^2}{\sigma^2}\right)\frac{M_n}{2\lambda_{\max}} \\ &+ 2BL\left(1 + \frac{3\lambda_{\max}BL}{\sigma^2}\right)\frac{\zeta_n}{2\lambda_{\max}} + \left(2 + \frac{8BL\lambda_{\max}}{\sigma^2}\right)\frac{\sqrt{\gamma_n}\Psi_n}{2\lambda_{\max}}.\end{aligned}$$

Recall that, by definition, $S_n = \sum_{k=0}^{K_n} p_k$. Furthermore, by Cauchy-Schwartz inequality, $\sum_{k=0}^{K_n}\sqrt{p_k} \leq \sqrt{K_n \sum_{k=0}^{K_n} p_k}$. Simplifying this a little,

$$\sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k} \leq \frac{\beta_{n-1}}{2\sigma^2} + \mathcal{O}\left(\sqrt{K_n \sum_{k=0}^{K_n} p_k}\right) + \mathcal{O}\left(\sqrt{\left(\sum_{k=0}^{K_n} p_k\right)\log\left(\sum_{k=0}^{K_n} p_k\right)}\right). \tag{10.31}$$

Choosing z_k and p_k

We choose the exponential schedule $z_k = z_0 e^k$ and $p_k = z_k e^{rk}$, where r will be specified later. The left-hand side of (10.31) is

$$\sum_{k\geq\underline{k}}^{K_n} \frac{p_k}{z_k} = \sum_{k\geq\underline{k}}^{K_n} e^{rk} \geq e^{rK_n},$$

while the right-hand side is

$$\frac{\beta_{n-1}}{2\sigma^2} + \mathcal{O}\left(\sqrt{K_n \sum_{k=0}^{K_n} e^{(r+1)k}}\right) + \mathcal{O}\left(\sqrt{\left(\sum_{k=0}^{K_n} e^{(r+1)k}\right) \log\left(\sum_{k=0}^{K_n} e^{(r+1)k}\right)}\right)$$
$$\leq \frac{\beta_{n-1}}{2\sigma^2} + \mathcal{O}\left(\sqrt{K_n^2 e^{(r+1)K_n}}\right).$$

For $r > 1$, the resulting inequality yields $K_n \leq \mathcal{O}(\frac{1}{r}\log\beta_{n-1})$, i.e., $K_n \leq \mathcal{O}(\frac{1}{r}\log\log n)$ by definition of β_{n-1}. Let us recall (10.27):

$$\mathbb{E}\left[R_n^{\text{explore}}\right] \leq \underbrace{2BL\sum_{k<\underline{k}} p_k}_{\text{I}} + \underbrace{BL\underline{z}(\theta^\star)\sum_{k\geq\underline{k}:z_k<\bar{z}} \frac{p_k}{z_k - \underline{z}(\theta^\star)}}_{\text{II}} + \underbrace{v^\star(\theta^\star)\frac{\beta_{n-1}}{2\sigma^2}}_{\text{III}} + \underbrace{a_\lambda \mathbb{E}\left[\sum_{k\geq\underline{k}}^{K_n}\sqrt{p_k}\right]}_{\text{IV}}$$
$$+ \underbrace{BL\underline{z}(\theta)z^\star(\theta^\star)\mathbb{E}\left[\sum_{k:z_k\geq\bar{z}(\theta^\star)}^{K_n} \frac{p_k}{z_k - \underline{z}(\theta^\star)}\max\left\{\frac{c_\Theta\sqrt{2}}{\sigma\sqrt{z_k}}, \frac{1}{z_k}\right\}\right]}_{\text{V}} + \underbrace{\mathbb{E}\left[\mathcal{O}(\sqrt{S_n\log S_n})\right]}_{\text{VI}}. \tag{10.32}$$

We bound the remaining terms separately.

Term I

$$\sum_{k<\underline{k}} p_k = z_0\sum_{k<\underline{k}} e^{(r+1)k} \leq z_0 e^{(r+1)\log\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right)}\log\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right) = z_0\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right)^{r+1}\log\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right),$$

where we used that, from the definition of $\underline{k}$ and z_k, it must be that $k < \log(2\underline{z}(\theta^\star)/z_0)$. Thus, $\text{I} \leq 2BLz_0(2\underline{z}(\theta^\star)/z_0)^{r+1}\log(2\underline{z}(\theta^\star)/z_0)$.

Term II

$$\sum_{k\geq\underline{k}:z_k<\bar{z}(\theta^\star)} \frac{p_k}{z_k - \underline{z}(\theta^\star)} = \sum_{\log\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right)\leq k<\log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)} \frac{z_0 e^{(r+1)k}}{z_0 e^k - \underline{z}(\theta^\star)}$$
$$= \sum_{\log\left(\frac{2\underline{z}(\theta^\star)}{z_0}\right)\leq k<\log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)} \underbrace{\frac{z_0 e^k}{z_0 e^k - \underline{z}(\theta^\star)}}_{\leq 2} e^{rk} \leq 2(\bar{z}(\theta^\star)/z_0)^r\log(\bar{z}(\theta^\star)/z_0).$$

Thus,

$$\text{II} \leq 2BL\underline{z}(\theta^\star)(\bar{z}(\theta^\star)/z_0)^r\log(\bar{z}(\theta^\star)/z_0).$$

Term IV The total number of exploration rounds is

$$S_n = \sum_{k=0}^{K_n} p_k = z_0\sum_{k=0}^{K_n} e^{(r+1)k} \leq z_0 e^{(r+1)(K_n+1)} \leq \mathcal{O}((\log n)^{\frac{r+1}{r}}).$$

Therefore,

$$\mathrm{IV} \leq \sqrt{K_n \sum_{k=0}^{K_n} p_k} \leq \mathcal{O}((\log\log n)^{1/2} (\log n)^{\frac{r+1}{2r}}).$$

Term V We consider two cases, based on which of the inner terms is the maximum. In the first case, we need to bound

$$\begin{aligned}
\sum_{k:z_k \geq \bar{z}(\theta^\star)}^{K_n} \frac{p_k}{(z_k - \underline{z}(\theta^\star))\sqrt{z_k}} &= \sum_{k \geq \log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)}^{K_n} \frac{z_0 e^{(r+1)k}}{(z_0 e^k - \underline{z}(\theta^\star))\sqrt{z_0 e^k}} \\
&= \frac{1}{\sqrt{z_0}} \sum_{k \geq \log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)}^{K_n} \underbrace{\frac{z_0 e^k}{(z_0 e^k - \underline{z}(\theta^\star))}}_{\leq 2} e^{(r-1/2)k} \leq \frac{2}{\sqrt{z_0}} \sum_{k \geq \log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)}^{K_n} e^{(r-1/2)k} \\
&\leq \frac{2}{\sqrt{z_0}} \int_{\log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)}^{K_n+1} e^{(r-1/2)k} \mathrm{d}k = \frac{2}{\sqrt{z_0}} \left[\frac{e^{(r-1/2)k}}{r-1/2}\right]_{\log\left(\frac{\bar{z}(\theta^\star)}{z_0}\right)}^{K_n+1} \\
&= \frac{2}{(r-1/2)\sqrt{z_0}} \left(e^{(r-1/2)(K_n+1)} - (\bar{z}(\theta^\star)/z_0)^{r-1/2}\right).
\end{aligned}$$

Since $K_n \leq \mathcal{O}(\frac{1}{r}\log\log n)$, this term is $\mathcal{O}((\log n)^{\frac{r-1/2}{r}})$. If the other term is the maximum, then the same procedure yields a $\mathcal{O}((\log n)^{\frac{r-1}{r}})$ dependency. Thus,

$$\mathrm{V} \leq \mathcal{O}((\log n)^{\frac{r-1/2}{r}}).$$

Term VI We have $\mathrm{VI} \leq \mathcal{O}((\log n)^{\frac{r+1}{2r}})$ as in Term IV.

Final Bound Using $r = 2$, we obtain the following bound on the expected regret during exploration:

$$\begin{aligned}
\mathbb{E}\left[R_n^{\mathrm{explore}}\right] &\leq 2BLz_0(2\underline{z}(\theta^\star)/z_0)^3 \log(2\underline{z}(\theta^\star)/z_0) \\
&\quad + 2BL\underline{z}(\theta^\star)(\bar{z}(\theta^\star)/z_0)^2 \log(\bar{z}(\theta^\star)/z_0) + v^\star(\theta^\star)\frac{\beta_{n-1}}{2\sigma^2} + \mathcal{O}((\log\log n)^{\frac{1}{2}}(\log n)^{\frac{3}{4}}),
\end{aligned}$$

which is asymptotically optimal.

Numerical Simulations

We compare SOLID to LinUCB, LinTS, and OAM. For SOLID, we set $\beta_t = \sigma^2(\log(t) + d\log\log(n))$ and $\gamma_t = \sigma^2(\log(S_t) + d\log\log(n))$ (i.e., we remove all numerical constants) and we use the exponential schedule for phases defined in Theorem 10.5.1. For OAM, we use the same β_t for the explore/exploit test and we try different values for the forced-exploration parameter ϵ. LinUCB uses the confidence intervals from Theorem 2 in Abbasi-Yadkori et al. (2011) with the log-determinant of the design matrix, and LinTS is as defined

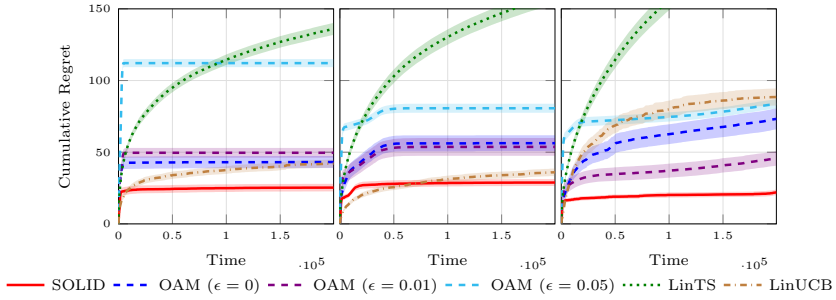

Figure 10.1: *Toy problem with 2 contexts and (left)* $\rho(x_1) = 0.5$*, (center)* $\rho(x_1) = 0.9$*, (right)* $\rho(x_1) = 0.99$*.*

in Agrawal and Goyal (2013) but without the extra-sampling factor $\sqrt{d}$ used to prove its frequentist regret. Both are without numerical constants as for SOLID. All plots are the results of 100 runs with 95% Student's t confidence intervals.

Synthetic Problems

Toy contextual linear bandit with structure. We start with a problem with $|\mathcal{X}| = 2$ and $|\mathcal{A}|, d = 3$. Let x_i (a_i) be the i-th context (arm). We have $\phi(x_1, a_1) = [1, 0, 0]$, $\phi(x_1, a_2) = [0, 1, 0]$, $\phi(x_1, a_3) = [1 - \xi, 2\xi, 0]$, $\phi(x_2, a_1) = [0, 0.6, 0.8]$, $\phi(x_2, a_2) = [0, 0, 1]$, $\phi(x_2, a_3) = [0, \xi/10, 1 - \xi]$ and $\theta^\star = [1, 0, 1]$. We consider a balanced context distribution $\rho(x_1) = \rho(x_2) = 0.5$. This is a two-context counterpart of the example presented by (Lattimore and Szepesvari, 2017) to show the asymptotic sub-optimality of optimism-based strategies. The intuition is that, for ξ small, an optimistic strategy pulls a_2 in x_1 and a_1 in x_2 only a few times since their gap is quite large, and suffers high regret (inversely proportional to ξ) to figure out which of the remaining arms is optimal. On the other hand, an asymptotically optimal strategy allocates more pulls to "bad" arms as they bring information to identify $\theta^\star$, which in turns avoids a regret scaling with ξ. This indeed translates into the empirical performance reported in Figure 10.1-(left), where SOLID effectively exploits the structure of the problem and significantly reduces the regret compared to LinTS and LinUCB. Actually, not only the regret is smaller but the "trend" is better. In fact, the regret curves of LinUCB and LinTS have a larger slope than SOLID's, suggesting that the gap may increase further with n, thus confirming the theoretical finding that the asymptotic performance of SOLID is better. OAM has a similar behavior, but the actual performance is worse than SOLID and it seems to be very sensitive to the forced exploration parameter, where the best performance is obtained for $\epsilon = 0.0$, which is not theoretically justified.

We also study the influence of the context distribution. We first notice that solving (P) leads to an optimal exploration strategy $\eta^\star$ where the only sub-optimal arm with non-zero pulls is a_1 in x_2 since it yields lower regret and similar information than a_2 in x_1. This means that the lower bound prescribes a greedy policy in x_1, deferring exploration to x_2 alone. In practice, tracking this optimal allocation might lead to poor finite-time performance when the context distribution is unbalanced towards x_1, in which case the algorithm would take time proportional to $1/\rho(x_2)$ before performing any meaningful ex-

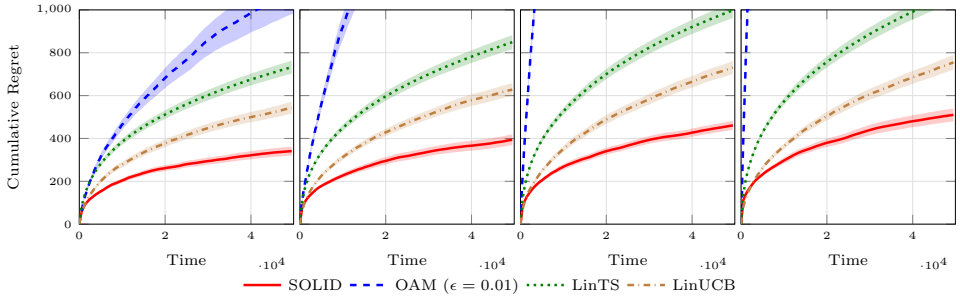

Figure 10.2: *Randomly generated bandit problems with* $d = 8, |\mathcal{X}| = 4$*, and* $|\mathcal{A}| = 4, 8, 16, 32$.

ploration. We verify these intuitions empirically by considering the case of $\rho(x_1) = 0.9$ and $\rho(x_1) = 0.99$ (middle and right plots in Figure 10.1 respectively). SOLID is consistently better than all other algorithms, showing that its performance is not negatively affected by $\rho_{\min}$. On the other hand, OAM is more severely affected by the context distribution. In particular, its performance with $\epsilon = 0$ significantly decreases when increasing $\rho(x_1)$ and the algorithm reduces to an almost greedy strategy, thus suffering linear regret in some problems. In this specific case, forcing exploration leads to slightly better finite-time performance since the algorithm pulls the informative arm a_2 in x_1, which is however not prescribed by the lower bound.

Random problems. We evaluate the impact of the number of actions $|\mathcal{A}|$ in randomly generated structured problems with $d = 8$ and $|\mathcal{X}| = 4$. We run each algorithm for $n = 50000$ steps. For OAM, we set forced-exploration $\epsilon = 0.01$ and solve (P) every 100 rounds to speed-up execution as computation becomes prohibitive. The plots in Figure 10.2 show the regret over time for $|\mathcal{A}| = 4, 8, 16, 32$. This test confirms the advantage of SOLID over the other methods. Interestingly, the regret of SOLID does not seem to significantly increase as a function of $|\mathcal{A}|$, thus supporting its theoretical analysis. On the other hand, the regret of OAM scales poorly with $|\mathcal{A}|$ since forced exploration pulls all arms in a round robin fashion.

Real Data

We use the Jester Dataset (Goldberg et al., 2001) which consists of joke ratings in a continuous range from -10 to 10 for a total of 100 jokes and 73421 users. We select a subset of 40 jokes and 19181 users rating all these 40 jokes.

We build a linear contextual problem as follows. We first extract separate 36-dimensional user (context) and joke (arm) features via a low-rank matrix factorization. Then, we concatenate these user and joke features (thus obtaining vectors with 72 entries) and fit a 64×64 neural-network with ReLU non-linearities to predict the ratings of a random subset of 75% of the users, using these feature vectors as inputs. We obtain $R^2 \simeq 0.95$ on the remaining 25% users. Finally, we take the features extracted in the last layer of the network as the features for our bandit problem and the parameters of the same layer as $\theta^\star$. Rewards in our bandit problem are generated from this linear model by perturbing the

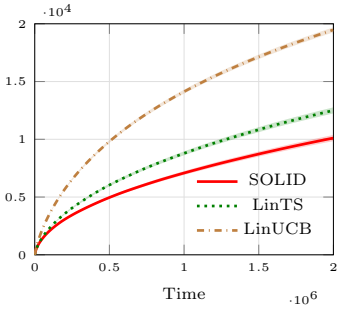

Figure 10.3: *Experiment on a real dataset (Jester).*

prediction with $\mathcal{N}(0, 0.5^2)$ noise. We thus obtain a problem with $d = 65$ (the 64 hidden neurons plus the bias term), 40 arms (the jokes), and a total of 19181 users.

We run the algorithms for $2 \cdot 10^6$ steps, with each run randomizing a subset of 1% of the total users (hence $|\mathcal{X}| = 191$) and using all 40 arms. For SOLID, we use the same parameters as in the experiment with random models. Due to the computational bottleneck demonstrated in the previous experiments, we could not run OAM on this problem. The results are shown in Figure 10.3 and confirm that SOLID achieves superior performance than the other baselines.

CHAPTER 11

Best Policy Identification in MDPs with Misspecified Structure

This chapter is based on the paper "Sequential Transfer in Reinforcement Learning with a Generative Model" co-authored with Riccardo Poiani and Marcello Restelli and published at ICML 2020.

Introduction

So far we have studied the problem of exploration in structured domains under the assumption of *known* structure. Unfortunately, this assumption rarely holds when the structure itself is learned from experience, e.g., from a sequence of tasks as in Section 8.3. In fact, as in any estimation problem, one must suffer an *estimation error* due the structure being learned from a finite set of previously-faced tasks (Brunskill and Li, 2013; Azar et al., 2013a) and possibly an *approximation error* due to a wrong modeling of the space of possible structures (see, e.g., Ghosh et al. (2017) for linear bandits). When the structure provided as input to the agent is *misspecified* (i.e., it is only an approximation of the true one), one must be more careful in exploiting it to bias the learning process. In fact, directly using a highly-misspecified structure as we have seen in the previous chapters could lead to the learning process being worse than learning without prior knowledge, that is, to *negative transfer*. Here we would expect a good agent to be able to take advantage of the structure whenever the level of misspecification is small, while falling back to the "no-transfer" case, without suffering any performance loss, whenever the misspecification

is large.

In this chapter, we study the problem of learning with misspecified structure in the context of *best policy identification* in MDPs, where the goal is to actively interact with the target MDP so as so identify a near-optimal policy while suffering a small sample complexity. In particular, we represent the structured domain as a finite set of possible MDPs that contains the one to be faced, and suppose that the agent is provided only with an approximation to this MDPs. For this structured/transfer setting, two classes of approaches exist. Approaches in the first class (Mann and Choe, 2013; Abel et al., 2018) seek *jump-start* initializers. The idea is to use the given structure to compute a provably optimistic initialization for any existing PAC algorithm (Strehl et al., 2009) which is directly used to learn a near-optimal policy for the target task. Assuming this initialization to be tighter than the one used when learning from scratch, the overall sample complexity is reduced. However, since these initializers are computed before receiving the new task, the agent, though starting from good performance, might still take a long time before converging to near-optimal behavior. An alternative approach is to spend some initial interactions with the target to gather information about the task itself, so as to better decide what to transfer. For instance, the agent could aim at identifying which of the MDPs in the approximate structure is most similar to the target. If the similarity is actually large, transferring its optimal policy would instantaneously lead to near-optimal behavior. This *task identification* problem has been studied by Dyagilev et al. (2008); Brunskill and Li (2013); Liu et al. (2016b). One downside is that these approaches do not actively seek information to reduce identification time (or, equivalently, sample complexity), in part because it is non-trivial to find which actions are the most informative given misspecified structural knowledge. In fact, as we have seen for bandits, when provided with structure, different state-action pairs might provide very different information about the underlying MDP model and, thus, about its solution.

In this chapter, we follow this second line of works. We propose Policy Transfer from Uncertain Models (PTUM), an algorithm that, under the availability of a generative model, actively explores the target MDP to identify a near-optimal policy while using the misspecified structure to reduce the sample complexity for this purpose. Our approach sheds light on how to actively explore the target MDP when provided with imprecise structure, i.e., on what state-action pairs are most informative for discriminating between imprecise MDP models. Furthermore, it can be readily combined with a structure learning mechanism in a multi-task/sequential transfer setting (see Section 8.3).[1] The idea behind PTUM is quite simple. The algorithm maintains a confidence set of all MDP models in the given structure that are "compatible" with the previously-observed samples. At each step, it queries the state-action pair where two MDP models in the confidence set are maximally distant, with the distance computed both in terms of their rewards and transition probabilities. This is used as a proxy for the state-action pair whose samples would be maximally informative for discriminating any "wrong" MDP from the true one being faced. We show that the sample complexity of PTUM is proportional to the one of an oracle strategy visiting the state-action pairs that yield the maximum information for discriminating between models, despite these state-action pairs being unknown a-priori.

[1] In the original paper (Tirinzoni et al., 2020c), we actually show how to combine PTUM with a spectral method for structure learning in a sequential setting with temporally-correlated tasks. Since this result mostly builds on top of the work of Azar et al. (2013a), it was not reported in this document.

Finally, we empirically verify our theoretical results, in particular by studying the different behaviors of identification and (optimistic) jumpstart strategies.

MDPs with Misspecified Structure

We consider a family of parameterized MDPs $\mathfrak{M}_\Theta = \{\mathcal{M}_\theta\}_{\theta\in\Theta}$, $\mathcal{M}_\theta = (\mathcal{S}, \mathcal{A}, P_\theta, U_\theta, \gamma)$. For simplicity, we assume that the set of realizable parameters Θ (and thus $\mathfrak{M}_\Theta$) is finite, though our approach and its analysis easily extend to the case of infinite/continuous parameters as in Chapter 9. We assume, without loss of generality, that reward distributions U_θ are supported in $[0, 1]$ for all MDPs in Θ and denote by r_θ the corresponding mean rewards. We use an infinite-horizon discounted formulation and denote by $V^\pi_\theta(s)$ the value function of a (deterministic) policy π in MDP $\mathcal{M}_\theta$, and similarly for $V^\star_\theta(s)$. We define $\sigma^r_\theta(s,a)^2 := \mathbb{V}\mathrm{ar}_{U_\theta(\cdot|s,a)}[R]$ as the variance of the reward in s, a for task $\mathcal{M}_\theta$, and $\sigma^p_\theta(s,a;\theta')^2 := \mathbb{V}\mathrm{ar}_{P_\theta(\cdot|s,a)}[V^\star_{\theta'}(S')]$ as the variance of the optimal value function of $\mathcal{M}_{\theta'}$ under the transition model of $\mathcal{M}_\theta$. To simplify the exposition, we shall alternatively use the standard vector notation. For instance, $V^\star_\theta \in \mathbb{R}^{|\mathcal{S}|}$ is the vector of optimal values, $p_\theta(s,a) \in \mathbb{R}^{|\mathcal{S}|}$ is the vector of transition probabilities from s, a, $r_\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the flattened reward matrix, and so on. We the define the following two measures of distance between two MDPs $\mathcal{M}_\theta, \mathcal{M}_{\theta'}$:

$$\Gamma^r_{s,a}(\theta,\theta') := |r_\theta(s,a) - r_{\theta'}(s,a)|, \tag{11.1}$$

$$\Gamma^p_{s,a}(\theta,\theta') := |(p_\theta(s,a) - p_{\theta'}(s,a))^T V^\star_\theta|. \tag{11.2}$$

This measures how much the expected return of an agent taking s, a and acting optimally in $\mathcal{M}_\theta$ changes when the first transition is under $\mathcal{M}_{\theta'}$.

Problem Setting. The agent faces an unknown task $\theta^\star \in \Theta$. Differently from the previous chapters, the agent has not access to the exact set of models $\mathfrak{M}_\Theta$ but only to an approximation $\widetilde{\mathfrak{M}}_\Theta = \{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, where we overload the notation by using tildes to indicate quantities related to approximate models. We assume that an upper bound to the approximation error is known.

Assumption 11.2.1 (Maximum approximation error)**.** *There exists a known scalar $\Delta \geq 0$ such that, for each $\theta \in \Theta$,*

$$\max_{s\in\mathcal{S},a\in\mathcal{A}} |r_\theta(s,a) - \tilde{r}_\theta(s,a)| \leq \Delta,$$

$$\max_{s\in\mathcal{S},a\in\mathcal{A}} \max_{\theta'\in\Theta} |(p_\theta(s,a) - \tilde{p}_\theta(s,a))^T \tilde{V}^\star_{\theta'}| \leq \Delta,$$

$$\max_{s\in\mathcal{S},a\in\mathcal{A}} |\sigma^r_\theta(s,a) - \tilde{\sigma}^r_\theta(s,a)| \leq \Delta,$$

$$\max_{s\in\mathcal{S},a\in\mathcal{A}} \max_{\theta'\in\Theta} |\sigma^p_\theta(s,a;\theta') - \tilde{\sigma}^p_\theta(s,a;\theta')| \leq \Delta.$$

The last two conditions are needed since the algorithm we shall propose explicitly considers the local variances to find informative state-action pairs. We note that this assumption can be satisfied in practice when learning the MDP models with suitable techniques, such as spectral methods (Azar et al., 2013a). Given $\epsilon, \delta > 0$ as input, the agent's goal is to

Algorithm 9 Policy Transfer from Uncertain Models (PTUM)

Require: Set of approximate MDPs $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, accuracy ϵ, confidence δ, number of samples n, maximum approximation error Δ

Ensure: An ϵ-optimal policy for $\mathcal{M}_{\theta^\star}$ with probability $1-\delta$

Initialize: $\tilde{\Theta}_0 \leftarrow \Theta$
// CHECK ACCURACY CONDITION
If $\Delta \geq \frac{\epsilon(1-\gamma)}{4(1+\gamma)} \rightarrow$ Stop and run (ϵ,δ)-PAC algorithm
// TRANSFER MODE
for $t = 1, 2, \dots, n$ **do**
  // STEP 1. BUILD EMPIRICAL MDP MODEL
  $\hat{r}_t(s,a) \leftarrow \frac{1}{N_{t-1}(s,a)} \sum_{h=1}^{t-1} \mathbb{1}\{S_h = s, A_h = a\} R_h$
  $\hat{p}_t(s'|s,a) \leftarrow \frac{1}{N_{t-1}(s,a)} \sum_{h=1}^{t-1} \mathbb{1}\{S_h = s, A_h = a, S'_h = s'\}$
  $\hat{\sigma}_t^r(s,a)^2 \leftarrow \frac{1}{N_{t-1}(s,a)-1} \sum_{h=1}^{t-1} \mathbb{1}\{S_h = s, A_h = a\} (R_h - \hat{r}_t(s,a))^2$
  $\hat{\sigma}_t^p(s,a;\theta')^2 \leftarrow \frac{1}{N_{t-1}(s,a)-1} \sum_{h=1}^{t-1} (\tilde{V}^\star_{\theta'}(S'_h) - \hat{p}_t(s,a)^\intercal \tilde{V}^\star_{\theta'})^2$
  // STEP 2. UPDATE CONFIDENCE SET
  $\tilde{\Theta}_t \leftarrow \left\{\theta \in \tilde{\Theta}_{t-1} \;\middle|\; \text{(11.3)-(11.6) hold for all } s,a \text{ and } \theta' \in \Theta\right\}$
  // STEP 3. CHECK STOPPING CONDITION
  If there exists $\theta \in \tilde{\Theta}_t$ such that for all $\theta' \in \tilde{\Theta}_t$ and $s \in \mathcal{S}$ we have $\tilde{V}^{\tilde{\pi}^\star_\theta}_{\theta'}(s) \geq \tilde{V}^\star_{\theta'}(s) - \epsilon + \frac{2\Delta(1+\gamma)}{1-\gamma} \rightarrow$ Stop and return $\tilde{\pi}^\star_\theta$
  // STEP 4. QUERY GENERATIVE MODEL
  $\mathcal{I}_t^r(s,a) \leftarrow \max_{\theta,\theta'\in\tilde{\Theta}_t} \mathcal{I}^r_{s,a}(\theta,\theta')$
  $\mathcal{I}_t^p(s,a) \leftarrow \max_{\theta,\theta'\in\tilde{\Theta}_t} \mathcal{I}^p_{s,a}(\theta,\theta')$
  $(S_t, A_t) \leftarrow \operatorname{argmax}_{s,a} \max\{\mathcal{I}_t^r(s,a), \mathcal{I}_t^p(s,a)\}$
  Obtain $S'_t \sim P_{\theta^\star}(\cdot|S_t,A_t)$ and $R_t \sim U_{\theta^\star}(\cdot|S_t,A_t)$
end for
If the algorithm did not stop $\rightarrow$ Run (ϵ,δ)-PAC algorithm

identify a policy that, with probability at least $1-\delta$, is ϵ-optimal for $\mathcal{M}_{\theta^\star}$. We recall that a policy π is ϵ-optimal for $\mathcal{M}_\theta$ if $V_\theta^\pi(s) \geq V_\theta^\star(s) - \epsilon$ for all states s. We assume that the agent can access a generative model of state-action pairs for $\mathcal{M}_{\theta^\star}$. Similarly to experimental optimal design (Pukelsheim, 2006), the agent can perform at most $n > 0$ *experiments* for identification, where each experiment consists in choosing an arbitrary state-action pair s, a and receiving a random next-state $S' \sim P_{\theta^\star}(\cdot|s,a)$ and reward $R \sim U_{\theta^\star}(\cdot|s,a)$. After this *identification phase*, the agent, if possible, has to output an ϵ-optimal policy and starts interacting with the environment in the standard online fashion.

Policy Transfer from Uncertain Models

We introduce Policy Transfer from Uncertain Models (PTUM),[2] whose pseudo-code is provided in Algorithm 9. Given the approximate models $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, whose maximum

[2]The name "policy transfer" derives from the sequential transfer setting in the original paper (Tirinzoni et al., 2020c) where the agent approximates the set of MDPs using the source tasks and uses PTUM to quickly identify a near-optimal policy to transfer to the target.

error is bounded by Δ from Assumption 11.2.1, and two values $\epsilon, \delta > 0$, PTUM returns a policy which, with probability at least $1 - \delta$, is ϵ-optimal for the target task $\mathcal{M}_{\theta^\star}$. The main intuition behind how the algorithm exploits structure is that, if the approximate models are accurate enough so that one is close to the target $\mathcal{M}_{\theta^\star}$, then it is possible to restrict the search for an ϵ-optimal policy to the set of optimal policies $\{\tilde{\pi}^\star_\theta\}_{\theta\in\Theta}$ for the MDPs in $\widetilde{\mathfrak{M}}_\Theta$. In order to achieve this, PTUM starts by checking if the approximate models are too inaccurate (i.e., Δ is large) in line 3. If this is the case, the transfer of a policy among the optimal ones of $\widetilde{\mathfrak{M}}_\Theta$ might actually lead to poor performance in the target task, i.e., *negative transfer* occurs. In this case, PTUM stops and it falls back to not exploiting structure at all by running any (ϵ, δ)-PAC[3] algorithm to obtain an ϵ-optimal policy. Although the condition at line 3 seems restrictive, as Δ is required to be below a factor of ϵ, we conjecture this dependency to be nearly-tight (at least in a worst-case sense). In fact, Feng et al. (2019) have recently shown that the sole knowledge of a poorly-approximate model cannot reduce the worst-case sample complexity of any agent seeking an ϵ-optimal policy. If the condition at line 3 fails, i.e., the models are accurate enough, we say that the algorithm enters the *transfer mode*. That is, the approximate models can be safely relied on and PTUM seeks one of their optimal policies to transfer. Here, the generative model is queried online until an ϵ-optimal policy is found. Similarly to existing works on model identification Dyagilev et al. (2008); Brunskill and Li (2013), the algorithm proceeds by elimination. At each time-step t (up to at most n), we keep a set $\tilde{\Theta}_t \subseteq \Theta$ of those models, called *active*, that are likely to be (close approximations of) the target $\theta^\star$. Then, the algorithm chooses the next state-action pair S_t, A_t to query the generative model so that the samples from S_t, A_t are informative to eliminate one of the "wrong" models from the active set. This process is iterated until the algorithm finds a policy that is ϵ-optimal for all active models, in which case the algorithm stops and returns such policy, or until n samples are reached, in which case the algorithm runs any (ϵ, δ)-PAC method. We shall discuss later how to make use of the information collected with the generative model whenever PTUM does not stop in the allowed sample budget. We now describe these main steps in detail.

Step 1. Building the empirical MDP. In order to find the set of active models $\tilde{\Theta}_t$ at time t, the algorithm builds an empirical MDP as a proxy for the true one. Let $N_{t-1}(s, a) := \sum_{h=1}^{t-1} \mathbb{1}\{S_h = s, A_h = a\}$ be the number of samples collected from s, a up to time $t - 1$. First, the algorithm estimates, for each s, a, the empirical rewards $\hat{r}_t(s, a)$ and transition probabilities $\hat{p}_t(s, a)$ (lines 7-8). If $N_{t-1}(s, a) = 0$, these quantities are arbitrarily initialized. Then, it computes the empirical variance of the rewards $\hat{\sigma}^r_t(s, a)^2$ and of the optimal value functions $\hat{\sigma}^p_t(s, a; \theta')^2$ for each $\theta' \in \Theta$ (lines 9-10). Similarly, if $N_{t-1}(s, a) < 2$, these quantities and arbitrarily initialized.

Step 2. Building the confidence set We define the confidence set $\tilde{\Theta}_t$ as the set of models that are "compatible" with the empirical MDP in *all* steps up to t. Formally, a model $\theta \in \Theta$ belongs to the confidence set $\tilde{\Theta}_t$ at time t if it was active before (i.e., $\theta \in \tilde{\Theta}_{t-1}$) and the

[3]We recall that an algorithm is (ϵ, δ)-PAC if, with probability $1 - \delta$, it computes an ϵ-optimal policy using a polynomial number of samples in the relevant problem-dependent quantities Strehl et al. (2009).

following conditions are satisfied for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $\theta' \in \Theta$:

$$|\hat{r}_t(s,a) - \tilde{r}_\theta(s,a)| \le C^r_{t,\delta}(s,a), \tag{11.3}$$

$$|(\hat{p}_t(s,a) - \tilde{p}_\theta(s,a))^T \tilde{V}^\star_{\theta'}| \le C^p_{t,\delta}(s,a,\theta'), \tag{11.4}$$

$$|\hat{\sigma}^r_t(s,a) - \tilde{\sigma}^r_\theta(s,a)| \le C^{\sigma_r}_{t,\delta}(s,a), \tag{11.5}$$

$$|\hat{\sigma}^p_t(s,a;\theta') - \tilde{\sigma}^p_\theta(s,a;\theta')| \le C^{\sigma_p}_{t,\delta}(s,a). \tag{11.6}$$

Intuitively, an approximate model belongs to the confidence set if its distance to the empirical MDP does not exceed, in any component, a suitable confidence interval $C_{t,\delta}(s,a)$. Alternatively, we say that a model is *eliminated* from the confidence set (i.e., it will never be active again) as soon as it is not compatible with the empirical MDP. These confidence intervals are obtained from standard applications of Bernstein's inequality Boucheron et al. (2003) as follows:

$$C^r_{t,\delta}(s,a) := \sqrt{\frac{2\hat{\sigma}^r_t(s,a)^2 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{7 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} + \Delta,$$

$$C^p_{t,\delta}(s,a) := \sqrt{\frac{2\hat{\sigma}^p_t(s,a;\theta')^2 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{\frac{7}{1-\gamma} \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} + \Delta,$$

$$C^{\sigma^r}_{t,\delta}(s,a) := \sqrt{\frac{2 \log \frac{4|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)-1}} + \Delta,$$

$$C^{\sigma^p}_{t,\delta}(s,a) := \frac{1}{1-\gamma}\sqrt{\frac{2 \log \frac{4|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)-1}} + \Delta.$$

We set the confidence intervals to infinity if $N_t(x,a) < 2$. As we shall see, these confidence values are chosen so that, with probability at least $1-\delta$, the target task $\theta^\star$ is never eliminated from $\tilde{\Theta}_t$. We note that the dependences on the sample budget n can be removed by using adaptive (e.g., $\log t$) schedules.

Step 3. Checking whether to stop After building the confidence set, the algorithm checks whether the optimal policy of some active model is $(\epsilon - 2\Delta\frac{1+\gamma}{1-\gamma})$-optimal in all other models in $\tilde{\Theta}_t$, in which case it stops and returns this policy. The extra $2\Delta\frac{1+\gamma}{1-\gamma}$ factor is needed to handle the model approximation error and, as we shall see in our analysis, it ensures that the returned policy is also ϵ-optimal for $\mathcal{M}_{\theta^\star}$.

Step 4. Deciding where to query the generative model The final step involves choosing the next state-action pair S_t, A_t from which to obtain a sample. This is a key point as the sampling rule is what directly determines the sample-complexity of the algorithm. As discussed previously, our algorithm eliminates the models from $\tilde{\Theta}_t$ until the stopping

condition is verified. Therefore, a good sampling rule should aim at minimizing the stopping time, i.e., it should aim at eliminating as soon as possible all models that prevent the algorithm from stopping. The design of our strategy is driven by this principle. Given the set of active models $\tilde{\Theta}_t$, we compute, for each s, a, a score $\mathcal{I}_t(s, a)$, which we refer to as the *index* of s, a, that is directly related to the information to discriminate between any two active models using s, a only (lines 16-18). Then, we choose the s, a that maximizes the index, which can be interpreted as sampling the state-action pair that allows us to discard an active model in the shortest possible time. We confirm this intuition in our analysis later. Formally, our information measure is defined as follows.

Definition 11.3.1 (Information for model discrimination)**.** *The information for discriminating between any two models θ and θ' using reward/transition samples from s, a are, respectively,*

$$\mathcal{I}^r_{s,a}(\theta, \theta') = \min \left\{ \left(\frac{\tilde{\Gamma}^r_{s,a}(\theta, \theta') - 8\Delta}{\tilde{\sigma}^r_\theta(s, a)} \right)^2 , \tilde{\Gamma}^r_{s,a}(\theta, \theta') - 8\Delta \right\},$$

$$\mathcal{I}^p_{s,a}(\theta, \theta') = \min \left\{ \left(\frac{\tilde{\Gamma}^p_{s,a}(\theta, \theta') - 8\Delta}{\tilde{\sigma}^p_\theta(s, a; \theta')} \right)^2 , \frac{\tilde{\Gamma}^p_{s,a}(\theta, \theta') - 8\Delta}{1/(1-\gamma)} \right\}.$$

The total information is the maximum of these two,

$$\mathcal{I}_{s,a}(\theta, \theta') = \max \left\{ \mathcal{I}^r_{s,a}(\theta, \theta'), \mathcal{I}^p_{s,a}(\theta, \theta') \right\}.$$

The information $\mathcal{I}$ is a fundamental tool for our analysis and it can be understood as follows. The terms on the left-hand side are ratios of the squared deviation between the means of the random variables involved and their variance. If these random variables were Gaussian, this would be proportional to the Kullback-Leibler divergence between the distributions induced by the two models, which in turn is related to their mutual information. The terms on the right-hand side arise from our choice of Bernstein's confidence intervals but have a minor role in the algorithm and its analysis.

Discussion

The sampling procedure of PTUM (Step 4) relies on the availability of a generative model to query informative state-action pairs. We note that the definition of informative state-action pair and all other components of the algorithm are independent on the assumption that a generative model is available. Therefore, one could use ideas similar to PTUM even in the absence of a generative model. For instance, taking inspiration from E^3 (Kearns and Singh, 2002), we could build a surrogate "exploration" MDP with high rewards in informative state-action pairs and solve this MDP to obtain a policy that autonomously navigates the true environment to collect information. Alternatively, we could use the information measure $\mathcal{I}_{s,a}$ as an exploration bonus (Jian et al., 2019). We conjecture that the analysis of this kind of approaches would follow quite naturally from the one of PTUM under the standard assumption of finite MDP diameter $D < \infty$ (Jaksch et al., 2010), for which the resulting bound would have an extra linear scaling in D as in prior works (Brunskill and Li, 2013).

PTUM calls an (ϵ, δ)-PAC algorithm whenever the models are too inaccurate or whenever n queries to the generative model are not sufficient to identify a near-optimal policy. This algorithm can be freely chosen among those available in the literature. For instance, we could choose the MaxQInit algorithm of Abel et al. (2018) which uses an optimistic value function to initialize the learning process of a PAC-MDP method (Strehl et al., 2009). In our case, the information about $\theta^\star$ collected through the generative model could be used to compute much tighter upper bounds to the optimal value function than those obtained solely from previous tasks, thus significantly reducing the overall sample complexity. Alternatively, we could use the Finite-Model-RL algorithm of Brunskill and Li (2013) or the Parameter ELimination (PEL) method of Dyagilev et al. (2008) by passing the set of survived models $\bar{\Theta}_n$ instead of Θ, so that the number of remaining eliminations is potentially much smaller than $|\Theta|$.

Sample-Complexity Bounds

We now analyze the sample complexity of Algorithm 9. The analysis is carried out under the following two assumptions.

Assumption 11.4.1. *The approximation error is such that* $\Delta < \frac{\epsilon(1-\gamma)}{4(1+\gamma)}$.

Assumption 11.4.2. *The sample budget* n *is large enough to allow Algorithm 9 to identify an* ϵ*-optimal policy.*

These two assumptions allow us to analyze only the core part of PTUM (i.e., the transfer mode), thus excluding trivial cases in which the chosen (ϵ, δ)-PAC algorithm is called. In fact, if Assumption 11.4.1 does not hold, the sample complexity for computing an ϵ-optimal policy is equivalent to the one of the chosen algorithm. Similarly, if Assumption 11.4.2 does not hold, the sample complexity is n (the samples collected by the generative model) plus the sample complexity of the chosen algorithm.

Main Result

We start by stating and discussing our main sample complexity bound.

Theorem 11.4.1. *Suppose that Assumption 11.4.1 and Assumption 11.4.2 hold. Let* τ *be the random stopping time and* π_τ *be the returned policy. Then, with probability at least* $1-\delta$*,* π_τ *is* ϵ*-optimal for* $\theta^\star$ *and the total number of queries to the generative model can be bounded by*

$$\tau - 1 \leq \frac{128 \min\{|\mathcal{S}||\mathcal{A}|, |\Theta|\} \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\min_{\theta \in \Theta_\epsilon} \max_{s,a} \mathcal{I}_{s,a}(\theta^\star, \theta)},$$

where, for $\kappa_\epsilon := \frac{(1-\gamma)\epsilon}{4} - \frac{\Delta(1+\gamma)}{2}$*, the set* $\Theta_\epsilon \subseteq \Theta$ *is*

$$\Theta_\epsilon := \left\{ \theta \in \Theta \;\middle|\; \|\tilde{r}_\theta - \tilde{r}_{\theta^\star}\| > \kappa_\epsilon \vee \|(\tilde{p}_\theta - \tilde{p}_{\theta^\star})^T \tilde{V}^\star_{\theta^\star}\| > \frac{\kappa_\epsilon}{\gamma} \right\}.$$

The proof, provided in the next section, combines standard techniques used to analyze PAC algorithms (Azar et al., 2013b; Zanette et al., 2019) with some of the novel ideas

that we used to analyze SAE in Chapter 9. At first glance, Theorem 11.4.1 looks quite different from the standard sample complexity bounds available in the literature Strehl et al. (2009). We shall see now that it reveals many interesting properties. First, this result implies that PTUM is (ϵ, δ)-PAC as the sample complexity is bounded by polynomial functions of all the relevant quantities. Next, we note that, except for logarithmic terms, the sample complexity scales with the minimum between the number of tasks and the number of state-action pairs. As in practice, we expect the former to be much smaller than the latter, we get a significant gain compared to the no-transfer case, where, even with a generative model, the sample complexity is at least linear in $|\mathcal{S}||\mathcal{A}|$ (Azar et al., 2013b). The set Θ_ϵ can be understood as the set of all models in Θ whose optimal policy cannot be guaranteed as ϵ-optimal for the target $\theta^\star$. As we shall see in our analysis, it is sufficient to eliminate all models in this set to ensure stopping. Our key result is that the sample complexity of PTUM is proportional to the one of an "oracle" strategy that knows in advance the *most informative* state-action pairs to achieve this elimination. Note, in fact, that the denominator involves the maximum information to discriminate any model in Θ_ϵ with $\theta^\star$, but the latter is not known to the algorithm. The following result provides further insights into the improvements over the no-transfer case.

Corollary 11.4.1. *Let Γ be the minimum gap between $\theta^\star$ and any other model in Θ,*

$$\Gamma := \min_{\theta \neq \theta^\star} \max \left\{ \|\tilde{r}_\theta - \tilde{r}_{\theta^\star}\|, \|(\tilde{p}_\theta - \tilde{p}_{\theta^\star})^T \tilde{V}^\star_{\theta^\star}\| \right\}.$$

Then, with probability at least $1 - \delta$,

$$\tau \leq \tilde{\mathcal{O}} \left(\frac{\min\{|\mathcal{S}||\mathcal{A}|, |\Theta|\} \log(1/\delta)}{\max\{\Gamma^2, \epsilon^2\}(1-\gamma)^4} \right).$$

Proof. We notice that each model $\theta \in \Theta_\epsilon$ is, by definition, such that either $\|\tilde{r}_\theta - \tilde{r}_{\theta^\star}\| \geq \max\{\Gamma, \kappa_\epsilon\}$ or $\|(\tilde{p}_\theta - \tilde{p}_{\theta^\star})^T \tilde{V}^\star_{\theta^\star}\| \geq \max\{\Gamma, \kappa_\epsilon\}$. By the transfer condition, we also have that $\kappa_\epsilon \geq \frac{(1-\gamma)\epsilon}{8}$. Then, it is easy to see that

$$\min_{\theta \in \Theta_\epsilon} \max_{s,a} \mathcal{I}_{s,a}(\theta^\star, \theta) \geq \max\{\Gamma^2, \kappa_\epsilon^2\}(1-\gamma)^2 \geq \frac{1}{8} \max\{\Gamma^2, \epsilon^2\}(1-\gamma)^4,$$

where we use the previous lower bounds and upper bounded the value-function variance by $1/(1-\gamma)^2$. Then, the result follows by rewriting in $\tilde{\mathcal{O}}$ notation. □

This result reveals that the sample complexity of Algorithm 9 does not scale with ϵ, which is typically regarded as the main term in PAC bounds. That is, when ϵ is small, the bound scales with the minimum gap Γ between the approximate models. Interestingly, the dependence on Γ is the same as the one obtained by Brunskill and Li (2013), but in our case it constitutes a worst-case scenario since Γ can be regarded as the *minimum* positive information for model discrimination, while Theorem 11.4.1 scales with the maximum one. Moreover, since our sample complexity bound is never worse than the one of Brunskill and Li (2013) and theirs achieves *robustness to negative transfer*, PTUM directly inherits this property. We note that $\Gamma > 0$ since, otherwise, two identical models would exist and one could be safely neglected. The key consequence is that one could set $\epsilon = 0$ and the algorithm would retrieve an optimal policy. However, this requires the models to be perfectly approximated so as to enter the transfer mode. Interestingly, our sample-complexity scales

only linearly with the number of models $|\Theta|$ while the bound derived by Brunskill and Li (2013) suffers a quadratic dependence. The intuition is that the algorithm of Brunskill and Li (2013) allocates samples to state-action pairs that discriminate among all the models, yielding in total $\mathcal{O}(|\Theta|^2)$ informative state-action pairs. On the other hand, the analysis of PTUM reveals that the algorithm always eliminates at least one model when allocating samples to a single informative state-action pair, yielding at most a linear dependence on $|\Theta|$. We remark that the optimal dependence on the discount factor was proved to be $\mathcal{O}(1/(1-\gamma)^3)$ Azar et al. (2013b). Here, we get a slightly sub-optimal result since, for simplicity, we naively upper-bounded the variances with their maximum value, but a more involved analysis (e.g., those by Azar et al. (2013b); Sidford et al. (2018)) should lead to optimal dependence. Finally, we note that tighter problem-dependent analyses of best policy identification algorithms have been concurrently provided (Zanette et al., 2019). How to combine these ideas with our structured setting is an interesting direction for future work.

Analysis

We define the event $E := \{\forall t = 1, ..., n : \theta^\star \in \tilde{\Theta}_t\}$ under which the true model is never eliminated from the active model set. We begin by showing that this event holds with high probability, i.e., that the true model is never eliminated from the confidence sets.

Lemma 11.4.1 (Valid confidence sets)**.** *For any $\delta > 0$, the event $E := \{\forall t = 1, ..., n : \theta^\star \in \tilde{\Theta}_t\}$ holds with probability at least $1 - \delta$.*

Proof. Take any step $t \geq 1$, any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, any model $\theta' \in \Theta$, and let $\delta' > 0$. We need to show that the conditions of (11.3)-(11.6) hold with high probability. First notice that these conditions trivially hold if $N_{t-1}(s, a) \leq 1$. Thus, suppose that $N_{t-1}(s, a) > 1$ so that the confidence intervals are well-defined. Using the triangle inequality we have that

$$|\hat{r}_t(s,a) - \tilde{r}_{\theta^\star}(s,a)| \leq |\hat{r}_t(s,a) - r_{\theta^\star}(s,a)| + \Delta,$$

$$|(\tilde{p}_{\theta^\star}(s,a) - \hat{p}_t(s,a))^T \tilde{V}^\star_{\theta'}| \leq |(p_{\theta^\star}(s,a) - \hat{p}_t(s,a))^T \tilde{V}^\star_{\theta'}| + \Delta,$$

$$|\hat{\sigma}^r_t(s,a) - \tilde{\sigma}^r_{\theta^\star}(s,a)| \leq |\hat{\sigma}^r_t(s,a) - \sigma^r_{\theta^\star}(s,a)| + \Delta,$$

$$|\hat{\sigma}^p_t(s,a;\theta') - \hat{\sigma}^p_{\theta^\star}(s,a;\theta')| \leq |\hat{\sigma}^p_t(s,a;\theta') - \sigma^p_{\theta^\star}(s,a;\theta')| + \Delta.$$

Using the empirical Bernstein's inequality (Maurer and Pontil, 2009), we have that, with probability at least $1 - \delta'$,

$$|\hat{r}_t(s,a) - r_{\theta^\star}(s,a)| \leq \sqrt{\frac{2\hat{\sigma}^r_t(s,a)^2 \log \frac{4}{\delta'}}{N_{t-1}(s,a)}} + \frac{7 \log \frac{4}{\delta'}}{3(N_{t-1}(s,a) - 1)}.$$

Similarly, for any $\theta' \in \Theta$, we have that, with probability at least $1 - \delta'$,

$$|(p_{\theta^\star}(s,a) - \hat{p}_t(s,a))^T \tilde{V}^\star_{\theta'}| \leq \sqrt{\frac{2\hat{\sigma}^p_t(s,a;\tilde{V}^\star_{\theta'})^2 \log \frac{4}{\delta'}}{N_{t-1}(s,a)}} + \frac{\frac{7}{1-\gamma} \log \frac{4}{\delta'}}{3(N_{t-1}(s,a) - 1)}.$$

From Theorem 10 of Maurer and Pontil (2009),

$$|\hat{\sigma}_t^r(s,a) - \sigma_{\theta^\star}^r(s,a)| \leq \sqrt{\frac{2\log\frac{2}{\delta'}}{N_{t-1}(s,a)-1}}$$

and

$$|\hat{\sigma}_t^p(s,a;\theta') - \sigma_{\theta^\star}^p(s,a;\theta')| \leq \frac{1}{1-\gamma}\sqrt{\frac{2\log\frac{2}{\delta'}}{N_{t-1}(s,a)-1}}$$

hold with probability at least $1-\delta'$, respectively. Taking union bounds over all state action pairs and over the maximum number of samples n, these four inequalities hold at the same time with probability at least $1-2|\mathcal{S}||\mathcal{A}|(|\Theta|+1)n\delta'$. The result follows after setting $\delta = 2|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)\delta'$ and rearranging. □

Next we bound the number of samples required from some state-action pair in order to eliminate a "wrong" model from the confidence set.

Lemma 11.4.2 (Model elimination)**.** *Let $\theta \in \Theta$, $(s,a) \in \mathcal{S}\times\mathcal{A}$, and define*

$$\bar{n}_\theta^r(s,a) := \min_{\theta''\in\tilde{\Theta}_t} \max\left\{\frac{\tilde{\sigma}_{\theta''}^r(s,a)^2}{[\tilde{\Gamma}_{s,a}^r(\theta^\star,\theta)-4\Delta]_+^2}, \frac{1}{[\tilde{\Gamma}_{s,a}^r(\theta^\star,\theta)-4\Delta]_+}\right\},$$

$$\bar{n}_\theta^p(s,a) := \min_{\theta'\in\Theta,\theta''\in\tilde{\Theta}_t} \max\left\{\frac{\tilde{\sigma}_{\theta''}^p(s,a;\theta')^2}{[\tilde{\Gamma}_{s,a}^p(\theta^\star,\theta)-4\Delta]_+^2}, \frac{1/(1-\gamma)}{[\tilde{\Gamma}_{s,a}^p(\theta^\star,\theta)-4\Delta]_+}\right\}.$$

Then, under event E, if

$$N_{t-1}(s,a) > \bar{n}_\theta(s,a) := 32\log\frac{8|\mathcal{S}||\mathcal{A}|n(1+|\Theta|)}{\delta}\min\{\bar{n}_\theta^r(s,a), \bar{n}_\theta^p(s,a)\},$$

we have that $\theta \notin \tilde{\Theta}_t$.

Proof. We split the proof into two parts, dealing with rewards and transitions separately. We then combine these results to obtain the final statement.

Elimination by rewards. Assuming $\theta \in \tilde{\Theta}_t$, we must have, for all state-action pairs and all $\theta'' \in \tilde{\Theta}_t$,

$$\begin{aligned}
&\tilde{\Gamma}_{s,a}^r(\theta^\star,\theta) \leq |\tilde{r}_\theta(s,a) - \hat{r}_t(s,a)| + |\tilde{r}_{\theta^\star}(s,a) - \hat{r}_t(s,a)| \leq 2C_{t,\delta}^r(s,a) \\
&\leq 2\sqrt{\frac{2\hat{\sigma}_t^r(s,a)^2\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{14\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} + 2\Delta \\
&\leq 2\sqrt{\frac{2\tilde{\sigma}_{\theta''}^r(s,a)^2\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{4\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{(N_{t-1}(s,a)-1)} + \frac{14\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} + 4\Delta \\
&\leq 2\sqrt{\frac{2\tilde{\sigma}_{\theta''}^r(s,a)^2\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{26\log\frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} + 4\Delta
\end{aligned}$$

where we applied the triangle inequality and used Lemma 11.4.1 to upper bound the empirical variance by the variance of a model θ'' in the confidence set. We note that, if the approximation error

Δ is too high and the denominators of $\bar{n}_\theta(s,a)$ are zero, it is not possible to eliminate θ from the rewards of this state-action pair. If this is not the case, for $N_{t-1}(s,a) \geq \bar{n}_\theta(s,a)$ we have that

$$2\sqrt{\frac{2\tilde{\sigma}^r_{\theta''}(s,a)^2 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} < \frac{\tilde{\Gamma}^r_{s,a}(\theta^\star,\theta) - 4\Delta}{2}$$

and

$$\frac{26 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)} < \frac{\tilde{\Gamma}^r_{s,a}(\theta^\star,\theta) - 4\Delta}{2}.$$

Plugging these two inequalities in the first upper bound leads to the contradiction $\tilde{\Gamma}^r_{s,a}(\theta^\star,\theta) < \tilde{\Gamma}^r_{s,a}(\theta^\star,\theta)$, hence it must be that $\theta \notin \tilde{\Theta}_t$.

Elimination by transition. The proof proceeds analogously to the previous case. Let $\theta' \in \Theta$ and $\theta'' \in \tilde{\Theta}_t$, then

$$\begin{aligned}
\tilde{\Gamma}^p_{s,a}(\theta^\star,\theta) &\leq |(\hat{p}_t(s,a) - \tilde{p}_{\theta^\star}(s,a))^T \tilde{V}^\star_{\theta'}| + |(\hat{p}_t(s,a) - \tilde{p}_\theta(s,a))^T \tilde{V}^\star_{\theta'}| \\
&\leq 2\sqrt{\frac{2\hat{\sigma}^p_t(s,a;\theta') \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{14 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)(1-\gamma)} + 2\Delta \\
&\leq 2\sqrt{\frac{2\tilde{\sigma}^p_{\theta''}(s,a;\theta') \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{N_{t-1}(s,a)}} + \frac{26 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)}{\delta}}{3(N_{t-1}(s,a)-1)(1-\gamma)} + 4\Delta,
\end{aligned}$$

where once again we applied the triangle inequality and Lemma 11.4.1 to upper bound the empirical variance. Hence, applying the same reasoning as before we obtain a contradiction, which in turns implies that $\theta \notin \tilde{\Theta}_t$. We finally note that if $\bar{n}_\theta(s,a) = +\infty$, i.e., the approximate models are too inaccurate, it is not possible to eliminate θ using this state-action pair. □

We now state different results to bound the deviation in value function between different MDPs. We need to define the discounted state-action visitation frequencies (Sutton et al., 2000) starting from state s and executing π in MDP $\mathcal{M}_\theta$ as

$$\nu^\pi_\theta(s',a';s) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi_\theta\{S_t = s', A_t = a' | S_0 = s\}. \tag{11.7}$$

Note that here, with some abuse of notation, the random variables S_t, A_t are not the state-action pairs chosen by PTUM but those generated by an interaction with $\mathcal{M}_\theta$ using policy π.

Lemma 11.4.3 (Simulation lemma)**.** *Let $\theta, \theta' \in \Theta$, $s \in \mathcal{S}$, and π be any policy. Then,*

$$|V^\pi_\theta(s) - V^\pi_{\theta'}(s)| \leq \sum_{s',a'} \nu^\pi_{\theta'}(s',a';s) \xi^\pi_{s',a'}(\theta,\theta'),$$

where

$$\xi^\pi_{s',a'}(\theta,\theta') := \Gamma^r_{s',a'}(\theta,\theta') + \gamma |(p_\theta(s',a') - p_{\theta'}(s',a'))^T V^\pi_\theta|.$$

Similarly, the optimal value functions of θ and θ' satisfy, for any $s \in \mathcal{S}$,

$$|V^\star_{\theta'}(s) - V^\star_\theta(s)| \leq \max_{\pi \in \{\pi^\star_\theta, \pi^\star_{\theta'}\}} \sum_{s',a'} \nu^\pi_{\theta'}(s',a';s) \left[\Gamma^r_{s',a'}(\theta,\theta') + \gamma \Gamma^p_{s',a'}(\theta,\theta')\right].$$

Proof. See, e.g., Lemma 3 of (Zanette et al., 2019) for the first inequality and Lemma 2 of (Azar et al., 2013b) or Lemma 2 of (Zanette et al., 2019) for the second one.[4] □

Corollary 11.4.2 (Value-function error decomposition)**.** *Let $\theta, \theta' \in \Theta$ and $s \in \mathcal{S}$. Then,*

$$|V^{\star}_{\theta'}(s) - V^{\pi^{\star}_{\theta}}_{\theta'}(s)| \leq 2 \max_{\pi \in \{\pi^{\star}_{\theta}, \pi^{\star}_{\theta'}\}} \sum_{s',a'} \nu^{\pi}_{\theta'}(s', a'; s) \left[\Gamma^{r}_{s',a'}(\theta, \theta') + \gamma \Gamma^{p}_{s',a'}(\theta, \theta')\right].$$

Proof. The proof is straightforward by using the triangle inequality,

$$|V^{\star}_{\theta'}(s) - V^{\pi^{\star}_{\theta}}_{\theta'}(s)| \leq \underbrace{|V^{\star}_{\theta'}(s) - V^{\star}_{\theta}(s)|}_{(a)} + \underbrace{|V^{\star}_{\theta}(s) - V^{\pi^{\star}_{\theta}}_{\theta'}(s)|}_{(b)},$$

and bounding (a) using the second inequality in Lemma 11.4.3 and (b) using the first inequality in Lemma 11.4.3 (note that $V^{\star}_{\theta} = V^{\pi^{\star}_{\theta}}_{\theta}$). □

Using this result, we now show that, if the algorithm does not stop at a certain time t, certain models belong to the confidence set. In other words, the following Lemma builds a set of models $\Theta_{\epsilon} \subseteq \Theta$ whose elimination is sufficient to guarantee that PTUM stops.

Lemma 11.4.4 (Stopping condition)**.** *Let τ be the random stopping time of Algorithm 9 and*

$$\Theta_{\epsilon} := \left\{ \theta \in \Theta \;\middle|\; \|\tilde{r}_{\theta} - \tilde{r}_{\theta^{\star}}\| > \kappa_{\epsilon} \vee \|(\tilde{p}_{\theta} - \tilde{p}_{\theta^{\star}})^{\top} \tilde{V}^{\star}_{\theta^{\star}}\| > \frac{\kappa_{\epsilon}}{\gamma} \right\},$$

where $\kappa_{\epsilon} := \frac{(1-\gamma)\epsilon}{4} - \frac{\Delta(1+\gamma)}{2}$. Then, under event E, for all $t < \tau$, there exists at least one model $\theta \in \Theta_{\epsilon}$ such that $\theta \in \tilde{\Theta}_t$.

Proof. We note that, for all $t < \tau$, under event E, it must be that

$$\exists \theta \in \tilde{\Theta}_t, s \in \mathcal{S} : \tilde{V}^{\tilde{\pi}^{\star}_{\theta^{\star}}}_{\theta}(s) < \tilde{V}^{\star}_{\theta}(s) - \epsilon + 2\Delta \frac{(1+\gamma)}{1-\gamma},$$

otherwise the algorithm would stop before τ since the optimal policy of $\mathcal{M}_{\theta^{\star}}$ (which belongs to the confidence set by event E) satisfies the stopping condition. This implies that $|\tilde{V}^{\star}_{\theta}(s) - \tilde{V}^{\tilde{\pi}^{\star}_{\theta^{\star}}}_{\theta}(s)| > \epsilon - 2\Delta\frac{(1+\gamma)}{1-\gamma}$ holds as well and, using Corollary 11.4.2,

$$2 \max_{\pi \in \{\tilde{\pi}^{\star}_{\theta^{\star}}, \tilde{\pi}^{\star}_{\theta}\}} \sum_{s',a'} \nu^{\pi}_{\theta}(s', a'; s) \left[\tilde{\Gamma}^{r}_{s',a'}(\theta^{\star}, \theta) + \gamma \tilde{\Gamma}^{p}_{s',a'}(\theta^{\star}, \theta)\right] > \epsilon - 2\Delta \frac{(1+\gamma)}{1-\gamma}$$

holds for some $\theta \in \tilde{\Theta}_t$ and $s \in \mathcal{S}$. Assume that all models in Θ_{ϵ} have been eliminated. Then, using that ν sums up to $1/(1-\gamma)$ and that, from the definition of Θ_{ϵ}, all models must be sufficiently close to $\theta^{\star}$ ($\tilde{\Gamma}^{r}_{s',a'}(\theta^{\star}, \theta) \leq \kappa_{\epsilon}$ and $\tilde{\Gamma}^{p}_{s',a'}(\theta^{\star}, \theta) \leq \kappa_{\epsilon}/\gamma$), the left-hand side of this inequality can be upper bounded by $\epsilon - 2\Delta\frac{(1+\gamma)}{1-\gamma}$. Hence, we obtain a contradiction and it must be that $\theta \in \tilde{\Theta}_t$ for some $\theta \in \Theta_{\epsilon}$. □

Before stating and proving the main result, we need to ensure that, at each step t in which PTUM does not stop, the algorithm collects strictly positive information.

[4] Note that, although the inequalities of, e.g., Azar et al. (2013b) and Zanette et al. (2019) relate the value functions of a fixed MDP with those of its empirical counterpart, they actually hold for any two MDPs.

Lemma 11.4.5 (Positive index)**.** *Let τ be the random stopping time of Algorithm 9, then, under event E,*

$$\forall t < \tau : \mathcal{I}_t(S_t, A_t) > 0$$

Proof. Recall that the algorithm enters the transfer mode if $\Delta < \frac{\epsilon(1-\gamma)}{4(1+\gamma)}$. Take any time $t < \tau$. Under event E, we have $\theta^\star \in \tilde{\Theta}_t$ and Lemma 11.4.4 implies that $\theta \in \tilde{\Theta}_t$ for some $\theta \in \Theta_\epsilon$. The definition of Θ_ϵ implies that either $\|\tilde{r}_\theta - \tilde{r}_{\theta^\star}\| > \kappa_\epsilon$ or $\|(\tilde{p}_\theta - \tilde{p}_{\theta^\star})^T \tilde{V}^\star_{\theta^\star}\| > \frac{\kappa_\epsilon}{\gamma}$ and both these quantities are strictly greater than zero since $\kappa_\epsilon > \frac{\epsilon(1-\gamma)}{8}$. Since the index contains a maximum over models involving these two, the result follows straightforwardly. □

The following lemma is the key result that allows us to bound the sample complexity of Algorithm 9. It shows that, at any time t, the number of times the chosen state-action action pair (S_t, A_t) has been chosen before is bounded by a quantity proportional to minimum number of samples required from any state-action pair to eliminate any of the active models.

Lemma 11.4.6 (Fundamental lemma)**.** *Let (S_t, A_t) be the state-action pair chosen at time t. Then, under event E, the number of queries to such couple prior to time t can be upper bounded by*

$$N_{t-1}(S_t, A_t) < \frac{128 \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\max_{s,a} \max_{\theta \in \tilde{\Theta}_t} \mathcal{I}_{s,a}(\theta^\star, \theta)}.$$

Proof. Let $F_t = \mathbb{1}\{\forall s, a \in \mathcal{S} \times \mathcal{A} : \mathcal{I}^r_t(S_t, A_t) \geq \mathcal{I}_t(s, a)\}$ be the event under which, at time t, the maximizer of the index is attained by the reward components. The proof is divided in two parts, based on whether F_t holds or not.

Event F_t holds. Let $\underline{\Theta}_t := \operatorname{argmax}_{\theta,\theta' \in \tilde{\Theta}_t} \mathcal{I}^r_t(S_t, A_t)$ be the set of active models that attain the maximum in the reward index. Similarly, define

$$\overline{\theta}_t := \operatorname*{argmax}_{\theta \in \underline{\Theta}_t} \tilde{\Gamma}^r_{S_t,A_t}(\theta^\star, \theta), \quad \underline{\theta}_t := \operatorname*{argmin}_{\theta \in \underline{\Theta}_t} \tilde{\Gamma}^r_{S_t,A_t}(\theta^\star, \theta),$$

as the farthest and closest models from $\theta^\star$ among those used to compute the index, respectively. Assume, without loss of generality, that the maximums/minimums are attained by single models. If more than one model attains them, the proof follows equivalently by choosing arbitrary ones. Furthermore, let θ^v_t be the (random) model among those in $\underline{\Theta}_t$ whose reward-variance is used to attain the maximum in the index. We proceed as follows. First, we prove that an upper bound to the index of the chosen state-action pair directly relates to the sample complexity for eliminating $\overline{\theta}_t$. Then, we use this result to guarantee that (S_t, A_t) cannot be chosen more than the stated quantity prior to time step t, otherwise $\overline{\theta}_t$ could not be an active model. By assumption we have

$$\begin{aligned}
\mathcal{I}_t(S_t, A_t) = \mathcal{I}^r_t(S_t, A_t) &= \min\left\{ \frac{(\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \underline{\theta}_t) - 8\Delta)^2}{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}, \tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \underline{\theta}_t) - 8\Delta \right\} \\
&\overset{(a)}{\leq} \min\left\{ \frac{\left(\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) + \tilde{\Gamma}^r_{S_t,A_t}(\theta^\star, \underline{\theta}_t) - 8\Delta\right)^2}{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}, \tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) + \tilde{\Gamma}^r_{S_t,A_t}(\theta^\star, \underline{\theta}_t) - 8\Delta \right\} \\
&\overset{(b)}{\leq} \min\left\{ \frac{(2\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 8\Delta)^2}{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}, 2\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 8\Delta \right\} \\
&\leq 4 \min\left\{ \frac{(\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta)^2}{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}, \tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta \right\},
\end{aligned}$$

where (a) follows from the triangle inequality and (b) from the definition of $\overline{\theta}_t$ (which was defined as the farthest from the estimate of $\theta^\star$). Note that to prove this inequalities we also need that $\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \underline{\theta}_t) - 8\Delta \geq 0$, which is implied by Lemma 11.4.5. Since (S_t, A_t) is chosen at time t, it must be that $\mathcal{I}_t(S_t, A_t) \geq \mathcal{I}_t(s, a)$ for all $s, a \in \mathcal{S} \times \mathcal{A}$. This implies that, for all $s, a \in \mathcal{S} \times \mathcal{A}$ and $\theta \in \tilde{\Theta}_t$,

$$4 \min \left\{ \frac{(\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta)^2}{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}, \tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta \right\} \geq \mathcal{I}_t(s, a) \geq \mathcal{I}_{s,a}(\theta^\star, \theta), \tag{11.8}$$

where the second inequality holds since the index of (s, a) is by definition larger than the one using the models $\theta^\star$ and θ. Note that Lemma 11.4.2 and 11.4.5 ensure that a number of queries to (S_t, A_t) of

$$32 \log \frac{8|\mathcal{S}||\mathcal{A}|n(|\Theta| + 1)}{\delta} \max \left\{ \frac{\tilde{\sigma}^r_{\theta^v_t}(S_t, A_t)^2}{(\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta)^2}, \frac{1}{\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) - 4\Delta} \right\}$$

suffices for eliminating $\tilde{\Theta}_t$. In particular, the positivity of the index from Lemma 11.4.5 implies that $\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \underline{\theta}_t) > 8\Delta$, which, in turn, implies that $\tilde{\Gamma}^r_{S_t,A_t}(\overline{\theta}_t, \theta^\star) > 4\Delta$. Therefore, Equation 11.8 above implies that a number of queries of

$$\frac{128 \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta| + 1)/\delta)}{\max_{s,a} \max_{\theta \in \tilde{\Theta}_t} \mathcal{I}_{s,a}(\theta^\star, \theta)}.$$

also suffices. We note that the maximums at the denominator can be introduced since (11.8) holds for all s, a and θ. Hence, it must be that $N_{t-1}(S_t, A_t)$ is strictly less than this quantity, otherwise the model $\overline{\theta}_t$ would be eliminated at time step $t - 1$ and it could not be active at time t. This concludes the first part of the proof.

Event F_t does not hold. In this case, the maximizer of the index must be attained using the transition components, thus $\mathcal{I}_t(S_t, A_t) = \mathcal{I}^p_t(S_t, A_t)$. The proof follows exactly the same steps as before and is therefore not reported. Since the result is the same, combining these two parts proves the main statement.

□

Before proving the main theorem, we need to ensure that, whenever PTUM terminates due its stopping condition, the returned policy is indeed ϵ-optimal for $\mathcal{M}_{\theta^\star}$.

Lemma 11.4.7 (Correctness)**.** *Let τ be the random stopping time of Algorithm 9 and π_τ be the returned policy. Then, under event E, π_τ is ϵ-optimal for $\mathcal{M}_{\theta^\star}$.*

Proof. Recall that π_τ is ϵ-optimal if, for all states, $V^{\pi_\tau}_{\theta^\star}(s) \geq V^\star_{\theta^\star}(s) - \epsilon$. Furthermore, π_τ is optimal for one of the active models at time τ, i.e., $\pi_\tau = \tilde{\pi}^\star_\theta$ for some $\theta \in \tilde{\Theta}_\tau$. Since under E we have $\theta^\star \in \tilde{\Theta}_\tau$, a sufficient condition is that $\|V^\star_{\theta'} - V^{\tilde{\pi}^\star_\theta}_{\theta'}\| < \epsilon$ holds for all $\theta' \in \bar{\Theta}_\tau$. Let us upper bound the left-hand side as

$$\|V^\star_{\theta'} - V^{\tilde{\pi}^\star_\theta}_{\theta'}\| \leq \underbrace{\|\tilde{V}^{\tilde{\pi}^\star_\theta}_{\theta'} - V^{\tilde{\pi}^\star_\theta}_{\theta'}\|}_{(a)} + \underbrace{\|V^\star_{\theta'} - \tilde{V}^\star_{\theta'}\|}_{(b)} + \underbrace{\|\tilde{V}^{\tilde{\pi}^\star_\theta}_{\theta'} - \tilde{V}^\star_{\theta'}\|}_{(c)}.$$

Using Lemma 11.4.3, we can bound the first term by

$$\begin{aligned}(a) &\leq \sum_{s',a'} \nu^{\tilde{\pi}^\star_\theta}_{\theta'}(s', a'; s)(|r_{\theta'}(s, a) - \tilde{r}_{\theta'}(s, a)| + \gamma|(p_{\theta'}(s, a) - \tilde{p}_{\theta'}(s, a))^T \tilde{V}^{\tilde{\pi}^\star_\theta}_{\theta'}|) \\ &\leq \sum_{s',a'} \nu^{\tilde{\pi}^\star_\theta}_{\theta'}(s', a'; s)(\Delta + \gamma\Delta) \leq \frac{\Delta(1 + \gamma)}{1 - \gamma}\end{aligned}$$

and the second term by

$$(b) \leq \max_{\pi \in \{\pi^\star_{\theta'}, \tilde{\pi}^\star_{\theta'}\}} \sum_{s',a'} \nu^\pi_{\theta'}(s',a';s)\Big(|r_{\theta'}(s,a) - \tilde{r}_{\theta'}(s,a)| + \gamma |(p_{\theta'}(s,a) - \tilde{p}_{\theta'}(s,a))^T \tilde{V}^\star_{\theta'}|\Big)$$
$$\leq \max_{\pi \in \{\pi^\star_{\theta'}, \tilde{\pi}^\star_{\theta'}\}} \sum_{s',a'} \nu^\pi_{\theta'}(s',a';s)(\Delta + \gamma\Delta) \leq \frac{\Delta(1+\gamma)}{1-\gamma}.$$

Therefore, the stopping condition, $(c) = \|\tilde{V}^{\tilde{\pi}^\star_{\theta}}_{\theta'} - \tilde{V}^\star_{\theta'}\| \leq \epsilon - 2\Delta\frac{(1+\gamma)}{1-\gamma}$, implies that $\|V^\star_{\theta'} - V^{\tilde{\pi}^\star_{\theta}}_{\theta'}\| \leq \epsilon$, which in turn implies the ϵ-optimality of π_τ. □

Proof of Theorem 11.4.1. Lemma 11.4.1 ensures that event E holds with probability at least $1-\delta$. Therefore, we shall carry out the proof conditioned on E. We split the proof into two parts. In the first one, we bound the number of times each state-action pair can be visited before the algorithm stops. In the second part, we directly bound the number of steps in which each model can be active.

Bound over $\mathcal{S}\times\mathcal{A}$. Take any state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$. For any sequence $\{n_t\}_{t\geq 1}$, its number of visits $N_{\tau-1}(s,a)$ can be written as

$$\sum_{t=1}^{\tau-1} \mathbb{1}\{S_t = s, A_t = a | E\} = \underbrace{\sum_{t=1}^{\tau-1} \mathbb{1}\{S_t = s, A_t = a, N_{t-1}(s,a) < n_t | E\}}_{(a)}$$
$$+ \underbrace{\sum_{t=1}^{\tau-1} \mathbb{1}\{S_t = s, A_t = a, N_{t-1}(s,a) \geq n_t | E\}}_{(b)}.$$

For

$$n_t := \frac{128 \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\max_{s,a}\max_{\theta\in\tilde{\Theta}_t} \mathcal{I}_{s,a}(\theta^\star,\theta)},$$

Lemma 11.4.6 ensures that, under event E, $(b) = 0$. Thus, we only need to bound (a). For all $t < \tau$, Lemma 11.4.4 implies that there exists a model $\theta \in \Theta_\epsilon$ which also belongs to the confidence set at time t, $\theta \in \tilde{\Theta}_t$. Therefore, for all $t < \tau$ and $(s',a') \in \mathcal{S}\times\mathcal{A}$, we have $\max_{\theta\in\tilde{\Theta}_t} \mathcal{I}_{s',a'}(\theta^\star,\theta) \geq \min_{\theta\in\Theta_\epsilon} \mathcal{I}_{s',a'}(\theta^\star,\theta)$. Since we removed all random quantities from n_t, we can now bound (a) as

$$(a) < \frac{128 \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\min_{\theta\in\Theta_\epsilon}\max_{s,a} \mathcal{I}_{s,a}(\theta^\star,\theta)}. \tag{11.9}$$

This immediately yields a bound on the total number of queries to the generative model,

$$\tau - 1 = \sum_{s,a} N_{\tau-1}(s,a) < \frac{128|\mathcal{S}||\mathcal{A}| \log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\max_{s,a}\min_{\theta\in\Theta_\epsilon} \mathcal{I}_{s,a}(\theta^\star,\theta)}.$$

Bound over Θ From the first part of the proof, we know that, for $t < \tau$, the confidence set $\tilde{\Theta}_t$ must contain a model that is also in Θ_ϵ, otherwise the algorithm would stop. Therefore, the stopping time can be bounded by

$$\tau \leq \sum_{t=1}^{n} \mathbb{1}\left\{\exists \theta \in \tilde{\Theta}_t : \theta \in \Theta_\epsilon | E\right\}.$$

By definition of the algorithm, the state-action pair chosen at each time step does not change until the set of active models $\underline{\Theta}_t$ (those that control the maximizer of the index as in the proof of Lemma 11.4.6) does not change. Furthermore, once a model has been eliminated, it cannot become active again. Consider a sequence $\{\tau_h\}_{h\geq 1}$ with $\tau_1 = 1$. We can partition the time line into different contiguous intervals (from now on called phases) $\mathcal{T}_h := [\tau_h, \tau_{h+1} - 1]$ such that the set of active models does not change within $\mathcal{T}_h$ and a change of phase occurs only when a model is eliminated. Let $\underline{\Theta}_h$ be the set of active models in phase h. We have $\tau_{h+1} = \inf_{t\geq 1}\{t \mid \exists \theta \in \underline{\Theta}_h : \theta \notin \tilde{\Theta}_t\}$. That is, the beginning of the new phase $h+1$ is the step where one of the previously-active models is eliminated. Let $\bar{\theta}_h$ be any such model and $h(t)$ be the (unique) phase containing time t. Note that, for each $\theta \in \Theta \setminus \{\theta^\star\}$, there exists at most one phase $h(\theta)$ where $\bar{\theta}_{h(\theta)} = \theta$. Then,

$$\begin{aligned}
\tau &\leq \sum_{\theta \in \Theta \setminus \{\theta^\star\}} \sum_{t=1}^{n} \mathbb{1}\left\{\bar{\theta}_{h(t)} = \theta \wedge \exists \theta' \in \tilde{\Theta}_t : \theta' \in \Theta_\epsilon | E\right\} \\
&\leq \sum_{\theta \in \Theta \setminus \{\theta^\star\}} \sum_{t=\tau_{h(\theta)}}^{\tau_{h(\theta)+1}-1} \mathbb{1}\left\{\bar{\theta}_{h(t)} = \theta \wedge \exists \theta' \in \tilde{\Theta}_t : \theta' \in \Theta_\epsilon | E\right\} \\
&\leq \frac{128(|\Theta| - 1)\log(8|\mathcal{S}||\mathcal{A}|n(|\Theta|+1)/\delta)}{\min_{\theta \in \Theta_\epsilon} \max_{s,a} \mathcal{I}_{s,a}(\theta^\star, \theta)},
\end{aligned}$$

where in the last inequality we applied Lemma 11.4.6 by noticing that, within the same phase, the chosen state-action pair does not change and used the fact that a model in Θ_ϵ still survives to upper bound the minimum over models in the confidence set. The proof follows by taking the minimum of the two bounds.

Numerical Simulations

The goal of our experiments is twofold. First, we analyze the performance of PTUM when the task models are known (i.e., $\Delta = 0$), focusing on the comparison between identification and jumpstart strategies. Then, we perform some ablation studies that confirm our theoretical results by verifying how the sample complexity of PTUM changes as a function of ϵ, $|\mathcal{S}|$, and the model misspecification Δ. All our plots report averages of 100 runs with 99% Student's t confidence intervals. We refer the reader to Tirinzoni et al. (2020c) for additional details.

Identification vs Jumpstart For this experiment, we adopt a standard 12×12 grid-world divided into two parts by a vertical wall (i.e., the 4-rooms domain of Sutton et al. (1999) with only two rooms). The agent starts in the left room and must reach a goal state in the right one, with the two rooms connected by a door. We consider 12 different tasks,

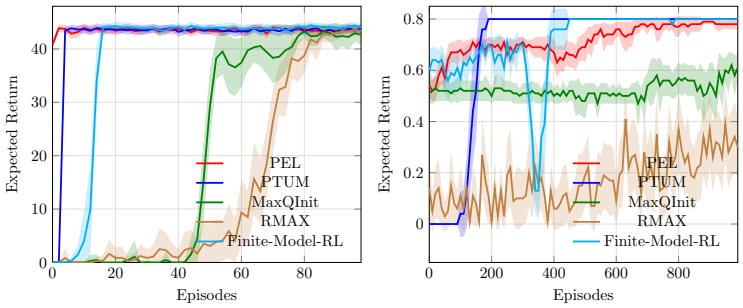

Figure 11.1: *Policy transfer from known models. (left) Jumpstart (optimistic) policies gain sufficient information. (right) High-reward states are poorly informative.*

whose models are known to the agent, with different goal and door locations. We compare PTUM with four baselines: RMAX (Brafman and Tennenholtz, 2002), which does not perform any transfer, RMAX with MaxQInit (Abel et al., 2018), in which the Q-function is initialized optimistically to achieve a jumpstart, PEL (Dyagilev et al., 2008), which performs model identification, and Finite-Model RL (Brunskill and Li, 2013), that is the most conceptually similar to PTUM. Not to give advantage to PTUM, as the other algorithms run episodes online with no generative model, in our plots we count each sample taken by PTUM as one episode and assign it the minimum reward value. Once PTUM returns a policy, we report its online performance. The results in Figure 11.1*(left)* show that the three identification methods find an optimal policy in a very small number of episodes. The jumpstart of MaxQInit is delayed since the optimistic Q-function directs the agent towards the wall, but finding the door takes many samples. PEL, which is also optimistic, instantly identifies the solution since attempts to cross the wall in locations without a door immediately discriminate between tasks. This is in fact an example where the optimism principle leads to both rewards and information. Unfortunately, we know that in structured settings this is not always the case and optimism might not be the best thing to do. To demonstrate this fact, we run the same experiment by removing the wall and adding multiple goal locations with different reward values. Some goals have uniformly high values over all tasks (thus they provide small information) while others are more variable (thus with higher information). In Figure 11.1*(right)*, we see that PEL, MaxQInit, and Finite-Model RL achieve a high jumpstart, as they immediately go towards one of the goals, but converge much more slowly than PTUM, which focuses on the informative ones. The performance drop of Finite-Model RL is due to the fact that, at some point, the algorithm eliminates a wrong MDP model whose optimal policy is actually near-optimal in the true task (hence causing the initial jumpstart).

Sample complexity vs accuracy. The setting in this case is a grid of dimension 12x12 where a reward of 1 is given when reaching a goal state and 0 otherwise. The agent has knowledge of 12 possible perfect models, each of which has a single goal state located in the opposite corner from the starting one. The door position is different in each task. We set γ to 0.99, δ to 0.01, n to 1000000, and test our algorithm for different values of ϵ. Figure 11.2(*left*) shows how the sample complexity changes as a function of ϵ. First, we

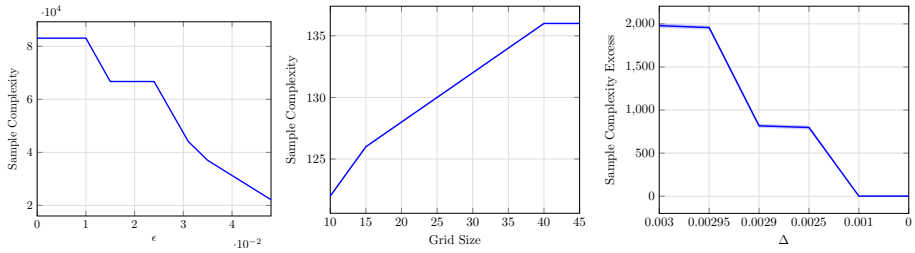

Figure 11.2: *Ablation studies for the main parameters of PTUM. (left) The sample complexity is a piece-wise constant and bounded function of ϵ. (middle) The sample complexity grows logarithmically with the grid size (i.e., the number of states). (right) The excess in sample complexity (w.r.t. the one with perfect models) decreases with the approximation error Δ.*

notice that the function is piece-wise constant, which is a direct consequence of the finite set of models. In fact, varying ϵ changes the set of models Θ_ϵ that must be discarded by the algorithm before stopping. Second, we notice that the function is bounded in ϵ, i.e., we are allowed to set $\epsilon = 0$ and the algorithm returns an optimal policy after discarding all the models different from the true one.

Sample complexity vs number of states. In this experiment, the agent faces grids of increasing sizes. The agent obtains reward 1 when ending up in a goal state and 0 otherwise. We consider three known models, each with the goal state in a corner different from the starting one and a different door position. We set ϵ to 0.0001, γ to 0.99, δ to 0.01, and n to 100000. Figure 11.2(*middle*) shows how the sample complexity changes when the grid size (i.e., the number of states) increases. As expected, we obtain a logarithmic growth due to the union bounds used to form the confidence sets. As before, the function is piece-wise due to the finite number of models.

Sample complexity vs model error A grid of fixed size 6x6 is considered. The agent has knowledge of 6 models, each of which has the goal state placed in the opposite corner w.r.t. the starting one. All models have reward 1 when reaching the goal state and 0 otherwise. Each task differs from the others in the door position. We set ϵ to 0.13, γ to 0.9, δ to 0.1, and n to 1000000. Here we study the sample complexity of PTUM with the exact set of models when varying the maximum uncertainty Δ. Figure 11.2(*right*) shows the excess in sample complexity w.r.t. the case with perfect models when the bound Δ on the approximation error decreases. Once again, we obtain a piece-wise constant function since, similarly to ϵ, a higher error bound Δ changes the set of models that must be discarded by the algorithm and hence its sample complexity. Notably, we do not require $\Delta = 0$ to recover the "oracle" sample complexity.

CHAPTER 12

Conclusion

We studied different aspects of knowledge transfer in reinforcement learning. From the very beginning, we introduced the concept of *structured domain*, defined by a collection of possibly-related tasks that could be faced by the agent together with a process generating them. We showed the generality of this concept, which allows to formalize the majority of transfer-related settings that are studied in the literature, including multi-task learning, meta-learning, lifelong learning, and curriculum learning. We framed all our problems into this context, while building our algorithms to exploit, either implicitly or explicitly, the properties of the underlying structured domain. In particular, we studied two main knowledge-transfer problems in this setting. The first one concerns how to *reuse old experience samples* collected in previous tasks to learn a new target task. The second is somewhat complementary: how to *generate new experience samples*, i.e., how to explore, in the new target task given knowledge from the source tasks.

Part I dealt with the first problem. In particular, we designed algorithms to transfer experience samples across tasks without any assumption on the underlying structured domain. One of our main objectives was the practical applicability of these methods, hence we built them on top of approaches for continuous MDPs. The general idea behind our methods was to transfer all source samples, while correcting the distribution mismatch between source and target models using importance sampling. This is in contrast to prior works, which either require an expensive sample selection process (Lazaric et al., 2008b), or make strong assumptions (Laroche and Barlier, 2017), or directly transfer samples without accounting for domain mismatch (Taylor et al., 2008). In Chapter 5, we designed IWFQI, an algorithm that instantiates this sample transfer technique in *batch* reinforce-

ment learning. We conducted a finite-sample analysis showing the effect of wrong importance weights onto the value-function errors of IWFQI, while in practice we computed these weights by directly estimating reward/transition models using Gaussian processes. In Chapter 6, we proposed IWPG, an extension of this algorithm to the *policy-gradient* setting. This required significant additional effort in designing importance-sampling estimators of the policy gradient that reuse entire trajectories. Interestingly, we established *robustness to negative transfer* for the most powerful of our estimators (a combination of multiple importance sampling scheme and control variates) with *exact* importance weights. For the realistic case where the weights are *unknown*, we derived a sound approach to estimate them by directly minimizing the mean-square error of the resulting estimator. For both IWFQI and IWPG, we provided numerical simulations that confirm the effectiveness of our methods in continuous control problems.

In Part II, we began studying the problem of exploration in structured domains from a *practical* perspective, with the aim of designing scalable and general methods that can be applied to large continuous MDPs. Similarly to Lazaric and Ghavamzadeh (2010), we assumed that tasks share structure in their optimal action-value functions. We used the source tasks to learn a prior distribution over these value functions and proposed a simple *variational* approximation of the corresponding posterior given samples from the target task. Using ideas from value-function randomization (Osband et al., 2014, 2019), we designed an algorithm that explores the target task by *posterior sampling* on these distributions. We derived a finite-sample analysis of the proposed method in simple settings and verified empirically that the algorithm extrapolates interesting exploration behavior from the source tasks and outperforms different baselines. The main advantage of this method is *generality*, in the sense that it can be combined with any value-function approximator (such as neural networks) and any posterior distribution class (such as Gaussian and mixtures of Gaussian).

We concluded in Part III with a theoretical study of the problem of exploration in structured domains. We considered a popular decomposition of the transfer problem (Brunskill and Li, 2013; Azar et al., 2013a) into *learning structure* from previous tasks and *exploiting structure* to quickly learn new tasks. While we discussed the main existing approaches for learning structure from experience, we concentrated our contributions on the problem of exploiting given structure. We started from bandit problems with *known* structure. In Chapter 9, we designed SAE, an arm elimination strategy which exploits a given structure in a general form (i.e., without any specific assumption). SAE uses this structure to build tighter confidence sets about the true (unknown) bandit problem being faced, which in turn allows eliminating some sub-optimal arms quickly. Our main result is a theoretical analysis that, differently from those in previous works (Azar et al., 2013a; Lattimore and Munos, 2014) shows the possibility to discard some sub-optimal arms using the information from other arms. In Chapter 10, we restricted our attention to linear contextual structures and, in particular, to the problem of designing asymptotically problem-dependent-optimal strategies. We took inspiration from works that derive algorithms from asymptotic problem-dependent lower bounds (Lattimore and Szepesvari, 2017; Combes et al., 2017; Hao et al., 2019; Degenne et al., 2020b), and designed SOLID, a computationally-efficient incremental algorithm that is asymptotically optimal for linear contextual problems. We confirmed this property by deriving both problem-dependent and worst-case finite-time regret bounds. Notably, we showed that SOLID is the first algo-

rithm to be simultaneously *asymptotically problem-dependent optimal* and *finite-time minimax optimal* for linear (non-contextual) bandits. Experiments on both real and synthetic data confirmed competitive (often superior) performance with respect to state-of-the-art baselines (Abbasi-Yadkori et al., 2011; Agrawal and Goyal, 2013). In Chapter 11, we considered the problem of best policy identification in MDPs with learned (approximate) structure under the assumption that a generative model of the target task is available. We assumed that the target task belongs to a finite set of MDPs which are known only up to some approximation error. Along the lines of Brunskill and Li (2013), we designed PTUM, an elimination-based algorithm which aims at discarding "wrong" MDP models so as to quickly identify one that yields a near-optimal policy. In particular, PTUM actively queries the generative model of the target for state-action pairs whose samples yield high information for discriminating between realizable (approximate) MDP models. Notably, our theoretical analysis revealed that the sample complexity of PTUM is proportional to the one of an oracle strategy that chooses the state-action pairs that yield the *maximum information* for discriminating between MDP models, though these state-action pairs are not known in advance.

Open Problems and Future Works

As usual, all our approaches have limitations which leave us with some open problems. We conclude by discussing the ones that we believe to be more interesting for future research.

Transfer of samples. Perhaps the most important open question is whether our sample transfer methods are provably robust to negative transfer when the importance weights are *estimated*. In fact, for IWFQI we only provided a sample complexity bound as a function of the estimated weights, but we did not provide a technique for estimating the weights that provably reduces this sample complexity; for IWPG, on the other hand, we ensured robustness to negative transfer in the ideal case of known models, while we did not report any theoretical results for the estimated weights. We conjecture that the MSE-aware model estimation procedure of Chapter 6 might enjoy some robustness guarantees, though we left its analysis as an interesting direction for future work.

Variational transfer. While we designed our approach specifically for value-based settings, the most immediate extension would be to learn and exploit prior distributions over different components than action-value functions. For instance, since the main goal of the learned prior distribution is to drive the exploration in the target task, one could learn distributions over optimal policies. Interestingly, the resulting mixture-based algorithm would be quite similar to policy distillation methods (Rusu et al., 2015; Teh et al., 2017; Yin and Pan, 2017; Czarnecki et al., 2019), with the target policy being learned using an additional regularization that keeps it close to the source optimal policies. More generally, one could directly model distributions over latent task parameters, while learning a task-conditioned optimal policy to enable posterior sampling. While approaches in this direction have been proposed by Yang et al. (2019) and Rakelly et al. (2019), they assume the possibility to repeatedly sample tasks at train time. This is not possible in our setting where only a small finite set of source tasks is available, so we wonder whether similar ideas can be still applied.

One of the main messages from Part III is that exploration techniques build on top of the optimism principle or Thompson sampling might not be the optimal way to approach structured domains. Therefore, the second big question is whether other exploration strategies than posterior sampling should be adopted. In particular, one might try to learn the exploration strategy itself from the source tasks. Some meta-learning approaches have been proposed in this direction (see, e.g., Zintgraf et al. (2019); Kamienny et al. (2020) and discussion therein). Once again, their extension to our setting might be highly non-trivial since a meta-learning method would require to interact with the source tasks using different exploration strategies to assess which one is most suitable for the given domain.

Theory of structured domains. In the structured-bandit setting (with known structure), the literature is now quite vast. We have algorithms for general structures with good finite-time performance (like SUCB Lattimore and Munos (2014) or our SAE) but without asymptotic optimality, and algorithms with asymptotic optimality (Combes et al., 2017; Degenne et al., 2020b) which might sacrifice finite-time behavior. The most natural question is whether we can obtain the best of these two worlds and obtain finite-time optimal algorithms. Unfortunately, at the present time, we have no idea about what the finite-time problem-dependent lower bound for general structures looks like. We do not exclude that, when the learning horizon is small, ignoring the information from sub-optimal arms and acting optimistically might be the optimal way to behave.

For contextual linear bandits, we made a step in this direction by establishing an optimal problem-dependent regret bound for SOLID together with strong finite-time worst-case guarantees. While the algorithm is minimax optimal in linear (non-contextual) bandits, in contextual problems SOLID suffers and extra linear dependence on the number of contexts. This is in contrast to standard algorithms such as LinUCB (Abbasi-Yadkori et al., 2011) or LinTS Agrawal and Goyal (2013), which apply even to infinite contexts. We wonder whether this dependence could be reduced or removed, thus making SOLID minimax optimal for contextual problems. Another interesting question is whether this approach could be extended to deal with infinite/continuous contexts. Unfortunately, at the present time we do not know the problem-dependent lower bound for continuous contexts. One idea would be to consider a class of parameterized policies and derive the lower bound with respect to the best in-class policy.

Differently from bandits, in structured MDPs there is still a considerable amount of work to be done. Regarding regret minimization, the only asymptotically optimal algorithm for general structures (Ok et al., 2018) has limitations. On the computational side, similarly to OSSB, the algorithm is quite inefficient as it requires solving the optimization problem of the lower bound. In this sense, one could use ideas from SPL (Degenne et al., 2020b) and SOLID to design an efficient incremental approach. On the theoretical side, the lower bound itself is derived for ergodic MDPs (Burnetas and Katehakis, 1996). That is, the assumption is that each state is visited by any policy with positive probability, which could make the optimal asymptotic exploration strategy very inefficient in finite time. One direction would be to extend this lower bound to, e.g., communicating MDPs (Puterman, 2014), though this might require significant theoretical efforts.

Regarding the best policy identification setting, the assumption that a generative model of the target task is available is quite restrictive. The most natural direction for future work would be to remove this assumption, as we already hinted in Section 11.3.1. While PTUM

provably reduces the sample complexity for learning new tasks by exploiting structure (as compared to the unstructured approaches), one might be wondering whether it is possible to design strategies that are optimal in some sense. For instance, one could take inspiration from asymptotically optimal strategies for pure exploration in bandits (Garivier and Kaufmann, 2016; Degenne et al., 2019, 2020a) and derive a lower bound on the minimal sample complexity needed for identifying an ϵ-optimal policy of the specific problem. This would lead to an optimal allocation of samples to collect from different state-action pairs that an algorithm could try to realize in practice.

Finally, existing methods for learning structure from previous tasks with theoretical guarantees have limitations. The clustering-based approach of (Brunskill and Li, 2013) requires strong assumptions on the number of underlying MDP models and on their minimal distance, while spectral methods (Azar et al., 2013a) are very computationally demanding. We wonder whether methods that overcome these limitations can be designed while still providing theoretical guarantees on the learned structures. This might be possible by considering non-parametric models (Bonilla et al., 2008; Passos et al., 2012) or by assuming specific structures, e.g., by learning shared features in linear models (Argyriou et al., 2007).

Concluding Remarks

At last, we have reached the final sentences of this (long) thesis. I hope to have convinced the reader that knowledge transfer is a fundamental component to enable reinforcement learning agents in the real world. While, as usual, we open more problems than we solve, there are definitely many promising and exciting directions for future research.

Bibliography

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320.

Abeille, M. and Lazaric, A. (2017). Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184.

Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. (2018). Policy and value transfer in lifelong reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 20–29, Stockholmsmassan, Stockholm Sweden. PMLR.

Agrawal, R., Teneketzis, D., and Anantharam, V. (1988). Asymptotically efficient adaptive allocation schemes for controlled markov chains: Finite parameter space. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 1198–1203. IEEE.

Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1.

Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135.

Agrawal, S. and Goyal, N. (2017). Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)*, 64(5):1–24.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. (2018). Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*.

Allen-Zhu, Z. (2017). Natasha 2: Faster non-convex optimization than sgd. *arXiv preprint arXiv:1708.08694*.

Alquier, P., Ridgway, J., and Chopin, N. (2016). On the properties of variational approximations of gibbs posteriors. *Journal of Machine Learning Research*, 17(239):1–41.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Amit, R. and Meir, R. (2018). Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *Proceedings of the 35th International Conference on Machine Learning*.

Ammar, H. B., Eaton, E., Ruvolo, P., and Taylor, M. (2014). Online multi-task learning for policy gradient methods. In *International Conference on Machine Learning*, pages 1206–1214.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014). Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012). A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058.

Antos, A., Szepesvári, C., and Munos, R. (2008). Fitted q-iteration in continuous action-space mdps. In *Advances in neural information processing systems*, pages 9–16.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Asadi, K. and Littman, M. L. (2017). An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252.

Asadi, M. and Huber, M. (2007). Effective control knowledge transfer through learning skill and representation hierarchies. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2054–2059.

Atan, O., Tekin, C., and van der Schaar, M. (2018). Global bandits. *IEEE transactions on neural networks and learning systems*, 29(12):5798–5811.

Audibert, J.-Y. and Bubeck, S. (2010). Best arm identification in multi-armed bandits.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Auer, P., Jaksch, T., and Ortner, R. (2009). Near-optimal regret bounds for reinforcement learning. In *Advances in neural information processing systems*, pages 89–96.

Auer, P. and Ortner, R. (2010). Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65.

Azar, M., Lazaric, A., and Brunskill, E. (2013a). Sequential transfer in multi-armed bandit with finite set of models. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2220–2228. Curran Associates, Inc.

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

Azar, M. G., Lazaric, A., and Brunskill, E. (2013c). Regret bounds for reinforcement learning with policy advice. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 97–112. Springer.

Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. *arXiv preprint arXiv:1703.05449*.

Azizzadenesheli, K., Brunskill, E., and Anandkumar, A. (2018). Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*.

Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. (2016). Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1602.07764*.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier.

Banerjee, B. and Stone, P. (2007). General game learning using knowledge transfer.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. (2018). Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4055–4065. Curran Associates, Inc.

Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems 30*, pages 908–919. Curran Associates, Inc.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bertsekas, D. P. (2001). *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific, 2nd edition.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160.

Boucheron, S., Lugosi, G., and Bousquet, O. (2003). Concentration inequalities. In *Summer School on Machine Learning*, pages 208–240. Springer.

Bouneffouf, D. and Rish, I. (2019). A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Brafman, R. I. and Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231.

Bräm, T., Brunner, G., Richter, O., and Wattenhofer, R. (2019). Attentive multi-task deep reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 134–149. Springer.

Brunskill, E. and Li, L. (2013). Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*.

Bubeck, S., Cesa-Bianchi, N., et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

Burnetas, A. N. and Katehakis, M. N. (1996). Optimal adaptive policies for sequential allocation problems. *Advances in Applied Mathematics*, 17(2):122–142.

Calandriello, D., Lazaric, A., and Restelli, M. (2014). Sparse multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 819–827.

Cassandra, A. R. (1998). Exact and approximate algorithms for partially observable markov decision processes.

Castelletti, A., Galelli, S., Restelli, M., and Soncini-Sessa, R. (2010). Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(9).

Catoni, O. (2007). Pac-bayesian supervised classification: the thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*.

Cesa-Bianchi, N. and Lugosi, G. (2012). Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422.

Chatzilygeroudis, K., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., and Mouret, J.-B. (2017). Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58. IEEE.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. (2000a). An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems*, pages 987–993.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. (2000b). Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer.

Combes, R., Magureanu, S., and Proutiere, A. (2017). Minimal exploration in structured stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 1763–1771.

Combes, R., Magureanu, S., Proutiere, A., and Laroche, C. (2015). Learning to rank: Regret lower bounds and efficient algorithms. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):231–244.

Combes, R. and Proutière, A. (2014). Unimodal bandits: Regret lower bounds and optimal algorithms. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 521–529. JMLR.org.

Cortes, C., Mansour, Y., and Mohri, M. (2010). Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450.

Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2008). Sample selection bias correction theory. In *International Conference on Algorithmic Learning Theory*, pages 38–53. Springer.

Cotter, N. E. and Conwell, P. R. (1990). Fixed-weight networks can learn. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 553–559. IEEE.

Crammer, K., Kearns, M., and Wortman, J. (2008). Learning from multiple sources. *Journal of Machine Learning Research*, 9(Aug):1757–1774.

Croonenborghs, T., Driessens, K., and Bruynooghe, M. (2007). Learning relational options for inductive transfer in relational reinforcement learning. In *International Conference on Inductive Logic Programming*, pages 88–97. Springer.

Csiszár, I. (1967). Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318.

Czarnecki, W. M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., and Jaderberg, M. (2019). Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340.

Da Silva, F. L. and Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703.

Degenne, R., Koolen, W. M., and Ménard, P. (2019). Non-asymptotic pure exploration by solving games. In *Advances in Neural Information Processing Systems*, pages 14465–14474.

Degenne, R., Ménard, P., Shang, X., and Valko, M. (2020a). Gamification of pure exploration for linear bandits. *arXiv preprint arXiv:2007.00953*.

Degenne, R., Shao, H., and Koolen, W. (2020b). Structure adaptive algorithms for stochastic bandits. In *ICML*, Preprint (personal communication).

Deisenroth, M. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. (2019). Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*.

Dorato, P., Abdallah, C. T., Cerone, V., and Jacobson, D. H. (1995). *Linear-quadratic control: an introduction*. Prentice Hall Englewood Cliffs, NJ.

Doshi-Velez, F. and Konidaris, G. (2016). Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Dyagilev, K., Mannor, S., and Shimkin, N. (2008). Efficient reinforcement learning in parameterized models: Discrete parameter case. In *European Workshop on Reinforcement Learning*, pages 41–54. Springer.

Efroni, Y., Mannor, S., and Pirotta, M. (2020). Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*.

Elvira, V., Martino, L., and Robert, C. P. (2018). Rethinking the effective sample size. *arXiv preprint arXiv:1809.04129*.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416.

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092.

Farahmand, A.-m. (2011). *Regularization in reinforcement learning*. PhD thesis, University of Alberta.

Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., and Mannor, S. (2009). Regularized fitted q-iteration for planning in continuous-space markovian decision problems. In *2009 American Control Conference*, pages 725–730. IEEE.

Farahmand, A. M. and Precup, D. (2012). Value pursuit iteration. In *Advances in Neural Information Processing Systems*, pages 1340–1348.

Feng, F., Yin, W., and Yang, L. F. (2019). Does knowledge transfer always help to learn a better policy? *arXiv preprint arXiv:1912.02986*.

Fernández, F. and Veloso, M. (2006). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727.

Ferrante, E., Lazaric, A., and Restelli, M. (2008). Transfer of task representation in reinforcement learning using policy-based proto-value functions.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Finn, C., Xu, K., and Levine, S. (2018). Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. (2018). An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*.

Garcke, J. and Vanck, T. (2014). Importance weighted inductive transfer learning for regression. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 466–481. Springer.

Garivier, A. and Cappé, O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pages 359–376.

Garivier, A. and Kaufmann, E. (2016). Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027.

Geist, M. and Scherrer, B. (2014). Off-policy learning with eligibility traces: A survey. *The Journal of Machine Learning Research*, 15(1):289–333.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

Ghosh, A., Chowdhury, S. R., and Gopalan, A. (2017). Misspecified linear bandits. *arXiv preprint arXiv:1704.06880*.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*.

Graves, T. L. and Lai, T. L. (1997). Asymptotically efficient adaptive choice of control laws incontrolled markov chains. *SIAM journal on control and optimization*, 35(3):715–743.

Guedj, B. (2019). A primer on pac-bayesian learning. *arXiv preprint arXiv:1901.05353*.

Guo, Z., Thomas, P. S., and Brunskill, E. (2017). Using options and covariance testing for long horizon off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pages 2492–2501.

Guo, Z. D., Doroudi, S., and Brunskill, E. (2016). A pac rl algorithm for episodic pomdps. In *Artificial Intelligence and Statistics*, pages 510–518.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018a). Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311.

Gupta, S., Joshi, G., and Yağan, O. (2018b). Exploiting correlation in finite-armed structured bandits. *arXiv preprint arXiv:1810.08164*.

Györfi, L., Kohler, M., Krzyzak, A., and Walk, H. (2006). *A distribution-free theory of nonparametric regression*. Springer Science & Business Media.

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

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018b). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hachiya, H., Akiyama, T., Sugiayma, M., and Peters, J. (2009). Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, 22(10):1399–1410.

Hachiya, H., Peters, J., and Sugiyama, M. (2011). Reward-weighted regression with sample reuse for direct policy search in reinforcement learning. *Neural Computation*, 23(11):2798–2832.

Hallak, A., Di Castro, D., and Mannor, S. (2015). Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*.

Hammersley, J. and Handscomb, D. (1964). *Monte Carlo Methods*. Methuen's monographs on applied probability and statistics. Methuen.

Hao, B., Lattimore, T., and Szepesvari, C. (2019). Adaptive exploration in linear contextual bandit. *arXiv preprint arXiv:1910.06996*.

Hasselt, H. V. (2010). Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621.

Hershey, J. R. and Olsen, P. A. (2007). Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. (2019). Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803.

Hesterberg, T. (1995). Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. (2019). Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*.

Ionides, E. L. (2008). Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311.

Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710.

Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600.

Jedra, Y. and Proutiere, A. (2020). Optimal best-arm identification in linear bandits. *arXiv preprint arXiv:2006.16073*.

Jian, Q., Fruit, R., Pirotta, M., and Lazaric, A. (2019). Exploration bonus for regret minimization in discrete and continuous average reward mdps. In *Advances in Neural Information Processing Systems*, pages 4890–4899.

Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 652–661, New York, New York, USA. PMLR.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. (2020). Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143.

Jong, N. K. and Stone, P. (2007). Model-based exploration in continuous state spaces. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 258–272. Springer.

Jun, K.-S. and Zhang, C. (2020). Crush optimism with pessimism: Structured bandits beyond asymptotic optimality. *arXiv preprint arXiv:2006.08754*.

Kamienny, P.-A., Pirotta, M., Lazaric, A., Lavril, T., Usunier, N., and Denoyer, L. (2020). Learning adaptive exploration strategies in dynamic environments through informed policy regularization. *arXiv preprint arXiv:2005.02934*.

Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *International conference on algorithmic learning theory*, pages 199–213. Springer.

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232.

Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems*, pages 6250–6261.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kober, J. and Peters, J. R. (2009). Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Kong, A. (1992). A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep*, 348.

Konidaris, G. and Barto, A. (2006). Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496.

Konidaris, G. and Barto, A. (2007). Building portable options: skill transfer in reinforcement learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 895–900.

Konidaris, G., Scheidwasser, I., and Barto, A. G. (2012). Transfer in reinforcement learning via shared features. *The Journal of Machine Learning Research*, 13(1):1333–1371.

Kuss, M. and Rasmussen, C. E. (2004). Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 751–758. MIT Press.

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

Lakshmanan, K., Ortner, R., and Ryabko, D. (2015). Improved regret bounds for undiscounted continuous reinforcement learning. In *International Conference on Machine Learning*, pages 524–532.

Lan, L., Li, Z., Guan, X., and Wang, P. (2019). Meta reinforcement learning with task embedding and shared policy. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2794–2800. AAAI Press.

Lange, S., Gabel, T., and Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer.

Laroche, R. and Barlier, M. (2017). Transfer reinforcement learning with shared dynamics. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Lattimore, T. and Munos, R. (2014). Bounded regret for finite-armed structured bandits. In *Advances in Neural Information Processing Systems*, pages 550–558.

Lattimore, T. and Szepesvari, C. (2017). The end of optimism? an asymptotic analysis of finite-armed linear bandits. In *Artificial Intelligence and Statistics*, pages 728–737.

Lattimore, T. and Szepesvari, C. (2019). Learning with good feature representations in bandits and in rl with a generative model. *arXiv preprint arXiv:1911.07676*.

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

Lazaric, A. (2012). Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer.

Lazaric, A. and Ghavamzadeh, M. (2010). Bayesian multi-task reinforcement learning.

Lazaric, A. and Restelli, M. (2011). Transfer from multiple mdps. In *Advances in Neural Information Processing Systems*, pages 1746–1754.

Lazaric, A., Restelli, M., and Bonarini, A. (2008a). Reinforcement learning in continuous action spaces through sequential monte carlo methods. In *Advances in neural information processing systems*, pages 833–840.

Lazaric, A., Restelli, M., and Bonarini, A. (2008b). Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551. ACM.

Le Cam, L. (1986). *Asymptotic methods in statistical decision theory*. Springer Science & Business Media.

Lehnert, L. and Littman, M. L. (2018). Transfer with model features in reinforcement learning. *arXiv preprint arXiv:1807.01736*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Li, L., Munos, R., and Szepesvari, C. (2015a). Toward Minimax Off-policy Value Estimation. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 608–616, San Diego, California, USA. PMLR.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for mdps.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Li, Y., Tian, X., Liu, T., and Tao, D. (2015b). Multi-task model and feature joint learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, H., Socher, R., and Xiong, C. (2019a). Taming maml: Efficient unbiased meta-reinforcement learning. In *International Conference on Machine Learning*, pages 4061–4071.

Liu, J. S. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119.

Liu, L. T., Dogan, U., and Hofmann, K. (2016a). Decoding multitask dqn in the world of minecraft.

Liu, M., Chowdhary, G., and How, J. P. (2012). Transfer learning for reinforcement learning with dependent dirichlet process and gaussian process.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5361–5371.

Liu, Y., Guo, Z., and Brunskill, E. (2016b). Pac continuous state online multitask reinforcement learning with identification. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 438–446.

Liu, Y., Hu, Y., Gao, Y., Chen, Y., and Fan, C. (2019b). Value function transfer for deep multi-agent reinforcement learning based on n-step returns. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 457–463. AAAI Press.

Liu, Y. and Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the 21st national conference on Artificial intelligence-Volume 1*, pages 415–420.

Magureanu, S., Combes, R., and Proutiere, A. (2014). Lipschitz bandits: Regret lower bounds and optimal algorithms. *arXiv preprint arXiv:1405.4758*.

Mahmood, A. R., van Hasselt, H. P., and Sutton, R. S. (2014). Weighted importance sampling for off-policy learning with linear function approximation. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3014–3022. Curran Associates, Inc.

Mahmud, M., Hawasly, M., Rosman, B., and Ramamoorthy, S. (2013). Clustering markov decision processes for continual transfer. *arXiv preprint arXiv:1311.3959*.

Maillard, O.-A., Munos, R., Lazaric, A., and Ghavamzadeh, M. (2010). Finite-sample analysis of bellman residual minimization. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 299–314.

Mann, T. A. and Choe, Y. (2013). Directed exploration in reinforcement learning with transferred knowledge. In *European Workshop on Reinforcement Learning*, pages 59–76.

Martino, L., Elvira, V., and Louzada, F. (2017). Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401.

Maurer, A. and Pontil, M. (2009). Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*.

Mehta, N., Natarajan, S., Tadepalli, P., and Fern, A. (2008). Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289.

Melo, F. S. (2001). Convergence of q-learning: A simple proof.

Melo, F. S. and Ribeiro, M. I. (2007). Convergence of q-learning with linear function approximation. In *2007 European Control Conference (ECC)*, pages 2671–2678. IEEE.

Micchelli, C. A. and Pontil, M. (2005). On learning vector-valued functions. *Neural computation*, 17(1):177–204.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Modi, A., Jiang, N., Singh, S., and Tewari, A. (2018). Markov decision processes with continuous side information. In *Algorithmic Learning Theory*, pages 597–618.

Monahan, G. E. (1982). State of the art–a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16.

Mülling, K., Kober, J., Kroemer, O., and Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062.

Munos, R. and Szepesvári, C. (2008). Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*.

Nedić, A. and Ozdaglar, A. (2009). Subgradient methods for saddle-point problems. *Journal of optimization theory and applications*, 142(1):205–228.

Neumann, G. and Peters, J. R. (2009). Fitted q-iteration by advantage weighted regression. In *Advances in neural information processing systems*, pages 1177–1184.

Ok, J., Proutiere, A., and Tranos, D. (2018). Exploration in structured reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8874–8882.

Osband, I., Van Roy, B., Russo, D. J., and Wen, Z. (2019). Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62.

Osband, I., Van Roy, B., and Wen, Z. (2014). Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*.

Owen, A. and Zhou, Y. (2000). Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Passos, A., Rai, P., Wainer, J., and Daume III, H. (2012). Flexible modeling of latent task structures in multitask learning. *arXiv preprint arXiv:1206.6486*.

Paterniti, C. (2020). *Learning from expert demonstrations on F1 simulators, with transfer learning across different vehicle setups*. Politecnico di Milano MS Thesis.

Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

Penner, A. (2002). The physics of putting. *Canadian Journal of Physics*, 80(2):83–96.

Peters, J. and Schaal, S. (2008a). Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

Peters, J. and Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

Precup, D. (2000). Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80.

Precup, D., Sutton, R., and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proceedings of the 18th International Conference on Machine Learning*.

Prokhorov, D. V., Feldkarnp, L., and Tyukin, I. Y. (2002). Adaptive behavior with fixed weights in rnn: an overview. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2018–2022. IEEE.

Pukelsheim, F. (2006). *Optimal design of experiments*. SIAM.

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

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340.

Ren, H., Garg, A., and Anandkumar, A. (2020). Context-based meta-reinforcement learning with structured latent space.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Rosman, B., Hawasly, M., and Ramamoorthy, S. (2016). Bayesian policy reuse. *Machine Learning*, 104(1):99–127.

Russo, D. and Van Roy, B. (2016). An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. (2015). Policy distillation. *arXiv preprint arXiv:1511.06295*.

Sæmundsson, S., Hofmann, K., and Deisenroth, M. P. (2018). Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015a). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015b). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schmidhuber, J. (1987). *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Seldin, Y., Laviolette, F., Cesa-Bianchi, N., Shawe-Taylor, J., and Auer, P. (2012). Pac-bayesian inequalities for martingales. *IEEE Transactions on Information Theory*.

Sidford, A., Wang, M., Wu, X., Yang, L., and Ye, Y. (2018). Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems*, pages 5186–5196.

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

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms.

Sinclair, S. R., Banerjee, S., and Yu, C. L. (2019). Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–44.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308.

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3-4):323–339.

Snel, M. and Whiteson, S. (2011). Multi-task reinforcement learning: Shaping and feature selection. In *European Workshop on Reinforcement Learning*, pages 237–248. Springer.

Song, Z. and Sun, W. (2019). Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*.

Spaan, M. T. (2012). Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer.

Strehl, A. L., Li, L., and Littman, M. L. (2009). Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density ratio estimation in machine learning*. Cambridge University Press.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. (2018). Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. *arXiv preprint arXiv:1811.08540*.

Sun, Y., Yin, X., and Huang, F. (2020). Temple: Learning template of transitions for sample efficient multi-task rl. *arXiv preprint arXiv:2002.06659*.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044.

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

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Szepesvári, C. (1998). The asymptotic convergence-rate of q-learning. In *Advances in Neural Information Processing Systems*, pages 1064–1070.

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

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer.

Tanaka, F. and Yamamura, M. (2003). Multitask reinforcement learning on the distribution of mdps. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*, volume 3, pages 1108–1113. IEEE.

Taneja, I. J. (2004). Generalized relative information and information inequalities. *Journal of Inequalities in Pure and Applied Mathematics*, 5(1):1–19.

Taylor, M. E., Jong, N. K., and Stone, P. (2008). Transferring instances for model-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 488–505. Springer.

Taylor, M. E. and Stone, P. (2005). Behavior transfer for value-function-based reinforcement learning. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 53–59.

Taylor, M. E. and Stone, P. (2007). Representation transfer for reinforcement learning.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685.

Taylor, M. E., Stone, P., and Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep):2125–2167.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506.

Thomas, P. and Brunskill, E. (2016). Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Thrun, S. and Pratt, L. (2012). *Learning to learn*. Springer Science & Business Media.

Tirinzoni, A., Lazaric, A., and Restelli, M. (2020a). A novel confidence-based algorithm for structured bandits. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3175–3185, Online. PMLR.

Tirinzoni, A., Pirotta, M., Restelli, M., and Lazaric, A. (2020b). An asymptotically optimal primal-dual incremental algorithm for linear contextual bandits. In *Advances in Neural Information Processing Systems 33*.

Tirinzoni, A., Poiani, R., and Restelli, M. (2020c). Sequential transfer in reinforcement learning with a generative model. In *Proceedings of the 37th International Conference on Machine Learning*.

Tirinzoni, A., Rodriguez Sanchez, R., and Restelli, M. (2018a). Transfer of value functions via variational methods. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 6179–6189. Curran Associates, Inc.

Tirinzoni, A., Salvini, M., and Restelli, M. (2019). Transfer of samples in policy search via multiple importance sampling. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6264–6274, Long Beach, California, USA. PMLR.

Tirinzoni, A., Sessa, A., Pirotta, M., and Restelli, M. (2018b). Importance weighted transfer of samples in reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4936–4945, Stockholmsmassan, Stockholm Sweden. PMLR.

Tosatto, S., Pirotta, M., d'Eramo, C., and Restelli, M. (2017). Boosted fitted q-iteration. In *International Conference on Machine Learning*, pages 3434–3443.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202.

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. (2017). Human learning in atari.

Van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning.

Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.

Veach, E. (1997). *Robust monte carlo methods for light transport simulation*. Number 1610. Stanford University PhD thesis.

Veach, E. and Guibas, L. J. (1995). Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428. ACM.

Vuorio, R., Sun, S.-H., Hu, H., and Lim, J. J. (2018). Toward multimodal model-agnostic meta-learning. *arXiv preprint arXiv:1812.07172*.

Vuorio, R., Sun, S.-H., Hu, H., and Lim, J. J. (2019). Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, pages 1–12.

Walsh, T. J., Li, L., and Littman, M. L. (2006). Transferring state abstractions between mdps.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016a). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wang, X., Bi, J., Yu, S., Sun, J., and Song, M. (2016b). Multiplicative multitask feature learning. *The Journal of Machine Learning Research*, 17(1):2820–2852.

Wang, Z., Dai, Z., Póczos, B., and Carbonell, J. (2019). Characterizing and avoiding negative transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11293–11302.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016c). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003.

Wang, Z., Zhou, R., and Shen, C. (2018). Regional multi-armed bandits. *arXiv preprint arXiv:1802.07917*.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1):9.

Williams, C. K. and Rasmussen, C. E. (2006). Gaussian processes for machine learning. *the MIT Press*, 2(3):4.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022.

Wilson, A., Fern, A., and Tadepalli, P. (2012). Transfer learning in sequential decision problems: A hierarchical bayesian approach. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 217–227.

Woodworth, R. S. and Thorndike, E. (1901). The influence of improvement in one mental function upon the efficiency of other functions.(i). *Psychological review*, 8(3):247.

Wulfmeier, M., Abdolmaleki, A., Hafner, R., Springenberg, J. T., Neunert, M., Hertweck, T., Lampe, T., Siegel, N., Heess, N., and Riedmiller, M. (2019). Compositional tran sf er in hierarchical reinforcement learning.

Yang, J., Petersen, B., Zha, H., and Faissol, D. (2019). Single episode policy transfer in reinforcement learning. *arXiv preprint arXiv:1910.07719*.

Yang, K., Yang, L. F., and Du, S. S. (2020). q-learning with logarithmic regret. *arXiv preprint arXiv:2006.09118*.

Yang, L. F. and Wang, M. (2019). Sample-optimal parametric q-learning using linearly additive features. *arXiv preprint arXiv:1902.04779*.

Yang, Z., Merrick, K., Abbass, H., and Jin, L. (2017). Multi-task deep reinforcement learning for continuous action control. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3301–3307.

Yin, H. and Pan, S. J. (2017). Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI conference on artificial intelligence*.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342.

Yu, J. Y. and Mannor, S. (2011). Unimodal bandits. In *ICML*, pages 41–48. Citeseer.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2019). Multi-task reinforcement learning without interference.

Zaki, M., Mohan, A., and Gopalan, A. (2020). Explicit best arm identification in linear bandits using no-regret learners. *arXiv preprint arXiv:2006.07562*.

Zanette, A., Brandfonbrener, D., Brunskill, E., Pirotta, M., and Lazaric, A. (2020). Frequentist regret bounds for randomized least-squares value iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 1954–1964.

Zanette, A., Kochenderfer, M. J., and Brunskill, E. (2019). Almost horizon-free structure-aware best policy identification with a generative model. In *Advances in Neural Information Processing Systems*, pages 5625–5634.

Zhan, Y., Ammar, H. B., et al. (2016). Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. *arXiv preprint arXiv:1604.03986*.

Zhang, Y. and Zavlanos, M. M. (2020). Transfer reinforcement learning under unobserved contextual information. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pages 75–86. IEEE.

Zhao, T., Hachiya, H., Tangkaratt, V., Morimoto, J., and Sugiyama, M. (2013). Efficient sample reuse in policy gradients with parameter-based exploration. *Neural computation*, 25(6):1512–1547.

Zhou, W., Pinto, L., and Gupta, A. (2019). Environment probing interaction policies. *arXiv preprint arXiv:1907.11740*.

Zhu, Z., Lin, K., and Zhou, J. (2020). Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. (2019). Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*.

APPENDIX $\mathcal{A}$

Proofs of Chapter 6

Proof of Proposition 6.4.2

Suppose our current dataset $\mathcal{D}$ contains n trajectories, with the target distribution being $p(\tau) = p(\tau|\theta_0, \mathcal{M}_0)$ and the mixture of source proposals being $q_\alpha(\tau) = \sum_{j=1}^m \alpha_j p_j(\tau)$, with $p_j(\tau) = p(\tau|\theta_j, \mathcal{M}_j)$. Notice that $q_\alpha(\tau)$ does not necessarily contain the target distribution p since the number of defensive samples has still to be computed. If we add n_0 defensive samples, the resulting ESS (according to the lower bound of Proposition 6.4.1) is

$$\begin{aligned}\mathrm{ESS}(n_0; \mathcal{D}) &= \frac{n+n_0}{\int \frac{p(\tau)^2}{\sum_{j=1}^m \frac{n_j}{n+n_0} p_j(\tau) + \frac{n_0}{n+n_0} p(\tau)} \mathrm{d}\tau} = \frac{n+n_0}{\mathbb{E}_{\tau\sim q_\alpha(\tau;n_0)}\left[\frac{p(\tau)^2}{q_\alpha(\tau;n_0)^2}\right]} \\ &= \frac{n+n_0}{1+\mathbb{V}\mathrm{ar}_{\tau\sim q_\alpha(\tau;n_0)}\left[\frac{p(\tau)}{q_\alpha(\tau;n_0)}\right]} = \frac{n+n_0}{1+\int q_\alpha(\tau;n_0)\left(\frac{p(\tau)}{q_\alpha(\tau;n_0)}-1\right)^2},\end{aligned}$$

where we use $q_\alpha(\tau; n_0)$ to denote the updated mixture after collecting n_0 defensive trajectories from p. Let us now approximate the variance term using our current samples. To simplify the notation, let us index the samples with $i = 1, \ldots, n$, while dropping the index j of the proposal which generated

the sample itself (under the balance heuristics, the weight does not depend on j). We have

$$\begin{aligned}\mathbb{V}\mathrm{ar}_{\tau\sim q_\alpha(\tau;n_0)}\left[\frac{p(\tau)}{q_\alpha(\tau;n_0)}\right] &\simeq \frac{1}{n}\sum_{i=1}^{n}\frac{q_\alpha(\tau_i;n_0)}{q_\alpha(\tau_i)}\left(\frac{p(\tau_i)}{q_\alpha(\tau_i;n_0)}-1\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{p(\tau_i)^2}{q_\alpha(\tau_i)q_\alpha(\tau_i;n_0)} + \frac{1}{n}\sum_{i=1}^{n}\frac{q_\alpha(\tau_i;n_0)}{q_\alpha(\tau_i)} - \frac{2}{n}\sum_{i=1}^{n}\frac{p(\tau_i)}{q_\alpha(\tau_i)} \\ &= (n+n_0)\sum_{i=1}^{n}\frac{a_i^2}{b_i(b_i+a_in_0)} + \frac{n}{n+n_0} + \frac{n_0}{n+n_0}\sum_{i=1}^{n}\frac{a_i}{b_i} - 2\sum_{i=1}^{n}\frac{a_i}{b_i},\end{aligned}$$

where we defined the constants $a_i := p(\tau_i)$ and $b_i := \sum_{j=1}^{m} n_j p_j(\tau_i)$. Thus, the ESS improvement as a function of n_0 can be approximated as

$$\begin{aligned}\widehat{\mathrm{ESS}}(n_0;\widetilde{\mathcal{D}}) &= \frac{n+n_0}{1+(n+n_0)\sum_{i=1}^{n}\frac{a_i^2}{b_i(b_i+a_in_0)} + \frac{n}{n+n_0} + \frac{n_0}{n+n_0}\sum_{i=1}^{n}\frac{a_i}{b_i} - 2\sum_{i=1}^{n}\frac{a_i}{b_i}} \\ &= \frac{1}{\sum_{i=1}^{n}\frac{a_i^2}{b_i(b_i+a_in_0)} + \frac{n}{(n+n_0)^2} + \frac{n_0}{(n+n_0)^2}\sum_{i=1}^{n}\frac{a_i}{b_i} + \frac{1}{n+n_0}\left(1-2\sum_{i=1}^{n}\frac{a_i}{b_i}\right)}.\end{aligned}$$

Note that $\widehat{\mathrm{ESS}}(0;\widetilde{\mathcal{D}}) = \frac{n}{1+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{p(\tau_i)}{q_\alpha(\tau_i)}-1\right)^2}$, which is exactly our ESS measure. Furthermore, this function is strictly increasing for $n_0 \geq 0$ and $\lim_{n_0\to\infty}\frac{\mathrm{ESS}(n_0;\mathcal{D})}{n_0} = 1$, i.e., the asymptotic increase rate is linear with slope 1. Therefore, $\widehat{\mathrm{ESS}}(n_0;\widetilde{\mathcal{D}}) \geq \widehat{\mathrm{ESS}}(0;\widetilde{\mathcal{D}}) + n_0 \inf_{x\in(0,+\infty)}\widehat{\mathrm{ESS}}'(x;\widetilde{\mathcal{D}})$. It is easy to check that $\inf_{x\in(0,+\infty)}\widehat{\mathrm{ESS}}'(x;\widetilde{\mathcal{D}})$ is either 1, when the function grows at a rate that is always grater than the asymptotic one, or c, when the initial rate is smaller. Thus,

$$\widehat{\mathrm{ESS}}(n_0;\widetilde{\mathcal{D}}) \geq \frac{n}{1+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{p(\tau_i)}{q_\alpha(\tau_i)}-1\right)^2} + \min\{1,c\}n_0.$$

Using this equation and setting the right-hand side to $\mathrm{ESS}_{\min}$, we get that the number n_0 of defensive samples to guarantee an ESS of at least $\mathrm{ESS}_{\min}$ can be approximated as

$$n_0 = \left\lceil \frac{\mathrm{ESS}_{\min} - \frac{n}{1+\widehat{\mathbb{V}\mathrm{ar}}[w]}}{\min\{1,c\}} \right\rceil.$$

Finally, we clip this value to $n_{\min}$ below, as required by our algorithm, and to $\mathrm{ESS}_{\min}$ above since the collecting $\mathrm{ESS}_{\min}$ samples is sufficient to guarantee an ESS of at least such value. In fact, if we have $\mathrm{ESS}_{\min}$ samples from the target p in our dataset, $d_2(p\|q_\alpha) \leq \frac{n}{\mathrm{ESS}_{\min}}$ since the importance weights are bounded by $\frac{1}{\alpha_0} = \frac{n}{\mathrm{ESS}_{\min}}$. Hence, $\frac{n}{d_2(p\|q_\alpha)} \geq \mathrm{ESS}_{\min}$.

Proof of Theorem 6.5.2

Since the proof relies on the theory of f-divergences (Csiszár, 1967), let us first recall some basic definitions. Let $f : (0,\infty) \to \mathbb{R}$ be a convex function such that $f(1) = 0$. Then, the f-divergence between P and Q is defined as:

$$D_f(P\|Q) = \int f\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)\mathrm{d}Q. \tag{A.1}$$

An example of f-divergence, which we will adopt in the remaining, is the chi-square divergence (Pearson, 1900), which is given by

$$D_{\chi^2}(P||Q) = \int \left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)^2 \mathrm{d}Q - 1, \tag{A.2}$$

or, equivalently, as the f-divergence with $f_{\chi^2}(w) = (w-1)^2$. Then,

$$d_2(P\|Q_\alpha) = \int \left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)^2 \mathrm{d}Q = D_{\chi^2}(P||Q_\alpha) + 1. \tag{A.3}$$

Let us now introduce an f-divergence $\Delta_\alpha(P||Q)$ defined by the function

$$f_{\Delta_\alpha}(w) = \frac{(w-1)^2}{\frac{\alpha}{1-\alpha}w + 1}. \tag{A.4}$$

This diverge can be seen as a skewed version of the triangular discrimination $\Delta(P||Q)$ (Le Cam, 1986), which is given by (A.4) for the particular case $\alpha = \frac{1}{2}$. Furthermore, it is easy to check that $D_{\chi^2}(P||Q_\alpha) = (1-\alpha)\Delta_\alpha(P||Q)$. Thus, in order to bound $d_2(P\|Q_\alpha)$ we only need to bound $\Delta_\alpha(P||Q)$.

Note that f_{Δ_α} is twice differential on $(0,\infty)$ and that, by assumption, $\operatorname{ess\,sup} \frac{dP}{dQ} \leq C$. Then, we can apply Theorem 3.1 of Taneja (2004)[1] to obtain

$$\Delta_\alpha(P||Q) \leq D_{\mathrm{KL}}(P||Q) \sup_{w\in(0,C)} w f''_{\Delta_\alpha}(w). \tag{A.5}$$

Let us now compute the constant multiplying the KL divergence on the right-hand side. A simple algebra shows that the first derivative of f_{Δ_α} is

$$f'_{\Delta_\alpha}(w) = \frac{(w-1)\left(\frac{\alpha}{1-\alpha}w + \frac{\alpha}{1-\alpha} + 2\right)}{\left(\frac{\alpha}{1-\alpha}w + 1\right)^2},$$

while the second derivative is

$$f''_{\Delta_\alpha}(w) = \frac{2\left(\frac{\alpha}{1-\alpha} + 1\right)^2}{\left(\frac{\alpha}{1-\alpha}w + 1\right)^3}.$$

Let us define $g_{\Delta_\alpha}(w) := w f''_{\Delta_\alpha}(w)$. Then,

$$g'_{\Delta_\alpha}(w) = \frac{2\left(\frac{\alpha}{1-\alpha} + 1\right)^2 \left(1 - 2\frac{\alpha}{1-\alpha}w\right)}{\left(\frac{\alpha}{1-\alpha}w + 1\right)^4}.$$

This function is positive for $w \leq \frac{1-\alpha}{2\alpha}$ and negative for $w \geq \frac{1-\alpha}{2\alpha}$. Thus,

$$\sup_{w\in(0,C)} g_{\Delta_\alpha}(w) = g_{\Delta_\alpha}(C) = \frac{2C\left(\frac{\alpha}{1-\alpha} + 1\right)^2}{\left(\frac{\alpha}{1-\alpha}C + 1\right)^3}$$

[1] Technically speaking, Taneja (2004) consider only discrete spaces. However, their result generalizes straightforwardly to general probability measures.

for $C \leq \frac{1-\alpha}{2\alpha}$, while

$$\sup_{w \in (0,C)} g_{\Delta_\alpha}(w) = g_{\Delta_\alpha}\left(\frac{1-\alpha}{2\alpha}\right) = \frac{8}{27}\frac{1}{\alpha(1-\alpha)}$$

in the opposite case. The theorem follows after multiplying these two constants by $1-\alpha$ and plugging everything into (A.3).

Proof of Proposition 6.5.1

From Theorem 6.5.2 we know that $d_2\left(p(\cdot|\theta_0, \widetilde{f}_0)\|q_\alpha(\cdot|\widetilde{f}_0)\right) \leq 1 + u(\alpha) D_{\mathrm{KL}}\left(p(\cdot|\theta_0, \widetilde{f}_0)\|\bar{q}_\alpha(\cdot|\widetilde{f}_0)\right)$, where we write $\bar{q}$ to denote the normalized mixture of proposals without the defensive component $p(\cdot|\theta_0, \widetilde{f}_0)$. Neglecting the constant term (which does not depend on $\widetilde{f}$), we have

$$\begin{aligned}
D_{\mathrm{KL}}\left(p(\cdot|\theta_0, \widetilde{f}_0), \bar{q}_\alpha(\cdot|\widetilde{f}_0)\right) &\leq \frac{1}{1-\alpha_0}\sum_{j=1}^{m} \alpha_j D_{\mathrm{KL}}\left(p(\cdot|\theta_0, \widetilde{f}_0)\|p(\cdot|\theta_j, \widetilde{f}_j)\right) \\
&= \frac{1}{1-\alpha_0}\sum_{j=1}^{m} \alpha_j \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)}\left[\log \frac{p(\tau|\theta_0, \widetilde{f}_0)}{p(\tau|\theta_j, \widetilde{f}_j)}\right] \\
&= \frac{1}{1-\alpha_0}\sum_{j=1}^{m} \alpha_j \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)}\left[\sum_{t=0}^{T-1} \log \frac{\widetilde{P}_0(S_{t+1}|S_t, A_t)}{\widetilde{P}_j(S_{t+1}|S_t, A_t)}\right] \\
&\quad + \frac{1}{1-\alpha_0}\sum_{j=1}^{m} \alpha_j \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)}\left[\sum_{t=0}^{T-1} \log \frac{\pi_{\theta_0}(A_t|S_t)}{\pi_{\theta_j}(A_t|S_t)}\right] \\
&= \frac{1}{1-\alpha_0}\sum_{j=0}^{m} \alpha_j \sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)}\left[D_{\mathrm{KL}}(\widetilde{P}_0(\cdot|S_t, A_t), \widetilde{P}_j(\cdot|S_t, A_t))\right] + \text{const} \\
&= \frac{1}{1-\alpha_0}\frac{1}{2\sigma^2}\sum_{t=0}^{T-1} \mathbb{E}_{\tau \sim p(\cdot|\theta_0, \widetilde{f}_0)}\left[\sum_{j=0}^{m} \alpha_j \|\widetilde{f}_0(x_t) - \widetilde{f}_j(x_t)\|_2^2\right] + \text{const},
\end{aligned}$$

where the first inequality follows from the convexity of the function $1/x$ and Jensen's inequality. Note that the expected KL divergence between policies can be considered constant since, according to the approximation introduced in Section 6.5.2, the expectation is not computed under the current model $\widetilde{f}_0$. The last term in (6.17) can be safely regarded as a constant since the integrand does not depend on $\widetilde{f}_0$, and so do all terms involving only source models. Noting that the bias term remained unchanged, the proposition is obtained after renaming the constants.

Auxiliary Results

Lemma A.4.1. *Let $Q_1, \ldots, Q_m$ be probability measures over $(\mathcal{X}, \mathscr{F})$, $Q_\alpha = \sum_{j=1}^m \alpha_j Q_j$ be a mixture of these measures with coefficients $\alpha_j \geq 0$ such that $\sum_{j=1}^m \alpha_j = 1$, and $f : \mathcal{X} \to \mathbb{R}$ be any measurable function. Consider $\hat{\mu} = \frac{1}{n}\sum_{j=1}^m \sum_{i=1}^{n_j} f(x_{i,j})$ where $x_{i,j}$ are i.i.d. samples and $n = \sum_{j=1}^m n_j$. Then, choosing $\alpha_j = \frac{n_j}{n}$, for each $j \in \{1, \ldots, m\}$, $\mathbb{V}\mathrm{ar}_{x_{i,j} \sim Q_j}[\hat{\mu}] \leq \mathbb{V}\mathrm{ar}_{x_{i,j} \sim Q_\alpha}[\hat{\mu}]$.*

Proof. Let $\mu = \mathbb{E}_{x\sim Q_\alpha}[f(x)]$ and $\mu_j = \mathbb{E}_{x\sim Q_j}[f(x)]$. Then,

$$\begin{aligned}
\mathbb{V}\mathrm{ar}_{x_{i,j}\sim Q_\alpha}[\hat{\mu}] &= \frac{1}{n^2}\sum_{j=1}^{m} n_j \int q_j(x)(f(x)-\mu)^2 dx \\
&= \frac{1}{n^2}\sum_{j=1}^{m} n_j \int q_j(x)(f(x)-\mu\pm\mu_j)^2 dx \\
&= \frac{1}{n^2}\sum_{j=1}^{m} n_j \int q_j(x)(f(x)-\mu_j)^2 dx + \frac{1}{n^2}\sum_{j=1}^{m} n_j \int q_j(x)(\mu_j-\mu)^2 dx \\
&= \mathbb{V}\mathrm{ar}_{x_{i,j}\sim Q_j}[\hat{\mu}] + \frac{1}{n}\sum_{j=1}^{m}\alpha_j(\mu_j-\mu)^2 \\
&\geq Var_{x_{i,j}\sim Q_j}[\hat{\mu}].
\end{aligned}$$

□

APPENDIX $\mathcal{B}$

Proofs of Chapter 7

Proof of Lemma 7.4.1

From Hoeffding's inequality we have

$$\mathbb{P}\left\{\left|\mathbb{E}_{\nu,P_0}\left[\left\|\widetilde{B}_\omega\right\|_{\mathcal{D}}^2\right]-\left\|\widetilde{B}_\omega\right\|_{\mathcal{D}}^2\right|>\epsilon\right\}\leq 2\exp\left(-\frac{2n\epsilon^2}{\left(2\frac{r_{\max}}{1-\gamma}\right)^4}\right)$$

which implies that, for any $\delta>0$, with probability at least $1-\delta$:

$$\left|\mathbb{E}_{\nu,P_0}\left[\left\|\widetilde{B}_\omega\right\|_{\mathcal{D}}^2\right]-\left\|\widetilde{B}_\omega\right\|_{\mathcal{D}}^2\right|\leq 4\frac{r_{\max}^2}{(1-\gamma)^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}.$$

Under independence assumptions, the expected TD error can be re-written as

$$\begin{aligned}
\mathbb{E}_{\nu,P_0}\left[\left\|\widetilde{B}_\omega\right\|_D^2\right] &= \mathbb{E}_{\nu,P_0}\left[\frac{1}{n}\sum_{i=1}^n(r_i+\gamma\operatorname*{mm}_{a'}Q_\omega(s_i',a')-Q_\omega(s_i,a_i))^2\right]\\
&= \mathbb{E}_{\nu,P_0}\left[(r_0(s,a)+\gamma\operatorname*{mm}_{a'}Q_\omega(s',a')-Q_\omega(s,a))^2\right]\\
&= \mathbb{E}_{\nu}\left[\mathbb{E}_{P_0}\left[\widetilde{b}(\omega)^2\right]\right]\\
&= \mathbb{E}_{\nu}\left[\mathbb{V}\mathrm{ar}_{P_0}\left[\widetilde{b}(\omega)\right]+\mathbb{E}_{P_0}\left[\widetilde{b}(\omega)\right]^2\right]\\
&= v(\omega)+\left\|\widetilde{B}_\omega\right\|_\nu^2,
\end{aligned}$$

Thus,

$$\left| \left\| \widetilde{B}_\omega \right\|_\nu^2 + v(\omega) - \left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right| \leq 4 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}. \tag{B.1}$$

From the change of measure inequality (Seldin et al., 2012), we have that, for any measurable function $f(\omega)$ and any two probability measures p and q,

$$\log \mathbb{E}_p \left[e^{f(\omega)} \right] \geq \mathbb{E}_q \left[f(\omega) \right] - D_{\mathrm{KL}}(q \| p).$$

Thus, multiplying both sides of (B.1) by $\lambda^{-1} n$ and applying the change of measure inequality with $f(\omega) = \lambda^{-1} n \left| \left\| \widetilde{B}_\omega \right\|_\nu^2 + v(\omega) - \left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right|$, we obtain

$$\mathbb{E}_q \left[f(\omega) \right] - D_{\mathrm{KL}}(q \| p) \leq \log \mathbb{E}_p \left[e^{f(\omega)} \right] \leq 4 \frac{r_{\max}^2 \lambda^{-1} n}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}},$$

where the second inequality holds since the right-hand side of (B.1) does not depend on ω. Finally, we can explicitly write:

$$\mathbb{E}_q \left[\left| \left\| \widetilde{B}_\omega \right\|_\nu^2 + v(\omega) - \left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right| \right] \leq \frac{\lambda}{n} D_{\mathrm{KL}}(q \| p) + 4 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}$$

from which the lemma follows straightforwardly.

Proof of Lemma 7.4.2

Let us use Lemma 7.4.1 for the specific choice $q = q_{\widehat{\xi}}$. Using the first inequality in Lemma 7.4.1,

$$\begin{aligned}
\mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] &\leq \mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right] - \mathbb{E}_{q_{\widehat{\xi}}} \left[v(\omega) \right] + \frac{\lambda}{n} D_{\mathrm{KL}}(q_{\widehat{\xi}} \| p) + 4 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \\
&\leq \mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right] + \frac{\lambda}{n} D_{\mathrm{KL}}(q_{\widehat{\xi}} \| p) + 4 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \\
&= \inf_{\xi \in \Xi} \left\{ \mathbb{E}_{q_\xi} \left[\left\| \widetilde{B}_\omega \right\|_{\mathcal{D}}^2 \right] + \frac{\lambda}{n} D_{\mathrm{KL}}(q_\xi \| p) \right\} + 4 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}},
\end{aligned}$$

where the second inequality holds since $v(\omega) > 0$, while the equality holds from the definition of $\widehat{\xi}$. We can now use the second ineuquality in Lemma 7.4.1 to bound $\mathbb{E}_{q_\xi} \left[\left\| \widetilde{B}_\omega \right\|_D^2 \right]$, thus obtaining:

$$\mathbb{E}_{q_{\widehat{\xi}}} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] \leq \inf_{\xi \in \Xi} \left\{ \mathbb{E}_{q_\xi} \left[\left\| \widetilde{B}_\omega \right\|_\nu^2 \right] + \mathbb{E}_{q_\xi} \left[v(\omega) \right] + 2 \frac{\lambda}{n} D_{\mathrm{KL}}(q_\xi \| p) \right\} + 8 \frac{r_{\max}^2}{(1-\gamma)^2} \sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

This concludes the proof.

APPENDIX C

Proofs of Chapter 9

Proof of Lemma 9.3.1

Using the union bound, we can decompose $\mathbb{P}\{E^c\}$ into

$$\mathbb{P}\left\{\exists h = 1, \dots, \lceil \log_2 n \rceil, \exists a \in \mathcal{A} : |\hat{\mu}_{h-1}(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta}\sqrt{\frac{\alpha \log n}{N_{h-1}(a)}} \wedge N_{h-1}(a) > 0\right\}$$
$$\leq \sum_{h=1}^{\lceil \log_2 n \rceil} \sum_{a \in \mathcal{A}^*(\Theta)} \mathbb{P}\left\{|\hat{\mu}_{h-1}(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta}\sqrt{\frac{\alpha \log n}{N_{h-1}(a)}} \wedge N_{h-1}(a) > 0\right\},$$

where the sum starts from $h = 1$ since in phase 0 no arm has been pulled and all models are therefore contained in the confidence set. Furthermore, $\mathcal{A}$ can be replaced by $\mathcal{A}^*(\Theta)$ since arms that are sub-optimal for all models are never pulled and so the corresponding event above never holds. Let us now consider the inner term for a fixed phase h and arm a. Notice that, at the end of phase $h-1$, the possible number of pulls of arm a are

$$k_s := \left\lceil \frac{\alpha \log n}{\tilde{\Gamma}_s^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil$$

for $s = 0, 1, \dots, h-1$. Thus, by taking a further union bound on the possible values of $N_{h-1}(a)$ and using Chernoff-Hoeffding inequality, we obtain

$$\begin{aligned}
&\mathbb{P}\left\{|\hat{\mu}_{h-1}(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta}\sqrt{\frac{\alpha \log n}{N_{h-1}(a)}}\right\} \\
&= \mathbb{P}\left\{\bigcup_{s=0}^{h-1} |\hat{\mu}_{h-1}(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta}\sqrt{\frac{\alpha \log n}{N_{h-1}(a)}} \wedge N_{h-1}(a) = k_s\right\} \\
&\leq \sum_{s=0}^{h-1} \mathbb{P}\left\{|\hat{\mu}_{i,k_s} - \mu_{\theta^*}(a)| \geq \frac{1}{\beta}\sqrt{\frac{\alpha \log n}{k_s}}\right\} \\
&\leq \sum_{s=0}^{h-1} 2e^{-2k_s \frac{\alpha \log n}{\beta^2 k_s}} = 2hn^{-2\frac{\alpha}{\beta^2}}.
\end{aligned}$$

Notice that, with some abuse of notation, we define $\hat{\mu}_{a,k_s}$ as the empirical mean of arm a after k_s pulls of such arm. Putting everything together,

$$\mathbb{P}\{E^c\} \leq \sum_{h=1}^{\lceil \log_2 n \rceil} \sum_{a \in \mathcal{A}^*(\Theta)} 2hn^{-2\frac{\alpha}{\beta^2}} = 2|\mathcal{A}^*(\Theta)|n^{-2\frac{\alpha}{\beta^2}} \sum_{h=1}^{\lceil \log_2 n \rceil} h \leq |\mathcal{A}^*(\Theta)|n^{-2\frac{\alpha}{\beta^2}} (\log_2 n + 2)^2,$$

which concludes the proof.

Proof of Proposition 9.3.1 and Proposition 9.3.2

Proposition 9.3.1. First notice that each sub-optimal arm a is also in the set of arms $\mathcal{A}_a^*$ available to remove a itself. Consider now any model $\bar{\theta}_a \in \Theta_a^*$ that must be removed from the confidence set in order to eliminate a. We have two cases.

1) $\bar{\theta}_a$ is an optimistic model w.r.t. θ^* This implies that $\mu^*_{\bar{\theta}_a} = \mu_{\bar{\theta}_a}(a) > \mu^*_{\theta^*}$ which, in turns, implies that $\Gamma_a(\bar{\theta}_a, \theta^*) > \Delta_{\theta^*}(a)$. Therefore, the regret for such arms can be upper bounded by

$$\frac{c\Delta_{\theta^*}(a) \log n}{\max_{a' \in \underline{\mathcal{A}}_{\bar{h}_a} \cup \{a\}} \Gamma^2_{a'}(\bar{\theta}_a, \theta^*)} + c' \leq \frac{c\Delta_{\theta^*}(a) \log n}{\Gamma^2_a(\bar{\theta}_a, \theta^*)} + c' \leq \frac{c \log n}{\Delta_{\theta^*}(a)} + c'.$$

2) $\bar{\theta}_a$ is not an optimistic model w.r.t. θ^* This implies that $\mu^*_{\bar{\theta}_a} = \mu_{\bar{\theta}_a}(a) \leq \mu^*_{\theta^*}$. If $\mu_{\bar{\theta}_a}(a) \geq \mu^*_{\theta^*} - \frac{\Delta_{\theta^*}(a)}{2}$, then $\Gamma_a(\bar{\theta}_a, \theta^*) \geq \frac{\Delta_{\theta^*}(a)}{2}$. If, on the other hand, $\mu_{\bar{\theta}_a}(a) \leq \mu^*_{\theta^*} - \frac{\Delta_{\theta^*}(a)}{2}$, then $\Gamma_{a^*}(\bar{\theta}_a, \theta^*) \geq \frac{\Delta_{\theta^*}(a)}{2}$ since $\mu_{\bar{\theta}_a}(a^*) < \mu_{\bar{\theta}_a}(a)$. Furthermore, under event E, $a^* \in \tilde{\mathcal{A}}_h$ for all $h \geq 0$ (and thus $a^* \in \underline{\mathcal{A}}_h$). Therefore,

$$\frac{c\Delta_{\theta^*}(a) \log n}{\max_{a' \in \underline{\mathcal{A}}_{\bar{h}_a} \cup \{a\}} \Gamma^2_{a'}(\bar{\theta}_a, \theta^*)} + c' \leq \frac{c\Delta_{\theta^*}(a) \log n}{\max\left\{\Gamma^2_a(\bar{\theta}_a, \theta^*), \Gamma^2_{a^*}(\bar{\theta}_a, \theta^*)\right\}} + c' \leq \frac{2c \log n}{\Delta_{\theta^*}(a)} + c'.$$

This concludes the proof of Proposition 9.3.1.

Proposition 9.3.2. In the proof of Proposition 9.3.1, we have already shown that the model gaps with respect to optimistic models are always larger than the action gaps. Therefore,

$$\inf_{\theta \in \Theta_a^* \setminus \Theta_a^+} \max_{a' \in \mathcal{A}_a^*} \Gamma_{a'}(\theta, \theta^*) \geq \inf_{\theta \in \Theta_a^+} \max_{a' \in \mathcal{A}_a^*} \Gamma_{a'}(\theta, \theta^*) \geq \Delta_{\theta^*}(a).$$

The proof follows straightforwardly.

Proofs of Section 9.4

Throughout this section, we override the notation of the previous results to account for the periods introduced in Algorithm 7. We use $N_{k,h}(a)$ to denote the number of pulls of arm a at the end of phase h in period k. Furthermore, we define $N_k(a)$ as the number of pulls of a at the end of period k. Similarly, $T_{k,h}(a)$ denotes the number of pulls of a at end of phase h but counting only those pulls occurred in period k. For all other period- and phase-dependent random variables, we shall use a superscript k to denote the period and a subscript h to denote the phase. For variables depending only on the period, we shall move k to a subscript. We will make these dependencies explicit whenever not clear from the context.

Proof of Theorem 9.4.1

We first extend Lemma 9.3.1 to bound the probability that the true model is not contained in the confidence set by a margin in some phase of period k.

Lemma C.3.1. *Let $\alpha > 0$, $\beta \geq 1$, $k \geq 0$, and E_k denote the following event:*

$$E_k := \left\{ \forall h = 0, \dots, \lceil \log_2 \tilde{n}_k \rceil : \theta^* \in \tilde{\Theta}_h^k \right\}. \tag{C.1}$$

Then, the probability that E_k does not hold can be upper bounded by

$$\mathbb{P}\{E_k^c\} \leq |\mathcal{A}^*(\Theta)| (\log_2 \tilde{n}_k + 3)^2 \tilde{n}_k^{-2\frac{\alpha}{\beta^2}} \sum_{k'=0}^{k-1} \tilde{n}_{k'}.$$

Proof. First assume that $k > 0$. Using the union bound, we have that $\mathbb{P}\{E_k^c\}$ can be written as

$$\mathbb{P}\left\{ \exists h = 0, \dots, \lceil \log_2 \tilde{n}_k \rceil, \exists i \in \mathcal{A} : |\hat{\mu}_{h-1}^k(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta} \sqrt{\frac{\alpha \log \tilde{n}_k}{N_{k,h-1}(a)}} \wedge N_{k,h-1}(a) > 0 \right\}$$
$$\leq \sum_{h=0}^{\lceil \log_2 \tilde{n}_k \rceil} \sum_{a \in \mathcal{A}^*(\Theta)} \mathbb{P}\left\{ |\hat{\mu}_{h-1}^k(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta} \sqrt{\frac{\alpha \log \tilde{n}_k}{N_{k,h-1}(a)}} \wedge N_{k,h-1}(a) > 0 \right\},$$

where $\mathcal{A}$ can be replaced by $\mathcal{A}^*(\Theta)$ since arms that are sub-optimal for all models are never pulled and so the corresponding event above never holds. Let us now consider the inner term for a fixed phase h and arm a. The number of pulls of a can be decomposed into $N_{k,h-1}(a) = N_{k-1}(a) + T_{k,h-1}(a)$. $N_{k-1}(a)$ could be any value s between 1 and $\bar{s}_k := \sum_{k'=0}^{k-1} \tilde{n}_{k'}$. On the other hand, $T_{k,h-1}(a)$ can lead only to $h+1$ different number of pulls,

$$p_u := \left\lceil \frac{\alpha \log \tilde{n}_k}{\tilde{\Gamma}_{u-1}^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil$$

for $u = 1, \dots, h$ and $p_u = 0$ for $u = 0$. Therefore, the number of pulls of a given s pulls up to period $k-1$ and p_u pulls in period k are $q_{s,u} = \max\{s, p_u\}$. Thus, by taking a further union bound on the possible values of $N_{k,h-1}(a)$ and using Chernoff-Hoeffding inequality, we obtain

$$\begin{aligned}
\mathbb{P}\left\{ |\hat{\mu}_{h-1}^k(a) - \mu_{\theta^*}(a)| \geq \frac{1}{\beta} \sqrt{\frac{\alpha \log \tilde{n}_k}{N_{k,h-1}(a)}} \right\} &= \mathbb{P}\left\{ \bigcup_{s=1}^{\bar{s}_k} \bigcup_{u=0}^{h} |\hat{\mu}_{a,q_{s,u}} - \mu_{\theta^*}(a)| \geq \frac{1}{\beta} \sqrt{\frac{\alpha \log \tilde{n}_k}{q_{s,u}}} \right\} \\
&\leq \sum_{s=1}^{\bar{s}_k} \sum_{u=0}^{h} \mathbb{P}\left\{ |\hat{\mu}_{a,q_{s,u}} - \mu_{\theta^*}(a)| \geq \frac{1}{\beta} \sqrt{\frac{\alpha \log \tilde{n}_k}{q_{s,u}}} \right\} \\
&\leq \sum_{s=1}^{\bar{s}_k} \sum_{u=0}^{h} 2e^{-2q_{s,u} \frac{\alpha \log \tilde{n}_k}{\beta^2 q_{s,u}}} = 2(h+1) \tilde{n}_k^{-2\frac{\alpha}{\beta^2}} \bar{s}_k.
\end{aligned}$$

Notice that, with some abuse of notation, we define $\hat{\mu}_{a,s}$ as the empirical mean of arm a after s pulls of such arm. Putting everything together,

$$\begin{aligned}\mathbb{P}\left\{E_k^c\right\} &\leq \sum_{h=0}^{\lceil \log_2 \tilde{n}_k \rceil} \sum_{a \in \mathcal{A}^*(\Theta)} 2(h+1)\tilde{n}_k^{-2\frac{\alpha}{\beta^2}} \bar{s}_k = 2|\mathcal{A}^*(\Theta)|\tilde{n}_k^{-2\frac{\alpha}{\beta^2}} \bar{s}_k \sum_{h=0}^{\lceil \log_2 \tilde{n}_k \rceil} (h+1) \\ &\leq |\mathcal{A}^*(\Theta)|(\log_2 \tilde{n}_k + 3)^2 \tilde{n}_k^{-2\frac{\alpha}{\beta^2}} \bar{s}_k.\end{aligned}$$

Notice that for $k = 0$ the bound is even smaller since we can avoid the union bound over the pulls in previous periods. This concludes the proof. □

Proof of Theorem 9.4.1. Let $L_k := \sum_{t=\bar{s}_k+1}^{\tilde{n}_k} \Delta_{\theta^*}(A_t)$, with $\bar{s}_k := \sum_{k'=0}^{k-1} \tilde{n}_{k'}$, be the regret incurred in period k. Then,

$$\begin{aligned}R_n = \mathbb{E}\left[\sum_{t=1}^{n} \Delta_{\theta^*}(A_t)\right] &\overset{(a)}{\leq} \mathbb{E}\left[\sum_{k=0}^{\bar{k}} L_k\right] = \mathbb{E}\left[\sum_{k=0}^{\bar{k}} L_k \mathbb{1}\left\{E_k = 1\right\}\right] + \mathbb{E}\left[\sum_{k=0}^{\bar{k}} L_k \mathbb{1}\left\{E_k = 0\right\}\right] \\ &\overset{(b)}{\leq} \underbrace{\sum_{k=0}^{\bar{k}} \mathbb{E}\left[L_k | E_k = 1\right]}_{(i)} + \underbrace{\sum_{k=0}^{\bar{k}} \mathbb{P}\left\{E_k = 0\right\} \tilde{n}_k}_{(ii)},\end{aligned}$$

where (a) follows from the definition of the maximum period $\bar{k} = \min_{k \in \mathbb{N}^+}\{k | \tilde{n}_k \geq n\}$ and (b) by bounding the regret of each period by $\tilde{n}_k$. We now bound the two terms separately.

Term (i). Fix a period k. We have

$$\mathbb{E}\left[L_k | E_k = 1\right] = \sum_{a \in \mathcal{A}^*(\Theta)} \Delta_{\theta^*}(a) \mathbb{E}\left[N_k(a) - N_{k-1}(a) | E_k = 1\right],$$

where we recall $N_k(a)$ is the total number of pulls of a at the end of period k (not necessarily only in period k), so that $N_k(a) - N_{k-1}(a)$ is the total number of pulls occurred in period k. Fix a sub-optimal arm a. Let

$$\bar{h}_a := \min_{h \in \mathbb{N}^+} \left\{ h \mid \tilde{\Gamma}_h \leq \inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma_{a'}(\theta, \theta^*) \right\}.$$

Lemma 9.3.3, together with the fact that a^* is pulled in all phases, ensures that if $a \in \tilde{\mathcal{A}}_{\bar{h}_a}^k$, a will not be pulled again in period k. Therefore,

$$\begin{aligned}N_k(a) - N_{k-1}(a) &\overset{(a)}{\leq} \left\lceil \frac{\alpha \log \tilde{n}_k}{\tilde{\Gamma}_{\bar{h}_a}^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \overset{(b)}{=} \left\lceil \frac{4\alpha \log n}{\tilde{\Gamma}_{\bar{h}_a - 1}^2} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \\ &\overset{(c)}{\leq} \left\lceil \frac{4\alpha \log \tilde{n}_k}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma_{a'}^2(\theta, \theta^*)} \left(1 + \frac{1}{\beta}\right)^2 \right\rceil \\ &\overset{(d)}{\leq} \frac{16\alpha \log \tilde{n}_k}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma_{a'}^2(\theta, \theta^*)} + 1 \\ &\overset{(e)}{\leq} \frac{24\alpha \log \tilde{n}_k}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma_{a'}^2(\theta, \theta^*)},\end{aligned}$$

where (a) follows from the previous comments, (b) from $\tilde{\Gamma}_h = \frac{\tilde{\Gamma}_{h-1}}{2}$, (c) from the definition of $\bar{h}_a$, (d) after setting $\beta = 1$, and (e) by noticing that $1 \leq \frac{3}{2}\log \tilde{n}_k$ for all $k \geq 0$. This allows us to bound the expected regret due to arms in $\mathcal{A}^*(\Theta)$ by

$$(i) \leq \sum_{a \in \mathcal{A}^*(\Theta)} \frac{24\alpha\Delta_{\theta^*}(a)\sum_{k=0}^{\bar{k}} \log \tilde{n}_k}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma^2_{a'}(\theta, \theta^*)} \leq \sum_{a \in \mathcal{A}^*(\Theta)} \frac{96\alpha\Delta_{\theta^*}(a)\log n}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma^2_{a'}(\theta, \theta^*)}.$$

To understand the second inequality, notice that $\tilde{n}_k = 2^{2^k}$ for all $k \geq 0$ since $\eta = 1$. Furthermore, since $\bar{k} < \log_2 \log_2 n + 1$, $\sum_{k=0}^{\bar{k}} \log \tilde{n}_k = (\log 2)\sum_{k=0}^{\bar{k}} 2^k \leq 2^{\bar{k}+1}\log 2 \leq 4 \log n$.

Term (ii). We have

$$\begin{aligned}
(ii) &\overset{(a)}{\leq} |\mathcal{A}^*(\Theta)| \sum_{k=0}^{\bar{k}} \tilde{n}_k^{2-2\frac{\alpha}{\beta^2}} (\log_2 \tilde{n}_k + 3)^2 \overset{(b)}{=} |\mathcal{A}^*(\Theta)| \sum_{k=0}^{\bar{k}} 2^{2^k(2-2\frac{\alpha}{\beta^2})}(2^k+3)^2 \\
&\overset{(c)}{\leq} |\mathcal{A}^*(\Theta)| \sum_{k=0}^{2} 2^{2^k(2-2\frac{\alpha}{\beta^2})}(2^k+3)^2 + |\mathcal{A}^*(\Theta)| \sum_{k=3}^{\infty} \frac{1}{2^{2^k(2\frac{\alpha}{\beta^2}-3)}} \\
&\overset{(d)}{\leq} 5.76|\mathcal{A}^*(\Theta)| + 0.026|\mathcal{A}^*(\Theta)| \leq 6|\mathcal{A}^*(\Theta)|,
\end{aligned}$$

where (a) follows from Lemma C.3.1 and $\sum_{k'=0}^{k-1} \tilde{n}_{k'} \leq \tilde{n}_k$, (b) from the definition of $\tilde{n}_k$, (c) from the fact that for $k \geq 3$ we have $(2^k+3)^2 \leq 2^{2^k}$, and (d) after setting $\alpha = 2$, $\beta = 1$, and some numerical calculations.

Combining (i) and (ii), we obtain the stated bound on R_n.

Proof of Theorem 9.4.2

As for Theorem 9.4.1, we define $L_k := \sum_{t=\bar{s}_k+1}^{\tilde{n}_k} \Delta_{\theta^*}(A_t)$ to be the regret incurred in period k. Similarly to Lattimore and Munos (2014), we decompose the expected regret into that incurred up to a fixed (constant in n) period $\underline{k}$ and that incurred in the remaining periods. Let $O_k := \{\exists a \neq a^* : a \in \tilde{\mathcal{A}}_0^k\}$ be the event under which some sub-optimal arm is pulled in period k. Then,

$$\begin{aligned}
R_n = \mathbb{E}\left[\sum_{t=1}^{n} \Delta_{\theta^*}(A_t)\right] &\overset{(a)}{\leq} \mathbb{E}\left[\sum_{k=0}^{\bar{k}} L_k\right] \overset{(b)}{\leq} \sum_{k=0}^{\bar{k}} \mathbb{E}\left[L_k | E_k = 1\right] + \sum_{k=0}^{\bar{k}} \mathbb{P}\left\{E_k = 0\right\}\tilde{n}_k \\
&\overset{(c)}{=} \sum_{k=0}^{\underline{k}} \mathbb{E}\left[L_k | E_k = 1\right] + \sum_{k=\underline{k}+1}^{\bar{k}} \mathbb{E}\left[L_k | E_k = 1\right] + \sum_{k=0}^{\bar{k}} \mathbb{P}\left\{E_k = 0\right\}\tilde{n}_k \\
&\overset{(d)}{\leq} \underbrace{\sum_{k=0}^{\underline{k}} \mathbb{E}\left[L_k | E_k = 1\right]}_{(i)} + \underbrace{\sum_{k=\underline{k}+1}^{\bar{k}} \tilde{n}_k \mathbb{P}\left\{O_k = 1 | E_k = 1\right\}}_{(ii)} + \underbrace{\sum_{k=0}^{\bar{k}} \mathbb{P}\left\{E_k = 0\right\}\tilde{n}_k}_{(iii)},
\end{aligned}$$

where (a) and (b) are as in the proof of Theorem 9.4.1, (c) is trivial, and (d) follows since if $O_k = 0$ then only the optimal arm is pulled in period k and thus no regret is incurred.

Using exactly the same argument as done in the proof of Theorem 9.4.1,

$$(i) \leq \sum_{a \in \mathcal{A}^*(\Theta)} \frac{24\alpha\Delta_{\theta^*}(a)\sum_{k=0}^{\underline{k}} \log \tilde{n}_k}{\inf_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma^2_{a'}(\theta, \theta^*)}.$$

Similarly, we obtain $(iii) \leq 3|\mathcal{A}^*(\Theta)|$, where the smaller constant is due to the fact that we increased α.

Let us now deal with (ii). First, we define $\underline{k}$ as

$$\underline{k} := \min_{k \in \mathbb{N}^+} \left\{ k \;\middle|\; \left\lfloor \frac{\tilde{n}_k}{|\mathcal{A}^*(\Theta)|} \right\rfloor \geq \frac{10 \log \tilde{n}_{k+1}}{\Gamma_*^2} \right\}.$$

By the union bound,

$$\begin{aligned}(ii) &\leq \sum_{k=\underline{k}+1}^{\bar{k}} \tilde{n}_k \mathbb{P}\left\{O_k = 1 \wedge E_{k-1} = 1 | E_k = 1\right\} + \sum_{k=\underline{k}+1}^{\bar{k}} \tilde{n}_k \mathbb{P}\left\{O_k = 1 \wedge E_{k-1} = 0 | E_k = 1\right\} \\ &\leq \underbrace{\sum_{k=\underline{k}+1}^{\bar{k}} \tilde{n}_k \mathbb{P}\left\{O_k = 1 | E_k = 1 \wedge E_{k-1} = 1\right\}}_{(iv)} + \underbrace{\sum_{k=\underline{k}+1}^{\bar{k}} \tilde{n}_k \mathbb{P}\left\{E_{k-1} = 0 | E_k = 1\right\}}_{(v)}.\end{aligned}$$

By recalling that $\tilde{n}_k = \tilde{n}_{k-1}^2$ and that α was increased to $\frac{5}{2}$, (v) can be bounded by $6|\mathcal{A}^*(\Theta)|$ as done for (iii) in Theorem 9.4.1. It only remains to bound (iv). Fix a period $k \geq \underline{k} + 1$. We have

$$\begin{aligned}\mathbb{P}\left\{O_k = 1 | E_k = 1 \wedge E_{k-1} = 1\right\} &= \mathbb{P}\left\{\exists a \neq a^* : a \in \tilde{\mathcal{A}}_0^k | E_k = 1 \wedge E_{k-1} = 1\right\} \\ &\overset{(a)}{=} \mathbb{P}\left\{\exists a \neq a^* : a \in \mathcal{A}^*(\tilde{\Theta}_0^k) | E_k = 1 \wedge E_{k-1} = 1\right\} \\ &\overset{(b)}{\leq} \mathbb{P}\left\{ N_{k-1}(a^*) < \frac{\alpha \log \tilde{n}_k}{\Gamma_*^2} \left(1 + \frac{1}{\beta}\right)^2 | E_k = 1 \wedge E_{k-1} = 1 \right\} \\ &\overset{(c)}{\leq} \mathbb{P}\left\{ N_{k-1}(a^*) < \left\lfloor \frac{\tilde{n}_{k-1}}{|\mathcal{A}^*(\Theta)|} \right\rfloor | E_k = 1 \wedge E_{k-1} = 1 \right\} \overset{(d)}{\leq} 0,\end{aligned}$$

where (a) follows from the definition of $\tilde{\mathcal{A}}_0^k$. In (b) we exploit the fact that, under event E_k, if a^* is pulled more than that quantity at the end of period $k-1$ then no model with a different optimal arm than a^* belongs to $\tilde{\Theta}_0^k$. (c) is from the definition of $\underline{k}$ and $k - 1 \geq \underline{k}$. (d) holds since, under E_{k-1}, a^* is pulled in all phases in period $k-1$. Therefore, even if all other arms are pulled as well, the round robin schedule of the pulls ensures $N_{k-1}(a^*) \geq \left\lfloor \frac{\tilde{n}_{k-1}}{|\mathcal{A}^*(\Theta)|} \right\rfloor$.

Therefore, $(ii) \leq 6|\mathcal{A}^*(\Theta)|$. Combining (i), (ii), and (iii) we obtain

$$R_n \leq \sum_{a \in \mathcal{A}^*(\Theta)} \frac{24\alpha\Delta_{\theta^*}(a) \sum_{k=0}^{\underline{k}} \log \tilde{n}_k}{\min_{\theta \in \Theta_a^*} \max_{a' \in \{a, a^*\}} \Gamma_{a'}^2(\theta, \theta^*)} + 9|\mathcal{A}^*(\Theta)|.$$

Since $\sum_{k=0}^{\underline{k}} \log \tilde{n}_k = \log 2 \sum_{k=0}^{\underline{k}} 2^k \leq 2^{\underline{k}+1} \log 2$, let us finally bound $\underline{k}$. From its definition,

$$\left\lfloor \frac{2^{2^{\underline{k}-1}}}{|\mathcal{A}^*(\Theta)|} \right\rfloor < \frac{20 \log 2^{2^{\underline{k}-1}}}{\Gamma_*^2} \quad \Longrightarrow \quad \frac{2^{2^{\underline{k}-1}}}{2^{\underline{k}-1}} \leq \frac{20|\mathcal{A}^*(\Theta)| \log 2}{\Gamma_*^2} + 2|\mathcal{A}^*(\Theta)|.$$

Since $\underline{k} - 1 \leq 2^{\underline{k}-2}$, we obtain

$$\underline{k} \leq \log_2 \log_2 \left(\frac{20|\mathcal{A}^*(\Theta)| \log 2}{\Gamma_*^2} + 2|\mathcal{A}^*(\Theta)| \right) + 2.$$

Therefore,

$$\sum_{k=0}^{\underline{k}} \log \tilde{n}_k \leq 2^{\underline{k}+1} \log 2 \leq 8 \log \left(\frac{20|\mathcal{A}^*(\Theta)| \log 2}{\Gamma_*^2} + 2|\mathcal{A}^*(\Theta)| \right),$$

which concludes the proof.

APPENDIX $\mathcal{D}$

Proofs of Chapter 10

Proof of Lemma 10.3.1

Feasibility of (P_z)**.** We start from the first result in Lem. 10.3.1, which states the minimal value of z for which (P_z) is feasible. Clearly, the maximal value that the left-hand side of the KL constraint can assume is

$$\max_{\omega\in\Omega}\inf_{\theta'\in\Theta_{\mathrm{alt}}}\mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}}\omega(x,a)d_{x,a}(\theta^\star,\theta')\right],$$

which can also be interpreted as the solution to the associated pure-exploration (or best-arm identification) problem (e.g., Degenne et al., 2019). Therefore,

$$\begin{aligned}\underline{z}(\theta^\star) &:= \min\{z>0 : (\mathrm{P}_z) \text{ is feasible}\}\\ &= \min\left\{z>0 : \max_{\omega\in\Omega}\inf_{\theta'\in\Theta_{\mathrm{alt}}}\mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}}\omega(x,a)d_{x,a}(\theta^\star,\theta')\right]\geq\frac{1}{z}\right\}\\ &= \frac{1}{\max_{\omega\in\Omega}\inf_{\theta'\in\Theta_{\mathrm{alt}}}\mathbb{E}_\rho\left[\sum_{a\in\mathcal{A}}\omega(x,a)d_{x,a}(\theta^\star,\theta')\right]}.\end{aligned}$$

This proves the first statement in Lem. 10.3.1.

Connection between (P) and (P_z). In order to prove the second result, let us rewrite (P_z) in the following more convenient form:

$$\begin{aligned} \underset{\eta(x,a)\geq 0}{\text{minimize}} \quad & \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \eta(x,a)\Delta_{\theta^\star}(x,a) \\ \text{subject to} \quad & \inf_{\theta'\in\Theta_{\text{alt}}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \eta(x,a) d_{x,a}(\theta^\star,\theta') \geq 1, \\ & \sum_{a\in\mathcal{A}} \eta(x,a) = z \quad \forall x\in\mathcal{X}. \end{aligned} \tag{P'_z}$$

Note that (P'_z) is obtained from (P_z) in the main paper by performing the change of variables $\eta(x,a) = z\omega(x,a)$, hence the two problems are equivalent. Recall that $v^\star(\theta^\star)$ is the optimal value of (P) and $u^\star(z,\theta^\star)$ is the optimal value of (P'_z) and (P_z) (if there exists one). We are interested in bounding the deviation between $u^\star(z,\theta^\star)$ and $v^\star(\theta^\star)$ as a function of z.

Let us first define the following set of *confusing* models:

$$\widetilde{\Theta}_{alt} := \left\{\theta' \in \Theta_{alt} : \forall x\in\mathcal{X}, \mu^\star_{\theta^\star}(x) = \mu_{\theta'}(x,a^\star_x)\right\},$$

where, for the sake of readability, we abbreviate $a^\star_x = a^\star_{\theta^\star}(x)$. These models are indistinguishable from $\theta^\star$ by pulling only optimal arms. The following proposition, which was proved in (Degenne et al., 2020b), connects models in the alternative set Θ_{alt} with the confusing ones in $\widetilde{\Theta}_{\text{alt}}$.

Proposition D.1.1 ((Degenne et al., 2020b)). *There exists a constant $c_\Theta > 0$ such that, for all $\theta' \in \Theta_{\text{alt}}$, there exists $\theta'' \in \widetilde{\Theta}_{\text{alt}}$ such that,*

$$\forall x\in\mathcal{X}, a\in\mathcal{A} \qquad |\mu_{\theta'}(x,a) - \mu_{\theta''}(x,a)| \leq c_\Theta |\mu^\star_{\theta^\star}(x) - \mu_{\theta'}(x, a^\star_{\theta^\star}(x))|.$$

We now prove the bound on $u^\star(z,\theta)$ reported in Lem. 10.3.1.

Second result of Lemma 10.3.1. We start from the Lagrangian version of (P'_z).

$$\min_{\eta\geq 0} \left\{ \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \eta(x,a)\Delta_{\theta^\star}(x,a) + \lambda^\star(z,\theta^\star) \left(1 - \inf_{\theta'\in\Theta_{\text{alt}}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \eta(x,a) d_{x,a}(\theta^\star,\theta')\right)\right\},$$

subject to $\sum_{a\in\mathcal{A}} \eta(x,a) = z$ for each context $x\in\mathcal{X}$. The optimal value of this problem is $u^\star(z,\theta)$. Here $\lambda^\star(z,\theta^\star)$ is the optimal value of the Lagrange multiplier for the same problem. We distinguish two cases.

Case 1: $z < \max_{x\in\mathcal{X}} \frac{1}{\rho(x)} \sum_{a\neq a^\star_{\theta^\star}(x)} \eta^\star(x,a)$. Let

$$\overline{\eta}(x,a) = z \cdot \begin{cases} \frac{\eta^\star(x,a)/\rho(x)}{\max_{x\in\mathcal{X}} \frac{1}{\rho(x)} \sum_{a\neq a^\star_{\theta^\star}(x)} \eta^\star(x,a)} & \text{if } a \neq a^\star_{\theta^\star}(x) \\ 1 - \frac{\sum_{a\neq a^\star_{\theta^\star}(x)} \eta^\star(x,a)/\rho(x)}{\max_{x\in\mathcal{X}} \frac{1}{\rho(x)} \sum_{a\neq a^\star_{\theta^\star}(x)} \eta^\star(x,a)} & \text{otherwise} \end{cases}$$

where $\eta^\star$ is the optimal solution of (P). Since $\sum_a \overline{\eta}(x,a) = z$, we have that $u^\star(z,\theta^\star)$ is less or equal to the value of the Lagrangian for $\eta = \overline{\eta}$, i.e.,

$$u^\star(z,\theta^\star) \leq v^\star(\theta^\star) + \lambda^\star(z,\theta^\star) \left(1 - \inf_{\theta'\in\Theta_{\text{alt}}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \overline{\eta}(x,a) d_{x,a}(\theta^\star,\theta')\right),$$

where we used the fact that

$$\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)\Delta_{\theta^\star}(x,a)=\underbrace{\frac{z}{\max_{x\in\mathcal{X}}\frac{1}{\rho(x)}\sum_{a\neq a^\star_{\theta^\star}(x)}\eta^\star(x,a)}}_{<1}\underbrace{\sum_{x\in\mathcal{X}}\sum_{a\neq a^\star_{\theta^\star}(x)}\eta^\star(x,a)\Delta_{\theta^\star}(x,a)}_{=v^\star(\theta^\star)}.$$

since $\Delta_{\theta^\star}(x,a^\star_{\theta^\star}(x))=0$. Since the KL divergence $d_{x,a}(\theta^\star,\theta')$ is lower-bounded by zero, in case 1 we have

$$u^\star(z,\theta^\star)\leq v^\star(\theta^\star)+\lambda^\star(z,\theta^\star).$$

Case 2: $z\geq\max_{x\in\mathcal{X}}\frac{1}{\rho(x)}\sum_{a\neq a^\star_{\theta^\star}(x)}\eta^\star(x,a)$**.** Let

$$\overline{\eta}(x,a)=\begin{cases}\eta^\star(x,a)/\rho(x) & \text{if } a\neq a^\star_{\theta^\star}(x)\\ z-\sum_{a\neq a^\star_{\theta^\star}(x)}\eta^\star(x,a)/\rho(x) & \text{otherwise}\end{cases}$$

where, as before, $\eta^\star$ is the optimal solution of (P). Since $z\geq\sum_{a\neq a^\star_{\theta^\star}(x)}\eta^\star(x,a)/\rho(x)$ for any $x\in\mathcal{X}$, $\overline{\eta}$ is well defined. Since $\overline{\eta}$ also sums to z for each context, we have that $u^\star(z,\theta)$ is less or equal to the value of the Lagrangian for $\eta=\overline{\eta}$, i.e.,

$$u^\star(z,\theta^\star)\leq v^\star(\theta^\star)+\lambda^\star(z,\theta^\star)\left(1-\inf_{\theta'\in\Theta_{\text{alt}}}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)d_{x,a}(\theta^\star,\theta')\right).$$

We first lower bound the infimum on the right hand side. We have

$$\inf_{\theta'\in\Theta_{\text{alt}}}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)d_{x,a}(\theta^\star,\theta')=\min\Bigg\{\underbrace{\inf_{\theta'\in\widetilde{\Theta}_{\text{alt}}}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)d_{x,a}(\theta^\star,\theta')}_{I_{\widetilde{\Theta}_{\text{alt}}}},\tag{D.1}$$

$$\underbrace{\inf_{\theta'\in\Theta_{\text{alt}}\setminus\widetilde{\Theta}_{\text{alt}}}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)d_{x,a}(\theta^\star,\theta')}_{I_{\Theta_{\text{alt}}\setminus\widetilde{\Theta}_{\text{alt}}}}\Bigg\}.\tag{D.2}$$

By definition of $\overline{\eta}$ and $\eta^\star$, the infimum over the set of confusing models can be written as

$$I_{\widetilde{\Theta}_{\text{alt}}}=\inf_{\theta'\in\widetilde{\Theta}_{\text{alt}}}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\overline{\eta}(x,a)d_{x,a}(\theta^\star,\theta')=\inf_{\theta'\in\widetilde{\Theta}_{\text{alt}}}\sum_{x\in\mathcal{X}}\sum_{a\neq a^\star_x}\eta^\star(x,a)d_{x,a}(\theta^\star,\theta')\geq 1,\tag{D.3}$$

where the equality holds since the KLs are zero in the optimal arms, which are the only arms where the values of $\overline{\eta}$ differ from those of $\eta^\star$, and the inequality holds since $\eta^\star$ is feasible. Regarding the infimum over the non-confusing models,

$$I_{\Theta_{\text{alt}}\setminus\widetilde{\Theta}_{\text{alt}}}=\inf_{\theta'\in\Theta_{\text{alt}}\setminus\widetilde{\Theta}_{\text{alt}}}\left(\underbrace{\sum_{x\in\mathcal{X}}\rho(x)\overline{\eta}(x,a^\star_x)d_{x,a^\star_x}(\theta^\star,\theta')}_{(i)}+\underbrace{\sum_{x\in\mathcal{X}}\sum_{a\neq a^\star_x}\eta^\star(x,a)d_{x,a}(\theta^\star,\theta')}_{(ii)}\right).\tag{D.4}$$

We partition the set of non-confusing models in two subsets:

$$\widetilde{\Theta}_{\text{alt}}^{(1)} := \Big\{\theta' \in \Theta_{\text{alt}} \setminus \widetilde{\Theta}_{\text{alt}} : \forall x \in \mathcal{X}, |\mu_{\theta^\star}^\star(x) - \mu_{\theta'}(x, a_{\theta^\star}^\star(x))| < \epsilon_z\Big\}, \tag{D.5}$$

$$\widetilde{\Theta}_{\text{alt}}^{(2)} := \Big\{\theta' \in \Theta_{\text{alt}} \setminus \widetilde{\Theta}_{\text{alt}} : \exists x \in \mathcal{X}, |\mu_{\theta^\star}^\star(x) - \mu_{\theta'}(x, a_{\theta^\star}^\star(x))| \geq \epsilon_z\Big\}. \tag{D.6}$$

The value of ϵ_z will be specified later. We have, for $\theta'' \in \widetilde{\Theta}_{\text{alt}}$,

$$\inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(1)}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \overline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') \overset{(a)}{\geq} \inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(1)}} \sum_{x\in\mathcal{X}} \sum_{a\neq a_x^\star} \eta^\star(x,a) d_{x,a}(\theta^\star, \theta') \tag{D.7}$$

$$\overset{(b)}{\geq} \inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(1)}} \sum_{x\in\mathcal{X}} \sum_{a\neq a_x^\star} \eta^\star(x,a) \left(d_{x,a}(\theta^\star, \theta'') - \frac{1}{\sigma^2} |\mu_{\theta'}(x,a) - \mu_{\theta''}(x,a)| \right) \tag{D.8}$$

$$\overset{(c)}{\geq} 1 - \frac{1}{\sigma^2} \sup_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(1)}} \sum_{x\in\mathcal{X}} \sum_{a\neq a_x^\star} \eta^\star(x,a) |\mu_{\theta'}(x,a) - \mu_{\theta''}(x,a)| \tag{D.9}$$

$$\overset{(d)}{\geq} 1 - \frac{c_\Theta}{\sigma^2} \sup_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(1)}} \sum_{x\in\mathcal{X}} \sum_{a\neq a_x^\star} \eta^\star(x,a) \underbrace{|\mu_{\theta^\star}^\star(x) - \mu_{\theta'}(x, a_x^\star)|}_{<\epsilon_z} \tag{D.10}$$

$$\overset{(e)}{\geq} 1 - \frac{c_\Theta \epsilon_z}{\sigma^2} \sum_{x\in\mathcal{X}} \sum_{a\neq a_x^\star} \eta^\star(x,a), \tag{D.11}$$

where (a) uses the fact that $(i) \geq 0$ and the definition of $\overline{\eta}$, (b) uses the Lipschitz property of the KL divergence between Gaussians, (c) uses the fact that $\overline{\eta}$ is feasible for confusing models (see Eq. D.3), (d) uses Prop. D.1.1 and (e) uses the definition of $\widetilde{\Theta}_{\text{alt}}^{(1)}$. Regarding the second set of alternative models,

$$\inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(2)}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \overline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') \tag{D.12}$$

$$\overset{(f)}{\geq} \inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(2)}} \sum_{x\in\mathcal{X}} \rho(x) \left(z - \sum_{a\neq a_{\theta^\star}^\star(x)} \eta^\star(x,a)/\rho(x) \right) d_{x,a_x^\star}(\theta^\star, \theta') \tag{D.13}$$

$$\overset{(g)}{=} \inf_{\theta' \in \widetilde{\Theta}_{\text{alt}}^{(2)}} \sum_{x\in\mathcal{X}} \rho(x) \left(z - \sum_{a\neq a_{\theta^\star}^\star(x)} \eta^\star(x,a)/\rho(x) \right) \frac{1}{2\sigma^2} \underbrace{(\mu_{\theta^\star}(x, a_x^\star) - \mu_{\theta'}(x, a_x^\star))^2}_{\geq \epsilon_z^2} \tag{D.14}$$

$$\overset{(k)}{=} \frac{\epsilon_z^2}{2\sigma^2} \left(z - \sum_{x\in\mathcal{X}} \sum_{a\neq a_{\theta^\star}^\star(x)} \eta^\star(x,a) \right). \tag{D.15}$$

where (f) uses the fact that $(ii) \geq 0$ and the definition of $\overline{\eta}$, (g) uses the definition of KL for Gaussian distributions and (k) uses the definition of $\widetilde{\Theta}_{\text{alt}}^{(2)}$. Let $z^\star(\theta^\star) := \sum_{x\in\mathcal{X}} \sum_{a\neq a_{\theta^\star}^\star(x)} \eta^\star(x,a)$. Putting together the results so far, we have

$$\inf_{\theta' \in \Theta_{\text{alt}}} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \overline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') \geq \min\left\{1, 1 - \frac{c_\Theta \epsilon_z z^\star(\theta^\star)}{\sigma^2}, \frac{\epsilon_z^2}{2\sigma^2}\Big(z - z^\star(\theta^\star)\Big)\right\}. \tag{D.16}$$

Setting $\epsilon_z = \sqrt{\frac{2\sigma^2}{z}}$,

$$\inf_{\theta' \in \Theta_{\text{alt}}} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \overline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') \geq \max\left\{ \min\left\{ 1 - \frac{c_\Theta \sqrt{2} z^\star(\theta^\star)}{\sigma\sqrt{z}}, 1 - \frac{z^\star(\theta^\star)}{z} \right\}, 0 \right\}, \tag{D.17}$$

Therefore, in case 2 we have

$$u^\star(z,\theta^\star) \leq v^\star(\theta^\star) + \lambda^\star(z,\theta^\star) \min\left\{ \max\left\{ \frac{c_\Theta \sqrt{2} z^\star(\theta^\star)}{\sigma\sqrt{z}}, \frac{z^\star(\theta^\star)}{z} \right\}, 1 \right\}.$$

Bounding $\lambda^\star(z,\theta^\star)$. Finally, we show that the optimal multiplier $\lambda^\star(z,\theta^\star)$ is bounded (regardless of which case z falls into). Let $\underline{\eta} = z\underline{\omega}$, where $\underline{\omega} = \omega^\star_{\underline{z},\theta^\star}$ is the pure-exploration solution obtained solving problem (P_z) with $\underline{z}(\theta^\star)$. Recall from the first statement of Lem. 10.3.1 that

$$\inf_{\theta' \in \Theta_{\text{alt}}} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \underline{\omega}(x,a) d_{x,a}(\theta^\star, \theta') = \frac{1}{\underline{z}(\theta^\star)}.$$

Thus, $\underline{\eta}$ is strictly feasible for problem ($\tilde{\mathrm{P}}_z$) and has constraint value

$$\inf_{\theta' \in \Theta_{\text{alt}}} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \underline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') = \frac{z}{\underline{z}(\theta^\star)} > 1 \tag{D.18}$$

since $z > \underline{z}(\theta^\star)$ by assumption. Using the Slater's condition (see e.g., Lem. 3 in (Nedić and Ozdaglar, 2009)),

$$0 \leq \lambda^\star(z,\theta^\star) \leq \frac{\sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \Delta_{\theta^\star}(x,a)(\underline{\eta}(x,a) - \eta^\star_z(x,a))}{\inf_{\theta' \in \Theta_{\text{alt}}} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \underline{\eta}(x,a) d_{x,a}(\theta^\star, \theta') - 1} \tag{D.19}$$

$$\leq \frac{z}{\frac{z}{\underline{z}(\theta^\star)} - 1} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \underbrace{\Delta_{\theta^\star}(x,a)}_{\geq 0} \left(\underline{\omega}(x,a) - \underbrace{\eta^\star_z(x,a)/z}_{\geq 0} \right) \tag{D.20}$$

$$\leq \frac{z}{\frac{z}{\underline{z}(\theta^\star)} - 1} \sum_{x \in \mathcal{X}} \rho(x) \sum_{a \in \mathcal{A}} \underbrace{\Delta_{\theta^\star}(x,a)}_{\in [0, 2BL]} \underline{\omega}(x,a) \leq 2BL \frac{z\underline{z}(\theta^\star)}{z - \underline{z}(\theta^\star)}. \tag{D.21}$$

Auxiliary Results

Concentration Inequalities

Lemma D.2.1 (Concentration of ρ during exploration)**.** *For any context $x \in \mathcal{X}$,*

$$\sum_{t \geq 1} \sum_{x \in \mathcal{X}} \mathbb{P}\left\{ E_t, |\widehat{\rho}_t(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}| S_t^2)}{2S_t}} \right\} \leq \frac{\pi^2}{3}. \tag{D.22}$$

Proof. The proof follows Lem. B.1 in (Combes and Proutière, 2014). Fix some $\overline{t} \geq 1$ and $x \in \mathcal{X}$. Then,

$$\sum_{t=1}^{\overline{t}} \mathbb{1}\left\{ E_t, |\widehat{\rho}_t(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}| S_t^2)}{2S_t}} \right\} \leq \sum_{s \geq 1} \mathbb{1}\left\{ |\widehat{\rho}_{\tau_s}(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}| s^2)}{2s}}, \tau_s \leq \overline{t} \right\}.$$

where τ_s is the random time the s-th exploration round occurs. Thus, by taking the expectation of both sides,

$$\sum_{t=1}^{\bar{t}} \mathbb{P}\left\{E_t, |\widehat{\rho}_t(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}|S_t^2)}{2S_t}}\right\} \leq \sum_{s\geq 1} \mathbb{P}\left\{|\widehat{\rho}_{\tau_s}(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}|s^2)}{2s}}, \tau_s \leq \bar{t}\right\}.$$

Since τ_s is a stopping-time upper bounded by $\bar{t}$ and the number of samples used to compute $\widehat{\rho}_{\tau_s}(x)$ is at least s, we can apply Lemma 4.3 of (Combes and Proutière, 2014):

$$\sum_{t=1}^{\bar{t}} \mathbb{P}\left\{E_t, |\widehat{\rho}_t(x) - \rho(x)| > \sqrt{\frac{\log(|\mathcal{X}|S_t^2)}{2S_t}}\right\} \leq \sum_{s\geq 1} 2e^{-2s\frac{\log(|\mathcal{X}|s^2)}{2s}} = \frac{2}{|\mathcal{X}|}\sum_{s\geq 1}\frac{1}{s^2} = \frac{\pi^2}{3|\mathcal{X}|}.$$

The reasoning above holds for any $\bar{t}$ and $x \in \mathcal{X}$. Summing over $\mathcal{X}$ concludes the proof. □

Lemma D.2.2 (Confidence set for exploration)**.** *With some abuse of notation, let $\gamma_t := c_{n,1/S_t^2}$. Then, under the same conditions as in Theorem 10.2.1,*

$$\sum_{t=1}^{n} \mathbb{P}\left\{E_t, \|\widehat{\theta}_{t-1} - \theta^\star\|_{\overline{V}_{t-1}} > \sqrt{\gamma_t}\right\} \leq \frac{\pi^2}{6}.$$

Proof. Let $\{\tau_s\}_{s\geq 1}$ be a sequence of stopping times with respect to $\mathcal{F}$ such that if $\tau_s = t$, then the s-th exploration round occurs at time $t+1$. Then,

$$\sum_{t=1}^{n} \mathbb{1}\left\{E_t, \|\widehat{\theta}_{t-1} - \theta^\star\|_{\overline{V}_{t-1}} > \sqrt{\gamma_t}\right\} \leq \sum_{s\geq 1} \mathbb{1}\left\{\|\widehat{\theta}_{\tau_s} - \theta^\star\|_{\overline{V}_{\tau_s}} > \sqrt{\gamma_{\tau_s+1}}, \tau_s \leq n\right\}. \quad \text{(D.23)}$$

Since $S_{\tau_s+1} = s$, we have $\gamma_{\tau_s+1} = c_{n,1/s^2}$. Taking expectations and applying Theorem 10.2.1,

$$\sum_{s\geq 1} \mathbb{P}\left\{\|\widehat{\theta}_{\tau_s} - \theta^\star\|_{\overline{V}_{\tau_s}} > \sqrt{\gamma_{\tau_s+1}}, \tau_s \leq n\right\} \leq \sum_{s\geq 1}\frac{1}{s^2} = \frac{\pi^2}{6}.$$

□

The following result is immediate from the definition of good event.

Lemma D.2.3. *Let $t \in [n]$ be a time step in which the good event G_t holds. Then,*

$$\forall x \in \mathcal{X}, a \in \mathcal{A} : |\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) - \mu_{\theta^\star}(x,a)| \leq \sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}.$$

Proof. Fix any $x \in \mathcal{X}$ and $a \in \mathcal{A}$. We distinguish three cases.

Case 1. If $\mu_{\widehat{\theta}_{t-1}}(x,a) \in [-BL, BL]$, we have $\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) = \mu_{\widehat{\theta}_{t-1}}(x,a)$. Thus, by Cauchy-Schwartz inequality, we can write

$$\begin{aligned}|\mu_{\widehat{\theta}_{t-1}}(x,a) - \mu_{\theta^\star}(x,a)| &= |\phi(x,a)^T(\widehat{\theta}_{t-1} - \theta^\star)| = |\phi(x,a)^T\bar{V}_{t-1}^{-1/2}\bar{V}_{t-1}^{1/2}(\widehat{\theta}_{t-1} - \theta^\star)| \\ &\leq \|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}\|\widehat{\theta}_{t-1} - \theta^\star\|_{\bar{V}_{t-1}} \leq \sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}.\end{aligned}$$

Case 2. If $\mu_{\widehat{\theta}_{t-1}}(x,a) < -BL$, then $\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) = -BL$. Since $\mu_{\theta^\star}(x,a) \geq -BL$, it must be that

$$|\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) - \mu_{\theta^\star}(x,a)| < |\mu_{\widehat{\theta}_{t-1}}(x,a) - \mu_{\theta^\star}(x,a)| \leq \sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}},$$

and the result follows as in Case 1.

Case 3. Finally, if $\mu_{\widehat{\theta}_{t-1}}(x,a) > BL$, then $\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) = BL$ and the result follows using the same argument as in Case 2. □

Lemma D.2.4. *Let $\gamma_t := c_{n,1/S_t^2}$ and $n \geq 3$. Then, for any time step t in which the good event G_t (see App. 10.6.3) holds,*

$$f_t(\omega) := \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\left(\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) + \sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}\right) \geq \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\mu_{\theta^\star}(x,a), \tag{D.24}$$

and

$$g_t(\omega) := \inf_{\theta'\in\overline{\Theta}_{t-1}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\left(\bar{d}_{x,a}\left(\widehat{\theta}_{t-1},\theta'\right) + \frac{2LB}{\sigma^2}\sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}\right) \geq \inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star,\theta'). \tag{D.25}$$

Proof. Since $\widehat{\rho}$ and ω are non-negative, the first inequality is trivial by upper bounding the true mean $\mu_{\theta^\star}(x,a)$ for each x,a by using the definition of G_t^θ and Lemma D.2.3. Let us prove the second one. Fix any model $\theta' \in \Theta$. By using the definition of KL divergence of Gaussians with fixed variance, we have that:

$$\begin{aligned} d_{x,a}(\theta^\star,\theta') = \frac{(\mu_{\theta'}(x,a) - \mu_{\theta^\star}(x,a))^2}{2\sigma^2} &\leq \bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') + \frac{2LB}{\sigma^2}|\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) - \mu_{\theta^\star}(x,a)| \\ &\leq \bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') + \frac{2LB}{\sigma^2}\sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}, \end{aligned}$$

where the first inequality is from $|(a-c)^2 - (b-c)^2| = |(a+b-2c)(a-b)| \leq 4LB|a-b|$ and the second one is once again from the definition of G_t and Lemma D.2.3. Therefore,

$$\begin{aligned} &\inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a) d_{x,a}(\theta^\star,\theta') \qquad \text{(D.26)} \\ &\leq \inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\left(\bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') + \frac{2LB}{\sigma^2}\sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}\right). \end{aligned}$$

We now upper bound the infimum over models in the alternative set. Note that such set can be fully specified once we assign an optimal arm to each context. Let $\{a_x\}_{x\in\mathcal{X}}$ and define

$$\Theta(\{a_x\}_{x\in\mathcal{X}}) = \{\theta' \in \Theta | \exists x \in \mathcal{X} : a^\star_{\theta'}(x) \neq a_x\}.$$

Note that $\Theta_{alt} = \Theta(\{a^\star_{\theta^\star}(x)\}_{x\in\mathcal{X}})$. Then,

$$\begin{aligned} &\inf_{\theta'\in\Theta_{alt}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') \qquad \text{(D.27)} \\ &\leq \max_{\{a_x\}_{x\in\mathcal{X}}} \inf_{\theta'\in\Theta(\{a_x\}_{x\in\mathcal{X}})} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta') \qquad \text{(D.28)} \\ &\leq \inf_{\theta'\in\overline{\Theta}_{t-1}} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega(x,a)\bar{d}_{x,a}(\widehat{\theta}_{t-1},\theta'). \qquad \text{(D.29)} \end{aligned}$$

To see the last inequality, note that for all $\{a_x\}_{x\in\mathcal{X}}$ which do not contain only the optimal arms of $\widehat{\theta}_{t-1}$ (i.e., $\{a_x\}_{x\in\mathcal{X}} \neq \{a^\star_{\widehat{\theta}_{t-1}}(x)\}_{x\in\mathcal{X}}$), we have $\widehat{\theta}_{t-1} \in \Theta(\{a_x\}_{x\in\mathcal{X}})$, and therefore the infimum is zero. Thus, the maximum must be attained by $\{a^\star_{\widehat{\theta}_{t-1}}(x)\}_{x\in\mathcal{X}}$, which yields $\Theta(\{a^\star_{\widehat{\theta}_{t-1}}(x)\}_{x\in\mathcal{X}}) = \overline{\Theta}_{t-1}$. This concludes the proof. □

Lemma D.2.5. *For all time steps t,*

$$\sum_{s\le t:E_s,G_s}\sum_{x\in\mathcal{X}}|\hat{\rho}_{s-1}(x)-\rho(x)|\le 4|\mathcal{X}|\left(\sqrt{S_t\log(|\mathcal{X}|S_t^2)}+\log S_t+1\right). \tag{D.30}$$

Proof. Using the definition of G_s,

$$\begin{aligned}
\sum_{s\le t:E_s,G_s}\sum_{x\in\mathcal{X}}|\hat{\rho}_{s-1}(x)-\rho(x)| &\le |\mathcal{X}|\sum_{s\le t:E_s,G_s}2\max\left(\sqrt{\frac{\log(|\mathcal{X}|S_s^2)}{2S_s}},\frac{2}{s}\right)\\
&\le 2|\mathcal{X}|\sum_{s=1}^{S_t}\max\left(\sqrt{\frac{\log(|\mathcal{X}|s^2)}{2s}},\frac{2}{s}\right)\\
&\le 2|\mathcal{X}|\sqrt{\frac{\log(|\mathcal{X}|S_t^2)}{2}}\sum_{s=1}^{S_t}\frac{1}{\sqrt{s}}+4|\mathcal{X}|\sum_{s=1}^{S_t}\frac{1}{s}\\
&\le 4|\mathcal{X}|\left(\sqrt{S_t\log(|\mathcal{X}|S_t^2)}+\log S_t+1\right),
\end{aligned}$$

where the last inequality holds since

$$\sum_{t=1}^{m}\sqrt{\frac{1}{t}}\le 1+\int_1^m x^{-1/2}dx=1+[2x^{1/2}]_1^m=2\sqrt{m}-1<2\sqrt{m}$$

and $\sum_{t=1}^m\frac{1}{t}\le\log m+1$. □

Lemma D.2.6. *Let t be such that both E_t and G_t occur and suppose $\nu\ge 1$. Define*

$$\Psi_t:=\sum_{s\le t:E_s}\sum_{x\in\mathcal{X}}\hat{\rho}_{s-1}(x)\sum_{a\in\mathcal{A}}\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}}.$$

Then,

$$\Psi_t\le\frac{4L|\mathcal{X}|}{\sqrt{\nu}}\left(\sqrt{S_t\log(|\mathcal{X}|S_t^2)}+\log S_t+1\right)+\frac{M_tL}{\sqrt{\nu}}+\frac{L}{\nu}\sqrt{S_t\log S_t}.+\sqrt{2dS_t\log\frac{\nu+S_tL^2/d}{\nu}}.$$

Proof. We start by noticing that, for all x,a and $s\ge 0$,

$$\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}}^2=\phi(x,a)^T\bar{V}_{s-1}^{-1}\phi(x,a)\le\sigma_{\max}(\bar{V}_{s-1}^{-1})\underbrace{\|\phi(x,a)\|_2^2}_{\le L}\le\frac{L^2}{\sigma_{\min}(\bar{V}_{s-1})}\le\frac{L^2}{\nu},$$

and thus $\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}}\le L/\sqrt{\nu}$. Here $\sigma_{\max}(\cdot)$ and $\sigma_{\min}(\cdot)$ denote the maximum and minimum

eigenvalue of a matrix, respectively. Splitting the steps where the good event does and does not hold,

$$\begin{aligned}
\Psi_t &= \sum_{s\le t:E_s,G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\hat{\rho}_{s-1}(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} + \sum_{s\le t:E_s,\neg G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\hat{\rho}_{s-1}(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} \\
&\le \sum_{s\le t:E_s,G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}(\hat{\rho}_{s-1}(x)-\rho(x))\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} + \frac{M_t L}{\sqrt{\nu}} \\
&\quad + \sum_{s\le t:E_s,G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} \\
&\le \frac{L}{\sqrt{\nu}}\sum_{s\le t:E_s,G_s}\sum_{x\in\mathcal{X}}|\hat{\rho}_{s-1}(x)-\rho(x)| + \frac{M_t L}{\sqrt{\nu}} + \sum_{s\le t:E_s,G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} \\
&\le \frac{4L|\mathcal{X}|}{\sqrt{\nu}}\left(\sqrt{S_t\log(|\mathcal{X}|S_t^2)}+\log S_t+1\right) + \frac{M_t L}{\sqrt{\nu}} + \sum_{s\le t:E_s,G_s}\sum_{\substack{x\in\mathcal{X}\\a\in\mathcal{A}}}\rho(x)\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}},
\end{aligned}$$

where in the first and second inequality we bounded the expected feature-norms by their maximum value and added/subtracted the first term with the true context distribution. In the last step we applied Lemma D.2.5. We now focus exclusively on the third term. Using the fact that the good event holds at time t,

$$\begin{aligned}
\sum_{s\le t:E_s,G_s}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} &\le \sum_{s\le t:E_s}\sum_{x\in\mathcal{X}}\rho(x)\sum_{a\in\mathcal{A}}\omega_s(x,a)\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} \\
&\le \sum_{s\le t:E_s}\|\phi(X_s,A_s)\|_{\bar{V}_{s-1}^{-1}} + \frac{L}{\nu}\sqrt{S_t\log S_t}.
\end{aligned}$$

Finally, let $\bar{V}_{e,t} := \sum_{s\le t:E_s}\phi(X_s,A_s)\phi(X_s,A_s)^T + \nu I$ denote the regularized design matrix computed using only the exploration rounds. Then, we have $\bar{V}_t \succeq \bar{V}_{e,t}$ (since sum of rank-one matrices), which implies $\bar{V}_t^{-1} \preceq \bar{V}_{e,t}^{-1}$ and thus $\|\phi(x,a)\|_{\bar{V}_{s-1}^{-1}} \le \|\phi(x,a)\|_{\bar{V}_{e,s-1}^{-1}}$. Here $\succeq$ denotes the Loewner ordering, i.e., for two symmetric matrices A, B we have $A \succeq B$ ($A \succ B$) if $A - B$ is positive semi-definite (positive definite). Therefore,

$$\begin{aligned}
\sum_{s\le t:E_s}\|\phi(X_s,A_s)\|_{\bar{V}_{s-1}^{-1}} \le \sum_{s\le t:E_s}\|\phi(X_s,A_s)\|_{\bar{V}_{e,s-1}^{-1}} &\overset{(a)}{\le} \sqrt{S_t\sum_{s\le t:E_s}\|\phi(X_s,A_s)\|^2_{\bar{V}_{e,s-1}^{-1}}} \\
&\overset{(b)}{\le} \sqrt{2S_t\log\frac{\det(\bar{V}_{e,t})}{\nu^d}} \overset{(d)}{\le} \sqrt{2dS_t\log\frac{\nu+S_tL^2/d}{\nu}},
\end{aligned}$$

where in (a) we equivalently rewritten the first term as a sum over exploration rounds, (b) is from Cauchy-Schwartz inequality, in (c) we used Lemma 11 of Abbasi-Yadkori et al. (2011), and in (d) we used the determinant-trace inequality (Lemma 10 of Abbasi-Yadkori et al. (2011)) to bound the determinant of $\bar{V}_{e,t}$ by $(\nu + S_tL^2/d)^d$. The final statement follows by combining the previous bounds. □

Online Convex Optimization

Here we recall some basic results from online convex optimization. See (e.g., Beck and Teboulle, 2003) for detailed proofs and discussion of these results.

Lemma D.2.7 (Recursion bound for subgradient descent)**.** *Let* $\sup_{t\geq 1:E_t} |g_t(\omega_t, z_k)|^2 \leq b_\lambda$*. For any phase* $k \geq 0$*,* $t \in \mathcal{T}_k^E$*, and* $\lambda \in \mathbb{R}_+$*, the incremental updates to the Lagrange multiplier* $\{\lambda_t\}_{t\in\mathcal{T}_k^E}$ *of Algorithm* **??** *satisfy*

$$\sum_{s\leq t: s\in\mathcal{T}_k^E} g_s(\omega_s, z_k)(\lambda_s - \lambda) \leq \frac{1}{2\alpha_k^\lambda}(\lambda - \lambda_1)^2 + \frac{\alpha_k^\lambda b_\lambda^2}{2} S_{t,k}.$$

Proof. Recall that the optimization process is reset at the beginning of each phase. Let $\tau_{s,k}$ be a random variable indicating the time at which the s-th exploration round of phase k occurs. Note that $\lambda_{\tau_{1,k}} = \lambda_1$. In order to simplify the exposition, and with some abuse of notation, let $\lambda_s = \lambda_{\tau_{s,k}}$ and $g_s = g_{\tau_{s,k}}(\omega_{\tau_{s,k}}, z_k)$. By definition of the update rule, for each $s \geq 1$,

$$\begin{aligned}(\lambda_{s+1} - \lambda)^2 &= (\min\{[\lambda_s - \alpha_k^\lambda g_s]_+, \lambda_{\max}\} - \lambda)^2 = \min\{[\lambda_s - \alpha_k^\lambda g_s]_+ - \lambda, \lambda_{\max} - \lambda\}^2 \\ &\leq (\lambda_s - \alpha_k^\lambda g_s - \lambda)^2 = (\lambda_s - \lambda)^2 + (\alpha_k^\lambda g_s)^2 + 2\alpha_k^\lambda(\lambda - \lambda_s)g_s.\end{aligned}$$

Dividing by $2\alpha_k^\lambda$ and rearranging,

$$(\lambda_s - \lambda)g_s \leq \frac{(\lambda_s - \lambda)^2 - (\lambda_{s+1} - \lambda)^2}{2\alpha_k^\lambda} + \frac{\alpha_k^\lambda}{2} g_s^2.$$

Summing over all s up to S_t and noting that the first sum on the right-hand side is telescopic,

$$\sum_{s=1}^{S_t}(\lambda_s - \lambda)g_s \leq \frac{1}{2\alpha_k^\lambda}(\lambda_1 - \lambda)^2 - \frac{1}{2\alpha_k^\lambda}(\lambda_{S_t+1} - \lambda)^2 + \frac{\alpha_k^\lambda}{2}\sum_{s=1}^{S_t} g_s^2.$$

The proof is concluded by upper-bounding the second term by zero and mapping the exploration counter s back to time steps. □

Lemma D.2.8. *[Recursion bound for Online Mirror Descent (OMD)] Let* ω_1 *be the uniform distribution over actions for each context and* $\sup_{t\geq 1:E_t} \|q_t\|_\infty \leq b_\omega$*. For any phase* $k \geq 0$*,* $t \in \mathcal{T}_k^E$*, and* $\omega \in \Omega$*, the OMD updates of Algorithm* **??** *satisfy*

$$\sum_{s\leq t: s\in\mathcal{T}_k^E} h_s(\omega_s, \lambda_s, z_k) - \sum_{s\leq t: s\in\mathcal{T}_k^E} h_s(\omega, \lambda_s, z_k) \geq -\frac{\log|\mathcal{A}|}{\alpha_k^\omega} - \frac{\alpha_k^\omega b_\omega^2}{2} S_{t,k}.$$

Proof. We can follow the same steps as before, mapping time steps to exploration counters and then applying the standard recursion bound for OMD (e.g., Beck and Teboulle, 2003). □

Corollary D.2.1. *[Recursion bound for primal-dual algorithm] For any phase* $k \geq 0$*,* $t \in \mathcal{T}_k^E$*,* $\omega \in \Omega$*, and* $\lambda \in \mathbb{R}_+$*, under the same conditions as in Lemma D.2.8 and D.2.7,*

$$\begin{aligned}\sum_{s\leq t: s\in\mathcal{T}_k^E} f_s(\omega_s) \geq \sum_{s\leq t: s\in\mathcal{T}_k^E} h_s(\omega, \lambda_s, z_k) - \lambda \sum_{s\leq t: s\in\mathcal{T}_k^E} g_s(\omega_s, z_k) - \frac{\log|\mathcal{A}|}{\alpha_k^\omega} - \frac{\alpha_k^\omega b_\omega^2}{2} S_{t,k} \\ - \frac{1}{2\alpha_k^\lambda}(\lambda - \lambda_1)^2 - \frac{\alpha_k^\lambda b_\lambda^2}{2} S_{t,k}.\end{aligned}$$

Proof. The proof is straightforward by expanding $\sum_{s\leq t: s\in\mathcal{T}_k^E} h_s(\omega_s, \lambda_s, z_k) = \sum_{s\leq t: s\in\mathcal{T}_k^E}(f_s(\omega_s) + \lambda_s g_s(\omega_s, z_k))$ and combining Lemma D.2.8 with Lemma D.2.7. □

Worst-case Analysis (Proof of Theorem 10.5.2)

Outline

The proof follows a similar argument as the one of Thm. 10.5.1 but it is considerably simpler and shorter. In particular, the main simplifications come from two worst-case arguments. (1) While bounding the regret during exploration rounds, we use the naive bound $S_n \leq n$. This is equivalent to assuming that SOLID never enters the exploitation step and it allows us to entirely avoid the bound on the number of phases of Sec. 10.6.4. (2) We completely ignore the sequence z_k and proceed as if the optimization problem (P_z) was infeasible in all phases. This makes the multiplier saturate to $\lambda_{\max}$ and facilitate the analysis of the resulting Lagrangian[1]. An outline of the proof, together with the main differences w.r.t. the one of Thm. 10.5.1, is as follows.

1. We decompose the regret suffered during exploitation and exploration rounds. Using the same steps as in Sec. 10.6.4, we bound the former by a constant and reduce the latter to the sum of objective values.
2. Instead of relating to the objective values of the optimal policies $\omega^\star_{z_k}$ at each phase k (as was done in Sec. 10.6.4, we reduce our bound to the optimal solution of our bandit problem, i.e., the policy that only pulls optimal arms. This makes the sum of objective values cancel since the optimal policy achieves zero regret.
3. Using the results of Sec. 10.6.4, we show that the sum of constraints is $\mathcal{O}(\log n)$.
4. We use the naive bound $S_n \leq n$ to conclude the proof.

Proof

We start from the same regret decomposition as in Sec. 10.6.4,

$$R_n = \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t) \mathbb{1}\left\{\neg E_t\right\} + \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t) \mathbb{1}\left\{E_t\right\} = R_n^{\mathrm{exploit}} + R_n^{\mathrm{explore}}.$$

The regret suffered during the exploitation rounds was bounded in Sec. 10.6.4 as $\mathbb{E}\left[R_n^{\mathrm{exploit}}\right] \leq 2LB$. Regarding the regret suffered during the exploration rounds, we have

$$R_n^{\mathrm{explore}} := \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t) \mathbb{1}\left\{E_t\right\} \leq \sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t) \mathbb{1}\left\{E_t, G_t\right\} + 2LB \underbrace{\sum_{t=1}^{n} \mathbb{1}\left\{E_t, \neg G_t\right\}}_{:=M_n}. \tag{D.31}$$

Refer to Sec. 10.6.3 for the definition of G_t. The second term is M_n, the number of exploration rounds in which the good event does not hold, and can be bounded in expectation by using Lem. 10.6.2. The first one can be bounded by using the good event. Suppose, without loss of generality, that E_n and G_n hold (if they do not, the following reasoning can be repeated for the last time

[1] Recall that the regret of SOLID is not defined in terms of the optimization problem (P_z) or its Lagrangian, but only in terms of the rewards of the chosen arms compared to those of the optimal arms. This makes it possible to obtain good regret guarantees even when solving an infeasible optimization problem.

step at which these events hold). Then, using G_t^Δ (see Sec. 10.6.3),

$$\begin{aligned}\sum_{t=1}^{n} \Delta_{\theta^\star}(X_t, A_t)\mathbb{1}\left\{E_t, G_t\right\} &\leq \sum_{t\leq n: E_t} \Delta_{\theta^\star}(X_t, A_t) \\ &\leq \sum_{t\leq n: E_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\Delta_{\theta^\star}(x,a) + 2LB\sqrt{S_n \log S_n}.\end{aligned} \tag{D.32}$$

We now proceed using similar steps as in Sec. 10.6.4, except that we ignore the phases. We decompose the first term as

$$\begin{aligned}&\sum_{t\leq n: E_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\Delta_{\theta^\star}(x,a) \\ &= \sum_{t\leq n: E_t, G_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\Delta_{\theta^\star}(x,a) + \sum_{t\leq n: E_t, \neg G_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a)(\mu^\star_{\theta^\star}(x) - \mu_{\theta^\star}(x,a)) \\ &\leq \sum_{t\leq n: E_t, G_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\Delta_{\theta^\star}(x,a) + M_n\mu^\star - \sum_{t\leq n: E_t, \neg G_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\mu_{\theta^\star}(x,a).\end{aligned}$$

Here we defined

$$\mu^\star := \sum_{x\in\mathcal{X}} \rho(x)\mu^\star_{\theta^\star}(x). \tag{D.33}$$

The last term can be bounded by M_nBL. Regarding the remaining two,

$$\begin{aligned}&\sum_{t\leq n: E_t, G_t} \sum_{x\in\mathcal{X}} \rho(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\Delta_{\theta^\star}(x,a) + M_n\mu^\star \\ &= (S_n - M_n)\mu^\star + M_n\mu^\star - \sum_{t\leq n: E_t, G_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \rho(x)\omega_t(x,a)\mu_{\theta^\star}(x,a) \\ &= S_n\mu^\star + \underbrace{\sum_{\substack{t\leq n:\\ E_t, G_t}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} (\widehat{\rho}_{t-1}(x) - \rho(x))\omega_t(x,a)\mu_{\theta^\star}(x,a)}_{(a)} - \underbrace{\sum_{\substack{t\leq n:\\ E_t, G_t}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \widehat{\rho}_{t-1}(x)\omega_t(x,a)\mu_{\theta^\star}(x,a)}_{(b)}.\end{aligned}$$

Term (a) can be bounded as

$$(a) \leq LB \underbrace{\sum_{t\leq n: E_t, G_t} \sum_{x\in\mathcal{X}} \left|\widehat{\rho}_{t-1}(x) - \rho(x)\right|}_{\zeta_n}.$$

For the sake of readability, we keep the dependence on ζ_n explicit. We will bound this term by Lem. D.2.5 at the end of the proof. Regarding term (b), using the definition of G_t and Lem. D.2.3,

$$\begin{aligned}(b) &\geq \sum_{t\leq n: E_t, G_t} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\left(\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a) - \sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}\right) \\ &\pm \sum_{\substack{t\leq n:\\ E_t, \neg G_t}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \widehat{\rho}_{t-1}(x)\omega_t(x,a)\underbrace{\bar{\mu}_{\widehat{\theta}_{t-1}}(x,a)}_{|\cdot|\leq LB} \pm \sum_{t\leq n: E_t} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \widehat{\rho}_{t-1}(x)\omega_t(x,a)\sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}} \\ &\geq \sum_{t\leq n: E_t} f_t(\omega_t) - M_nBL - 2\sum_{t\leq n: E_t} \sum_{x\in\mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a\in\mathcal{A}} \omega_t(x,a)\sqrt{\gamma_t}\|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}} \\ &\geq \sum_{t\leq n: E_t} f_t(\omega_t) - M_nBL - 2\sqrt{\gamma_n}\Psi_n.\end{aligned}$$

We recall that $\sqrt{\gamma_t} \leq \sqrt{\gamma_n}$ and $\Psi_n := \sum_{t \leq n:E_t} \sum_{x \in \mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a \in \mathcal{A}} \omega_t(x,a) \|\phi(x,a)\|_{\bar{V}_{t-1}^{-1}}$. As for ζ_n, we keep the dependence on Ψ_n explicit and defer bounding this term to the end of the proof. Using the bounds on (a) and (b) and plugging everything back into (D.32) and then into (D.31), we obtain

$$R_n^{\text{explore}} \leq S_n \mu^\star - \sum_{t \leq n:E_t} f_t(\omega_t) + 4M_n BL + \zeta_n BL + 2\sqrt{\gamma_n}\Psi_n + 2BL\sqrt{S_n \log S_n}. \tag{D.34}$$

We now lower bound the sum of objective values. Here we proceed in a slightly different way with respect to the proof of the asymptotically optimal regret bound. Instead of relating to the objective values of the optimal policies $\omega_{z_k}^\star$ at each phase k, we reduce our bound to the optimal solution of our bandit problem, i.e., the policy that only pulls optimal arms. Let

$$\omega_{\theta^\star}^\star(x,a) := \begin{cases} 1 & \text{if } a = a_{\theta^\star}^\star(x) \\ 0 & \text{otherwise} \end{cases} \tag{D.35}$$

Recall that $\sum_{t \leq n:E_t} f_t(\omega_t) = \sum_{k=0}^{K_n} \sum_{t \in \mathcal{T}_k^E} f_t(\omega_t)$. Fix some phase index $k \geq 0$ and let $\lambda \geq 0$ be arbitrary. Using Corollary D.2.1 with $\alpha_k^\lambda = \alpha_k^\omega = 1/\sqrt{p_k}$ and $\omega = \omega_{\theta^\star}^\star$,

$$\sum_{t \in \mathcal{T}_k^E} f_t(\omega_t) \geq \sum_{t \in \mathcal{T}_k^E} h_t(\omega_{\theta^\star}^\star, \lambda_t, z_k) - \lambda \sum_{t \in \mathcal{T}_k^E} g_t(\omega_t, z_k) - a_\lambda \sqrt{p_k}, \tag{D.36}$$

where $a_\lambda := \left(\log|\mathcal{A}| + \frac{b_\omega^2 + b_\lambda^2}{2} + \frac{(\lambda - \lambda_1)^2}{2}\right)$ and b_λ and b_ω are the maximum sub-gradients in λ and ω, respectively. Note that, since we apply Corollary D.2.1 to bound the sum of objective values over the whole phase, we have $S_{n,k} = p_k$. We now lower-bound the first term on the right-hand side. We have

$$\begin{aligned} \sum_{t \in \mathcal{T}_k^E} h_t(\omega_{\theta^\star}^\star, \lambda_t, z_k) &\overset{(c)}{=} \sum_{t \in \mathcal{T}_k^E} f_t(\omega_{\theta^\star}^\star) + \sum_{t \in \mathcal{T}_k^E} \lambda_t g_t(\omega_{\theta^\star}^\star, z_k) \overset{(d)}{\geq} \sum_{t \in \mathcal{T}_k^E} f_t(\omega_{\theta^\star}^\star) - \sum_{t \in \mathcal{T}_k^E} \frac{\lambda_t}{z_k} \\ &\overset{(e)}{\geq} \sum_{t \in \mathcal{T}_k^E} f_t(\omega_{\theta^\star}^\star) - \frac{\lambda_{\max} S_{n,k}}{z_k}, \end{aligned} \tag{D.37}$$

where (c) uses the definition of h_t and g_t (see Eq. 10.9 and Eq. 10.10), (d) uses the positivity of KL divergences and confidence intervals, and (e) uses $\lambda_t \leq \lambda_{\max}$ and $S_{n,k} := |\mathcal{T}_k^E|$. Let us focus on the sum of objective values. Since $f_t(\omega_{\theta^\star}^\star) \geq -LB$, we have $\sum_{t \in \mathcal{T}_k^{E:\neg G_t}} f_t(\omega_{\theta^\star}^\star) \geq -M_{n,k} BL$. For any step $t \in \mathcal{T}_k^E$ in which G_t holds, the optimism property (see Sec. 10.6.3 and Lem. D.2.4) yields

$$\begin{aligned} \sum_{t \in \mathcal{T}_k^{E:G_t}} f_t(\omega_{\theta^\star}^\star) &\geq \sum_{t \in \mathcal{T}_k^{E:G_t}} \sum_{x \in \mathcal{X}} \widehat{\rho}_{t-1}(x) \sum_{a \in \mathcal{A}} \omega_{\theta^\star}^\star(x,a) \mu_{\theta^\star}(x,a) \\ &= \sum_{t \in \mathcal{T}_k^{E:G_t}} \sum_{x \in \mathcal{X}} (\widehat{\rho}_{t-1}(x) - \rho(x)) \underbrace{\sum_{a \in \mathcal{A}} \omega_{\theta^\star}^\star(x,a) \mu_{\theta^\star}(x,a)}_{|\cdot| \leq BL} + \sum_{t \in \mathcal{T}_k^{E:G_t}} \underbrace{f(\omega_{\theta^\star}^\star)}_{=\mu^\star} \\ &\geq (S_{n,k} - M_{n,k})\mu^\star - BL \underbrace{\sum_{t \in \mathcal{T}_k^{E:G_t}} \sum_{x \in \mathcal{X}} |\widehat{\rho}_{t-1}(x) - \rho(x)|}_{:= \zeta_{n,k}}, \end{aligned}$$

where we used the fact that $f(\omega^\star_{\theta^\star}) = \mu^\star$ by definition (D.35) and (D.33) and $\sum_{t\in\mathcal{T}_k^E} \mathbb{1}\{G_t\} = \sum_{t\in\mathcal{T}_k} \mathbb{1}\{E_t\} - \sum_{t\in\mathcal{T}_k} \mathbb{1}\{E_t, \neg G_t\} = S_{n,k} - M_{n,k}$. Plugging this back into (D.37) and then into (D.36),

$$\sum_{t\in\mathcal{T}_k^E} f_t(\omega_t) \geq (S_{n,k} - M_{n,k})\mu^\star - BL\zeta_{n,k} - \frac{\lambda_{\max} S_{n,k}}{z_k} - \lambda \sum_{t\in\mathcal{T}_k^E} g_t(\omega_t, z_k) - a_\lambda\sqrt{p_k} - M_{n,k}BL.$$

Summing over all phases and recalling that $\sum_{k=0}^{K_n} S_{n,k} = S_n$, $\sum_{k=0}^{K_n} M_{n,k} = M_n$, and $\sum_{k=0}^{K_n} \zeta_{n,k} = \zeta_n$, we obtain

$$\sum_{t\leq n:E_t} f_t(\omega_t) \geq (S_n - M_n)\mu^\star - BL\zeta_n - \sum_{k=0}^{K_n} \frac{\lambda_{\max} S_{n,k}}{z_k} - \lambda \sum_{t\leq n:E_t} g_t(\omega_t, z_{K_t}) \tag{D.38}$$

$$- a_\lambda \sum_{k=0}^{K_n} \sqrt{p_k} - M_n BL. \tag{D.39}$$

Using the definition of g_t (see Eq. 10.9),

$$\sum_{t\leq n:E_t} g_t(\omega_t, z_{K_t}) := \sum_{t\leq n:E_t} \inf_{\theta'\in\overline{\Theta}_{t-1}} \sum_{\substack{x\in\mathcal{X}\\ a\in\mathcal{A}}} \widehat{\rho}_{t-1}(x)\omega_t(x,a)\left(d_{x,a}\left(\widehat{\theta}_{t-1}, \theta'\right) + \frac{2LB}{\sigma^2}\sqrt{\gamma_t}\|\phi(x,a)\|_{\overline{V}_{t-1}^{-1}}\right)$$
$$- \sum_{t\leq n:E_t} \frac{1}{z_{K_t}}.$$

By the definition of phase, the second term is $\sum_{t\leq n:E_t} \frac{1}{z_{K_t}} = \sum_{k=0}^{K_n} \frac{S_{n,k}}{z_k}$. The first term can be bounded using exactly the same steps as in Sec. 10.6.4.[2] We obtain

$$\sum_{t\leq n:E_t} g_t(\omega_t, z_{K_t}) \leq \frac{\beta_{n-1}}{2\sigma^2} - \sum_{k=0}^{K_n} \frac{S_{n,k}}{z_k} + \frac{2L^2B^2}{\sigma^2} M_n + \frac{6LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n + \frac{2L^2B^2}{\sigma^2}\zeta_n$$
$$+ \frac{2B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right). \tag{D.40}$$

If we now set $\lambda = \lambda_{\max}$ and plug (D.40) into (D.38),

$$\sum_{t\leq n:E_t} f_t(\omega_t) \geq (S_n - M_n)\mu^\star - BL\left(1 + \frac{2\lambda_{\max}BL}{\sigma^2}\right)(\zeta_n + M_n) + \underbrace{\sum_{k=0}^{K_n} \frac{\lambda_{\max} S_{n,k}}{z_k} - \sum_{k=0}^{K_n} \frac{\lambda_{\max} S_{n,k}}{z_k}}_{=0}$$
$$- \frac{\lambda_{\max}\beta_{n-1}}{2\sigma^2} - a_{\lambda_{\max}} \sum_{k=0}^{K_n} \sqrt{p_k} - \frac{6\lambda_{\max}LB}{\sigma^2}\sqrt{\gamma_n}\Psi_n - \frac{2\lambda_{\max}B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right).$$

We can finally plug this into (D.34), thus obtaining

$$R_n^{\text{explore}} \leq M_n \underbrace{\mu^\star}_{|\cdot|\leq BL} + BL\left(5 + \frac{2\lambda_{\max}BL}{\sigma^2}\right)(\zeta_n + M_n) + \frac{\lambda_{\max}\beta_{n-1}}{2\sigma^2} + a_{\lambda_{\max}} \sum_{k=0}^{K_n} \sqrt{p_k}$$
$$+ \left(2 + \frac{6\lambda_{\max}LB}{\sigma^2}\right)\sqrt{\gamma_n}\Psi_n + \frac{2\lambda_{\max}B^2L^2}{\sigma^2}\left(\sqrt{dS_n \log(dS_n)} + 1\right) + 2BL\sqrt{S_n \log S_n}.$$

[2] Note that the bound on the sum of constraints of Sec. 10.6.4 uses only the properties of the confidence intervals and of the exploitation test. Thus, it is applicable regardless of the feasibility of the optimization problems at each phase.

Let $\bar{k}_n := \min\{k : p_k \geq n\}$, then $K_n \leq \bar{k}_n$. Using the exponential schedule $p_k = e^{rk}$, $\bar{k}_n = \lceil \frac{1}{r} \log n \rceil$ and

$$\sum_{k=0}^{K_n} \sqrt{p_k} \leq \sum_{k=0}^{\bar{k}_n} e^{\frac{r}{2}k} \leq \int_0^{\bar{k}_n+1} e^{\frac{r}{2}x} \mathrm{d}x = \left[\frac{2}{r} e^{\frac{r}{2}x}\right]_0^{\bar{k}_n+1} = \frac{2}{r} e^{\frac{r}{2}(\lceil \frac{1}{r}\log n \rceil + 1)} - \frac{2}{r} \leq \frac{2e^r}{r}\sqrt{n}.$$

Taking expectations of both sides of the regret bound above and using $S_n \leq n$ and $\mathbb{E}[M_n] \leq \frac{3\pi^2}{2}$ by Lem. 10.6.2,

$$\mathbb{E}\left[R_n^{\text{explore}}\right] \leq \frac{3BL\pi^2}{2}\left(6 + \frac{2\lambda_{\max}BL}{\sigma^2}\right) + \frac{\lambda_{\max}\beta_{n-1}}{2\sigma^2} + \frac{2e^r a_{\lambda_{\max}}}{r}\sqrt{n} + 2BL\sqrt{n\log n}$$
$$+ \left(2 + \frac{6\lambda_{\max}LB}{\sigma^2}\right)\mathbb{E}\left[\sqrt{\gamma_n}\Psi_n\right] + \frac{2\lambda_{\max}B^2L^2}{\sigma^2}\left(\sqrt{nd\log(nd)} + 1\right) + BL\left(5 + \frac{2\lambda_{\max}BL}{\sigma^2}\right)\mathbb{E}\left[\zeta_n\right].$$

After bounding $S_n \leq n$, by Lem. D.2.6, $\Psi_n \leq \mathcal{O}\left(L|\mathcal{X}|\sqrt{n\log n} + \sqrt{nd\log n}\right)$ while, by Lem. D.2.5, $\zeta_n \leq \mathcal{O}\left(|\mathcal{X}|\sqrt{n\log n}\right)$. Therefore, recalling that the regret during exploitation rounds was bounded by $2BL$ and noting that $2 < \frac{3\pi^2}{2}$,

$$\mathbb{E}[R_n] \leq 3BL\pi^2\left(4 + \frac{\lambda_{\max}BL}{\sigma^2}\right) + \frac{2e^r\lambda_{\max}^2}{r}\sqrt{n} + C_{\text{sqrt}}\left(1 + \frac{\lambda_{\max}BL}{\sigma^2}\right)\log(n)\sqrt{n},$$

where $C_{\text{sqrt}} = lin_{\geq 0}(|\mathcal{X}|, \sqrt{d}, B, L)$. Here we included $\frac{\lambda_{\max}\beta_{n-1}}{2\sigma^2}$ and the components of $a_{\lambda_{\max}}$ (except $\lambda_{\max}^2$ which is kept explicit) into the last term above. This concludes the proof.