ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

Data Science and Computation

Ciclo XXXIV

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

UNSUPERVISED REINFORCEMENT LEARNING VIA STATE ENTROPY MAXIMIZATION

Presentata da: **Mirco Mutti**

Coordinatore Dottorato

Prof. Daniele Bonacorsi

Supervisore

Prof. Marcello Restelli

Esame finale anno 2022

Acknowledgements

Before starting our journey into the technicalities of unsupervised reinforcement learning via state entropy maximization, I would like to spend a few words to thank the people who contributed, in various ways, to make this dissertation possible in the first place.

First and foremost, I thank my Ph.D. advisor Marcello for his supervision on this thesis, as well as on every other aspect of being a researcher. I especially thank him for advising me to study unsupervised reinforcement learning and for working side-by-side on the first contribution of this thesis, which is featured in Chapter 7.

I thank Mohammad Gheshlaghi Azar and Lerrel Pinto for accepting to review this dissertation and for providing valuable comments. I am humbled and grateful that my work has been appreciated by renowned researchers as they are.

I also thank Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest for their "Provably Efficient Maximum Entropy Exploration". I remember myself circling around a proper formulation of an unsupervised exploration objective when their paper came out. Their work was a major source of inspiration for the contributions of this thesis.

Whereas my dissertation is based on a series of papers that I primarily led, these works relied on the contribution of my brilliant co-authors Lorenzo, Mattia, Riccardo, Piersilvio. Thank you for bringing crucial value to this thesis.

I would like to mention also my students Giovanni, Stefano, Pietro, Majid, Juan who collaborated on projects that are not featured in this document. Working with them has been anyway important to sharpen the expertise I brought to this thesis.

I thank Hao Liu for the useful conversations around the state of the unsupervised reinforcement learning research and for enhancing the capabilities of state entropy maximization with his amazing work. I especially thank him for providing valuable feedback on this dissertation and the papers it is based on.

I want to thank the anonymous reviewers who provided feedback on the papers that are featured in this dissertation. I especially thank them for appreciating our work even beyond

our own expectations sometimes.

I thank Alberto, Matteo, Andrea for their valuable advice. They are extremely talented researchers, and having the chance to follow their footsteps has been a blessing.

I am grateful to my office mates Lorenzo, Luca, Nico, Pierre, Edoardo, Ahmad, and then Marco, Amarildo for their companionship and for sharing this journey with me. Thanks Giorgia for mentoring me sometimes. Despite claiming my desk, you have been a great addition to the team. I thank Francesco for introducing me to teaching and for advice of all kinds. I want to thank also Carlo and Davide, Simone, Alessio, Giuseppe, then Alberto, Andrea, Alessandro, Matteo, Martino, Federico, and finally Riccardo (x2), Alessio, Gianluca, Luca, Paolo, Davide for making the lab such a great place to work and thrive.

I thank my friends Lele, Seggia, Agata, Ale, Cesco, Ardu, Bonni, Fiore, Bono, Meggy, Riki, Simone, Francesca, Malvo, Nicola, Danny and all the others who, while not being technically involved in this thesis, made the last few years working on it way more fun.

Finally, I am immensely grateful to my family, Mamma, Papà, Fabio, Nadia, and Nonna for always supporting me unconditionally.

As it usually happens, these acknowledgements have been written a few hours before submitting the manuscript, which means I am likely forgetting to thank someone deserving. However, if you supported me during this journey, there is a good chance you have been in my thoughts while working on this thesis. I want to thank you all.

Mirco Mutti
February 10th, 2023

Abstract

Reinforcement learning provides a powerful framework to address sequential decision-making problems in which the transition dynamics is unknown or too complex to be represented. The reinforcement learning approach is based on speculating what is the best decision to make given sample estimates obtained from previous interactions, a recipe that led to several breakthroughs in various domains, ranging from game playing to robotics. Despite their undeniable success, current reinforcement learning methods hardly generalize from one task to another, and achieving the kind of generalization obtained through unsupervised pre-training in non-sequential problems seems unthinkable.

Unsupervised reinforcement learning has recently emerged as a way to improve generalization of reinforcement learning methods. Just as its non-sequential counterpart, the unsupervised reinforcement learning framework comprises two phases: An unsupervised pre-training phase, in which the agent interacts with the environment without external feedback, and a supervised fine-tuning phase, in which the agent aims to efficiently solve a task in the same environment by exploiting the knowledge acquired during pre-training.

In this thesis, we study unsupervised reinforcement learning via state entropy maximization, in which the agent makes use of the unsupervised interactions to pre-train a policy that maximizes the entropy of its induced state distribution.

First, we provide a theoretical characterization of the learning problem by considering a convex reinforcement learning formulation that subsumes state entropy maximization. Our analysis shows that maximizing the state entropy in finite trials is inherently harder than reinforcement learning. Then, we study the state entropy maximization problem from an optimization perspective. Especially, we show that the primal formulation of the corresponding optimization problem can be (approximately) addressed through tractable linear programs. Finally, we provide the first practical methodologies for state entropy maximization in complex domains, both when the pre-training takes place in a single environment as well as multiple environments. The procedures are based on flexible non-parametric entropy estimators, which have become de facto standard in subsequent works.

With this thesis, we hope to shed light on the potential of unsupervised pre-training via state entropy maximization in the pursuit of generalization in reinforcement learning.

Contents

CHAPTER 1

Introduction

Let assume you are a tennis coach, and you are supposed to teach a kid how to play tennis. The kid follows you on the court, holding a racket firmly in her hand, full of will and desire to learn. You are a lazy coach though, and so you tell her - Just familiarize with the racket today - and you leave her training alone, hitting a ball against the wall at the bottom of the court. You do the same the next day, and the day after. Perhaps you do it for a week, because you really are lazy. Then, someday you come back and you tell her it is time to practice her forehand. You give very simple instructions - When I throw the ball at you, let it bounce on the court, then hit it along this line. - You definitely do not instruct her on all the muscle activation required to lift the racket backward at first, and then move it upfront to smoothly hit the ball with the right timing. Nevertheless, she is already hitting a very passable forehand after a few unsuccessful trials. Then you challenge her with a different exercise - We are going to hit some volleys: Come closer to the net, when I throw the ball at you, just hit it quickly on the other side. - Again, the results are quite promising. You end the practice for the day and leave the court quite pleased with your coaching ability.

A few meters away, a similar tennis practice takes place on another court. There is a coach, a zealous one, teaching a kid how to hit a forehand. This coach does not tell the kid to initially practice alone. Instead, he starts giving him very precise instructions: How to hold the racket in his hand, how to place his feet in preparation for the shot, what is the exact position at which he should hit the ball, and how to move his arms and wrist during the shot. However, the results are quite bleak at first, the forehand does not come natural at

all. It takes a painful week of practice for this unfortunate kid to hit some good forehands.

What is happening here is quite common in sports and outside of it. The first kid, the one with the lazy coach, is taking benefits from an unsupervised training process. While hitting against the wall, she does not know exactly what kind of exercises she will be tasked with, but she is already discovering the things she can and cannot do with the racket. This proves to be beneficial for her learning curve in the actual practice with the coach. The other kid starts with a precise task right away, but he did not have the chance to familiarize with the racket, the bounce of the ball, and all of the other nuances first. This negatively affects his learning process, which results way slower, despite his zealous coach.

Now, this thesis is not about tennis,[1] so let us translate this metaphor in the language of artificial intelligence and, specifically, reinforcement learning, which we care about. We call a kid a learning *agent*. The movements of the kid are referred as *actions* she/he can take. The court, the racket, the ball, and basically anything else are part of the *environment*. A specific configuration of the environment, such as the relative positions of the ball, the racket, the kid on the court is called a *state*. The coach provides feedback on the actions of the kid, which we call a *reward*.[2] The agent takes sequential actions in the environment trying to maximize the reward in the long term, hence learning how to play tennis.

Reinforcement learning provides a powerful framework to model the learning process of reward maximization in a sequential problem. Instead, *how can we model the unsupervised training into the reinforcement learning framework?* In this thesis, we will address various facets of this important question, with contributions ranging from a theoretical characterization of the problem to practical methodologies for complex domains.

Unsupervised Reinforcement Learning Reinforcement Learning (RL, Sutton & Barto, 2018) is a powerful framework to address sequential decision making problems through sampled interactions. Crucially, with its trial-and-error nature, RL does not require access to the dynamics of the problem, which is often unknown or too complex to be represented efficiently. This is a key reason why, in the last decade, the RL approach led to remarkable breakthroughs, such as mastering Atari games (Mnih et al., 2015), Go (Silver et al., 2016), Dota 2 (Berner et al., 2019), and robotic manipulation (Andrychowicz et al., 2020).

Despite its recent success, RL counts its own drawbacks, ranging from the sample efficiency (Kakade, 2003) to generalization (Kirk et al., 2021). Especially, the RL framework relies on the presence of a reward function that perfectly encodes the task. In principle, the reward arises naturally from the problem. In practice, the reward is usually hand-crafted, and it requires a very careful design in order to drive the learning process towards a desirable outcome. This poses a serious roadblock on the way of autonomous learning, as any task requires a costly and specific formulation. Moreover, the synergy between solving one RL problem and another is very limited, significantly reducing generalization.

Instead, supervised learning made notable strides in terms of generalization as of late. Especially, a procedure that prescribes unsupervised pre-training on massive unlabelled datasets, and then fine-tuning of the pre-trained model on the specific task, has led to outstanding empirical results. This is the recipe behind the recent breakthroughs in language

[1] If you came to this dissertation eager to improve your tennis, you might turn to different textbooks, such as "The Inner Game of Tennis" by W. Timothy Gallwey.

[2] Rewards in reinforcement learning are sometimes as lazy as tennis coaches.

modelling (Brown et al., 2020; Alayrac et al., 2022), and generative models for static images (Ramesh et al., 2022) or video clips (Singer et al., 2022). It is still an open question whether similar generalization capabilities can be obtained in sequential problems by integrating unsupervised pre-training within the RL framework.

The unsupervised RL framework, which has been formalized in (Laskin et al., 2021) although various seeds appeared in the literature before (e.g., Hazan et al., 2019; Mutti & Restelli, 2020), is actually dedicated to answering this question. In the unsupervised RL framework, the learning agent interacts with the environment in two separate and distinct phases. In a first phase, which we call *unsupervised pre-training*, the environment does not include an external reward function to be maximized, and the goal of the agent is to pre-train a model to incorporate the acquired knowledge for later use. Examples of such models are a representation of the transition dynamics (e.g., Jin et al., 2020), an exploratory policy that guarantees good coverage of the environment's states (e.g., Hazan et al., 2019), a reduced set of policies that produce relevant interactions with environment (e.g., Mutti et al., 2022d), an encoding of abstract state representations (Agarwal et al., 2020, e.g.,), a dataset of samples that includes all the relevant interactions (Yarats et al., 2022).

The purpose of the unsupervised pre-training is to equip the agent with a model that can help addressing any downstream task in the same environment. This ability is tested during a second phase, called *supervised fine-tuning*, in which a reward function is revealed to the agent. In the fine-tuning phase, the agent tries to efficiently learn a reward-maximizing policy by exploiting the pre-trained model, such as doing planning with a pre-trained transition model, or collecting samples through a pre-trained exploratory policy. In this thesis, we focus on the unsupervised pre-training phase, while the supervised fine-tuning will serve for the evaluation of the pre-trained model. Especially, we will focus on a specific formulation of the pre-training objective, which is the state entropy maximization problem.

State Entropy Maximization In the unsupervised pre-training via state entropy maximization, the agent looks for a policy that maximizes the entropy of its induced state distribution (Hazan et al., 2019). Notably, this is fairly different from a policy assigning equal probability to all the actions in any given state. Indeed, such a policy neglects the sequential nature of the problem, which often requires performing a specific sequence of actions to reach certain states with high probability, making the problem of state entropy maximization actually non-trivial. While being non-trivial to obtain this policy, state entropy maximization clearly does not require any feedback embedded in the environment, making its corresponding learning process fully unsupervised.

A policy pre-trained to maximize the entropy of the induced state distribution can provide interesting benefits to the subsequent fine-tuning phase. From a theoretical viewpoint, the importance of the state coverage in the data collection for offline RL (Levine et al., 2020) has been extensively demonstrated (Antos et al., 2008; Chen & Jiang, 2019; Jin et al., 2021; Foster et al., 2021; Zhan et al., 2022). More recently, a similar notion of state coverage has been linked to the learning efficiency of online RL methods as well (Xie et al., 2022), while Xie et al. (2021) shows the impact of the state coverage properties of the initial policy in a policy fine-tuning setting. A state entropy maximizing policy has been also demonstrated to provide provably efficient reward identification (Tarbouriech et al., 2021) and transition dynamics representations (Jin et al., 2020, Appendix B).

Whereas state entropy maximization is arguably sub-optimal for the aforementioned objectives, including the notion of state coverage expressed through a concentrability coefficient (Antos et al., 2008), it also showcases a feature that most of the other theoretically-driven unsupervised exploration approaches do not have yet: Practical upside. Thanks to the work in this thesis, we know that practical algorithms for state entropy maximization can be developed, paving the way for efficient unsupervised pre-training of exploratory policies in complex domains.

Contributions In this thesis, we first address a theoretical characterization of the state entropy maximization problem by casting it into the broader convex RL framework (Hazan et al., 2019; Zhang et al., 2020a). Convex RL is a generalization of the standard RL setting, in which the objective can be represented through any convex function of the state distribution. This formulation subsumes state entropy maximization, as well as other relevant sequential problems including imitation learning (Abbeel & Ng, 2004) and risk-averse RL (Garcıa & Fernández, 2015) among the others. Through a formal study of the convex RL problem, we reveal an essential mismatch between its infinite trials formulation analyzed in theory, and its finite trials formulation optimized in practice (see Chapter 4). Then, we prove the importance of non-Markovian policies to optimize the finite trials convex RL formulation, which was previously neglected (see Chapter 5).

Shifting our focus to the specific state entropy maximization objective, which was introduced by (Hazan et al., 2019), we analyze its corresponding optimization problem. Whereas previous works demonstrated that its primal formulation is intractable (Hazan et al., 2019), we derive a lower bound to the primal state entropy objective that is amenable to optimization, as it can be formulated through a linear program (see Chapter 7).

Prior to our work, algorithms for state entropy maximization required either precise representations of the transition dynamics (Tarbouriech & Lazaric, 2019) or state density estimation (Hazan et al., 2019; Lee et al., 2019) to compute the entropy, which hardly scale to the most complex domains. In this thesis, we provide the first practical algorithm for state entropy maximization, which scales to continuous and high-dimensional domains by combining non-parametric entropy estimation (Singh et al., 2003) with policy optimization (see Chapter 8). Non-parametric entropy estimation resulted a key feature in all the subsequent works address state entropy maximization in complex domains (Liu & Abbeel, 2021b; Seo et al., 2021; Yarats et al., 2021).

Finally, our work introduces a novel formulation of the unsupervised pre-training over multiple environments. We provide a practical algorithm specifically designed for this setting, which crucially optimizes the mean of the lower tail of the distribution of the state entropy achieved across multiple environments. We show that this conservative approach allows for efficient fine-tuning even in unfavorable environments within the set (see Chapter 9).

Overview Chapter 2 introduces the basics of sequential decision making and reinforcement learning. Chapter 3 is dedicated to unsupervised RL, for which we present a general problem formulation and an overview of the literature approaches.

Before going through the details of unsupervised pre-training via state entropy maximization, in Part I of this thesis we make a step back to consider the convex RL framework,

which subsumes state entropy maximization. In Chapter 4, we first introduce the convex RL problem, and we reveal a fundamental mismatch between the infinite trials formulation considered in theory, and the finite trials formulation addressed in practice. In Chapter 5, we provide a crucial result on the importance of non-Markovian policies to optimize the finite trials convex RL formulation. Nevertheless, we also show that solving the latter optimization is NP-hard in general.

Part II of this thesis is dedicated to the state entropy maximization objective for unsupervised RL. In Chapter 6, we present the objective function that constitutes the bedrock of this thesis, borrowing contributions from (Hazan et al., 2019). In Chapter 7, we provide an optimization perspective on state entropy maximization, for which we provide a family of tractable optimization programs that trade off computational complexity with tightness of the results.

Part III of this thesis is dedicated to the practical methods for state entropy maximization. In Chapter 8, we provide the first algorithm that is able to address state entropy maximization in continuous high-dimensional domains. Then, in Chapter 9, we consider the problem of unsupervised pre-training over multiple environments, and we provide a practical methodology specifically designed for this setting.

Finally, Chapter 10 wraps up the dissertation while mentioning some interesting directions for future works in unsupervised RL via state entropy maximization. Appendix A provides all the proofs that are omitted in the previous chapters of the thesis. Appendix B provides additional experimental details for the practical methodologies presented in Part III.

CHAPTER 2

Reinforcement Learning

2.1 Introduction

In this chapter, we introduce the fundamentals of reinforcement learning as a methodology to address sequential decision-making problems. Our overview is admittedly brief and leaves out some all-important aspects of RL that are not critical to this dissertation but might interest the willing yet novice reader. We refer those to (Puterman, 2014) for a thorough coverage of sequential decision-making fundamentals, and to (Sutton & Barto, 2018) for an introduction to RL. The expert reader can instead make use of this chapter to familiarize with the notation, or directly jump to Chapter 3 to avoid a tedious read.

Contents The chapter is organized as follows. In Section 2.2, we present the fundamentals of sequential decision-making. Especially, we introduce Markovian processes as a framework to model sequential decision-making problem (Section 2.2.1), together with the accessory objects and definitions (Sections 2.2.2-2.2.4) as well as an overview of solution algorithms (Section 2.2.5). Then, Section 2.3 is devoted to reinforcement learning as a tool to address Markovian processes, for which we briefly report the main algorithmic paradigms, i.e., value-based, policy-based, and actor-critic methods.

2.2 Fundamentals of Sequential Decision-Making

In this section, we report the basics of sequential decision-making with the notation we will employ, with slight modifications here and there, in the remainder of this thesis.

2.2.1 Markov Processes

In the realm of Markov processes, we call a Controlled Markov Process (CMP) a process in which the transition dynamics can be (partially) controlled through a set of actions. Specifically, we formulate a CMP, denoted by $\mathcal{M}$, through the tuple $\mathcal{M} := \big(\mathcal{S}, \mathcal{A}, P, \mu, (\gamma \vee T)\big)$ in which

- $\mathcal{S}$ denotes a set of states, called the *state space*;
- $\mathcal{A}$ denotes a set of actions, called the *action space*;
- P is a function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$[1] such that $P(s'|s, a)$ denotes the probability of transitioning to state s' by taking action a in state s. P is called the *transition model*;
- μ denotes the initial state distribution $\mu \in \Delta(\mathcal{S})$;
- $\gamma \in [0, 1]$ is a *discount factor* such that the probability of the process ending at the next step is $1 - \gamma$;
- T is a finite time horizon $T \in \mathbb{N}$.

We indicate $(\gamma \vee T)$ to denote that the presence of a discount factor or a time horizon is usually alternative. We call a process with $\gamma < 1$ and without time horizon a *discounted* process. We call a process with $\gamma = 1$ and without time horizon an *undiscounted* process. We call a process with $T \in \mathbb{N}$ and no discount a *finite-horizon* or *episodic* process.

For any kind of Markovian process, the Markov property states that the future evolution of the process at a time step t does not depend on the realization of the process in the previous steps $t' < t$. The Markov property is evident in the definition of the transition model P, for which the transition to the next state s' only depends on the current state s and the taken action a.

A CMP can sometimes include external feedback that assigns a scalar value to an action taken in a given state. In the latter case, we call the process a Markov Decision Process (MDP, Puterman, 2014), denoted by $\mathcal{M}^R$, which is defined through the tuple $\mathcal{M}^R := \big(\mathcal{S}, \mathcal{A}, P, \mu, (\gamma \vee T), R\big)$ where

- $\big(\mathcal{S}, \mathcal{A}, P, \mu, (\gamma \vee T)\big)$ is a CMP as defined before;
- R is a function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $R(s, a) < \infty$ is the *reward* collected taking action a in state s.[2]

[1] We denote as $\Delta(\mathcal{X})$ the simplex of a space $\mathcal{X}$.

[2] The reward function can sometimes be bounded between 0 and 1, i.e., $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and it can sometimes be defined as a function of states $R(s)$ or state-action-state triplets $R(s, a, s')$.

The evolution of an MDP proceeds as follows.[3] First, an initial state $s_0 \in \mathcal{S}$ is drawn from the initial state distribution $s_0 \sim \mu$. Then, an agent interacting with the process observes the state s_0 and picks an action $a_0 \in \mathcal{A}$, therefore collecting the reward $R(s_0, a_0)$, while the MDP transitions to the next state s_1 drawn from $P(\cdot|s_0, a_0)$. If the process is episodic, this interaction is repeated for each step $t \in [T]$,[4] until the last reward $r(s_T, a_T)$ is collected and the episode ends. If the process is discounted, the interaction is repeated indefinitely, for a number of steps T_γ that is distributed according to a geometric distribution $T_\gamma - 1 \sim \text{Geo}(1-\gamma)$. If the process is undiscounted, the interaction is repeated infinitely many times. We call either *history* or *trajectory* a sequence h of state-action pairs $(s_j, a_j)_{j\in[t]}$ obtained in an interaction episode of length t. We further denote as $\mathcal{H}_t$ the set of all the histories of length t and with $\mathcal{H}$ the space of the histories of arbitrary length.

2.2.2 Policies

While describing the evolution of a Markovian process in the previous section, we have gently introduced the presence of an *agent* that takes actions interacting with the process. A policy π defines the action-selection strategy of an agent interacting with a CMP (or MDP). π consists of a sequence of decision rules $\pi := (\pi_t)_{t=0}^{\infty}$. Each of them is a map between histories $h \in \mathcal{H}_t$ and a probability distribution over actions, i.e., $\pi_t : \mathcal{H}_t \to \Delta(\mathcal{A})$, such that $\pi_t(a|h)$ defines the conditional probability of taking action $a \in \mathcal{A}$ having experienced the history $h \in \mathcal{H}_t$. We denote as Π the set of all the policies, and as Π^{D} the set of deterministic policies $\pi = (\pi_t)_{t=1}^{\infty}$ such that $\pi_t : \mathcal{H}_t \to \mathcal{A}$. We further define:

- *Non-Markovian* (NM) policies Π_{NM}, where each $\pi \in \Pi_{\text{NM}}$ collapses to a single time-invariant decision rule $\pi = (\pi, \pi, \dots)$ such that $\pi : \mathcal{H} \to \Delta(\mathcal{A})$;
- *Non-Stationary* (NS) policies Π_{NS}, where each $\pi \in \Pi_{\text{NS}}$ is defined through a sequence of Markovian decision rules $\pi = (\pi_t)_{t=0}^{\infty}$ such that $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$;
- *Markovian* (M) policies Π_{M},[5] where each $\pi \in \Pi_{\text{M}}$ collapses to a single, time-invariant, Markovian decision rule $\pi = (\pi, \pi, \dots)$ such that $\pi : \mathcal{S} \to \Delta(\mathcal{A})$.

The pairing of a CMP $\mathcal{M}$ and a policy π induces a Markov Chain (MC, Levin & Peres, 2017) with state-to-state transition model $P^\pi(s'|s) = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s,a)$. Similarly, the pairing of an MDP $\mathcal{M}^R$ and a policy π induces a Markov reward process, which is a Markov chain with rewards.

2.2.3 State Distributions

A policy π interacting with a CMP (or MDP) induces a distribution over the states of the CMP (MDP). Especially, we denote as $d_t^\pi(s) := Pr(s_t = s \mid \pi)$ the t-step state distribution induced by π, which we can also express through the flow equation

$$d_t^\pi(s) = \sum_{s'\in\mathcal{S}} d_{t-1}^\pi(s') \sum_{a'\in\mathcal{A}} \pi(a'|s')P(s|s',a').$$

[3] The evolution of a CMP is identical to the one of an MDP in which the reward does not exist.

[4] We denote with $[T]$ the set of integers $\{0, \dots, T-1\}$.

[5] Note that we will sometimes denote as Π the set of Markovian policies, especially in the contexts where the distinction between non-Markovian, non-stationary, Markovian policy is not crucial.

Under common assumptions (Puterman, 2014), a Markovian policy $\pi \in \Pi_{\mathrm{M}}$ interacting with an undiscounted process $\mathcal{M}$ induces a *stationary state distribution* $d^{\pi}_{\infty}(s) := \lim_{t\to\infty} d^{\pi}_{t}(s)$, which is also the steady state of the MC induced by π over $\mathcal{M}$.

Similarly, a policy π interacting with a discounted process induces a *discounted state distribution* $d^{\pi}_{\gamma}(s) := \sum_{t=0}^{\infty} \gamma^t d^{\pi}_{t}(s)$, which can be also expressed through the recursive equation

$$d^{\pi}_{\gamma}(s) = (1-\gamma)\mu(s) + \gamma \sum_{s'\in\mathcal{S}} d^{\pi}_{\gamma}(s') \sum_{a'\in\mathcal{A}} \pi(a'|s')P(s|s',a').$$

Finally, a policy π interacting with an episodic process over T steps induces a *marginal state distribution* $d^{\pi}_{T}(s) := \frac{1}{T}\sum_{t=0}^{T-1} d^{\pi}_{t}(s)$, in which $d^{\pi}_{0}(s) = \mu(s)$.[6]

2.2.4 Objectives, Value Functions, and Bellman Equations

The goal of an agent interacting with an MDP is expressed as the sum of the rewards collected in the *long term*. By specifying long term, we mean that the agent is far-sighted, i.e., he does not care for the immediate reward alone, but he looks at the rewards he can collect in the future as well. This intuition can be mathematically expressed through the concept of *value function*.

For an episodic MDP, we define the value $V^{\pi}_{t}(s)$ of being in state s at the time step t while following the policy π as

$$V^{\pi}_{t}(s) := \mathop{\mathbb{E}}_{\pi}\left[\sum_{t'=t}^{T-1} R(s_{t'}, a_{t'}) \,\middle|\, s_t = s\right]. \tag{2.1}$$

Similarly, we can define the value $Q^{\pi}_{h}(s,a)$ of the state-action pair s, a at step t while following the policy π as $Q^{\pi}_{t}(s,a) := \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^{\pi}_{t+1}(s')]$. Then, we can define the objective function for an episodic MDP $\mathcal{M}^R$ as

$$\mathcal{J}_{\mathcal{M}^R}(\pi) := \mathop{\mathbb{E}}_{s\sim\mu}\left[V^{\pi}_{0}(s)\right]. \tag{2.2}$$

For a discounted MDP, we analogously define the value $V^{\pi}_{\gamma}(s)$ of being in state s while following the policy π as

$$V^{\pi}_{\gamma}(s) := \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right], \tag{2.3}$$

and the state-action value function $Q^{\pi}_{\gamma}(s,a) := \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^{\pi}_{\gamma}(s')]$. Notably, the value function $V^{\pi}_{\gamma}(s)$ at state s depends to the value function $V^{\pi}_{\gamma}(s')$ of the subsequent state s' through $V^{\pi}_{\gamma}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi}_{\gamma}(s,a)]$. Indeed, we can write a recursive formulation for the state value function

$$V^{\pi}_{\gamma}(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[R(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}\left[V^{\pi}_{\gamma}(s')\right]\right], \tag{2.4}$$

[6] Note that we will sometimes denote the state distribution simply as d^{π}, especially when it is clear from the context whether we refer to the stationary, discounted, or marginal state distribution.

which takes the name of *Bellman expectation equation* (Bellman, 1957). Similarly, we can define a corresponding *Bellman expectation operator* T^π (Bellman, 1957) that applies the recursive relation (2.4) to any function $f : \mathcal{S} \to \mathbb{R}$, i.e.,[7]

$$\left(T^\pi f\right)(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[R(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}\left[f(s')\right]\right]. \tag{2.5}$$

Finally, we define the objective function for a discounted MDP $\mathcal{M}^R$ as[8][9]

$$\mathcal{J}_{\mathcal{M}^R}(\pi) := \mathop{\mathbb{E}}_{s\sim\mu}\left[V^\pi_\gamma(s)\right]. \tag{2.6}$$

2.2.5 Solving Markov Decision Processes

Having introduced the objective function for an agent interacting with an MDP, we can now discuss how to *solve* an MDP, i.e., to find a policy that maximizes the objective function. This can be translated to the optimization problem[10]

$$\max_{\pi\in\Pi} \; \mathcal{J}_{\mathcal{M}^R}(\pi). \tag{2.7}$$

Puterman (2014) guarantees the existence of a deterministic Markovian policy π^* such that $\pi^* \in \arg\max_{\pi\in\Pi} J_{\mathcal{M}^R}(\pi)$, which we call the optimal policy. How can we extract the optimal policy π^* having full knowledge of the MDP $\mathcal{M}^R$? Is the optimization problem (2.7) tractable in the first place? In this section, we revise three alternative approaches.

MDPs as Linear Programs

It can be shown that the problem (2.7) can be translated into the following linear program (Puterman, 2014).

$$\begin{aligned}
&\underset{\substack{V\in\mathbb{R}^{|\mathcal{S}|}\\ \rho\in\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}}}{\text{maximize}} \quad \mathop{\mathbb{E}}_{s\sim\mu}\left[V(s)\right] \\
&\text{subject to} \quad V(s) = \mathop{\mathbb{E}}_{a\sim\rho(\cdot|s)}\left[R(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}\left[V(s)\right]\right], \quad \forall s \in \mathcal{S}, \\
&\qquad\qquad \sum_{a\in\mathcal{A}} \rho(a|s) = 1, \quad \forall s \in \mathcal{S}, \qquad \rho(a|s) > 0, \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.
\end{aligned}$$

The solution of the former program is the value function of the optimal policy $V^*(s) := V^{\pi^*}(s)$, which is also called the optimal value function. Since solving an MDP is equivalent to solving a linear program with $|\mathcal{S}| + |\mathcal{S}||\mathcal{A}|$ variables, $|\mathcal{S}||\mathcal{A}|$ inequality constraints,

[7] Similar Bellman expectation equation and operator can be defined for the state-action value function $Q^\pi_\gamma(s,a)$.

[8] Note that the following definition overloads the notation of the objective (2.2), but it will be clear from the context whether we refer to the discounted objective or the episodic objective.

[9] All of the reported definitions can be extended to undiscounted MDPs. However, we omit these definitions, which can be found in (Puterman, 2014) as they are not crucial for the remainder of the thesis.

[10] For the ease of presentation, we will focus on the discounted MDP setting in this section.

and $|\mathcal{S}| + |\mathcal{S}||\mathcal{A}|$ equality constraints, we can conclude that solving an MDP is tractable in general (Papadimitriou & Tsitsiklis, 1987). Notably, recovering the optimal policy π^* from the value V^* requires some additional calculations. Instead, we can define an alternative dual program formulation (Puterman, 2014; De Farias & Van Roy, 2003) from which the optimal policy can be directly recovered.

$$\begin{aligned} \underset{\omega \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}}{\text{maximize}} \quad & \mathop{\mathbb{E}}_{(s,a)\sim\omega} \big[R(s,a)\big] \\ \text{subject to} \quad & \sum_{a\in\mathcal{A}} \omega(s,a) = \sum_{s'\in\mathcal{S}} \sum_{a'\in\mathcal{A}} \omega(s',a') P(s|s',a'), \quad \forall s \in \mathcal{S}, \\ & \sum_{a\in\mathcal{A}} \omega(s,a) = 1, \quad \forall s \in \mathcal{S}, \qquad \omega(s,a) > 0, \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}. \end{aligned}$$

The latter is a linear program with $|\mathcal{S}||\mathcal{A}|$ variables, $|\mathcal{S}||\mathcal{A}|$ inequality constraints, and $|\mathcal{S}| + |\mathcal{S}||\mathcal{A}|$ equality constraints. Let $\omega^*(s,a)$ the solution of the program, we can recover an optimal policy as

$$\pi^*(a|s) = \frac{\omega^*(s,a)}{\sum_{a'\in\mathcal{A}} \omega^*(s,a')}, \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

Dynamic Programming

Dynamic Programming (DP, Bellman, 1957) provides a flexible alternative to the linear program formulation. Especially, it can be easily incorporated into sample-based approaches, and it can also be extended to work for MDPs of infinite size, as we will see in the next sections.

The first dynamic programming algorithm that we introduce is *policy iteration* (Sutton & Barto, 2018). Policy iteration is based on a repeated iteration of a *policy evaluation* step followed by a *policy improvement* step. The former computes the value function of the current policy, which is a randomly initialized policy at first. Especially, it exploits a classic result stating that the value function $V^\pi(s)$ of a policy π is the unique fixed point of the Bellman expectation operator T^π. The policy improvement step instead exploits the policy improvement theorem (Sutton & Barto, 2018) to provably improve the current policy through a *greedyfication*. We report the pseudocode of policy iteration below.

Algorithm 1 Policy Iteration

Input: Randomly initialized policy π_0
for $i = 0, \dots,$ until convergence **do**
  randomly initialize $V_0(s)$, $\forall s \in \mathcal{S}$
  for $j = 0, \dots,$ until convergence **do**
    compute $V_{j+1}(s) = (T^\pi V_j)(s)$, $\forall s \in \mathcal{S}$
  end for
  compute $Q^{\pi_i}(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[V_j(s')]$, $\forall (s,a) \in \mathcal{S}\times\mathcal{A}$
  compute the greedy policy $\pi_{i+1}(s) \in \arg\max_{a\in\mathcal{A}} Q^{\pi_i}(s,a)$, $\forall s \in \mathcal{S}$
end for
Output: Optimal policy π_i

The inner loop for policy evaluation in the policy iteration algorithm can result quite expensive in practice. Especially, if the goal is to find an optimal policy for the MDP $\mathcal{M}^R$, there is little additional benefit in precise evaluations of the policies π_i. The *value iteration* algorithm (Sutton & Barto, 2018) effectively applies a unique sweep of the Bellman expectation operator T^π and then proceeds to the greedyfication step right away. This is made possible by the *Bellman optimality equation* (Bellman, 1957)

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \in P(\cdot|s,a)} \left[V^*(s') \right] \right\}, \tag{2.8}$$

and its corresponding *Bellman optimality operator* T^* (Bellman, 1957) that applies the recursive relation to any function $f : \mathcal{S} \to \mathbb{R}$, i.e.,

$$\left(T^* f\right)(s) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \in P(\cdot|s,a)} \left[f(s') \right] \right\}. \tag{2.9}$$

The value iteration algorithm exploits the fact that the optimal value function V^* is the unique fixed point of the Bellman optimality operator. Hence, it repeatedly applies T^* to a randomly initialized function to obtain V^*. We report the pseudocode of value iteration below.

Algorithm 2 Value Iteration

Input: Randomly initialized function $f_0 : \mathcal{S} \to \mathbb{R}$
for $i = 0, \ldots,$ until convergence **do**
  compute $f_{i+1} = (T^* f_i)(s),\ \forall s \in \mathcal{S}$
end for
compute $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma\, \mathbb{E}_{s' \in P(\cdot|s,a)}[f_i(s')] \right\}$
Output: Optimal policy π^*

2.3 Reinforcement Learning Algorithms

All the methods to solve MDPs that we have presented in the previous section crucially rely on the full knowledge of the transition model P. However, in several relevant decision-making problems the transition model is unknown, such as the laws governing the human body in a medical treatment application, or too complex to be represented, e.g., the game of chess. Here is where reinforcement learning comes into play. RL allows to *learn* an (approximately) optimal policy from sampled interactions with the MDP. In the following sections, we briefly present some of the most common RL algorithms, while we leave to textbooks (e.g., Szepesvári, 2010; Sutton & Barto, 2018) a more detailed overview.

2.3.1 Value-Based Methods

The *value-based* methods for RL are essentially inspired by dynamic programming, for which they provide sample-based extensions. Just as the dynamic programming methods compute value functions from which the optimal policy can be recovered, in value-based RL methods the value function is the learning target.

Here we present *Q-learning* (Watkins & Dayan, 1992), which is a popular sampled-based version of the value iteration algorithm.

Algorithm 3 Q-learning

Input: learning rate α, behavioral policy π_b
randomly initialize $Q_0(s,a),\ \forall (s,a) \in \mathcal{S} \times \mathcal{A}$
sample the initial state $s_0 \sim \mu$
for $t = 0, \dots,$ until convergence **do**
  perform the action $a_t \sim \pi_b(\cdot|s_t)$
  collect the reward $R(s_t, a_t)$ and observe $s_{t+1} \sim P(\cdot|s_t, a_t)$
  compute the update

$$Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha\Big(R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q_t(s_{t+1}, a')\Big)$$

  (optional) update the behavioral policy π_b
end for
compute $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} Q_t(s,a),\ \forall s \in \mathcal{S}$
Output: Optimal policy π^*

The algorithm is centered around the so-called Q-learning update, in which the current estimate of the value function Q_t is updated with the fresh information coming from the latest sampled interaction (s_t, a_t, s_{t+1}), while the learning rate α controls the combination between current estimate and new information. Crucially, the value function Q_t does not refer to any policy, as it is estimated taking the maximum over the actions at the next step. This allows the algorithm to work off-policy, which means that the data are collected through a behavioral policy π_b that can be different from the greedy policy for the current estimate Q_t. Under standard conditions on the learning rate α and the behavioral policy π_b, the Q-learning algorithm is guaranteed to converge to the optimal value function Q^* asymptotically and, as a consequence, to the optimal policy π^*.

Whereas Q-learning provides a flexible procedure to solve MDPs from sampled interactions, its implementation still requires storing a vector of $|\mathcal{S}||\mathcal{A}|$ values to represent the Q function. This becomes clearly impractical when the MDP has infinitely many states. In the latter case, we can turn to *function approximation* (Sutton & Barto, 2018) by expressing the Q function through a linear combination $Q_\omega(s,a) = \sum_{i=1}^N \omega_i f_{\theta i}(s,a)$ of parametric functions $f_{\theta i}$. Then, we can compute the update over the parameters ω_i, θ_i by minimizing the mean squared error $\sum_{(s,a,s') \in \mathcal{D}} (Q_\omega(s,a) - R(s,a) - \gamma \max_{a' \in \mathcal{A}} Q_\omega(s', a'))^2$, in which $\mathcal{D}$ is a dataset of interactions. In the function approximation setting, a version of the Q-learning algorithm in which the Q function is approximated through a deep neural network, which is called a *Deep Q-Network* (DQN), has achieved outstanding results in complex domains (Mnih et al., 2015).

2.3.2 Policy-Based Methods

In the last section, we have seen how the Q-learning algorithm can be combined with function approximation to address MDPs with large state spaces. Nevertheless, the Q-learning procedure still requires performing the maximization $\max_{a \in \mathcal{A}} Q(s,a)$, which can

be cumbersome when the number of actions is also large, and the Q function is expressed through a non-linear approximation. How can we address domains with infinitely many states and actions, such as continuous control problems, through RL?

The family of Policy Optimization (PO, Deisenroth et al., 2013) methods provides an answer to the latter question. In PO, we turn to a policy-based perspective on the MDP objective function. Let us consider an episodic MDP setting with horizon T, we can define the probability of experiencing a trajectory $h \in \mathcal{H}_T$ by deploying the policy π as

$$p_\pi(h) = \mu(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t), \tag{2.10}$$

from which we can rewrite the objective (2.2) as

$$\mathcal{J}_{\mathcal{M}^R}(\pi) = \mathop{\mathbb{E}}_{s\sim\mu}\big[V_0^\pi(s)\big] = \mathop{\mathbb{E}}_{h\sim p_\pi}\big[R(h)\big], \tag{2.11}$$

where we denote $R(h) = \sum_{t=0}^{T-1} R(s_t, a_t)$ with a slight overload of notation. Then, we can look at (2.11) as the problem of finding the policy inducing high-rewarding trajectories in expectation. However, when the MDP has infinitely many states and actions, we cannot even represent the policy π through $|\mathcal{S}||\mathcal{A}|$ real numbers. Instead, we can represent the policy space with a set of parametric policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^q\}$. PO is about solving the problem

$$\max_{\pi_{\boldsymbol{\theta}}\in\Pi_\Theta} \mathcal{J}_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}})$$

through a direct search in the parametric policy space, e.g., via gradient ascent. It is straightforward to derive (see Peters & Schaal, 2008) the gradient of $\mathcal{J}_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}})$ w.r.t. the policy parameters as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{h\sim p_{\pi_{\boldsymbol{\theta}}}}\big[R(h)\big] \\
&= \nabla_{\boldsymbol{\theta}} \int_{\mathcal{H}_T} p_{\pi_{\boldsymbol{\theta}}}(h) R(h)\,\mathrm{d}h \\
&= \int_{\mathcal{H}_T} p_{\pi_{\boldsymbol{\theta}}}(h) \nabla_{\boldsymbol{\theta}} \log p_{\pi_{\boldsymbol{\theta}}}(h) R(h)\,\mathrm{d}h \\
&= \int_{\mathcal{H}_T} p_{\pi_{\boldsymbol{\theta}}}(h) \Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\Big)\Big(\sum_{t=0}^{T-1} R(s_t, a_t)\Big)\,\mathrm{d}h \\
&= \mathop{\mathbb{E}}_{h\sim p_{\pi_{\boldsymbol{\theta}}}}\left[\Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\Big)\Big(\sum_{t=0}^{T-1} R(s_t, a_t)\Big)\right].
\end{aligned}$$

By noting the conditional independence between the action selected at step $\bar{t}$ and the reward collected at the previous steps $\sum_{t=0}^{\bar{t}-1} R(s_t, a_t)$, we have

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}}) = \mathop{\mathbb{E}}_{h\sim p_{\pi_{\boldsymbol{\theta}}}}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) Q^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)\right], \tag{2.12}$$

Algorithm 4 G(PO)MDP

Input: learning rate α, parameter space Θ
randomly initialize $\boldsymbol{\theta}_0 \in \Theta$
for $i = 0, \dots,$ until convergence **do**
draw a batch of trajectories $\{h_j\}_{j=1}^{N}$ with $\pi_{\boldsymbol{\theta}i}$
estimate the policy gradient as

$$\widehat{\nabla}_{\boldsymbol{\theta}i} J_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}i}) = \sum_{j=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}i} \log \pi_{\boldsymbol{\theta}i}(a_t^{(j)}|s_t^{(j)}) \left(\sum_{k=t}^{T-1} R(s_k^{(j)}, a_k^{(j)}) \right)$$

update the parameters $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha \widehat{\nabla}_{\boldsymbol{\theta}i} J_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}i})$
end for
Output: (Approximately) optimal policy $\pi_{\boldsymbol{\theta}i}$

which is known as the *Policy Gradient Theorem* (Sutton et al., 1999). We report below the pseudocode of an algorithm, called *G(PO)MDP* (Baxter & Bartlett, 2001), which implements PO via gradient ascent through Monte-Carlo estimates of (2.12).

While being remarkably easy to implement, G(PO)MDP is an effective algorithm for several continuous control tasks. However, it can suffer from significant variance in practice, due to the Monte-Carlo estimate of the policy gradient (2.12). Several mechanisms have been later introduced to reduce the variance of PO methods, such as through an additive baseline for the gradient estimate (Peters & Schaal, 2008) and sample reuse through importance sampling techniques (Papini et al., 2018; Metelli et al., 2018b) among others.

Another approach to improve the stability of PO methods is to consider *conservative* updates in the gradient direction. Of special consideration in this regard is *Trust-Region Policy Optimization* (TRPO, Schulman et al., 2015). TRPO, which is inspired by previous works on conservative updates for policy iteration (Kakade & Langford, 2002; Pirotta et al., 2013), computes the next policy parameters $\boldsymbol{\theta}'$ by solving a local optimization problem within a trust region around the current parameters $\boldsymbol{\theta}$, i.e., $\max_{s \in \mathcal{S}} \mathrm{KL}(\pi_{\boldsymbol{\theta}} || \pi_{\boldsymbol{\theta}'}) \leq \lambda$, where KL is the Kullback-Leibler divergence and λ is a positive constant.

2.3.3 Actor-Critic Methods

Finally, another family of algorithms, which are called *actor-critic*, has been designed to combine the benefit of value-based methods and policy-based methods (Peters & Schaal, 2008). Indeed, actor-critic methods support learning policies over continuous actions spaces as the policy-based methods, while achieving the reduced variance of value-based methods in practice.

One can see actor-critic as an extension of policy-based methods in which the policy gradient (2.12) estimate is computed through a Q function that is learned from sampled interactions as well, or as an extension of value-based methods (e.g., policy iteration) in which the policy greedyfication is substituted by several steps in the gradient direction.

Within the actor-critic methods, the Soft Actor-Critic (SAC, Haarnoja et al., 2018) algorithm resulted in remarkable empirical success by optimizing an entropy-regularized

objective in place of the standard objective (2.11).

CHAPTER 3

Unsupervised Reinforcement Learning

3.1 Introduction

In the previous chapter, we have presented the RL approach (Sutton & Barto, 2018) to address sequential decision-making problems in challenging domains. In the last decade, the RL approach led to outstanding results in remarkable tasks, such as Atari games (Mnih et al., 2015), Go (Silver et al., 2016), Dota 2 (Berner et al., 2019), and dexterous manipulation (Andrychowicz et al., 2020).

Nonetheless, the learning process usually requires a considerable amount of human supervision to accomplish these feats, especially in the form of a reward function that the agent can maximize. In principle, the reward function is inherent to the environment and perfectly encodes the learning task. In practice, the reward is usually hand-crafted, and designing it to make the agent learn a desirable behavior is often a huge challenge. This poses a serious roadblock on the way of autonomous learning, as any task requires a costly and specific formulation, while the synergy between solving one RL problem and another is very limited, as most of the successful approaches are designed to learn from scratch.

The unsupervised RL formulation (Laskin et al., 2021) recently emerged as a powerful framework to address this crucial limitation. In this framework, which was originally envisioned in (Hazan et al., 2019; Mutti & Restelli, 2020), the learning process takes place in two sequential phases. In the first *unsupervised fine-tuning* phase, the agent interacts with a CMP with the goal of incorporating the collected knowledge into a pre-trained model,

which can be alternatively a representation of the transition dynamics, an abstract state representation, a policy or a set of policies, a dataset of interactions. The purpose of the pre-trained model is to improve the efficiency of the subsequent phase, called *supervised fine-tuning*, in which the agent is tasked with a (previously unknown) reward function in the same CMP. Crucially, the agent can exploit the pre-trained model to solve a wide range of tasks in the same CMP, instead of addressing each task from scratch, which have been demonstrated to yield significant empirical (e.g., Laskin et al., 2021) and theoretical (Jin et al., 2020; Xie et al., 2021) benefits.

In this thesis, we will focus on a specific approach for unsupervised pre-training, in which the agent's policy is pre-trained to maximize the entropy it induces over the states of the environment (Hazan et al., 2019). Before going through the details of state entropy maximization in the next chapters, in the following sections we provide an overview of unsupervised RL, its objective functions, and common approaches.

Contents The chapter is organized as follows. In Section 3.2, we present the unsupervised RL framework, and we introduce a general mathematical formulation for the unsupervised pre-training objective (Section 3.2.1) and for the supervised fine-tuning objective (Section 3.2.2). Then, in Section 3.3, we provide an overview of the unsupervised RL literature, providing a list of the common approaches, as well as an informal taxonomy and categorization. Especially, in Section 3.3.5, we focus on the unsupervised pre-training of exploration policies, which is central in the remainder of this thesis.

3.2 Problem Formulation

We now present the problem formulation of unsupervised RL (see Laskin et al., 2021). As depicted in Figure 3.1, the learning process involves an agent interacting with a CMP $\mathcal{M}$ in two distinct phases.

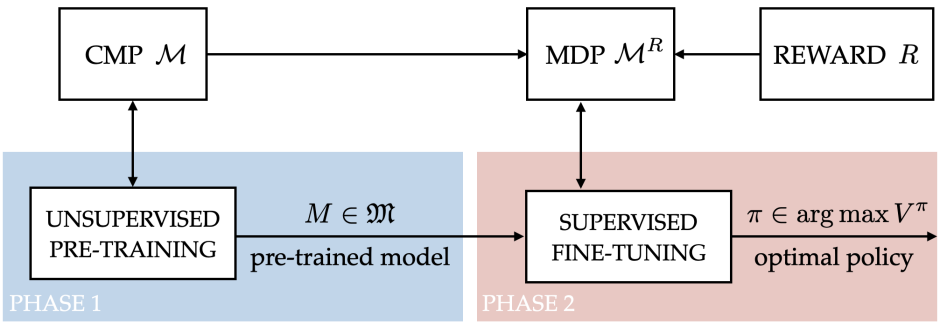


Figure 3.1: *Unsupervised reinforcement learning.*

In the first phase, the CMP $\mathcal{M}$ does not include any reward function. Instead, the agent incorporates the knowledge it collects on the CMP $\mathcal{M}$ by pre-training M, which can alternatively be a model of the transition dynamics, states representations, a policy space or just a single policy, a dataset of interactions. Since there is no externally defined objective in this phase, we call it the *unsupervised pre-training* phase (see Section 3.2.1).

In the second phase, the agent is provided with a reward function R on the same CMP $\mathcal{M}$, thus producing a standard MDP $\mathcal{M}^R$. The agent then leverages the pre-trained model M in order to find a reward-maximizing policy for $\mathcal{M}^R$. The latter task can be sometimes completed through simple planning with the pre-trained model M (e.g., when M is a good representation of the transition dynamics), or by taking further interactions with the MDP (e.g., when M is an exploratory policy). We call this second phase the *supervised fine-tuning* phase (see Section 3.2.2).

The aim of unsupervised RL is to improve the efficiency of the supervised fine-tuning, where the sampled interactions are assumed to be expensive, leveraging the knowledge acquired in the unsupervised pre-training phase, where the sampled interactions are generally assumed to be cheap. Crucially, one can use the same pre-trained model M to tackle several supervised fine-tuning tasks defined on $\mathcal{M}$. Thus, even if the unsupervised pre-training can be computationally and statistically inefficient, any benefit it provides over RL from scratch can propagate across a wide range of tasks.

A notable generalization of the unsupervised RL formulation considers unsupervised pre-training of a single model M over multiple CMPs, such that the agent can be tasked with any reward in any CMP during the fine-tuning phase (Parisi et al., 2021; Mutti et al., 2022e,c; Ye et al., 2022). However, in this thesis we will mostly focus on the single-environment setting, with the exception of Chapter 9. In Section 3.2.1, we present a formulation of the unsupervised pre-training objective, whereas in Section 3.2.2 we present a formulation of the supervised fine-tuning objective.

3.2.1 Unsupervised Pre-Training

Let us denote as $\mathfrak{M}$ a class of models to be pre-trained, we can (informally) define the unsupervised pre-training objective as follows.

Unsupervised Pre-Training Objective

$$\max_{M \in \mathfrak{M}} \; \mathcal{F}_{\text{pre-train}}(M, \mathcal{M}), \tag{3.1}$$

in which $\mathcal{F}_{\text{pre-train}}$ is a function that maps a pre-trained model $M \in \mathfrak{M}$ and a controlled Markov process $\mathcal{M}$ to reals, i.e., $\mathcal{F}_{\text{pre-train}} : \mathfrak{M} \times \mathcal{M} \to \mathbb{R}$. In a typical example, which we will further expand throughout this thesis, the class of models $\mathfrak{M}$ is the space of Markovian policies Π, a model M is a policy $\pi \in \Pi$, and the function $\mathcal{F}_{\text{pre-train}}$ is the entropy H of the state distribution d^π induced by π over the CMP $\mathcal{M}$ (Hazan et al., 2019). Other than the pre-training of policies, a wide range of model classes $\mathfrak{M}$ and pre-training functions $\mathcal{F}_{\text{pre-train}}$ have been considered in the literature, for which we provide an (incomplete) overview in Section 3.3. Note that the evaluation of the pre-training function $\mathcal{F}_{\text{pre-train}}$ can be arbitrarily complicated, such as including an inner planning procedure (e.g., Jin et al., 2020). Moreover, the unsupervised pre-training objective can be alternatively defined over a class of CMPs $\boldsymbol{\mathcal{M}}$ instead of a single $\mathcal{M}$, as we will show in Chapter 9. As a final remark on the unsupervised pre-training phase, we note that it does include a supervision of the learning process through the surrogate function $\mathcal{F}_{\text{pre-train}}$, but the process is *unsupervised* w.r.t. the objective of the fine-tuning phase, which we will introduce in the next section.

3.2.2 Supervised Fine-Tuning

In the supervised fine-tuning phase, we assume the existence of a reward function R in the CMP $\mathcal{M}$, such that the objective function can be (informally) defined over the corresponding MDP $\mathcal{M}^R$ as follows.

Supervised Fine-Tuning Objective

$$\max_{\pi \in \Pi} \; \mathcal{J}_{\mathcal{M}^R}(\pi) \qquad \text{given } M \in \mathfrak{M}, \tag{3.2}$$

where the objective is akin to the standard RL objective in $\mathcal{M}^R$, with the addition of the model M pre-trained a priori according to (3.1). In the aforementioned typical example, M can be a pre-trained policy π that acts as the initial policy of a standard RL algorithm, as opposed to a random initialization without pre-training. Whereas the unsupervised pre-training does not fundamentally change the nature of the objective (3.2), the benefit brought by the pre-trained model has to be found in the comparison of the *regret* suffered by a learning algorithm with and without pre-training. Especially, we can define the regret suffered by a learning algorithm $\mathfrak{A}$ in K episodes collected from the MDP $\mathcal{M}^R$ as

$$\mathrm{Reg}_K(\mathcal{M}^R, \mathfrak{A}) = \mathop{\mathbb{E}}_{s \sim \mu} \left[\sum_{k=0}^{K} V^*(s) - V^{\pi_k}(s) \right],$$

where V^* is the optimal value function in $\mathcal{M}^R$, V^{π_k} is the value function in $\mathcal{M}^R$ of the policy π_k, which is deployed by the algorithm $\mathfrak{A}$ at the episode k. Then, the benefit of the pre-training can be measured through the relative difference between the regret of $\mathfrak{A}$ when it is given the pre-trained model M and when it is randomly initialized (Ye et al., 2022). Crucially, the regret suffered from an RL algorithm learning from scratch cannot be smaller than $\Omega(\sqrt{|\mathcal{S}||\mathcal{A}|HK})$ (Jaksch et al., 2010). Instead, the regret can be smaller than a constant ϵ if the pre-trained model M allows for zero-shot near-optimal planning in every $\mathcal{M}^R$, and it can still be significantly reduced in its multiplicative factors through carefully pre-trained policies (Ye et al., 2022).

3.3 Literature Overview

We now provide an overview of the literature related to unsupervised RL. While the unsupervised RL field is relatively recent, the body of works is rapidly growing, and the intent of this section is not to give a comprehensive list of all the relevant works, but rather a roadmap to the kinds of approaches that have been considered in the literature. Table 3.1 summarizes the approaches we will mention in this section, together with a taxonomy and an informal categorization.

A first important dichotomy to consider is the nature of the works, which are generally split between theoretical contributions and methodologies of practical interest (see column "Guarantees" in Table 3.1). The former are dedicated to demonstrating the sample efficiency of the pre-training phase and the provable benefits it brings to the subsequent fine-tuning phase. Those theoretical guarantees are mostly devoted to tabular settings, or to continuous settings with strong structural hypotheses, which limit their practical upside.

Approach	Pre-training	Fine-tuning	Guarantees	Practical	Multi CMPs
Reward-free RL (Jin et al., 2020; Ménard et al., 2021) and others	transition model	non-interactive	✓	✗	✗
Task-agnostic RL (Zhang et al., 2020b) and others	transition model	non-interactive	✓	✗	✗
Forward-backward representation (Touati & Ollivier, 2021)	transition model	non-interactive	✓	✓	✗
Systematic generalization (Mutti et al., 2022c)	transition model	non-interactive	✓	✗	✓
World models (Ha & Schmidhuber, 2018)	transition model	interactive	✗	✓	✗
Curiosity (Schmidhuber, 1991; Pathak et al., 2017) and others	transition model	interactive	✗	✓	✗
Low-rank MDPs (Agarwal et al., 2020; Modi et al., 2021)	representations	non-interactive	✓	✗	✗
RND (Burda et al., 2019b)	representations	interactive	✗	✓	✗
Contrastive representations (Laskin et al., 2020) and others	representations	interactive	✗	✓	✗
Policy space compression (Mutti et al., 2022d)	policy space	non-interactive	✓	✗	✗
Policy collection-elimination (Ye et al., 2022)	policy space	interactive	✓	✗	✓
Skills discovery (Eysenbach et al., 2019) and others	policy space	interactive	✗	✓	✗
Policy fine-tuning (Xie et al., 2021)	policy	interactive	✓	✗	✗
State entropy maximization (Hazan et al., 2019) and others	policy	interactive	✗	✓	✓
Fine-tuning mechanisms (Campos et al., 2021; Pislar et al., 2021)	policy	interactive	✗	✓	✗
Reward-free data collection (Kaufmann et al., 2021) and others	dataset	non-interactive	✓	✗	✗
ExORL (Yarats et al., 2022)	dataset	non-interactive	✗	✓	✗
Explore2Offline (Lambert et al., 2022)	dataset	non-interactive	✗	✓	✗
Count-based (Bellemare et al., 2016) and others	dataset	interactive	✗	✓	✗

Table 3.1: *Overview of the literature in unsupervised RL. The **Approach** column reports either a specific work or a stream of similar works. The **Pre-training** column reports the model to be pre-trained. The **Fine-tuning** column reports whether the approach can perform zero-shot fine-tuning (non-interactive) or it requires further interactions with the MDP (interactive). The **Guarantees** column reports whether the works provide sub-optimality or regret guarantees for the fine-tuning phase. The column **Practical** reports whether the works provide a practical methodology for pre-training in continuous and high-dimensional environments. The **Multi CMPs** column reports whether the works address pre-training over a set of CMPs or a single CMP.*

Instead, the empirical works are dedicated to providing methodologies for unsupervised pre-training in continuous and/or high-dimensional domains, often relying on heuristics and implementation tricks that trade theoretical guarantees for empirical success.

A second significant dichotomy is whether the pre-trained model allows for zero-shot adaptation or planning in the supervised fine-tuning phase, or it requires further interactions to be collected from the environment (see column "Fine-tuning" in Table 3.1). Clearly, the *non-interactive* approaches are generally more demanding in the pre-training phase, often limiting their tractability in practice. Instead, the *interactive* approaches include two actual learning processes, which complicate their theoretical analysis.

Another important categorization relates the unsupervised RL approaches with the model they aim to pre-train in the unsupervised phase, and then transfer to the subsequent supervised phase (see column "Pre-training" in Table 3.1). In the following Sections 3.3.1-3.3.5 we provide a list of some common classes of models for pre-training.

Finally, a few approaches in unsupervised RL have considered pre-training over a class of CMPs instead of a single CMP (see column "Multi CMPs" in Table 3.1). Those are mostly addressing a policy or a policy space as a learning target (Mutti et al., 2022e; Ye et al., 2022), with the notable exception of (Mutti et al., 2022c), which adopts a causal viewpoint to pre-train a transition model that can be transferred across multiple CMPs.

3.3.1 Pre-Training of Transition Models

Several works have considered the estimation of the transition model of the CMPs as a valuable target for unsupervised pre-training. Indeed, a good estimate of the transition model allows for efficient planning once the reward is revealed in the supervised fine-tuning phase. In this setting, the model class comprises the functions that map a state-action pair to a next state distribution, i.e., $\mathfrak{M}_p = \{p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})\}$. Typically, the pre-training objective is either a distance minimization between the pre-trained model and the true model

$$\min_{p \in \mathfrak{M}_p} \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \|P(s'|s,a) - p(s'|s,a)\|_2 \tag{3.3}$$

or a sub-optimality minimization between the optimal planning policy of the pre-trained model and the true model for a worst-case reward

$$\min_{p \in \mathfrak{M}_p} \sup_{R \in \{\mathcal{S} \times \mathcal{A} \to [0,1]\}} \mathbb{E}_{s \sim \mu} \left[|V_P^*(s) - V_p^*(s)| \right]. \tag{3.4}$$

The latter (3.4) have been preferred by theoretically-driven approaches, such as reward-free RL (Jin et al., 2020; Kaufmann et al., 2021; Ménard et al., 2021; Zhang et al., 2021c) and task-agnostic[1] RL (Zhang et al., 2020b), but it also underlies some interesting practical methods (Touati & Ollivier, 2021).

Variations of the former (3.3) have been considered both in theoretical works (Tarbouriech et al., 2020) and empirical works that address transition model representations in complex

[1] Similar to reward-free RL objective 3.4, but the reward function is oblivious to the pre-training process rather than worst-case.

domains (e.g., Ha & Schmidhuber, 2018). Surrogate objectives can be also employed to reduce the estimation error on the model learning, such as curiosity-driven bonuses (Schmidhuber, 1991; Pathak et al., 2017; Burda et al., 2019a) that specifically assign high values to state-action pairs inducing unexpected transitions.

3.3.2 Pre-Training of Representations

When dealing with RL tasks in MDPs with rich observations (such as images), it is well-known that good state (or state-action) representations are paramount to efficiently learn the optimal policy. Thus, the unsupervised pre-training phase can focus on learning task-agnostic representations that allow for efficient supervised fine-tuning downstream. In this setting, the model class typically comprises the set of mapping between high-dimensional (s, a) pairs to lower-dimensional representations $\mathfrak{M}_\phi = \{\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d\}$.

The works in (Misra et al., 2020; Agarwal et al., 2020; Modi et al., 2021) learn state-action representations $\phi(s, a)$ that allow for a lower-rank decomposition of the CMP transition model $P(s'|s, a) = \phi(s, a) \cdot \psi(s'), \forall s \in \mathcal{S}, a \in \mathcal{A}, s' \in \mathcal{S}$ under convenient structural assumptions. Given these representations, the agent can directly plan on the corresponding abstract MDP in the supervised fine-tuning phase, getting the optimal policy without further interactions. Especially, sample complexity results for the pre-training of representations, as well as sub-optimality guarantees for the fine-tuning policy, are provided for the model-based (Agarwal et al., 2020) and the model-free (Modi et al., 2021) setting.

On the methodological side, some approaches have been developed to extract lower-dimensional representations without imposing structural assumptions on the underlying CMP. Those generally encode the state-action representations through the bottleneck of a neural network that is trained to minimize a convenient loss, such as a contrastive loss (CURL, Laskin et al., 2020) or a reconstruction loss (RND, Burda et al., 2019b). Those methodologies hardly provide any theoretical guarantee on the quality of the pre-trained representations, but they have been employed with great benefits in challenging benchmarks (Laskin et al., 2021).

3.3.3 Pre-Training of Policy Spaces

Another category of approaches considers pre-training the policy space through unsupervised interactions, so that the subsequent fine-tuning phase can be addressed through a more compact policy space, which usually benefits sample efficiency (Papini et al., 2019; Metelli et al., 2021). In this setting, the typical model class comprises all the subsets of policies of a given policy space $\mathfrak{M}_\Pi = \{\Pi_{\text{red}} \in \mathscr{P}(\Pi)\}$.[2]

Theoretical works generally extract a compact policy space Π_{red} through an objective that measures covering of the state-action distributions induced by the policies in Π_{red} w.r.t. the policies in Π (Mutti et al., 2022d; Ye et al., 2022). Then, the supervised fine-tuning phase can directly search the optimal policy within the compact policy space, with provable benefits in the regret is suffers (Ye et al., 2022).

A large stream of methodological works instead operates the pre-training of the policy

[2] $\mathscr{P}(\Pi)$ denotes the power set of the policy space Π.

space by maximizing a heuristic measure of diversity, typically the mutual information, within a small set of policies (Gregor et al., 2017; Eysenbach et al., 2019; Hansen et al., 2019; Sharma et al., 2020; Campos et al., 2020; Liu & Abbeel, 2021a; He et al., 2022; Zahavy et al., 2022). The supervised fine-tuning phase can then use the learned policies for exploration or adapt them to the given task.

3.3.4 Pre-Training of Datasets

Another relevant approach prescribes using the unsupervised pre-training phase to collect a dataset of interactions instead of pre-training an actual model. The dataset can be then exploited in the supervised fine-tuning phase for direct planning or through offline RL algorithms (Levine et al., 2020). In this setting, the model class $\mathfrak{M}_{\mathcal{D}}$ can be informally defined as all the possible datasets $\mathcal{D}$ of N transitions (s, a, s'). The question is how to optimally collect those transitions through interactions with the CMP, and how large should the dataset be.

Some model-free works (Wang et al., 2020; Zanette et al., 2020) in the reward-free RL setting consider an objective function similar to (3.4), where the planning is not performed on an estimated model, but on a given dataset of samples

$$\min_{\mathcal{D} \in \mathfrak{M}_{\mathcal{D}}} \sup_{R \in \{\mathcal{S} \times \mathcal{A} \to [0,1]\}} \mathop{\mathbb{E}}_{s \sim \mu} \left[|V_P^*(s) - \widehat{V}_{\mathcal{D}}(s)| \right],$$

in which $\widehat{V}_{\mathcal{D}}(s)$ denotes the value function of the policy provided by an offline algorithm taking $\mathcal{D}$ as input. Those works provide compelling theoretical guarantees in the linear MDP setting, where the transition model can be linearly decomposed through a given state-action representation.[3]

On the empirical side, practical methodologies to perform unsupervised data collection in complex domains have been proposed (Yarats et al., 2022; Lambert et al., 2022). The latter lose the theoretical guarantees of the reward-free RL methods, but they do not require any structural assumption on the CMP, while they can be used in combination with any offline RL algorithm downstream.

3.3.5 Pre-Training of Policies

Of all the possible targets for pre-training mentioned in Table 3.1, in this thesis we will specifically cover unsupervised pre-training of policies. Thus, it is worth having a deeper look into it. Let the class of models $\mathfrak{M}$ be a policy space Π, we can formulate the policy pre-training objective as follows.

Policy Pre-Training Objective

$$\max_{\pi \in \Pi} \; \mathcal{F}(d^\pi), \tag{3.5}$$

[3]The structural hypothesis is similar to the one of low-rank MDPs (e.g., Agarwal et al., 2020) but the representation $\phi(s, a)$ is assumed to be known by the agent.

where d^π is the state distribution induced by the policy π in the CMP $\mathcal{M}$, and $\mathcal{F}$ is a function that maps state distributions to reals, i.e., $\mathcal{F} : \Delta(\mathcal{S}) \to \mathbb{R}$. The goal of optimizing the surrogate objective (3.5) is to obtain a pre-trained policy that can jump-start the subsequent supervised fine-tuning phase (Uchendu et al., 2022). Most of the existing approaches (see Table 3.1) aim to pre-train policies that can serve as exploratory initializations to the downstream RL algorithm, whereas little attention has been dedicated to how to best use the pre-trained policies in the fine-tuning phase. The supervised fine-tuning can catastrophically forget the pre-trained policy in a few iterations while maximizing objective (3.2), which can sometimes waste the capability of the pre-trained model. This problem of careful fine-tuning received relatively little attention w.r.t. the pre-training (the work by Campos et al. (2021) and Pislar et al. (2021) make interesting exceptions).

Similarly, in this thesis we will focus on the pre-training phase. While we are going to report fine-tuning results as well, we will consider naïve mechanisms to exploit the pre-trained policy, mostly for the purpose of evaluation of the pre-training rather than as a conceptual contribution. In Chapters 4, 5 we study the optimization problem (3.5) when $\mathcal{F}$ is any concave[4] function of d^π. In Chapter 6 we present the specific instance of the optimization problem (3.5) where the function $\mathcal{F}$ is the entropy function H (Hazan et al., 2019), which is also the focus of the remainder of this thesis. In the next paragraph, we briefly report the most relevant works in state entropy maximization, though most of them will be discussed in greater length in the following chapters.

Unsupervised Pre-Training via State Entropy Maximization

Hazan et al. (2019) were the first to consider an entropic measure over the state distribution as a sensible unsupervised pre-training. Especially, they propose an algorithm, called MaxEnt (see Section 6.2.1), that learns a mixture of policies collectively maximizing the Shannon entropy of the discounted state distribution. The final mixture is learned through a conditional gradient method, in which the algorithm iteratively estimates the state distribution of the current mixture to define an intrinsic reward function, and then identifies the next policy to be added by solving a specific RL sub-problem with this reward. A similar methodology has been obtained by Lee et al. (2019) from a game-theoretic perspective on unsupervised pre-training. Their algorithm, called SMM, targets the Shannon entropy of the marginal state distribution instead of the discounted distribution of MaxEnt. Another approach based on the conditional gradient method is FW-AME (Tarbouriech & Lazaric, 2019), which learns a mixture of policies to maximize the entropy of the stationary state-action distribution. As noted in (Tarbouriech & Lazaric, 2019), the mixture of policies might suffer a slow mixing to the asymptotic distribution for which the entropy is maximized. In (Mutti & Restelli, 2020), we present a method (IDE^{3}AL, see Section 7.4) to pre-train a single policy that simultaneously accounts for the entropy of the stationary state-action distribution and the mixing time.

Even if they are sometimes evaluated in continuous domains, especially (Hazan et al., 2019; Lee et al., 2019), the methods we mentioned require an accurate estimate of either the state distribution (Hazan et al., 2019; Lee et al., 2019) or the transition model (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), which hardly scales to the most com-

[4]We assume $\mathcal{F}$ is a concave function when the problem is formulated as a maximization, whereas we assume $\mathcal{F}$ is convex when the problem is formulated as a minimization instead.

Algorithm	Entropy	Distribution	Space	Mixture	Non-Par
MaxEnt (Hazan et al., 2019)	Shannon	discounted	state	✓	✗
FW-AME (Tarbouriech & Lazaric, 2019)	Shannon	stationary	state-action	✓	✗
SMM (Lee et al., 2019)	Shannon	marginal	state	✓	✗
IDE^{3}AL (Mutti & Restelli, 2020)	Shannon	stationary	state-action	✗	✗
MEPOL (Mutti et al., 2021)	Shannon	marginal	state	✗	✓
MaxRényi (Zhang et al., 2021a)	Rényi	discounted	state-action	✗	✗
GEM (Guo et al., 2021)	geometry-aware	marginal	state	✗	✗
APT (Liu & Abbeel, 2021b)	Shannon	marginal	state	✗	✓
RE3 (Seo et al., 2021)	Shannon	marginal	state	✗	✓
Proto-RL (Yarats et al., 2021)	Shannon	marginal	state	✗	✓
αMEPOL (Mutti et al., 2022e)	Shannon	marginal	state	✗	✓
KME (Nedergaard & Cook, 2022)	Shannon	discounted	state	✗	✓

Table 3.2: *Overview of the algorithms for unsupervised pre-training via state entropy maximization. For each algorithm, we report the nature of the objective, i.e., the entropy formulation (**Entropy**), whether it considers stationary, discounted, or marginal distributions (**Distribution**), and if it accounts for the state space $\mathcal{S}$ or the state-action space $\mathcal{S} \times \mathcal{A}$ (**Space**). We also specify if the method learns a single policy rather than a mixture of policies (**Mixture**), and if it supports non-parametric entropy estimation (**Non-Par**).*

plex, often high-dimensional, domains. In a subsequent work (Mutti et al., 2021), we propose an approach to estimate the entropy of the state distribution through a non-parametric method, and then to directly optimize the estimated entropy via policy optimization. The algorithm, called MEPOL (see Chapter 8), is able to learn a single exploration policy that maximizes the entropy of the marginal state distribution in challenging continuous control domains. Liu & Abbeel (2021b) combine non-parametric entropy estimation with learned state representations into an algorithm, called APT, that successfully addresses unsupervised pre-training in visual-inputs domains. Seo et al. (2021) shows that even random state representations are sufficient to pre-train such policies from visual inputs. On a similar line, Yarats et al. (2021) consider simultaneously learning state representations and a basis for the latent space (or prototypical representations) to help reduce the variance of the entropy estimates. The latter three works (Liu & Abbeel, 2021b; Seo et al., 2021; Yarats et al., 2021) are discussed in Section 8.6.

Whereas all of the previous approaches account for the Shannon entropy in their objectives, recent works (Zhang et al., 2021a; Guo et al., 2021) consider alternative formulations. Zhang et al. (2021a) argue that the Rényi entropy provides a superior incentive to cover all of the corresponding space than the Shannon entropy, and they propose a method to optimize the Rényi of the state-action distribution via gradient ascent (MaxRényi). On an orthogonal direction, the authors of (Guo et al., 2021) consider a reformulation of the entropy function that accounts for the underlying geometry of the space. They present a method, called GEM, to learn an optimal policy for the geometry-aware entropy objective. The work by Nedergaard & Cook (2022) maximizes a lower bound to the state entropy approximation through a clustering approach. They especially relate the entropy of the state distribution with the radius of the clusters obtained by a k-means procedure.

Finally, in a follow-up work to (Mutti et al., 2021), we introduce the problem of pre-training via state entropy maximization over a class of multiple CMPs (Mutti et al., 2022e). Especially, we present an algorithm, called αMEPOL (see Chapter 9), that maximizes the mean of a critical percentile α of the entropy realizations achieved across the class.

Part I

A Step Back to Convex Reinforcement Learning

CHAPTER 4

Convex Reinforcement Learning

The content of this chapter is based on the paper "Challenging Common Assumptions in Convex Reinforcement Learning" co-authored with Riccardo De Santi, Piersilvio De Bartolomeis, and Marcello Restelli, published at NeurIPS 2022.[1]

4.1 Introduction

In this chapter, we provide a formal study of the Convex Reinforcement Learning (CRL) formulation, as it subsumes the state entropy maximization problem that is the core focus of this thesis. The standard RL problem, which we have already introduced in Chapter 2, concerns sequential decision-making problems in which the utility can be expressed through a linear combination of scalar reward terms. The coefficients of this linear combination are given by the state visitation distribution induced by the agent's policy. Thus, the objective function can be equivalently written as the inner product between the mentioned state distribution and a reward vector. However, not all the relevant objectives can be encoded through this linear representation (Abel et al., 2021). Several works have thus extended the standard RL formulation to address non-linear objectives of practical interest. These include, among others, imitation learning (Hussein et al., 2017; Osa et al., 2018), or the problem of finding a policy that minimizes the distance between the induced state dis-

[1] A complete reference can be found in the bibliography (Mutti et al., 2022a).

tribution and the state distribution provided by experts' interactions (Abbeel & Ng, 2004; Ho & Ermon, 2016; Kostrikov et al., 2019; Lee et al., 2019; Ghasemipour et al., 2020; Dadashi et al., 2020), risk-averse RL (Garcıa & Fernández, 2015), in which the objective is sensitive to the tail behavior of the agent's policy (Tamar & Mannor, 2013; Prashanth & Ghavamzadeh, 2013; Tamar et al., 2015a; Chow et al., 2015, 2017; Bisi et al., 2020; Zhang et al., 2021b), diverse skills discovery (Gregor et al., 2017; Eysenbach et al., 2019; Hansen et al., 2019; Sharma et al., 2020; Campos et al., 2020; Liu & Abbeel, 2021a; He et al., 2022; Zahavy et al., 2022), constrained RL (Altman, 1999; Achiam et al., 2017; Brantley et al., 2020; Miryoosefi et al., 2019; Qin et al., 2021; Yu et al., 2021; Bai et al., 2022), and, most importantly, the state entropy maximization problem that we address in this thesis. All this large body of work has been recently unified into a unique framework, called *convex* RL (Zhang et al., 2020a; Zahavy et al., 2021; Geist et al., 2022), which admits as an objective any convex function of the state distribution induced by the agent's policy. The convex RL problem has been shown to be largely tractable either computationally, as it admits a dual formulation akin to standard RL (Puterman, 2014), or statistically, as principled algorithms achieving sub-linear regret rates that are slightly worse than standard RL have been developed (Zhang et al., 2020a; Zahavy et al., 2021).

However, we note that the usual convex RL formulation makes an implicit *infinite trials* assumption which is rarely met in practice. Indeed, the objective is written as a function of the state distribution, which is an expectation over the empirical state distributions that are actually obtained by running the policy in a given episode. In practice, we always run our policy for a finite number of episodes (or *trials*), which in general prevents the empirical state distribution from converging to its expectation. This has never been a problem in standard RL: Due to the scalar objective, optimizing the policy over infinite trials or finite trials is equivalent, as it leads to the same optimal policy. Crucially, we can show that this property does not hold for the convex RL formulation: A policy optimized over infinite trials can be significantly sub-optimal when deployed over finite trials (Figure 4.1). In light of this observation, we reformulate the convex RL problem from a *finite trials* perspective, developing insights that can be used to partially rethink the way convex objectives have been previously addressed in RL, with potential ripple effects to research areas of significant interest, such as imitation learning, risk-averse RL, state entropy maximization, and others.

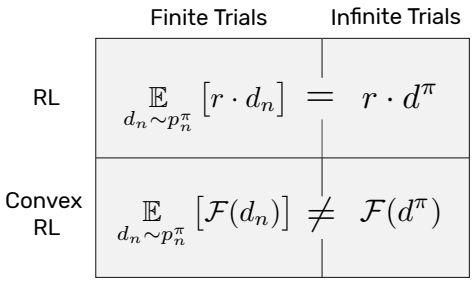


Figure 4.1: *Summary of the main result of this chapter: The equivalence between finite and infinite trials objectives does not hold for the convex RL formulation.*

Contents In this chapter, we formalize the notion of a finite trials RL problem, in which the objective is a function of the empirical state distribution induced by the agent's policy over n trials rather than its expectation over infinite trials. As an illustrative example, consider a financial application, in which we aim to optimize a trading strategy. In the real world, we can only deploy the strategy over a single trial. Thus, we are only interested in the performance of the strategy in the real-world realization, rather than the performance

of the strategy when averaging different realizations. Following this intuition, we first define the (linear) finite trial RL formulation (Section 4.3), for which it is trivial to prove the equivalence with standard RL. In Section 4.4, we provide the finite trial convex RL formulation, for which we prove an upper bound on the approximation error made by optimizing the infinite trials as a proxy of the finite trials objective. In light of this finding, we challenge the hidden assumption that **(1)** *convex RL can be equivalently addressed with an infinite trials formulation*, even if the setting is inherently finite trials. We corroborate this result with an additional numerical analysis showing that the approximation bound is non-vacuous for relevant applications (Section 4.6). Finally, in Section 4.5 we include an in-depth analysis of the *single trial* convex RL, which suggests that other common assumptions in the convex RL literature, i.e., that **(2)** *the problem is always computationally tractable* and that **(3)** *stationary policies are in general sufficient*, should be reconsidered as well. The proofs of the reported results can be found in Appendix A.1.

4.2 Definitions

In this section, we recall useful notation for the remainder of the chapter. As usual, we denote with $[H]$ a set of integers $\{1, \dots, H\}$, and with a lowercase letter a a scalar or a vector, according to the context. For two vectors $a, b \in \mathbb{R}^d$, we denote with $a \cdot b = \sum_{i=1}^{d} a_i b_i$ the inner product between a, b.

Probabilities Let $\mathcal{X}$ denote a measurable space, $\Delta(\mathcal{X})$ is the probability simplex over $\mathcal{X}$ and $p \in \Delta(\mathcal{X})$ is a probability measure over $\mathcal{X}$. For two probability measures p, q over $\mathcal{X}$, we define their ℓ^n-distance as $\|p - q\|_n := \left(\sum_{x \in \mathcal{X}} \left| p(x) - q(x) \right|^n \right)^{1/n}$, and their Kullback-Leibler (KL) divergence as $\mathrm{KL}(p||q) := \sum_{x \in \mathcal{X}} p(x) \log \big(p(x)/q(x) \big)$.

Percentiles Let X be a random variable distributed according to p, having a cumulative density function $F_X(x) = Pr(X \leq x)$. We denote with $\mathbb{E}[X]$ its expected value, and its α-percentile is denoted as $\mathrm{VaR}_\alpha(X) = \inf \left\{ x \mid F_X(x) \geq \alpha \right\} = F_X^{-1}(\alpha)$, where $\alpha \in (0, 1)$ is a confidence level, and VaR_α stands for Value at Risk (VaR) at level α. We denote the expected value of X within its α-percentile as $\mathrm{CVaR}_\alpha(X) = \mathbb{E}\left[X \mid X \leq \mathrm{VaR}_\alpha(X) \right]$, where CVaR_α stands for Conditional Value at Risk (CVaR) at level α.

Policies A policy π interacting with an MDP $\mathcal{M}^R := (\mathcal{S}, \mathcal{A}, P, \mu, T, R)$ consists of a sequence of decision rules $(\pi_t)_{t=0}^{\infty}$ that maps the current trajectory[2] $h_t = (s_i, a_i)_{i=0}^{t-1} \in \mathcal{H}_t$ with a distribution over actions $\pi_t : \mathcal{H}_t \to \Delta(\mathcal{A})$, where $\mathcal{H}_t$ denotes the set of trajectories of length t. A non-stationary policy is a sequence of decision rules $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$. A stationary (Markovian) policy is a time-consistent decision rule $\pi : \mathcal{S} \to \Delta(A)$.

State Distributions A trajectory h, obtained from an interaction episode, induces an empirical distribution d over the states of the MDP $\mathcal{M}^R$, such that $d(s) = \frac{1}{|h|} \sum_{s_t \in h} \mathbb{1}(s_t = s)$. We denote with p^π the probability of drawing d by following the policy π. For $n \in \mathbb{N}$, we denote with d_n the empirical distribution $d_n(s) = \frac{1}{n} \sum_{i=1}^{n} d_i(s)$, and with p_n^π the probability of drawing d_n by following the policy π for n episodes. Finally, we call the expectation $d^\pi = \mathbb{E}_{d \sim p^\pi}[d]$ the state distribution induced by π.

[2] We will call a sequence of states and actions a trajectory or a history indifferently.

4.3 Reinforcement Learning in Finite Trials

In the standard RL formulation (Sutton & Barto, 2018), an agent aims to learn an optimal policy by interacting with an unknown MDP $\mathcal{M}^R$. An optimal policy is a decision strategy that maximizes the expected sum of rewards collected during an episode. Especially, we can represent the value of a policy π through the value function $V_t^\pi(s) := \mathbb{E}_\pi \left[\sum_{t'=t}^{T} r(s_{t'}) \;\middle|\; s_t = s \right]$. The value function allows us to write the RL objective as $\max_{\pi \in \Pi} \mathbb{E}_{s_1 \sim \mu}[V_1^\pi(s_1)]$, where Π is the set of all the stationary policies. Equivalently, we can rewrite the RL objective into its dual formulation (Puterman, 2014), i.e.,

RL

$$\max_{\pi \in \Pi} \; \Big(R \cdot d^\pi \Big) =: \mathcal{J}_\infty(\pi) \tag{4.1}$$

where we denote with $R \in \mathbb{R}^S$ a reward vector, and with d^π the state distribution induced by π. We call the problem (4.1) the *infinite trials* RL formulation. Indeed, the objective $\mathcal{J}_\infty(\pi)$ considers the sum of the rewards collected during an episode, i.e., $R \cdot d^\pi$, that we can achieve on the average of an infinite number of episodes drawn with π. This is due to the state distribution d^π being an expectation of empirical distributions $d^\pi = \mathbb{E}_{d \sim p^\pi}[d]$. However, in practice, we can never draw infinitely many episodes following a policy π. Instead, we draw a small batch of episodes $d_n \sim p_n^\pi$. Thus, we can instead conceive a *finite trials* RL formulation that is closer to what is optimized in practice.

Finite Trials RL

$$\max_{\pi \in \Pi} \; \Big(\mathop{\mathbb{E}}_{d_n \sim p_n^\pi} \big[R \cdot d_n \big] \Big) =: \mathcal{J}_n(\pi) \tag{4.2}$$

One could then wonder whether optimizing the finite trials objective (4.2) leads to results that differ from the infinite trials one (4.1). To this point, it is straightforward to see that the two objective functions are actually equivalent

$$\mathcal{J}_n(\pi) = \mathop{\mathbb{E}}_{d_n \sim p_n^\pi} \big[R \cdot d_n \big] = R \cdot \mathop{\mathbb{E}}_{d_n \sim p_n^\pi} \big[d_n \big] = R \cdot d^\pi = \mathcal{J}_\infty(\pi),$$

since R is a fixed vector and the expectation is a linear operator. It follows that the infinite trials and the finite trials RL formulations admit the same optimal policies. Hence, one can enjoy the computational tractability of the infinite trials formulation and, at the same time, optimize the objective function that is employed in practice. In the next section, we will show that a similar result does not hold true for the convex RL formulation.

4.4 Convex Reinforcement Learning in Finite Trials

Even though the RL formulation covers a wide range of sequential decision-making problems, several relevant applications cannot be expressed by means of the inner product between a linear reward vector R and a state distribution d^π (Abel et al., 2021; Silver et al., 2021). These include imitation learning, state entropy maximization, constrained

Table 4.1: *Various convex RL objectives and applications. The last column states the equivalence between infinite trials and finite trials settings, as derived in Proposition A.1.1 (Appendix A.1).*

OBJECTIVE $\mathcal{F}$		APPLICATION	INFINITE $\equiv$ FINITE
$R \cdot d$	$R \in \mathbb{R}^S, d \in \Delta_\mathcal{S}$	RL	✓
$\|d - d_E\|_p^p$ $\mathrm{KL}(d \| d_E)$	$d, d_E \in \Delta_\mathcal{S}$	IMITATION LEARNING	✗
$-d \cdot \log(d)$	$d \in \Delta_\mathcal{S}$	PURE EXPLORATION	✗
$\mathrm{CVaR}_\alpha[R \cdot d]$ $R \cdot d - \mathbb{V}\mathrm{ar}[R \cdot d]$	$R \in \mathbb{R}^S, d \in \Delta_\mathcal{S}$	RISK-AVERSE RL	✗
$R \cdot d$, S.T. $\lambda \cdot d \leq c$	$R, \lambda \in \mathbb{R}^S, c \in \mathbb{R}, d \in \Delta_\mathcal{S}$	LINEARLY CONSTRAINED RL	✓
$-\mathbb{E}_z \mathrm{KL}(d_z \| \mathbb{E}_k d_k)$	$z \in \mathbb{R}^d, d_z, d_k \in \Delta_\mathcal{S}$	DIVERSE SKILL DISCOVERY	✗

problems, and risk-sensitive objectives, among others. Recently, a *convex* RL formulation (Zhang et al., 2020a; Zahavy et al., 2021; Geist et al., 2022) has been proposed to unify these applications in a unique general framework, which is

Convex RL

$$\max_{\pi \in \Pi} \Big(\mathcal{F}(d^\pi) \Big) =: \zeta_\infty(\pi) \tag{4.3}$$

where $\mathcal{F} : \Delta(S) \to \mathbb{R}$ is a function[3] of the state distribution d^π. In Table 4.1, we recap some of the most relevant problems that fall under the convex RL formulation along with their specific $\mathcal{F}$ function. As it happens for linear RL, in any practical simulated or real-world scenario, we can only draw a finite number of episodes with a policy π. From these episodes, we obtain an empirical distribution $d_n \sim p_n^\pi$ rather than the actual state distribution d^π, where n is the number of episodes (trials). This can cause a mismatch between the objective that is typically considered in convex RL (Zhang et al., 2020a; Zahavy et al., 2021) and what can be optimized in practice. To overcome this mismatch, we define a finite trials formulation of the convex RL objective as we did in the previous section for the linear RL formulation.

Finite Trials Convex RL

$$\max_{\pi \in \Pi} \Big(\mathop{\mathbb{E}}_{d_n \sim p_n^\pi} \big[\mathcal{F}(d_n) \big] \Big) =: \zeta_n(\pi) \tag{4.4}$$

Comparing objectives (4.3) and (4.4), one can notice that both of them include an expectation over the episodes, being $d^\pi = \mathbb{E}_{d \sim p^\pi}[d]$. Especially, we can write

$$\zeta_\infty(\pi) = \mathcal{F}(d^\pi) = \mathcal{F}\big(\mathop{\mathbb{E}}_{d_n \sim p_n^\pi} [d_n] \big) \leq \mathop{\mathbb{E}}_{d_n \sim p_n^\pi} [\mathcal{F}(d_n)] = \zeta_n(\pi)$$

through Jensen's inequality. As a consequence, optimizing the infinite trials objective $\zeta_\infty(\pi)$ does not necessarily lead to an optimal behavior for the finite trials objective $\zeta_n(\pi)$.

[3] In this context, we use the term *convex* to distinguish it from the standard *linear* RL objective. However, in the following, we will consider functions $\mathcal{F}$ that are either convex, concave, or even non-convex. In general, problem (4.3) takes the form of a max problem for concave $\mathcal{F}$, or a min problem for convex $\mathcal{F}$.

From a mathematical perspective, this is due to the fact that the empirical distributions d_n induced by the policy π are averaged by the expectation d^π before computing the $\mathcal{F}$ function into objective (4.3), thus losing a measure of the performance $\mathcal{F}$ for each batch of episodes, which we instead keep in the objective (4.4).

4.4.1 Approximating the Finite Trials Objective with Infinite Trials

Despite the evident mismatch between the finite trials and the infinite trials formulation of the convex RL problem, most existing works consider (4.3) as the standard objective, even if only a finite number of episodes can be drawn in practice. Thus, it is worth investigating how much we can lose by approximating a finite trials objective with an infinite trials one. First, we report a useful assumption on the structure of the function $\mathcal{F}$.

Assumption 4.4.1 (Lipschitz)**.** *A function $\mathcal{F} : \mathcal{X} \to \mathbb{R}$ is Lipschitz-continuous if it holds for some constant L*

$$\left|\mathcal{F}(x) - \mathcal{F}(y)\right| \leq L\left\|x - y\right\|_1, \qquad \forall (x, y) \in \mathcal{X}^2.$$

Then, we provide an upper bound on the approximation error in the following theorem.

Theorem 4.4.2 (Approximation Error)**.** *Let $n \in \mathbb{N}$ be a number of trials, let $\delta \in (0, 1]$ be a confidence level, let $\pi^\dagger \in \arg\max_{\pi \in \Pi} \zeta_n(\pi)$ and $\pi^\star \in \arg\max_{\pi \in \Pi} \zeta_\infty(\pi)$. Then, it holds with probability at least $1 - \delta$*

$$err := \left|\zeta_n(\pi^\dagger) - \zeta_n(\pi^\star)\right| \leq 4LT\sqrt{\frac{2S\log(4T/\delta)}{n}}$$

The previous result establishes an approximation error rate $err = O(LT\sqrt{S/n})$ that is polynomial in the number of episodes n. Unsurprisingly, the guarantee over the approximation error scales with $O(1/\sqrt{n})$, as one can expect the empirical distribution d_n to concentrate around its expected value for large n (Weissman et al., 2003). This implies that approximating the finite trials objective $\zeta_n(\pi)$ with the infinite trials $\zeta_\infty(\pi)$ can be particularly harmful in those settings in which n is necessarily small. As an example, consider training a robot through a simulator and deploying the obtained policy in the real world, where the performance measures are often based on a single episode ($n = 1$). The performance that we experience from the deployment can be much lower than the expected $\zeta_\infty(\pi)$, which might result in undesirable or unsafe behaviors. However, Theorem 4.4.1 only reports an instance-agnostic upper bound, and it does not necessarily imply that there would be a significant approximation error in a specific instance, i.e., a pairing of a CMP $\mathcal{M}$ and a function $\mathcal{F}$. Nevertheless, in this paper, we argue that the upper bound of the approximation error is not vacuous in several relevant applications, and we provide an illustrative numerical corroboration of this claim in Section 4.6.

Challenged assumption 1. *The convex RL problem can be equivalently addressed with an infinite trials formulation.*

Finally, in Figure 4.2 we report a visual representation[4] of the approximation error defined

[4]Note that it is not possible to represent the objective functions in two dimensions in general. Nevertheless, we provide an abstract one-dimensional representation of the policy space to bring the intuition.

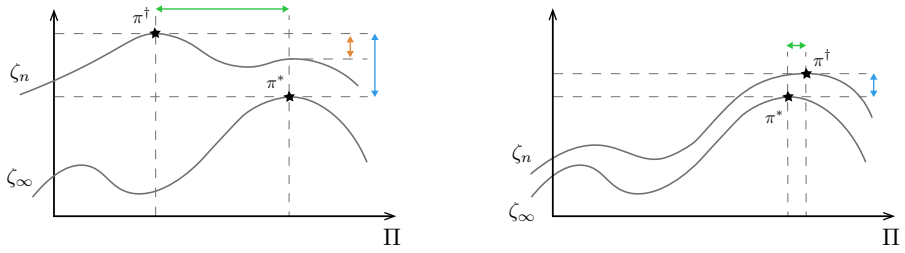


Figure 4.2: *The two illustrations report an abstract visualization of ζ_n and ζ_∞ for small values of n (left) and large values of n (right) respectively. The green bar visualize the distance $\left\|d_n - d^{\pi^\star}\right\|_1$, in which $d_n \sim p_n^{\pi^\dagger}$. The blue bar visualize the distance $\left|\zeta_n(\pi^\dagger) - \zeta_\infty(\pi^\star)\right|$. The orange bar visualize the approximation error, i.e., the distance $\left|\zeta_n(\pi^\dagger) - \zeta_n(\pi^\star)\right|$.*

in Theorem 4.4.1. Notice that the finite trials objective ζ_n converges uniformly to the infinite trials objective ζ_∞ as a trivial consequence of Theorem 4.4.1. This is particularly interesting as it results in $\pi^\dagger$ converging to $\pi^\star$ in the limit of large n as shown in Figure 4.2.

4.5 Single Trial Convex Reinforcement Learning

Having established a significant mismatch between the infinite trials convex RL setting that is usually considered in previous works, i.e., $\zeta_\infty(\pi)$, and the finite trials formulation that is actually targeted in practice, i.e., $\zeta_n(\pi)$, it is now worth taking a closer look at the finite trials optimization problem (4.4). Indeed, to avoid the approximation error that can occur by optimizing (4.4) through the infinite trials formulation (Theorem 4.4.1), one could instead directly address the optimization of (4.4). Especially, how does the finite trials convex RL problem compare to its infinite trials formulation and the linear RL problem? What kind of policies do we need to optimize the finite trials objective? Is the underlying learning process statistically harder than infinite trials convex RL? In this section, we investigate the answers to these relevant questions. To this purpose, we will focus on a *single trial* setting, i.e., $\zeta_n(\pi)$ with $n = 1$, which allows for a clearer analysis, while analogous considerations should extend to a general number of trials $n > 1$.

Single Trial Convex RL

$$\max_{\pi \in \Pi} \left(\mathop{\mathbb{E}}_{d \sim p^\pi} \left[\mathcal{F}(d) \right] \right) =: \zeta_1(\pi) \tag{4.5}$$

Taking inspiration from (Zahavy et al., 2021), we can cast the problem (4.5) defined over a CMP $\mathcal{M}$ into a *convex* MDP $\mathcal{CM} := (\mathcal{S}, \mathcal{A}, P, T, \mu, \mathcal{F})$, where $\mathcal{S}, \mathcal{A}, P, T, \mu$ are defined as in a standard MDP, and $\mathcal{F} : \Delta(\mathcal{S}) \to \mathbb{R}$ is a convex function that defines the objective $\zeta_1(\pi)$. Is solving a convex MDP $\mathcal{CM}$ significantly harder than solving an MDP $\mathcal{M}^R$?

4.5.1 Extended MDP Formulation of Single Trial Convex RL

We can show that any finite-horizon convex MDP $\mathcal{CM}$ can be actually translated into an equivalent MDP $\widetilde{\mathcal{M}}_T^R = (\mathcal{S}_\ell, \mathcal{A}_\ell, P_\ell, \mu_\ell, R_\ell)$, which we call an *extended* MDP. The main idea is to temporally extend $\mathcal{CM}$ so that each state contains the information of the full trajectory leading to it, so that the convex objective can be cast into a linear reward. To do this, we define the extended state space $\mathcal{S}_\ell$ to be the set of all the possible histories up to length T, so that $s_\ell \in \mathcal{S}_\ell$ now represents a history. Then, we can keep $\mathcal{A}_\ell, P_\ell, \mu_\ell$ equivalent to $\mathcal{A}, P, \mu$ of the original $\mathcal{CM}$, where for the extended transition model $P_\ell(s'_\ell|s_\ell, a)$ we solely consider the last state in the history s_ℓ to define the conditional probability to the next history s'_ℓ. Finally, we just need to define a scalar reward function $R_\ell : \mathcal{S}_\ell \to \mathbb{R}$ such that $R_\ell(s_\ell) = \mathcal{F}(d_{s_\ell})$ for all the histories s_ℓ of length T and $R_\ell(s_\ell) = 0$ otherwise, where we denoted with d_{s_ℓ} the empirical state distribution induced by s_ℓ.

Notably, the problem of finding an optimal policy for the extended MDP $\widetilde{\mathcal{M}}_T^R$, i.e., $\pi^* \in \arg\max_{\pi\in\Pi} R_\ell \cdot d^\pi$, is equivalent to solving the problem (4.5). Indeed, we have

$$R_\ell \cdot d^\pi = \sum_{s_\ell \in \mathcal{S}_\ell} R_\ell(s_\ell) d^\pi(s_\ell) = \sum_{s_\ell \in \mathcal{S}_\ell} \mathcal{F}(d_{s_\ell}) \mathbb{1}(|s_\ell| = H) p^\pi(d_{s_\ell}) = \mathop{\mathbb{E}}_{d_{s_\ell} \sim p^\pi} [\mathcal{F}(d_{s_\ell})].$$

Whereas $\widetilde{\mathcal{M}}_T^R$ can be solved with classical MDP methods (Puterman, 2014), the size of the policy $\pi : \mathcal{S}_\ell \to \Delta(\mathcal{A}_\ell)$ to be learned does scale with the size of $\widetilde{\mathcal{M}}_T^R$, which grows exponentially in the episode horizon as we have $|\mathcal{S}_\ell| > S^T$. Thus, the extended MDP formulation and the resulting insight cast some doubts on the notion that the convex RL is not significantly harder than standard RL (Zhang et al., 2020a; Zahavy et al., 2021).

Challenged assumption 2. *Convex RL is only slightly harder than the standard RL formulation.*

4.5.2 POMDP Formulation of Single Trial Convex RL

Instead of temporally extending the convex MDP $\mathcal{CM}$ as in the previous section, which causes the policy space to grow exponentially with the episode horizon T, we can alternatively formulate $\mathcal{CM}$ as a Partially Observable MDP (POMDP) (Åström, 1965; Kaelbling et al., 1998) $\mathcal{M}_\Omega^R = (\mathcal{S}_\ell, \mathcal{A}_\ell, P_\ell, \mu_\ell, R_\ell, \Omega, O)$, in which Ω denotes an observation space, and $O : \mathcal{S}_\ell \to \Delta(\Omega)$ is an observation function. The process to build $\mathcal{M}_\Omega^R$ is rather similar to the one we employed for the extended MDP $\widetilde{\mathcal{M}}_T^R$, and the components $\mathcal{S}_\ell, \mathcal{A}_\ell, P_\ell, \mu_\ell, R_\ell$ remain indeed unchanged. However, in a POMDP the agent does not directly access a state $s_\ell \in \mathcal{S}_\ell$, but just a partial observation $o \in \Omega$ that is given by the observation function O. Here $O(s_\ell) = o$ is a deterministic function such that the given observation o is the last state in the history s_ℓ. Since the agent only observes o, a stationary policy can be defined as a function $\pi : \Omega \to \Delta(\mathcal{A})$, for which the size depends on the number of states S of the convex MDP $\mathcal{CM}$, being $\Omega = \mathcal{S}$. However, it is well known (Kaelbling et al., 1998) that history-dependent policies should be considered for the problem of optimizing a POMDP. This is in sharp contrast with the current convex MDP literature, which only considers stationary policies due to the infinite trials formulation (Zhang et al., 2020a).

Challenged assumption 3. *The set of stationary randomized policies is sufficient for convex RL.*

4.5.3 Online Learning in Single Trial Convex RL

Let us assume to have access to a planning oracle that returns an optimal policy π^* for a given $\mathcal{CM}$ so that we can sidestep the concerns on the computational feasibility reported in previous sections. It is worth investigating the complexity of learning π^* from *online interactions* with an unknown $\mathcal{CM}$. A typical measure of this complexity is the online learning regret $\mathcal{R}(N)$, which is defined as

$$\mathcal{R}(N) := \sum_{t=1}^{N} V^* - V^{(t)},$$

where N is the number of learning episodes, $V^* = V_1^{\pi^*}(s_1)$ is the value of the optimal policy, $V^{(t)} = V^{\pi_t}$ is the value of the policy π_t deployed at the episode t. We now aim to assess whether there exists a principled algorithm that can achieve a sub-linear regret $\mathcal{R}(N)$ in the worst case. To this purpose, we can cast our learning problem in the Once-Per-Episode (OPE) RL formulation (Chatterji et al., 2021). In the latter setting, the agent interacts with the MDP for T steps, receiving a $0/1$ feedback at the end of the episode, where the feedback is computed according to a logistic model that is a function of the history. To translate our objective $\zeta_1(\pi) = \mathbb{E}_{d\sim p^\pi}[\mathcal{F}(d)]$ into the OPE framework (Chatterji et al., 2021), we have to encode $\mathcal{F}$ into a linear representation. With the following assumption, we state the existence of such representation.

Assumption 4.5.1 (Linear Realizability)**.** *The function $\mathcal{F}$ is linearly-realizable if it holds*

$$\mathcal{F}(d) = \mathbf{w}_*^\top \phi(d),$$

where $\mathbf{w}_* \in \mathbb{R}^{d_\mathbf{w}}$ is a vector of parameters such that $\| \mathbf{w}_* \|_2 \leq B$ for some known $B > 0$, and $\phi(d) = (\phi_j(d))_{j=1}^{d_\mathbf{w}}$ is a known vector of basis functions such that $\|\phi(d)\|_2 \leq 1, \forall d \in \Delta(\mathcal{S})$.

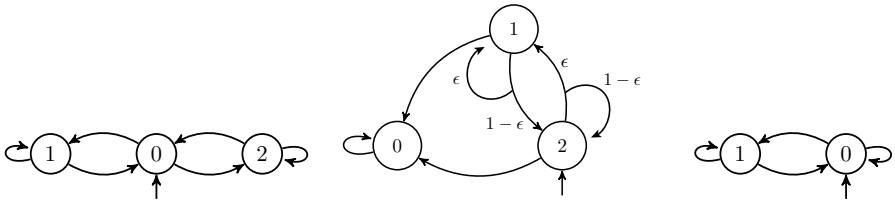
With the Assumption 4.5.1 and other minor changes that are detailed in the Appendix, we can invoke the analysis of OPE-UCBVI in (Chatterji et al., 2021) to provide an upper bound to the regret $\mathcal{R}(N)$ in our setting.

Theorem 4.5.2 (Regret)**.** *For any confidence $\delta \in (0, 1]$ and unknown convex MDP $\mathcal{CM}$, the regret of the* OPE-UCBVI *algorithm is upper bounded as*

$$\mathcal{R}(N) \leq O\Big(\Big[d_\mathbf{w}^{7/2} B^{3/2} T^2 S A^{1/2}\Big]\sqrt{N}\Big)$$

with probability $1 - \delta$.

The latter result states that the problem of learning an optimal policy in an unknown convex MDP is at least statistically efficient assuming linear realizability and access to a planning oracle. Those are fairly strong assumptions, but principled approximate solvers may be designed to overcome the planning oracle assumption, whereas in several convex RL settings the function $\mathcal{F}$ is assumed to be known, and thus trivially realizable.



(a) *Pure exploration* **(b)** *Risk-averse RL* **(c)** *Imitation learning*

Figure 4.3: *Visualization of the illustrative MDPs. In* ***(b)****, state* 0 *is a low-reward* (r) *low-risk state, state* 2 *is a high-reward* (R) *high-risk state, and state* 1 *is a penalty state with zero reward.*

4.6 Numerical Validation

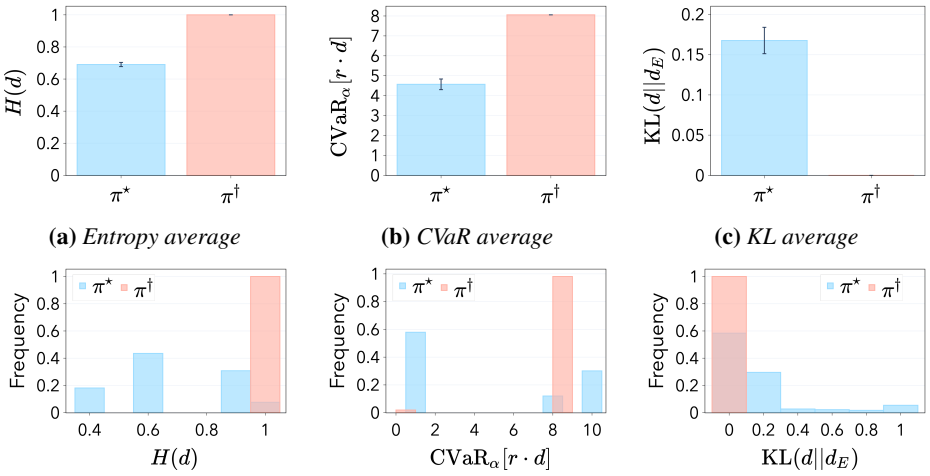
In this section, we evaluate the performance over the finite trials objective (4.4) achieved by a policy $\pi^\dagger \in \arg\max_{\pi\in\Pi} \zeta_n(\pi)$ maximizing the same finite trials objective (4.4) against a policy $\pi^\star \in \arg\max_{\pi\in\Pi} \zeta_\infty(\pi)$ maximizing the infinite trials objective (4.3) instead. The latter infinite trials π^* can be obtained by solving a dual optimization on the convex MDP (see Section 6.3),

$$\max_{\omega\in\Delta(\mathcal{S}\times\mathcal{A})} \mathcal{F}(\omega), \qquad \text{subject to } \sum_{a\in\mathcal{A}} \omega(s,a) = \sum_{s'\in\mathcal{S},a'\in\mathcal{A}} P(s|s',a')\omega(s',a'),\ \forall s \in \mathcal{S},$$

To get the finite trials $\pi^\dagger$, we first recover the extended MDP as explained in Section 4.5.1, and then we apply standard dynamic programming (Bellman, 1957). In the experiments, we show that optimizing the infinite trials objective can lead to sub-optimal policies across a wide range of applications. In particular, we cover examples from pure exploration, risk-averse RL, and imitation learning. We carefully selected MDPs that are as simple as possible in order to stress the generality of our results. For the sake of clarity, we restrict the discussion to the single trial setting ($n = 1$).

Pure Exploration For the pure exploration setting, we consider the state entropy objective (Hazan et al., 2019), i.e., $\mathcal{F}(d) = H(d) = -d \cdot \log d$, and the convex MDP in Figure 4.3a. In this example, the agent aims to maximize the state entropy over finite trajectories of T steps. Notice that this happens when a policy induces an empirical state distribution that is close to uniform. In Figure 4.4a, we compare the average entropy induced by the optimal finite trials policy $\pi^\dagger$ and the optimal infinite trials policy $\pi^\star$. An agent following the policy $\pi^\dagger$ always achieves a uniform empirical state distribution leading to the maximum entropy. Moreover, $\pi^\dagger$ is a non-Markovian deterministic policy. In contrast, the policy π^* is randomized in all three states. As a result, this policy induces sub-optimal empirical state distributions with *strictly positive* probability, as shown in Figure 4.4d.

Risk-Averse RL For the risk-averse RL setting, we consider a Conditional Value-at-Risk (CVaR) objective (Rockafellar et al., 2000) given by $\mathcal{F}(d) = \mathrm{CVaR}_\alpha[R \cdot d]$, where $R \in [0,1]^S$ is a reward vector, and the convex MDP in Figure 4.3b, in which the agent aims to maximize the CVaR over a finite-length trajectory of T steps. First, notice that financial semantics can be attributed to the given MDP. An agent, starting in state 2, can decide



(a) *Entropy average* **(b)** *CVaR average* **(c)** *KL average*

(d) *Entropy distribution* **(e)** *CVaR distribution* **(f)** *KL distribution*

Figure 4.4: *$\pi^\dagger$ denotes an optimal single trial policy, $\pi^\star$ denotes an optimal infinite trials policy. In **(a, d)** we report the average and the empirical distribution of the single trial utility $H(d)$ achieved in the pure exploration convex MDP ($T = 6$) of Figure 4.3a. In **(b, e)** we report the average and the empirical distribution of the single trial utility $\mathrm{CVaR}_\alpha[r \cdot d]$ (with $\alpha = 0.4$) achieved in the risk-averse convex MDP ($T = 5$) of Figure 4.3b. In **(c, f)** we report the average and the empirical distribution of the single trial utility $\mathrm{KL}(d||d_E)$ (with expert distribution $d_E = (1/3, 2/3)$) achieved in the imitation learning convex MDP ($T = 12$) of Figure 4.3c. For all the results, we provide 95 % c.i. over 1000 runs.*

whether to invest in risky assets, e.g., crypto-currencies, or in safe ones, e.g., treasury bills. Because of the stochastic transitions, a policy would need to be reactive to the realizations of the transition model in order to maximize the single trial objective (4.5). This kind of behavior is achieved by an optimal finite trials policy $\pi^\dagger$. Indeed, $\pi^\dagger$ is a non-Markovian deterministic policy, which can take decisions as a function of history, and thus takes into account the current realization. On the other hand, an optimal infinite trials policy π^* is a Markovian policy, and it cannot take into account the current history. As a result, the policy π^* induces sub-optimal trajectories with *strictly positive* probability (see Figure 4.4e). Finally, in Figure 4.4b we compare the single trial performance induced by the optimal single trial policy $\pi^\dagger$ and the optimal infinite trials policy $\pi^\star$. Overall, $\pi^\dagger$ performs significantly better than $\pi^\star$.

Imitation Learning For the imitation learning setting, we consider the distribution matching objective (Kostrikov et al., 2019), i.e., $\mathcal{F}(d) = \mathrm{KL}\left(d||d_E\right)$, and the convex MDP in Figure 4.3c. The agent aims to learn a policy π inducing an empirical state distribution d close to the empirical state distribution d_E demonstrated by an expert. In Figure 4.4c, we compare single trial performance induced by the optimal single trial policy $\pi^\dagger$ and the optimal infinite trials policy $\pi^\star$. An agent following $\pi^\dagger$ induces an empirical state distribution that perfectly matches the expert. In contrast, an agent following π^* induces

sub-optimal realizations with *strictly positive* probability (see Figure 4.4f).

4.7 Related Work

To the best of our knowledge, (Hazan et al., 2019) were the first to introduce the convex RL problem, as a generalization of the standard RL formulation to non-linear utilities, especially the entropy of the state distribution. They show that the convex RL objective, while being concave (convex) in the state distribution, can be non-concave (non-convex) in the policy parameters. Anyway, they provide a provably efficient algorithm that overcomes the non-convexity through a Frank-Wolfe approach. (Zhang et al., 2020a) study the convex RL problem under the name of RL with general utilities. Especially, they investigated a hidden convexity of the convex RL objective that allows for statistically efficient policy optimization in the infinite-trials setting. Recently, the infinite trials convex RL formulation has been reinterpreted from game-theoretic perspectives (Zahavy et al., 2021; Geist et al., 2022). The former (Zahavy et al., 2021) notes that the convex RL problem can be seen as a min-max game between the policy player and a cost player. The latter (Geist et al., 2022) shows that the convex RL problem is a subclass of mean-field games.

Another relevant branch of literature is the one investigating the expressivity of scalar (Markovian) rewards (Abel et al., 2021; Silver et al., 2021). Especially, (Abel et al., 2021) shows that not all the notions of task, such as inducing a set of admissible policies, a (partial) policy ordering, or a trajectory ordering, can be naturally encoded with a scalar reward function. Whereas the convex RL formulation extends the expressivity of scalar RL w.r.t. all these three notions of task, it is still not sufficient to cover any instance. Convex RL is powerful in terms of the policy ordering it can induce, but it is inherently limited on the trajectory ordering as it only accounts for the infinite trials state distribution. Instead, the finite trials convex RL setting that we presented in this paper is naturally expressive in terms of trajectory orderings, at the expense of a diminished expressivity on the policy orderings w.r.t. infinite trials convex RL.

Previous works concerning RL in the presence of trajectory feedback are also related to this work. Most of this literature assumes an underlying scalar reward model (e.g., Efroni et al., 2021) which only delays the feedback at the end of the episode. One notable exception is the once-per-episode formulation in (Chatterji et al., 2021), which we have already commented on in Section 4.5.

Finally, the work in (Cheung, 2019a,b) considers MDP with vectorial rewards as a mean to encode convex objectives in RL with a multi-objective flavor. They show that stationary policies are in general sub-optimal for the introduced online learning setting, where non-stationarity becomes essential. In this setting, they provide principled procedures to learn an optimal policy with sub-linear regret.

CHAPTER 5

The Importance of Non-Markovianity in Convex RL

The content of this chapter is based on the paper "The Importance of Non-Markovianity in Maximum State Entropy Exploration" co-authored with Riccardo De Santi and Marcello Restelli, published at ICML 2022, where it received the Outstanding Paper Award.[1]

5.1 Introduction

In the previous Chapter 4, we have introduced the Convex Reinforcement Learning (CRL) problem, and we have made a crucial distinction between its infinite trials and finite trials formulation, which induce fundamentally different learning problems. In this chapter, we aim to address a specific question that we left open in Section 4.5.2, i.e., whether the set of Markovian policies is sufficient to address CRL problems in finite trials.

All of the existing works in CRL solely focus on optimizing Markovian policies, which we recall are built with decision rules conditioned only on the current state of the environment rather than the full history of the visited states. There are good reasons for it, as the resulting learning problem is known to be, at least in tabular domains, provably efficient both computationally and statistically (e.g., Hazan et al., 2019; Zhang et al., 2020a). Moreover,

[1] A complete reference can be found in the bibliography (Mutti et al., 2022b).

this choice is common in RL, as it is well-known that an optimal deterministic Markovian strategy maximizes the usual objective, i.e., the cumulative sum of rewards (Puterman, 2014). Similarly, Hazan et al. (2019, Lemma 3.3) note that the class of Markovian strategies is *sufficient* for the infinite trials CRL objective. A carefully constructed Markovian strategy is able to induce the same state distribution of any history-based (non-Markovian) one by exploiting randomization. Crucially, this result does not hold only for asymptotic state distributions, but also for state distributions that are marginalized over a finite horizon (Puterman, 2014). Hence, there is little incentive to consider more complicated strategies as they are not providing any benefit to the value of the objective function.

However, we argue that prior work does not recognize the importance of non-Markovianity in CRL due to the common infinite trials assumption that we have extensively discussed in Chapter 4. Instead, we will show that non-Markovian strategies are crucial in the finite trials CRL formulation, by introducing a novel notion of *convex value gap*. Unfortunately, learning a non-Markovian strategy is in general much harder than a Markovian one, and we are able to show that it is indeed NP-hard in this setting. Nonetheless, we aim to highlight the importance of non-Markovinaity to optimize the finite trials CRL objective, thereby motivating the development of tractable formulations of the problem as future work.

Contents The chapter is organized as follows. Having recalled a few crucial definitions for understanding the content of this chapter (Section 5.2), we report a known result (Puterman, 2014) to show that the class of Markovian strategies is sufficient for any infinite trials CRL objective (Section 5.3). Then, in Section 5.4, we focus on the single trial CRL formulation and we introduce a novel notion of *convex value gap* to show that the class of non-Markovian strategies is sufficient, whereas the optimal Markovian strategy suffers a non-zero gap in this setting. However, in Section 5.5, we show that the problem of finding an optimal non-Markovian strategy for the single trial CRL objective is NP-hard in general. Despite the hardness result, in Section 5.6, we comment on some potential options to address the problem in a tractable way, which we leave as future work. The missing proofs of the reported results can be found in Appendix A.2.

5.2 Definitions

Before going into the technical details of this chapter, we recall some useful definitions and notation that are useful in the following sections. As usual, we will denote with $\Delta(\mathcal{X})$ the simplex of a space $\mathcal{X}$, with $[T]$ the set of integers $\{0, \dots, T-1\}$, and with $v \oplus u$ a concatenation of the vectors v, u.

Policies A (general) policy π consists of a sequence of decision rules $\pi := (\pi_t)_{t=0}^{\infty}$. Each of them is a map between histories $h := (s_j, a_j)_{j=0}^{t} \in \mathcal{H}_t$ and actions $\pi_t : \mathcal{H}_t \to \Delta(\mathcal{A})$, such that $\pi_t(a|h)$ defines the conditional probability of taking action $a \in \mathcal{A}$ having experienced the history $h \in \mathcal{H}_t$. We denote as $\mathcal{H}$ the space of the histories of arbitrary length. Within this chapter, we denote as Π the set of all the policies, and as Π^{D} the set of deterministic policies $\pi = (\pi_t)_{t=1}^{\infty}$ such that $\pi_t : \mathcal{H}_t \to \mathcal{A}$. We further define:

- *Non-Markovian* (NM) policies Π_{NM}, where each $\pi \in \Pi_{\mathrm{NM}}$ collapses to a single time-invariant decision rule $\pi = (\pi, \pi, \dots)$ such that $\pi : \mathcal{H} \to \Delta(\mathcal{A})$;
- *Markovian* (M) policies Π_{M}, where each $\pi \in \Pi_{\mathrm{M}}$ is defined through a sequence of

Markovian decision rules $\pi = (\pi_t)_{t=0}^{\infty}$ such that $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$. A Markovian policy that collapses into a single time-invariant decision rule $\pi = (\pi, \pi, \dots)$ is called a *stationary* policy.

State Distributions and Visitation Frequency We denote with $d_t^\pi(s) := Pr(s_t = s|\pi)$ the t-step state distribution induced by π, and with $d_t^\pi(s,a) := Pr(s_t = s, a_t = a|\pi)$ the corresponding t-step state-action distribution. When t goes to infinity, we call $d_\infty^\pi(s) := \lim_{t\to\infty} d_t^\pi(s)$ the *stationary state distribution*, and we call $d_\gamma^\pi(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_t^\pi(s)$ the *discounted state distribution*, where $\gamma \in (0,1)$ is the discount factor. With some abuse of notation, we will denote as $d_T^\pi(s) := \frac{1}{T}\sum_{t\in[T]} d_t^\pi(s)$ the *marginal state distribution*. The *state visitation frequency* $d_h(s) = \frac{1}{T}\sum_{t\in[T]}\mathbb{1}(s_t = s|h)$ is a realization of the marginal state distribution, such that $\mathbb{E}_{h\sim p_T^\pi}\left[d_h(s)\right] = d_T^\pi(s)$, where the distribution over histories $p_T^\pi \in \Delta(\mathcal{H}_T)$ is defined as

$$p_T^\pi(h) = \mu(s_0)\prod_{t\in[T-1]}\pi(a_t|h_t)P(s_{t+1}|a_t,s_t).$$

5.3 Infinite Trials: Non-Markovianity Does Not Matter

As we discussed in Chapter 4, previous works in CRL consider an objective of the kind

$$\max_{\pi\in\Pi}\ \left(\mathcal{F}(d^\pi)\right) =: \zeta_\infty(\pi), \tag{5.1}$$

where $d^\pi(\cdot)$ is either a stationary state distribution, a discounted state distribution, or a marginal state distribution. While it is well-known (Puterman, 2014) that there exists an optimal deterministic policy $\pi^* \in \Pi_{\mathrm{M}}^{\mathrm{D}}$ for the common average return objective, it is not pointless to wonder whether the objective in (5.1) requires a more powerful policy class than Π_{M}. Hazan et al. (2019, Lemma 3.3) confirm that the set of (randomized) Markovian policies Π_{M} is indeed sufficient for ζ_∞ defined over asymptotic (stationary or discounted) state distributions. In the following theorem and corollary, we report a common MDP result (Puterman, 2014) to show that Π_{M} suffices for ζ_∞ defined over (non-asymptotic) marginal state distributions as well.

Theorem 5.3.1. *Let $x \in \{\infty, \gamma, T\}$, and let $\mathcal{D}_{\mathrm{NM}}^x = \{d_x^\pi(\cdot) : \pi \in \Pi_{\mathrm{NM}}\}$, $\mathcal{D}_{\mathrm{M}}^x = \{d_x^\pi(\cdot) : \pi \in \Pi_{\mathrm{M}}\}$ the corresponding sets of state distributions over a MDP. We can prove that:*

(i) The sets of stationary state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^\infty \equiv \mathcal{D}_{\mathrm{M}}^\infty$;

(ii) The sets of discounted state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^\gamma \equiv \mathcal{D}_{\mathrm{M}}^\gamma$ for any γ;

(iii) The sets of marginal state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^T \equiv \mathcal{D}_{\mathrm{M}}^T$ for any T.

Proof Sketch. For any non-Markovian policy $\pi \in \Pi_{\mathrm{NM}}$ inducing distributions $d_t^\pi(\cdot)$, $d_t^\pi(\cdot,\cdot)$ over the states and the state-action pairs of the MDP, we can build a Markovian policy $\pi' \in \Pi_{\mathrm{M}}, \pi' = (\pi'_t)_{t=0}^{\infty}$ through the construction $\pi'_t(a|s) = d_t^\pi(s,a)\big/d_t^\pi(s), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$. From (Puterman, 2014, Theorem 5.5.1) we know that $d_t^\pi(s) = d_t^{\pi'}(\cdot)$ holds for any $t \geq 0$. This implies that $d_\infty^\pi(\cdot) = d_\infty^{\pi'}(\cdot)$, $d_\gamma^\pi(\cdot) = d_\gamma^{\pi'}(\cdot)$, $d_T^\pi(\cdot) = d_T^{\pi'}(\cdot)$, from which $\mathcal{D}_{\mathrm{NM}}^x \equiv \mathcal{D}_{\mathrm{M}}^x$ follows. See Appendix A.2 for a detailed proof. □

From the equivalence of the sets of induced distributions, it is straightforward to derive the optimality of Markovian policies for objective (5.1).

Corollary 5.3.2. *For every MDP, there exists a Markovian policy* $\pi^* \in \Pi_{\mathrm{M}}$ *such that* $\pi^* \in \arg\max_{\pi\in\Pi} \zeta_\infty(\pi)$.

As a consequence of Corollary 5.3.2, there is little incentive to consider non-Markovian policies when optimizing objective (5.1), since there is no clear advantage to make up for the additional complexity of the policy. This result might be unsurprising when considering asymptotic distributions, as one can expect a carefully constructed Markovian policy to be able to tie the distribution induced by a non-Markovian policy in the limit of the interaction steps. However, it is less evident that a similar property holds for the expectation of final-length interactions alike. Yet, we were able to show that a Markovian policy that properly exploits randomization can always achieve equivalent state distributions w.r.t. non-Markovian counterparts. Note that state distributions are actually *expected* state visitation frequency, and the expectation practically implies an infinite number of realizations. In this chapter, we show that this underlying infinite trials regime is the reason why the benefit of non-Markovianity, albeit backed up by intuition, does not matter.

5.4 Single Trial: Non-Markovianity Matters

In this section, we focus on the single trial CRL formulation, i.e.,

$$\max_{\pi\in\Pi} \Big(\mathop{\mathbb{E}}_{d_h\sim p^\pi_T} \big[\mathcal{F}(d_h)\big] \Big) =: \zeta_1(\pi). \tag{5.2}$$

We note that $\zeta_1(\pi) \geq \zeta_\infty(\pi)$ for any $\pi \in \Pi$, which is trivial by the convexity of $\mathcal{F}$ and the Jensen's inequality. Whereas (5.2) is ultimately an expectation as it is (5.1), the objective is not computed over the (infinite trials) state distribution $d^\pi_T(\cdot)$ but its realization $d_h(\cdot)$. Thus, to maximize $\zeta_1(\pi)$ we have to find a policy maximizing $\mathcal{F}$ within a single trajectory rather than over infinitely many trajectories. Crucially, while Markovian policies are as powerful as any other policy class in terms of induced state distributions (Theorem 5.3.1), this is no longer true when looking at induced trajectory distributions p^π_T. Indeed, we will show that non-Markovianity provides a superior policy class for objective (5.2). First, we define a performance measure to formally assess this benefit, which we call the *convex value gap*.[2]

Definition 5.4.1 (Convex Value Gap)**.** *Consider a policy* $\pi \in \Pi$ *interacting with an MDP over* $T - t$ *steps starting from the trajectory* h_t*. We define the convex value gap* $\mathcal{V}_{T-t}$*, i.e., from step* t *onwards, as*

$$\mathcal{V}_{T-t}(\pi, h_t) = \mathcal{F}^* - \mathop{\mathbb{E}}_{h_{T-t}\sim p^\pi_{T-t}} \big[\mathcal{F}\big(d_{h_t\oplus h_{T-t}}(\cdot)\big)\big],$$

where $\mathcal{F}^* = \max_{\pi^*\in\Pi} \mathbb{E}_{h^*_{T-t}\sim p^{\pi^*}_{T-t}} \big[\mathcal{F}\big(d_{h_t\oplus h^*_{T-t}}(\cdot)\big)\big]$ *is the convex value achieved by an optimal policy* π^**. The term* $\mathcal{V}_T(\pi)$ *denotes the convex value gap of a* T*-step trajectory* h_T *starting from* $s \sim d_0$.

[2]Note that the objective function does not enjoy additivity, thus we cannot split the gap per-state.

Despite its convoluted definition, the intuition behind the convex value gap is quite simple. Suppose to have drawn a trajectory h_t upon step t. If we take the subsequent action with the (possibly sub-optimal) policy π, by how much would we decrease (in expectation) the convex value $\mathcal{F}(d_{h_T}(\cdot))$ w.r.t. an optimal policy π^*? In particular, we would like to know how limiting the policy π to a specific policy class would affect the convex value gap and the $\zeta_1(\pi)$ we could achieve. The following theorem and subsequent corollary state that an optimal non-Markovian policy suffers zero convex value gap in any case, whereas an optimal Markovian policy suffers non-zero gap in general.

Theorem 5.4.2 (Non-Markovian Optimality)**.** *For every MDP $\mathcal{M}$ and trajectory $h_t \in \mathcal{H}_{[T]}$, there exists a deterministic non-Markovian policy $\pi_{\text{NM}} \in \Pi^{\text{D}}_{\text{NM}}$ that suffers zero convex value gap $\mathcal{V}_{T-t}(\pi_{\text{NM}}, h_t) = 0$, whereas for any $\pi_{\text{M}} \in \Pi_{\text{M}}$, $\mathcal{V}_{T-t}(\pi_{\text{M}}, h_t) \geq 0$.*

Proof. The result is a straightforward combination of Lemma 5.4.5, 5.4.7 that we provide in the following section. Especially, $\mathcal{V}_{T-t}(\pi_{\text{NM}}, h_t) = 0$ for a policy $\pi_{\text{NM}} \in \Pi^{\text{NM}}_{\text{D}}$ is a direct implication of Lemma 5.4.5, whereas $\mathcal{V}_{T-t}(\pi_{\text{M}}, h_t) \geq 0$ for any $\pi_{\text{M}} \in \Pi_{\text{M}}$ is given by Lemma 5.4.7, which states that even an optimal Markovian policy $\pi^*_{\text{M}} \in \arg\max_{\pi \in \Pi_{\text{M}}} \mathcal{E}(\pi)$ suffers convex value gap $\mathcal{V}_{T-t}(\pi^*_{\text{M}}) \geq 0$. □

Corollary 5.4.3 (Sufficient Condition)**.** *For every MDP $\mathcal{M}$ and trajectory $h_t \in \mathcal{H}_{[T]}$ for which any optimal Markovian policy $\pi_{\text{M}} \in \Pi_{\text{M}}$ is randomized (i.e., stochastic) in s_t, we have strictly positive gap $\mathcal{V}_{T-t}(\pi_{\text{M}}, h_t) > 0$.*

The result of Theorem 5.4.2 highlights the importance of non-Markovianity for optimizing the single trial objective (5.2), as the class of Markovian policies is dominated by the class of non-Markovian policies. Most importantly, Corollary 5.4.3 shows that non-Markovian policies are strictly better than Markovian policies in several MDP of practical interest, i.e., those in which any optimal Markovian policy has to be randomized in order to maximize (5.2). The intuition behind this result is that a Markovian policy would randomize to make up for the uncertainty over history, whereas a non-Markovian policy does not suffer from this partial observability, and it can deterministically select an optimal action. Clearly, this partial observability is harmless when dealing with the standard RL objective, in which the reward is fully Markovian and does not depend on the history, but it is instead relevant in the peculiar single trial CRL setting, in which the objective is a concave function of the state visitation frequency. In the following section, we report a sketch of the derivation underlying Theorem 5.4.2 and Corollary 5.4.3, while we refer to the Appendix A.1 for complete proofs as usual.

5.4.1 Gap Analysis

For the purpose of the analysis, we will consider the following assumption to ease the notation.[3]

[3]Note that this assumption could be easily removed by partitioning the action space in h_t as $\mathcal{A}(h_t) = \mathcal{A}_{opt}(h_t) \cup \mathcal{A}_{sub-opt}(h_t)$, such that $\mathcal{A}_{opt}(h_t)$ are optimal actions and $\mathcal{A}_{sub-opt}(h_t)$ are sub-optimal, and substituting any term $\pi(a^*|h_t)$ with $\sum_{a \in \mathcal{A}_{opt}(h_t)} \pi(a|h_t)$ in the results.

Assumption 5.4.4 (Unique Optimal Action)**.** *For every MDP $\mathcal{M}$ and trajectory $h_t \in \mathcal{H}_{[T]}$, there exists a unique optimal action $a^* \in \mathcal{A}$ w.r.t. the objective* (5.2)*.*

First, we show that the class of deterministic non-Markovian policies is sufficient for the minimization of the convex value gap, and thus for the maximization of (5.2).

Lemma 5.4.5. *For every MDP $\mathcal{M}$ and trajectory $h_t \in \mathcal{H}_{[T]}$, there exists a deterministic non-Markovian policy $\pi_{\mathrm{NM}} \in \Pi^{\mathrm{D}}_{\mathrm{NM}}$ such that $\pi_{\mathrm{NM}} \in \arg\max_{\pi \in \Pi_{\mathrm{NM}}} \zeta_1(\pi)$, which suffers convex value gap $\mathcal{V}_{T-t}(\pi_{\mathrm{NM}}, h_t) = 0$.*

Proof. The result $\mathcal{V}_{T-t}(\pi_{\mathrm{NM}}, h_t) = 0$ is straightforward by noting that the set of non-Markovian policies Π_{NM} with arbitrary history-length is as powerful as the general set of policies Π. To show that there exists a deterministic π_{NM}, we consider the extended MDP $\widetilde{\mathcal{M}}^R_T$ obtained from the MDP $\mathcal{M}$ as in Section 4.5.1, in which the extended reward function is $R_\ell(s_\ell, a_\ell) = \mathcal{F}(d_{s_\ell}(\cdot))$ for every $a_\ell \in \mathcal{A}_\ell$, and every $s_\ell \in \mathcal{S}_\ell$ such that $|s_\ell| = T$, and $R_\ell(s_\ell, a_\ell) = 0$ otherwise. Since a Markovian policy $\widetilde{\pi}_{\mathrm{M}} \in \Pi^{\mathrm{D}}_{\mathrm{M}}$ on $\widetilde{\mathcal{M}}^R_T$ can be mapped to a non-Markovian policy $\pi_{\mathrm{NM}} \in \Pi^{\mathrm{D}}_{\mathrm{NM}}$ on $\mathcal{M}$, and it is well-known (Puterman, 2014) that for any MDP there exists an optimal deterministic Markovian policy, we have that $\widetilde{\pi}_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \mathcal{J}_{\widetilde{\mathcal{M}}^R_T}(\pi)$ implies $\pi_{\mathrm{NM}} \in \arg\max_{\pi \in \Pi_{\mathrm{NM}}} \zeta_1(\pi)$. □

Then, in order to prove that the class of non-Markovian policies is also necessary for gap minimization, it is worth showing that Markovian policies can instead rely on randomization to optimize objective (5.2).

Lemma 5.4.6. *Let $\pi_{\mathrm{NM}} \in \Pi^{\mathrm{D}}_{\mathrm{NM}}$ be a deterministic non-Markovian policy such that $\pi_{\mathrm{NM}} \in \arg\max_{\pi \in \Pi} \zeta_1(\pi)$ on a CMP $\mathcal{M}$. For a fixed history $h_t \in \mathcal{H}_t$ ending in state s, the variance of the event of an optimal Markovian policy $\pi_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \zeta_1(\pi)$ taking $a^* = \pi_{\mathrm{NM}}(h_t)$ in s is given by*

$$\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s,t))\right] = \mathop{\mathbb{V}\mathrm{ar}}_{hs \sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right],$$

where $hs \in \mathcal{H}_t$ is any history of length t such that the final state is s, i.e., $hs := (h_{t-1} \in \mathcal{H}_{t-1}) \oplus s$, and $\mathcal{B}(x)$ is a Bernoulli with parameter x.

Proof Sketch. We can prove the result through the Law of Total Variance (LoTV) (see Bertsekas & Tsitsiklis, 2002), which gives

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s,t))\right] = \mathop{\mathbb{E}}_{hs \sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right] \\ + \mathop{\mathbb{V}\mathrm{ar}}_{hs \sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right], \qquad \forall s \in \mathcal{S}.\end{aligned}$$

Then, exploiting the determinism of π_{NM} (through Lemma 5.4.5), it is straightforward to see that $\mathbb{E}_{hs \sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right] = 0$, which concludes the proof.[4] □

[4]Note that the determinism of π_{NM} does not also imply $\mathbb{V}\mathrm{ar}_{hs \sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right] = 0$, as the optimal action $\overline{a} = \pi_{\mathrm{NM}}(hs)$ may vary for different histories, which results in the inner expectations $\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]$ being either 1 or 0.

Unsurprisingly, Lemma 5.4.6 shows that, whenever the optimal strategy for (5.2), i.e., the non-Markovian π_{NM}, requires to adapt its decision in a state s according to the history that led to it (hs), an optimal Markovian policy for the same objective, i.e., π_{M}, must necessarily be randomized. This is crucial to prove the following result, which establishes lower and upper bounds $\underline{\mathcal{V}}_{T-t}, \overline{\mathcal{V}}_{T-t}$ to the convex value gap of any Markovian policy that optimizes (5.2).

Lemma 5.4.7. *Let π_{M} be an optimal Markovian policy $\pi_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \zeta_1(\pi)$ on a MDP $\mathcal{M}$. For any $h_t \in \mathcal{H}_{[T]}$, it holds $\underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) \leq \mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \leq \overline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}})$ such that*

$$\underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) = \frac{\mathcal{F}^* - \mathcal{F}_2^*}{\pi_{\mathrm{M}}(a^*|s_t)} \operatorname*{\mathbb{V}ar}_{hs_t \sim p_t^{\pi_{\mathrm{NM}}}} \left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t)) \right] \right],$$
$$\overline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) = \frac{\mathcal{F}^* - \mathcal{F}_*}{\pi_{\mathrm{M}}(a^*|s_t)} \operatorname*{\mathbb{V}ar}_{hs_t \sim p_t^{\pi_{\mathrm{NM}}}} \left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t)) \right] \right],$$

where $\pi_{\mathrm{NM}} \in \arg\max_{\pi \in \Pi^{\mathrm{D}}_{\mathrm{NM}}} \zeta_1(\pi)$, and $\mathcal{F}_*, \mathcal{F}_2^*$ are given by

$$\mathcal{F}_* = \min_{h \in \mathcal{H}_{T-t}} \mathcal{F}(d_{h_t \oplus h}(\cdot)),$$
$$\mathcal{F}_2^* = \max_{h \in \mathcal{H}_{T-t} \setminus \mathcal{H}^*_{T-t}} \mathcal{F}(d_{h_t \oplus h}(\cdot)) \qquad s.t.\ \mathcal{H}^*_{T-t} = \arg\max_{h \in \mathcal{H}_{T-t}} \mathcal{F}(d_{h_t \oplus h}(\cdot)).$$

Proof Sketch. The crucial idea to derive lower and upper bounds to the regret-to-go is to consider the impact of a sub-optimal action in the best-case and the worst-case MDP respectively. This gives $\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \geq \mathcal{F}^* - \pi_{\mathrm{M}}(a^*|s_t)\mathcal{F}^* - \big(1 - \pi_{\mathrm{M}}(a^*|s_t)\big)\mathcal{F}_2^*$ and $\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \leq \mathcal{F}^* - \pi_{\mathrm{M}}(a^*|s_t)\mathcal{F}^* - \big(1 - \pi_{\mathrm{M}}(a^*|s_t)\big)\mathcal{F}_*$. Then, with Lemma 5.4.6 we get $\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s_t))\right] = \pi_{\mathrm{M}}(a^*|s_t)\big(1 - \pi_{\mathrm{M}}(a^*|s_t)\big) = \operatorname*{\mathbb{V}ar}_{hs \sim p_t^{\pi_{\mathrm{NM}}}} \left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t)) \right] \right]$, which concludes the proof. □

Finally, the result in Theorem 5.4.2 is a direct consequence of Lemma 5.4.7. Note that the upper and lower bounds on the value gap are strictly positive whenever $\pi_{\mathrm{M}}(a^*|s_t) < 1$, as it is stated in Corollary 5.4.3.

5.5 Complexity Analysis

Having established the importance of non-Markovianity in single trial CRL, it is worth considering how hard it is to optimize the objective (5.2) within the class of non-Markovian policies. Especially, we aim at characterizing the complexity of the problem:

$$\Psi_0 := \operatorname*{maximize}_{\pi \in \Pi_{\mathrm{NM}}} \zeta_1(\pi),$$

defined over a MDP $\mathcal{M}$. Before going into the details of the analysis, we provide a couple of useful definitions for the remainder of the section, whereas we leave to (Arora & Barak, 2009) an extended review of complexity theory.

Definition 5.5.1 (Many-to-one Reductions)**.** *We denote as $A \leq_m B$ a many-to-one reduction from A to B.*

Definition 5.5.2 (Polynomial-time Reductions)**.** *We denote as $A \leq_p B$ a polynomial-time (Turing) reduction from A to B.*

Then, we recall that Ψ_0 can be rewritten as the problem of finding a reward-maximizing Markovian policy, i.e., $\widetilde{\pi}_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \mathcal{J}_{\widetilde{\mathcal{M}}_T^R}(\pi)$, over a convenient extended MDP $\widetilde{\mathcal{M}}_T^R$ obtained from MDP $\mathcal{M}$ (see the proof of Lemma 5.4.5 and Section 4.5.1 for further details). We call this problem $\widetilde{\Psi}_0$ and we note that $\widetilde{\Psi}_0 \in \mathrm{P}$, as the problem of finding a reward-maximizing Markovian policy is well-known to be in P for any MDP (Papadimitriou & Tsitsiklis, 1987). However, the following lemma shows that it does not exist a many-to-one reduction from Ψ_0 to $\widetilde{\Psi}_0$.

Lemma 5.5.3. *A reduction $\Psi_0 \leq_m \widetilde{\Psi}_0$ does not exist.*

Proof. In general, coding any instance of Ψ_0 in the representation required by $\widetilde{\Psi}_0$, which is an extended MDP $\widetilde{\mathcal{M}}_T^R$, holds exponential complexity w.r.t. the input of the initial instance of Ψ_0, i.e., a MDP $\mathcal{M}$. Indeed, to build the extended MDP $\widetilde{\mathcal{M}}_T^R$ from $\mathcal{M}$, we need to define the transition probabilities $P_\ell(s'_\ell | s_\ell, a_\ell)$ for every $s'_\ell \in \mathcal{S}_\ell, a_\ell \in \mathcal{A}_\ell, s_\ell \in \mathcal{S}_\ell$. Whereas the action space remains unchanged $\mathcal{A}_\ell = \mathcal{A}$, the extended state space $\mathcal{S}_\ell$ has cardinality $|\mathcal{S}_\ell| = S^T$ in general, which grows exponentially in T. □

The latter result informally suggests that $\Psi_0 \notin \mathrm{P}$. Indeed, we can now prove the main theorem of this section, which shows that Ψ_0 is NP-hard under the common assumption that $\mathrm{P} \neq \mathrm{NP}$.

Theorem 5.5.4. Ψ_0 *is* NP*-hard.*

Proof Sketch. To prove the theorem, it is sufficient to show that there exists a problem $\Psi_c \in \mathrm{NP}$-hard so that $\Psi_c \leq_p \Psi_0$. We show this by reducing 3SAT, which is a well-known NP-complete problem, to Ψ_0. To derive the reduction we consider two intermediate problems, namely Ψ_1 and Ψ_2. Especially, we aim to show that the following chain of reductions holds

$$\Psi_0 \geq_m \Psi_1 \geq_p \Psi_2 \geq_p \mathrm{3SAT}.$$

First, we define Ψ_1 and we prove that $\Psi_0 \geq_m \Psi_1$. Informally, Ψ_1 is the problem of finding a reward-maximizing Markovian policy $\pi_{\mathrm{M}} \in \Pi_{\mathrm{M}}$ w.r.t. the convex objective (5.2) encoded through a reward function in a convenient POMDP $\mathcal{M}_\Omega^R$. We can build $\mathcal{M}_\Omega^R$ from the MDP $\mathcal{M}$ similarly as the extended MDP $\widetilde{\mathcal{M}}_T^R$ (see Section 4.5.2 and the proof of Lemma 5.4.5 for details), except that the agent only accesses the observation space Ω instead of the extended state space $\mathcal{S}_\ell$. In particular, we define $\Omega = \mathcal{S}$ (note that $\mathcal{S}$ is the state space of the original MDP $\mathcal{M}$), and $O(s_\ell) = o$ is a deterministic function such that the given observation o is the last state in the history s_ℓ.

Then, the reduction $\Psi_0 \geq_m \Psi_1$ works as follows. We denote as $\mathcal{I}_{\Psi_i}$ the set of possible instances of Ψ_i. We show that Ψ_0 is harder than Ψ_1 by defining the polynomial-time functions ψ and ϕ such that any instance of Ψ_1 can be rewritten through ψ as an instance of Ψ_0, and a solution $\pi^*_{\mathrm{NM}} \in \Pi_{\mathrm{NM}}$ for Ψ_0 can be converted through ϕ into a solution $\pi^*_{\mathrm{M}} \in \Pi_{\mathrm{M}}$ for the original instance of Ψ_1. The function ψ sets $\mathcal{S} = \Omega$ and derives the transition model of $\mathcal{M}$ from the one of $\mathcal{M}_\Omega^R$, while ϕ converts the optimal solution of Ψ_0

by computing $\pi^*_{\mathrm{M}}(a|o,t) = \sum_{ho \in \mathcal{H}_o} p_T^{\pi^*_{\mathrm{NM}}}(ho)\pi^*_{\mathrm{NM}}(a|ho)$, where $\mathcal{H}_o$ stands for the set of histories $h \in \mathcal{H}_t$ ending in the observation $o \in \Omega$. Thus, we have that $\Psi_0 \geq_m \Psi_1$.

We now define Ψ_2 as the policy existence problem w.r.t. the problem statement of Ψ_1. Hence, Ψ_2 is the problem of stating whether the value of a reward-maximizing Markovian policy $\pi^*_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \mathcal{J}_{\mathcal{M}^R_\Omega}(\pi)$ is greater than 0. Since computing an optimal policy in POMDPs is in general harder than the relative policy existence problem (Lusena et al., 2001, Section 3), we have that $\Psi_1 \geq_p \Psi_2$. For the last reduction, i.e., $\Psi_2 \geq_p$ 3SAT, we extend the proof of Theorem 4.13 in (Mundhenk et al., 2000), which states that the policy existence problem for POMDPs is NP-complete. In particular, we show that this holds within the restricted class of POMDPs defined in Ψ_1. Since the chain $\Psi_0 \geq_m \Psi_1 \geq_p \Psi_2 \geq_p$ 3SAT holds, we have that $\Psi_0 \geq_p$ 3SAT. Since 3SAT $\in$ NP-complete, we can conclude that Ψ_0 is NP-hard. □

Having established the hardness of the optimization of Ψ_0, one could now question whether the problem Ψ_0 is instead easy to verify ($\Psi_0 \in$ NP), from which we would conclude that $\Psi_0 \in$ NP-complete. Whereas we doubt that this problem is significantly easier to verify than to optimize, the focus of this work is on its optimization version, and we thus leave as future work a finer analysis to show that $\Psi_0 \notin$ NP.

5.6 Discussion

In the previous sections, we detailed the importance of non-Markovianity when optimizing a single trial CRL objective, but we also proved that the corresponding optimization problem is NP-hard in its general formulation. Despite the hardness result, we believe that it is not hopeless to learn CRL policies with some form of non-Markovianity, while still preserving an edge over Markovian strategies. In the following paragraphs, we discuss potential avenues to derive practical methods for relevant relaxations to the general class of non-Markovian policies.

Finite-Length Histories Throughout the paper, we considered non-Markovian policies that condition their decisions on histories of arbitrary length, i.e., $\pi : \mathcal{H} \to \Delta(\mathcal{A})$. However, the complexity of optimizing such policies grows exponentially with the length of the history. To avoid this exponential blowup, one can define a class of non-Markovian policies $\pi : \mathcal{H}_H \to \Delta(\mathcal{A})$ in which the decisions are conditioned on histories of a finite length $T > 1$ that are obtained from a sliding window on the full history. The optimal policy within this class would still retain better gap guarantees than an optimal Markovian policy, but it would not achieve zero gap in general. With the length parameter H one can trade off the learning complexity with the gap according to the structure of the domain.

Compact Representations of the History Instead of setting a finite length H, one can choose to perform function approximation on the full history to obtain a class of policies $\pi : f(\mathcal{H}) \to \Delta(\mathcal{A})$, where f is a function that maps a history h to some compact representation. An interesting option is to use the notion of *eligibility traces* (Sutton & Barto, 2018) to encode the information of h in a vector of length S, which is updated as $\boldsymbol{z}_{t+1} \leftarrow \lambda \boldsymbol{z}_t + \mathbf{1}_{s_t}$, where $\lambda \in (0,1)$ is a discount factor, $\mathbf{1}_{s_t}$ is a vector with a unit entry at the index s_t, and $\boldsymbol{z}_0 = 0$. The discount factor λ acts as a smoothed version of the length parameter H, and it can be dynamically adapted while learning. Indeed, this eligibility

traces representation is particularly convenient for policy optimization (Deisenroth et al., 2013), in which we could optimize in turn a parametric policy over actions $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{z}, \lambda)$ and a parametric policy over the discount $\pi_{\boldsymbol{\nu}}(\lambda)$. To avoid a direct dependence on S, one can define the vector $\boldsymbol{z}$ over a discretization of the state space.

Deep Recurrent Policies Another noteworthy way to do function approximation on the history is to employ recurrent neural networks (Williams & Zipser, 1989; Hochreiter & Schmidhuber, 1997) to represent the non-Markovian policy. This kind of recurrent architecture is already popular in RL. In this section, we are providing the theoretical ground to motivate the use of deep recurrent policies to address CRL.

Non-Markovian Control with Tree Search In principle, one can get a realization of actions from the optimal non-Markovian policy without ever computing it, e.g., by employing a Monte-Carlo Tree Search (MCTS) (Kocsis & Szepesvári, 2006) approach to select the next action to take. Given the current state s_t as a root, we can build the tree of trajectories from the root through repeated simulations of potential action sequences. With a sufficient number of simulations and a sufficiently deep tree, we are guaranteed to select the optimal action at the root. If the horizon is too long, we can still cut the tree at any depth and approximately evaluate a leaf node with the value induced by the path from the root to the leaf. The drawback of this procedure is that we require to access a simulator with reset (or a reliable estimate of the transition model) to actually build the tree.

Finally, in this section, we focus on the gap between non-Markovian and Markovian policies, which can be either stationary or time-variant. Future works might consider the role of stationarity (see also Akshay et al., 2013; Laroche et al., 2022), such as establishing under which conditions stationary strategies are sufficient in this setting. Notably, here we focus on state distributions, but similar results could be extended to state-action distributions with minor modifications.

Part II

Introduction to State Entropy Maximization

CHAPTER 6

The State Entropy Objective

The content of this chapter is mostly based on the paper "Provably Efficient Maximum Entropy Exploration" by Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest.[1] The reported formalizations and results shall not be intended as our original contribution.

6.1 Introduction

In the previous chapters, we have analyzed the Convex RL (CRL) problem, which can be traced back to (Hazan et al., 2019; Zhang et al., 2020a), and we especially unveiled a significant mismatch between its infinite trials and finite trials formulation (Chapter 4), as well as the importance of considering non-Markovian policies to optimize the latter (Chapter 5). This preliminary study serves to provide essential context and theoretical ground for an instance of CRL that draws our interest, i.e., the *state entropy maximization* problem, which will be the focus of the remainder of this thesis. In Chapters 6, 7 we will introduce and characterize, from an optimization perspective, a formulation of the state entropy maximization problem that derives from infinite trials CRL. Then, in Chapters 8, 9 we provide methodologies to address the state entropy maximization problem in relevant practical scenarios, which are inherently akin to a finite trials CRL formulation.

In their seminal paper, Hazan et al. (2019) propose the entropy of the state distribution

[1] A complete reference can be found in the bibliography (Hazan et al., 2019).

induced by a policy as an objective for exploration in the absence of external rewards. Specifically, they formalize the objective function as

$$H(d^\pi) := - \mathop{\mathbb{E}}_{s\sim d^\pi} \big[\log d^\pi(s) \big],$$

where H denotes the Shannon entropy (Shannon, 1948), and d^π is the discounted state distribution induced by π. Since the entropy H is a concave function of d^π, this is an instance of the broader CRL framework (Hazan et al., 2019; Zhang et al., 2020a). The infinite-horizon formulation makes the problem analogous to an infinite trials CRL problem in episodic settings.[2] Although we have already mentioned the existence of provably efficient algorithms to solve the infinite trials CRL problem (e.g., Zhang et al., 2020a; Zahavy et al., 2021) in previous chapters, it is worth reporting some problem-specific considerations on the computational complexity of the state entropy maximization. Before diving into the computational aspects, we provide a brief detour on why state entropy maximization has emerged as a powerful objective for unsupervised RL.

Motivation While the principal motivation for considering the state entropy maximization has to be found in its remarkable empirical success (see Laskin et al., 2021), in which the development of methodologies that are suitable to practical domains (see Chapters 8, 9) certainly played a role, it is worth considering some (informal) technical corroborations that can at least partially explain its empirical prowess.

In the offline RL literature (Levine et al., 2020), which is the RL field dealing with the problem of learning a near-optimal policy from a batch of samples collected a priori, it is well-known (Antos et al., 2008; Chen & Jiang, 2019; Jin et al., 2021; Foster et al., 2021; Zhan et al., 2022) that the *coverage* of the state space in the given batch fundamentally affects the sample complexity of the problem. The coverage condition is often expressed through a *concentrability* coefficient

$$C(\mathcal{D}) := \sup_{\pi\in\Pi, s\in\mathcal{S}} \frac{d^\pi(s)}{\mathcal{D}(s)},$$

where $\mathcal{D}$ denotes the data distribution within the batch. The intuition behind concentrability is that good coverage of the state space is necessary to have sufficient information to find out the optimal action in every state with high probability. More recently, the work by Xie et al. (2021) explicitly links the coverage condition of the initial policy to the sample complexity of the fine-tuning task, while Xie et al. (2022) generalizes the importance of coverage conditions to the online RL setting. Notably, the problem of unsupervised policy pre-training is akin to finding a policy inducing an optimal data distribution $\mathcal{D}$. Whereas a direct minimization of the concentrability coefficient in the space of all the data collection strategies (i.e., policies) is arguably far-fetched, maximizing the entropy of the data collection strategy to produce a flat $\mathcal{D}$ can be seen as a viable surrogate objective.

Other relevant problem formulations for which state entropy maximizing policies have been demonstrated to be provably efficient, albeit sub-optimal in comparison to problem-specific solutions, are the *reward discovery* problem (Tarbouriech et al., 2021) and the

[2] To see this point, let us consider a long trajectory from the discounted Markov chain induced by π. Since the dependence between the samples vanishes with their distance in time, we can extract imaginary episodes from the long trajectory, and then compute the average of the state visitations across episodes to obtain the discounted state distribution d^π.

reward-free RL formulation (Jin et al., 2020). The former can be formulated as the minimization of the number of exploration steps required to visit every state-action pair at least once (with high probability). The latter instead asks for an exploration phase that acquires enough information to be able to extract a near-optimal policy with high probability for every possible reward function.

Contents The chapter is organized as follows. In Section 6.2, we focus on the primal formulation of the state entropy maximization problem. We especially show that the primal optimization problem, albeit concave in the state distribution, is non-concave in the policy parameters, which makes direct optimization intractable. Then, in Section 6.2.1, we describe a general methodology to split the primal objective into a sequence of tractable optimization problems through the conditional gradient method. Finally, in Section 6.3, we discuss the dual formulation of the state entropy maximization problem. We can show that the dual formulation is amenable to optimization, but it is hardly suitable for scaling to more complex scenarios that require function approximation.

6.2 The Primal Formulation

In the primal formulation of the state entropy objective we directly look for the policy parameters that maximize the entropy of the induced state distribution. The resulting optimization problem can be written as follows.

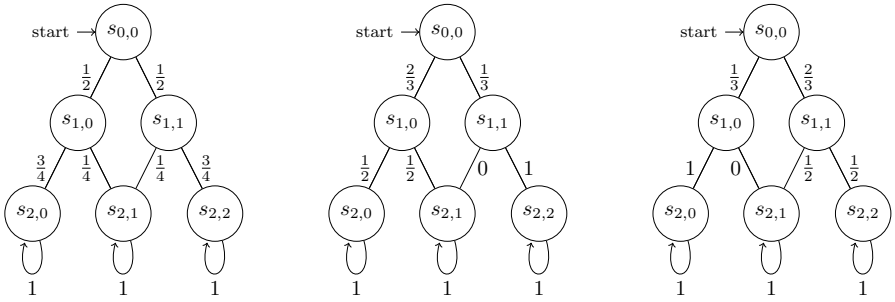
(Primal) State Entropy Maximization

$$\max_{\pi \in \Pi} \; H\big(d^\pi\big) \tag{6.1}$$

Although (6.1) may look amenable to optimization, being formulated as a maximization of a concave objective function, the relation between the optimization variables $\pi(a|s)$ and the objective function $H(d^\pi)$ is actually convoluted, as it holds the recursive relation

$$d^\pi(s) = (1-\gamma)\mu(s) + \gamma \int_{\mathcal{S}} \int_{\mathcal{A}} d^\pi(s')\pi(a'|s')P(s|s',a') \,\mathrm{d}s' \,\mathrm{d}a'.$$

Hazan et al. (2019) were able to show that the objective function is actually non-concave in the policy parameters, which means that directly optimizing the primal formulation (6.1) is intractable in general. For the sake of completeness, here we report their hard instance, which is remarkably easy to interpret (Figure 6.1). This is a rather negative result for the prospect of directly optimizing a policy for the primal state entropy objective, such as through a reworked policy gradient method (Sutton et al., 1999). Indeed, following the gradient of the state entropy w.r.t. the policy parameters, i.e., $\nabla_\pi H(d^\pi)$, would not guarantee eventual convergence to the global optimum due to the non-concave objective. In the next section, we report a method designed by Hazan et al. (2019) to address the non-concave objective through a conditional gradient method, while in Chapter 7 we discuss surrogate objectives that make the primal formulation amenable to optimization.



(a) $\pi_0,\ d^{\pi_0} = \left(\frac{3}{8}, \frac{1}{4}, \frac{3}{8}\right)$ **(b)** $\pi_1,\ d^{\pi_1} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ **(c)** $\pi_2,\ d^{\pi_2} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

Figure 6.1: *(Hazan et al., 2019). State distribution d^π induced by π_0, π_1, π_2 in a six-state deterministic MDP with binary actions. Each label denotes the probability of taking the corresponding action under the respective policy. Note that the discounted state distribution converges to the distribution of the bottom stage of the MDP for $\gamma \to 1$. Crucially, the policy $\pi_0 = (\pi_1 + \pi_2)/2$ induces a lower state entropy of the other two policies $H(d^{\pi_0}) < (H(d^{\pi_1}) + H(d^{\pi_2}))/2$, which means $H(d^\pi)$ is non-concave in π.*

6.2.1 MaxEnt: A Frank-Wolfe Approach

In this section, we comment on a methodology proposed by Hazan et al. (2019)[3] to sidestep the non-concavity of the objective function (6.1) through a conditional gradient method (also known as the Frank-Wolfe algorithm (Frank & Wolfe, 1956)). The approach, named *MaxEnt*, is based on formulating the (intractable) primal optimization problem into a sequence of tractable sub-problems, whose solutions can be combined to solve the original primal state entropy maximization. Those sub-problems cast each policy update as the problem of solving an MDP built on the original CMP by taking the reward function $R(s) = \big(\nabla_\pi H(d^\pi_{\text{mix}})\big)(s)$, where d^π_{mix} is a mixture of policies that is obtained from a convex combination of the solutions of previous sub-problems. Algorithm 5 reports the pseudocode of the MaxEnt algorithm, a more detailed description of the procedure, together with implementation details, can be found in (Hazan et al., 2019).

Assuming full knowledge of the MDP, especially of the transition model P, Hazan et al. (2019) show that MaxEnt is guaranteed to output a policy π_{mix} such that $H(d^{\pi_{\text{mix}}}) \geq \max_{\pi \in \Pi} H(d^\pi) - \epsilon$ within a number of iterations T that is $\text{POLY}\big(|\mathcal{S}|, |\mathcal{A}|, \frac{1}{\epsilon}, \frac{1}{1-\gamma}\big)$.[4] Thus, the approach is computationally efficient. When the transition model P is not known, the MaxEnt procedure is still viable, but it requires bot a provably efficient density estimator to obtain $\hat{d}^{\pi_{\text{mix}}}$ and a provably efficient planning solver to compute π_t. Assuming the existence of these two modules, the MaxEnt algorithm is guaranteed to output a nearly-optimal policy π_{mix} by taking $\tilde{O}\big(|\mathcal{S}|^2|\mathcal{A}|/\epsilon^3(1-\gamma)^2\big)$ samples from the environment, which makes the algorithm statistically efficient.

[3] A similar procedure has been concurrently studied in (Tarbouriech & Lazaric, 2019) for the problem of active exploration in MDPs.

[4] Note that the space of Markovian policies Π and the space of mixture of Markovian policies, which we can denote as Π_{mix}, are equally expressive in terms of state distribution they can induce, thus $\max_{\pi \in \Pi} H(d^\pi) = \max_{\pi_{\text{mix}} \in \Pi_{\text{mix}}} H(d^{\pi_{\text{mix}}})$.

Algorithm 5 MaxEnt (Hazan et al., 2019)

Input: Step size α, number of iterations T, tolerance ϵ
Take π_0 arbitrarily and set $\eta_0 = 1, C_0 = \{\pi_0\}, \pi_{\text{mix}} = \langle \eta_0, C_0 \rangle$
for $t = 0, \dots, T-1$ **do**
  Estimate the state distribution $\hat{d}^{\pi_{\text{mix}}}$ induced by π_{mix} up to error ϵ
  Compute the reward function $R(s) = \big(\nabla_{\pi_{\text{mix}}} H(\hat{d}^{\pi_{\text{mix}}})\big)(s)$ for all $s \in \mathcal{S}$
  Compute the ϵ-optimal policy π_t given the reward R through approximate value iteration
  Update the mixture $\pi_{\text{mix}} = \langle \eta_{t+1}, C_{t+1} \rangle$ where:

$$\eta_{t+1} = ((1-\alpha)\eta_t, \alpha)$$
$$C_{t+1} = (\pi_0, \dots, \pi_t)$$

end for
Output: State entropy maximizing policy π_{mix}

Although the MaxEnt algorithm is provably efficient both statistically and computationally, since the sample complexity is polynomial in all the relevant quantities and the algorithm runs in polynomial time, some considerations are due. First, the MaxEnt algorithm outputs a mixture of policies instead of a single Markovian policy. While the mixture can be distilled into a corresponding Markovian policy inducing a similar state distribution, the process could result expensive in practice. Alternatively, one could initialize the fine-tuning task with the mixture directly, but most of the existing RL algorithms are not designed to work with mixtures of policies. On the other hand, while it is not hard to obtain an effective density estimator in tabular domains, density estimation over continuous and possibly high-dimensional spaces is known to be hardly tractable, which casts some doubts over the scalability of MaxEnt. In the remainder of the thesis, we will both consider alternative tractable formulations of the primal state entropy objective (Chapter 7), as well as practical methodologies that gracefully scale to challenging continuous domains (Chapter 8). Before approaching these topics, it is worth introducing a dual formulation of the state entropy maximization problem, which we present in the next section.

6.3 The Dual Formulation

Since the objective function of (6.1) is essentially concave, but the convoluted relationship between the policy parameters and the state distribution makes the optimization problem ultimately non-concave, it is interesting to look for a dual formulation. In the dual formulation, we reverse roles between the optimization variables and the constraints by directly acting on the state distribution to maximize the objective function, while ensuring the optimization variables remain in the set of admissible state distributions. For this purpose, we first define the set of admissible state-action distributions as

$$\mathcal{K} = \Bigg\{ \omega \in \Delta(\mathcal{S} \times \mathcal{A}) \ : \ \sum_{a \in \mathcal{A}} \omega(s,a) = (1-\gamma)\mu(s) + \gamma \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} P(s|s',a')\omega(s',a'), \ \forall s \in \mathcal{S} \Bigg\}.$$

With a slight abuse of notation, we define $\boldsymbol{\omega} := \sum_{a \in \mathcal{A}} \omega(\cdot, a)$. Then, we formalize the dual optimization problem as follows.[5]

(Dual) State Entropy Maximization

$$\max_{\omega \in \mathcal{K}} \; H(\boldsymbol{\omega}) \tag{6.2}$$

Since the set of admissible state distribution $\mathcal{K}$ is known to be convex (Puterman, 2014), (6.2) is a linearly constrained concave problem with $|\mathcal{S}||\mathcal{A}|$ optimization variables and $2|\mathcal{S}||\mathcal{A}| + |\mathcal{S}|$ linear constraints, which is definitely amenable to optimization. Having obtained $\omega^* \in \arg\max_{\omega \in \mathcal{K}} H(\boldsymbol{\omega})$ through any convex solver, we can then recover a state entropy maximizing policy π_{ω^*} as

$$\pi_{\omega^*}(a|s) = \frac{\omega^*(s,a)}{\sum_{a' \in \mathcal{A}} \omega^*(s,a')}, \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A},$$

which is efficient to compute. Although the dual formulation (6.2) provides a tractable and technically compelling perspective of the state entropy maximization problem, it comes with its own shortcomings. While the dual problem is efficient to solve in small tabular domains, its computational complexity grows at a fast rate with the number of optimization variables in comparison to the linear program formulation of MDPs (Grötschel et al., 1993). Moreover, it is unclear how to scale the dual problem to continuous and possibly high-dimensional settings, in which the optimization variables grow to infinity.

[5]The dual problem takes inspiration from the linear program formulation of the MDP (Schweitzer & Seidmann, 1985; De Farias & Van Roy, 2003), as well as the work by (Tarbouriech & Lazaric, 2019) that presents a similar formulation in the setting of active exploration in MDPs.

CHAPTER 7

Optimization of the State Entropy Objective

The content of this chapter is based on the paper "An Intrinsically-Motivated Approach for Learning Highly Exploring and Fast Mixing Policies" co-authored with Marcello Restelli, published at AAAI 2020.[1]

7.1 Introduction

In the previous chapter, we have formalized the state entropy objective for unsupervised pre-training of exploration policies. Before addressing the learning problem associated with the introduced objective, which will be the focus of the remainder of this thesis, it is worth investigating the problem from an optimization perspective. As we reported in Section 6.2, the *primal* formulation of the state entropy maximization problem is non-concave, which makes the problem of directly optimizing an entropy maximizing policy mostly intractable. Hazan et al. (2019) show how to sidestep this inherent hardness through a Frank-Wolfe procedure, in which the primal optimization is split in a sequence of optimization problems such that their resulting policies are combined in a mixture (Section 6.2.1). Alternatively, one can write an equivalent *dual* formulation of the problem that is amenable to optimization (Section 6.3). While the dual formulation is interesting from a theoretic perspective, building a practical methodology to optimize the dual variables is far-fetched.

[1] A complete reference can be found in the bibliography (Mutti & Restelli, 2020).

Whether there exist tractable methods to address the primal formulation directly, which is instead closer to practice, remains an open question.

In this chapter, we aim to circumvent the fundamental intractability of the primal formulation of the state entropy maximization problem, presenting a family of surrogate primal objectives that are amenable to optimization. These surrogate formulations are based on a well-known result in the Markov chains literature, which states that a chain for which the transition matrix is *doubly stochastic*[2] converges to a uniform state distribution (hence, entropy maximizing). Taking inspiration from this result, we design a family of optimization problems in which we directly look for a policy inducing a state transition matrix that is close to a doubly stochastic. These problems are computationally tractable, and they often admit a linear program formulation. We crucially show that optimizing these problems is equivalent to maximizing a lower bound to the state entropy induced by the policy. The latter result allows us to conclude that a convenient reformulation of the primal state entropy objective can be directly optimized, which we believe is a crucial milestone before addressing how to design practical methodologies to learn state entropy maximizing policies through interactions with an unknown environment.

Contents The chapter is organized as follows. We start recalling useful definitions and the matrix notation of MDPs (Section 7.2). In Section 7.3, we present a family of tractable optimization problems for primal state entropy maximization. Specifically, we first propose surrogate formulations of the state entropy objective (Section 7.3.1), then we show how we can account also for the entropy over actions (Section 7.3.2) and the mixing time of the resulting policy (Section 7.3.3) through convenient linear constraints. In Section 7.4, we provide a simple model-based procedure employing the presented surrogate problems as building blocks. Finally, in Section 7.5, we deploy the procedure on a set of toy domains to numerically validate the surrogate objectives. The proofs of the theorems can be found in Appendix A.3.

7.2 Definitions

In the following, we will indifferently turn to scalar or matrix notation, where $\boldsymbol{v}$ denotes a vector, $\boldsymbol{M}$ denotes a matrix, and $\boldsymbol{v}^T$, $\boldsymbol{M}^T$ denote their transpose. A matrix is row (column) stochastic if it has non-negative entries and all of its rows (columns) sum to one. A matrix is *doubly stochastic* if it is both row and column stochastic. We denote with $\mathbb{P}$ the space of doubly stochastic matrices. The L_∞-norm $\|\boldsymbol{M}\|_\infty$ of a matrix is its maximum absolute row sum, while $\|\boldsymbol{M}\|_2 = \left(\max \text{eig } \boldsymbol{M}^T\boldsymbol{M}\right)^{\frac{1}{2}}$ and $\|\boldsymbol{M}\|_F = \left(\sum_i \sum_j (\boldsymbol{M}(i,j))^2\right)^{\frac{1}{2}}$ are its L_2 and Frobenius norms respectively. We denote with $\mathbf{1}_n$ a column vector of n ones and with $\mathbf{1}_{n\times m}$ a matrix of ones with n rows and m columns.

We now introduce a matrix notation for MDPs. The initial state distribution is represented through $\boldsymbol{d}_0$, which is a column vector of size $|\mathcal{S}|$ having elements $d_0(s)$, $\boldsymbol{P}$ is a row stochastic matrix of size $(|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|)$ that describes the transition model $\boldsymbol{P}((s,a),s') = P(s'|s,a)$, $\boldsymbol{\Pi}$ is a row stochastic matrix of size $(|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|)$ that contains the policy $\boldsymbol{\Pi}(s,(s,a)) = \pi(a|s)$, and $\boldsymbol{P}^\pi = \boldsymbol{\Pi}\boldsymbol{P}$ is a row stochastic matrix of size $(|\mathcal{S}| \times |\mathcal{S}|)$ that represents the state transition matrix under policy π. We denote with Π the space of all the

[2] A matrix is doubly stochastic if all of its entries are non-negative and both its rows and columns sum to one.

stationary Markovian policies.

Fixing a policy π over an MDP, we obtain a Markov Chain (MC) (Levin & Peres, 2017) with transition kernel $P^\pi(s'|s) = \boldsymbol{P}^\pi(s, s')$. Having defined the t-step transition matrix as $\boldsymbol{P}_t^\pi = (\boldsymbol{P}^\pi)^t$, the state distribution of the MC at time step t is $\boldsymbol{d}_t^\pi = (\boldsymbol{P}_t^\pi)^T \boldsymbol{d}_0$, while $\boldsymbol{d}^\pi = \lim_{t\to\infty} \boldsymbol{d}_t^\pi$ is the steady state distribution. If the MC is ergodic, i.e., aperiodic and recurrent, it admits a unique steady-state distribution, such that $\boldsymbol{d}^\pi = (\boldsymbol{P}^\pi)^T \boldsymbol{d}^\pi$. The mixing time t_{mix} of the MC describes how fast the state distribution converges to the steady state

$$t_{\text{mix}} = \min\left\{t \in \mathbb{N} : \sup_{\boldsymbol{d}_0} \left\|\boldsymbol{d}_t^\pi - \boldsymbol{d}^\pi\right\|_\infty \leq \epsilon\right\}, \tag{7.1}$$

where ϵ is the mixing threshold. An MC is reversible if the condition $\boldsymbol{P}^\pi \boldsymbol{d}^\pi = (\boldsymbol{P}^\pi)^T \boldsymbol{d}^\pi$ holds. Let $\boldsymbol{\lambda}_\pi$ be the eigenvalues of $\boldsymbol{P}^\pi$. For ergodic reversible MCs the largest eigenvalue is 1 with multiplicity 1. Then, we can define the second largest eigenvalue modulus $\lambda_\pi(2)$ and the spectral gap γ_π as:

$$\lambda_\pi(2) = \max_{\lambda_\pi(i)\neq 1} |\lambda_\pi(i)|, \qquad \gamma_\pi = 1 - \lambda_\pi(2). \tag{7.2}$$

7.3 Optimization Problems for Entropy Maximization

In this section, we define a set of optimization problems whose goal is to directly provide a stationary Markovian policy that maximizes the state entropy while ensuring sufficient exploration over actions and accounting for the mixing time. The optimization problem is introduced in three steps: First, we ask for a policy that maximizes some lower bound to the steady-state distribution entropy, then we foster exploration over the action space by adding a linear constraint on the minimum action probability, and finally we add another linear constraint to reduce the mixing time of the Markov chain induced by the policy.

7.3.1 Entropy Maximization over the State Space

As we already noted, direct optimization of the primal formulation of the state entropy maximization objective is not tractable (Section 6.2). However, one can turn to a Frank-Wolfe procedure to get a state entropy maximizing mixture of policies (Section 6.2.1), or a dual formulation to get the state distribution induced by a state entropy maximizing policy (Section 6.3). Here, we instead follow an alternative route that consists in directly optimizing a stationary Markovian policy that maximizes a lower bound to the state distribution entropy. In particular, in the following, we will consider three versions of the lower bound that lead to as many optimization problems (named Infinity, Frobenius, Column Sum) that show different trade-offs between tightness of the guarantee and computational complexity of the optimization.

Infinity

From the theory of Markov chains (Levin & Peres, 2017), we know a necessary and sufficient condition for a policy to induce a uniform steady-state distribution (i.e., to achieve the maximum possible entropy). We report this result in the following theorem.

Theorem 7.3.1. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP. The steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is uniform over $\mathcal{S}$ iff the matrix $\boldsymbol{P}^\pi = \boldsymbol{\Pi}\boldsymbol{P}$ is doubly stochastic.*

Unfortunately, given the constraints specified by the transition matrix $\boldsymbol{P}$, a stationary Markovian policy that induces a doubly stochastic $\boldsymbol{P}^\pi$ may not exist. On the other hand, it is possible to lower bound the entropy of the steady-state distribution induced by policy π as a function of the minimum L_∞-norm between $\boldsymbol{P}^\pi$ and any doubly stochastic matrix.

Theorem 7.3.2. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP and $\mathbb{P}$ the space of doubly stochastic matrices. The entropy of the steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^\pi) \geq \log|\mathcal{S}| - |\mathcal{S}| \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_\infty^2 .$$

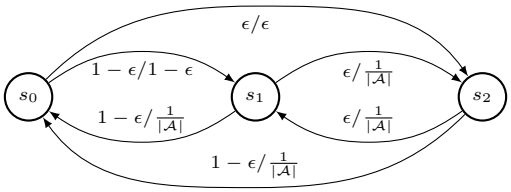
One can maximize the presented lower bound to obtain a policy that is guaranteed to achieve a state distribution entropy that is at least as good as the value of the lower bound. Intuitively, this requires finding a policy matrix π that drives the matrix $\boldsymbol{\Pi}\boldsymbol{P}$ closer to a doubly stochastic matrix $\boldsymbol{P}^u$ of any choice. The latter intuition can be formally translated in the following constrained optimization problem:

$$\underset{\boldsymbol{P}^u \in \mathbb{P}, \boldsymbol{\Pi} \in \Pi}{\text{minimize}} \quad \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_\infty \tag{7.3}$$

It is worth noting that this optimization problem can be reformulated as a linear program:

$$\begin{aligned}
\underset{\boldsymbol{P}^u, \boldsymbol{\Pi}, v}{\text{minimize}} \quad & v \\
\text{subject to} \quad & \sum_{s' \in \mathcal{S}} |P^u(s'|s) - P^\pi(s'|s)| \leq v, \quad \forall s \in \mathcal{S}, \\
& \boldsymbol{\Pi}(s,(s,a)) \geq 0, \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}, \\
& P^u(s'|s) \geq 0, \quad \forall s \in \mathcal{S}, \quad \forall s' \in \mathcal{S}, \\
& \sum_{a \in \mathcal{A}} \boldsymbol{\Pi}(s,(s,a)) = 1, \quad \forall s \in \mathcal{S}, \\
& \sum_{s' \in \mathcal{S}} P^u(s'|s) = 1, \quad \forall s \in \mathcal{S}, \\
& \sum_{s' \in \mathcal{S}} P^u(s|s') = 1, \quad \forall s \in \mathcal{S}.
\end{aligned} \tag{7.4}$$

The first set of inequality constraints can be transformed into a set of linear inequality constraints. Each constraint is obtained by removing the absolute values and considering a different permutation of the signs in front of the terms in the summation. As a result, if the original summation contains n elements, the number of linear constraints is 2^n. Since this process needs to be done for each state $s \in \mathcal{S}$, the first set of constraints can be replaced by $2^{|\mathcal{S}|}|\mathcal{S}|$ constraints. Thus, the linear program has $|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{A}| + |\mathcal{S}|$ optimization variables, $2^{|\mathcal{S}|}|\mathcal{S}| + |\mathcal{S}|^2 + |\mathcal{S}||\mathcal{A}|$ inequality constraints and $3|\mathcal{S}|$ equality constraints. In order to avoid the exponential growth of the number of constraints as a function of the number of states, we are going to introduce alternative optimization problems.



	$\boldsymbol{P}_1$	$\boldsymbol{P}_2$
Infinite	0.60	0.60
Frobenius	0.73	0.42
$H(\boldsymbol{d}^\pi)$	0.93	0.99

Figure 7.1: *Graphical representation of a Markov chain (left), having on the edges the transition probabilities $\boldsymbol{P}_1(s_i, s_j)/\boldsymbol{P}_2(s_i, s_j)$. On the right, a table providing the values of L_∞-norm, and Frobenius norm of the difference w.r.t. a uniform $\boldsymbol{P}^u$, along with state distribution entropies.*

Frobenius

It is worth noting that different transition matrices $\boldsymbol{P}^\pi$ having equal $\|\boldsymbol{P}^u - \boldsymbol{P}^\pi\|_\infty$ might lead to significantly different state distribution entropies $H(\boldsymbol{d}^\pi)$, as the L_∞-norm only accounts for the state corresponding to the maximum absolute row sum. The Frobenius norm can better capture the distance between $\boldsymbol{P}^u$ and $\boldsymbol{P}^\pi$ over all the states, as we can see from the simple example in Figure 7.1. For this reason, we have derived a lower bound to the policy entropy that replaces the L_∞-norm with the Frobenius one.

Theorem 7.3.3. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP and $\mathbb{P}$ the space of doubly stochastic matrices. The entropy of the steady-state distribution $\mathbf{d}^\pi$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^\pi) \geq \log|\mathcal{S}| - |\mathcal{S}|^2 \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_F^2 .$$

It can be shown that the lower bound based on the Frobenius norm cannot be tighter (i.e., larger) than the one with the Infinite norm.

Corollary 7.3.4. *The bound in Theorem 7.3.2 is never less than the bound in Theorem 7.3.3.*

However, we have the advantage that the resulting optimization problem has significantly fewer constraints than Problem (7.3):

$$\underset{\boldsymbol{P}^u \in \mathbb{P}, \boldsymbol{\Pi} \in \Pi}{\text{minimize}} \quad \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_F . \tag{7.5}$$

This problem is a (linearly constrained) quadratic problem with $|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{A}|$ optimization variables and $|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{A}|$ inequality constraints and $3|\mathcal{S}|$ equality constraints.

Column Sum

Problems (7.3) and (7.5) aim at finding a policy associated with a state transition matrix that is doubly stochastic. To achieve this result it is enough to guarantee that the column sums of the matrix $\boldsymbol{P}^\pi$ are all equal to one (Kirkland, 2010). A measure that can

be used to evaluate the distance to a doubly stochastic matrix can be the absolute sum of the difference between one and the column sums: $\sum_{s\in\mathcal{S}} |1 - \sum_{s'\in\mathcal{S}} P^{\pi}(s|s')| = \left\| \left(\boldsymbol{I} - (\boldsymbol{\Pi P})^{\boldsymbol{T}}\right) \cdot \mathbf{1}_{|\mathcal{S}|} \right\|_1$. The following theorem provides a lower bound to the policy entropy as a function of this measure.

Theorem 7.3.5. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP. The entropy of the steady-state distribution $\boldsymbol{d}^{\pi}$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^{\pi}) \geq \log|\mathcal{S}| - |\mathcal{S}| \left\| \left(\boldsymbol{I} - (\boldsymbol{\Pi P})^{\boldsymbol{T}}\right) \cdot \mathbf{1}_{|\mathcal{S}|} \right\|_1^2 .$$

The optimization of this lower bound leads to the following linear program:

$$\underset{\boldsymbol{\Pi}\in\Pi}{\text{minimize}} \quad \left\| \left(\boldsymbol{I} - (\boldsymbol{\Pi P})^{\boldsymbol{T}}\right) \cdot \mathbf{1}_{|\mathcal{S}|} \right\|_1 . \tag{7.6}$$

Unlike the other optimization problems presented, Problem (7.6) does not require to optimize over the space of all the doubly stochastic matrices, thus significantly reducing the number of optimization variables ($|\mathcal{S}| + |\mathcal{S}||\mathcal{A}|$) and constraints ($2|\mathcal{S}| + |\mathcal{S}||\mathcal{A}|$ inequalities and $|\mathcal{S}|$ equalities). The explicit linear program formulation is given by

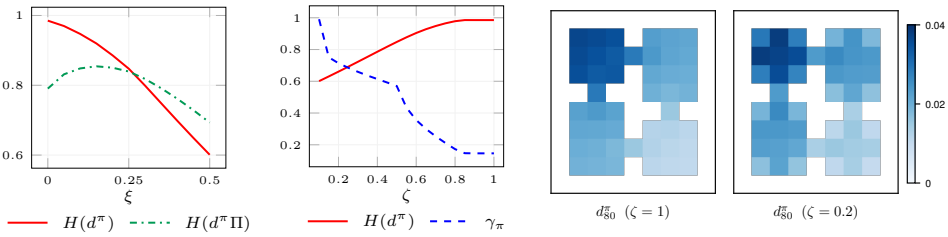
$$\begin{aligned}
\underset{\boldsymbol{\Pi},\boldsymbol{v}}{\text{minimize}} \quad & \sum_{s\in\mathcal{S}} v(s) \\
\text{subject to} \quad & 1 - \sum_{s'\in\mathcal{S}} P^{\pi}(s|s') \leq v(s), \quad \forall s \in \mathcal{S}, \\
& \sum_{s'\in\mathcal{S}} P^{\pi}(s|s') - 1 \leq v(s), \quad \forall s \in \mathcal{S}, \\
& \boldsymbol{\Pi}(s,(s,a)) \geq 0, \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}, \\
& \sum_{a\in\mathcal{A}} \boldsymbol{\Pi}(s,(s,a)) = 1, \quad \forall s \in \mathcal{S}.
\end{aligned} \tag{7.7}$$

7.3.2 Entropy Maximization over the State and Action Space

Although the policy resulting from the optimization of one of the above problems may lead to the most uniform exploration of the state space, exploring the action space might be just as crucial to support policy learning in the supervised fine-tuning phase. Thus, it is fair to wonder whether these pre-trained policies adequately cover the action space $\mathcal{A}$ in any visited state. Unfortunately, the optimization of Problems (7.3), (7.5), (7.6) does not guarantee even that the obtained policy is stochastic. However, we can embed in the problem a secondary objective that takes into account the exploration over $\mathcal{A}$. This can be done by enforcing a minimal entropy over actions in the policy to be learned, adding to (7.3), (7.5), (7.6) the following constraints:

$$\pi(a|s) \geq \xi, \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}, \tag{7.8}$$

where $\xi \in [0, \frac{1}{|\mathcal{A}|}]$. This secondary objective is actually in competition with the objective of uniform exploration over states. Indeed, an overblown incentive in the exploration over actions may limit the state distribution entropy of the optimal policy. Having a low



(a) *Varying* ξ **(b)** *Varying* ζ **(c)** *Varying* ζ

Figure 7.2: *Comparison of state distribution entropy* $H(\boldsymbol{d}^\pi)$ *and state-action distribution entropy* $H(\boldsymbol{d}^\pi \boldsymbol{\Pi})$ *for different values of* ξ *on the Single Chain domain* ***(a)****. Comparison of state distribution entropy* $H(\boldsymbol{d}^\pi)$ *and spectral gap* γ_π *for different values of* ζ *on the Single Chain domain* ***(b)****. Color-coded state distribution overlaid on a* 4*-rooms gridworld for different values of* ζ ***(c)****.*

probability of visiting a state decreases the likelihood of sampling an action from that state, hence, also reducing the exploration over actions. To illustrate that, Figure 7.2a shows state distribution entropies, i.e., $H(\boldsymbol{d}^\pi)$, and state-action distribution entropies, i.e., $H(\boldsymbol{d}^\pi \boldsymbol{\Pi})$, achieved by the optimal policy w.r.t. Problem (7.5) on the Single Chain domain (Furmston & Barber, 2010) for different values of ξ.

7.3.3 Entropy Maximization and Mixing Time

In many cases, such as in episodic tasks where the horizon for exploration is capped, we may have interest in trading inferior state entropy at the steady state for faster convergence to that steady state. Although the doubly stochastic matrices are equally valid in terms of steady-state distribution, the choice of the target $\boldsymbol{P}^u$ strongly affects the mixing properties of the $\boldsymbol{P}^\pi$ induced by the policy. Indeed, while an MC with a uniform transition matrix, i.e., transition probabilities $\boldsymbol{P}^u(s, s') = \frac{1}{|\mathcal{S}|}$ for any s, s', mixes in no time, an MC with probability one on the self-loops never converges to a steady state. This is evident considering that the mixing time t_{mix} of an MC is trapped as follows (Levin & Peres, 2017, Theorems 12.3 and 12.4):

$$\frac{1 - \gamma_\pi}{\gamma_\pi} \log \frac{1}{2\epsilon} \leq t_{\text{mix}} \leq \frac{1}{\gamma_\pi} \log \frac{1}{d^\pi_{\text{min}} \epsilon}, \tag{7.9}$$

where ϵ is the mixing threshold, d^π_{min} is a minorization of $\boldsymbol{d}^\pi$, and γ_π is the spectral gap of $\boldsymbol{P}^\pi$ (7.2). From the literature of MCs, we know that a variant of the Problems (7.3), (7.5) having the uniform transition matrix as target $\boldsymbol{P}^u$ and the L_2 as matrix norm, is equivalent to the problem of finding the fastest mixing transition matrix $\boldsymbol{P}^\pi$ (Boyd et al., 2004). However, the choice of this target may overly limit the entropy over the state distribution induced by the optimal policy. Instead, we look for a generalization that allows us to prioritize fast exploration at will. Thus, we consider a continuum of relaxations in the fastest mixing objective by embedding in Problems (7.3) and (7.5) (but not in Problem (7.6)) the following constraints:

$$\boldsymbol{P}^u(s, s') \leq \zeta, \quad \forall s, s' \in \mathcal{S}, \tag{7.10}$$

Algorithm 6 IDE^{3}AL

Input: ξ, ζ, batch size N
Initialize π_0 and transition counts $C \in \mathbb{N}^{|\mathcal{S}|^2 \times |\mathcal{A}|}$
for $i = 0, 1, 2, \dots,$ until convergence **do**
  Collect N steps with π_i and update C
  Estimate the transition model as:

$$\widehat{P}_i(s'|s,a) = \begin{cases} \frac{C(s'|s,a)}{\sum_{s'} C(s'|s,a)}, & \text{if } C(\cdot|s,a)>0 \\ 1/|\mathcal{S}|, & \text{otherwise} \end{cases}$$

  $\pi_{i+1} \leftarrow$ optimal policy for (7.3) (or (7.5) or (7.6)) given the parameters ξ, ζ, and matrix $\widehat{P}_i$
end for
Output: maximum entropy policy π_i

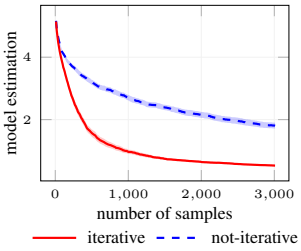


Figure 7.3: *Model estimation error on the Double Chain with $\xi = 0.1$, $\zeta = 0.7$, $N = 10$ (100 runs, 95% c.i.).*

where $\zeta \in [\frac{1}{|\mathcal{S}|}, 1]$. By setting $\zeta = \frac{1}{|\mathcal{S}|}$, we force the optimization problem to consider the uniform transition matrix as a target, thus aiming to reduce the mixing time, while larger values of ζ relax this objective, allowing us to get a higher steady-state distribution entropy. In Figure 7.2b we show how the parameter ζ affects the trade-off between high steady-state entropy and low mixing times (i.e., high spectral gaps), reporting the values obtained by optimal policies w.r.t. Problem (7.5) for different ζ.

7.4 A Model-Based Algorithm for Entropy Maximization

In this section, we provide a straightforward model-based approach, called *Intrinsically-Driven Effective and Efficient Exploration ALgorithm* (IDE^{3}AL), to learn a maximum entropy policy through the proposed optimization problems while interacting with an unknown environment. Since Problems (7.3), (7.5), (7.6) requires an explicit representation of the matrix $\boldsymbol{P}$, we need to estimate the transition model from samples before optimizing the objective (model-based approach). In tabular settings, this can be easily done by adopting the experienced transition frequencies as a proxy for the (unknown) transition probabilities, obtaining an estimated transition model $\widehat{P}(s'|s,a)$. In hard-exploration settings, it can be arbitrarily arduous to sample transitions from some of the states by relying on naïve exploration mechanisms, such as a random policy. To address this issue, we lean on an iterative approach in which we alternate model estimation phases with optimization sweeps of the objectives (7.3), (7.5), or (7.6). In this way, we combine the benefit of collecting samples with entropy maximizing policies to better estimate the transition model and the benefit of having a better-estimated model to learn superior entropy maximizing policies. In order to drive the policy towards (s,a) pairs that have never been sampled, we keep their corresponding distribution $\widehat{P}(\cdot|s,a)$ to be uniform over all possible states, thus making the pair (s,a) particularly valuable in the perspective of the optimization problem.[3] The algorithm converges whenever the exploratory policy remains unchanged during consecutive optimization sweeps and, if we know the size of the MDP, when all

[3]This trick is analogous to the R-max trick (Brafman & Tennenholtz, 2002), which assigns the maximum reward to never visited state-actions, translated to the maximum state entropy framework.

	ξ	$H(\boldsymbol{d}^\pi)$	$\min \boldsymbol{d}^\pi$
Frobenius		0.98	0.064
Infinity	0	0.98	0.064
Column Sum		0.98	0.06
Frobenius		0.94	0.041
Infinity	0.1	0.89	0.026
Column Sum		0.95	0.038

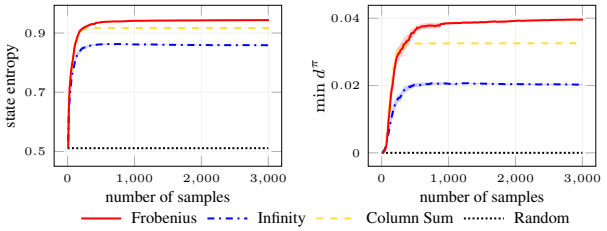


Figure 7.4: *State distribution entropy $H(\boldsymbol{d}^\pi)$ and probability of the least favorable state $\min \boldsymbol{d}^\pi$ for different objective formulations on the Single Chain domain. We report exact solutions with $\zeta = 0$ (left), and approximate optimizations with $\xi = 0.1$, $\zeta = 0.7$, $N = 10$ (100 runs, 95% c.i.) (right).*

state-action pairs have been sufficiently explored. In Algorithm 6, we report the pseudo-code of IDE^{3}AL. Finally, in Figure 7.3, we compare the iterative formulation against a not-iterative one, i.e., an approach that collects samples with a random policy and then optimizes the exploration objective offline. Considering an exploration task on the Double Chain domain (Furmston & Barber, 2010), we show that the iterative form has a clear edge in reducing the model estimation error $\|\boldsymbol{P} - \widehat{\boldsymbol{P}}\|_F$. For the sake of the experiment, both approaches employ a Frobenius formulation (7.5).

7.5 Experimental Evaluation

In this section, we provide the experimental evaluation of IDE^{3}AL. First, we show a set of experiments on the illustrative *Single Chain* and *Double Chain* domains (Furmston & Barber, 2010; Peters et al., 2010). The Single Chain consists of 10 states having 2 possible actions, one to climb up the chain from state 0 to 9, and the other to directly fall to the initial state 0. The two actions are flipped with a probability $p_{slip} = 0.1$, making the environment stochastic and reducing the probability of visiting the higher states. The Double Chain concatenates two Single Chain into a bigger one sharing the central state 9, which is the initial state. Thus, the chain can be climbed in two directions. These two domains, albeit rather simple from a dimensionality standpoint, are actually hard to explore uniformly, due to the high shares of actions returning to the initial state and preventing the agent from consistently reaching the higher states. Then, we present an experiment on the much more complex *Knight Quest* environment (Fruit et al., 2018, Appendix), having $|\mathcal{S}| = 360$ and $|\mathcal{A}| = 8$. This domain takes inspiration from classical arcade games, in which a knight has to rescue a princess in the shortest possible time without being killed by the dragon. To accomplish this feat, the knight has to perform an intricate sequence of actions. In the absence of any reward, it is a fairly challenging environment for exploration. In these domains, we address the task of learning the best exploratory policy in a limited number of samples. Especially, we evaluate these policies in terms of the induced state entropy $H(\boldsymbol{d}^\pi)$ and state-action entropy $H(\boldsymbol{d}^\pi \boldsymbol{\Pi})$.

We compare our approach with *MaxEnt* (Hazan et al., 2019) and a count-based approach

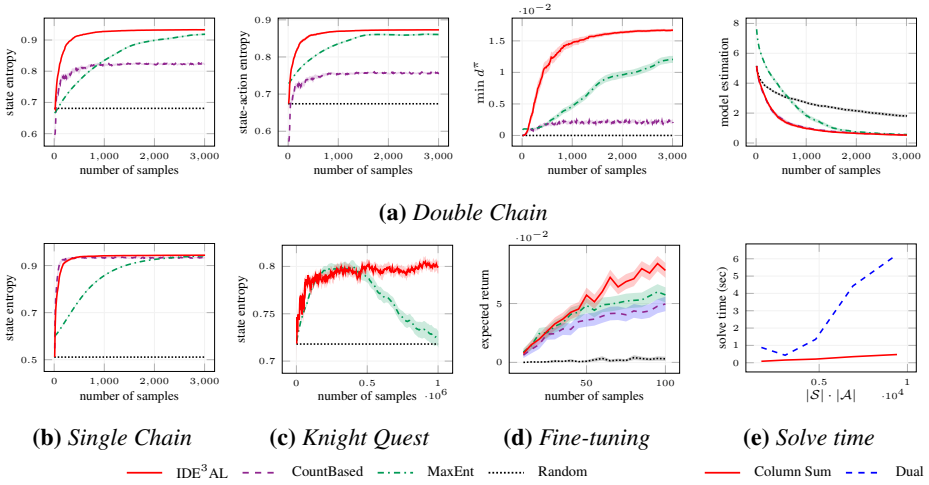


Figure 7.5: *Comparison of the algorithms on unsupervised pre-training (a, b, c) and supervise fine-tuning (d), with parameters $\xi = 0.1$, $\zeta = 0.7$, $N = 10$ (a, b, d) and $\xi = 0.01$, $\zeta = 1$, $N = 2500$ (c). (95% c.i. over 100 runs (a, b), 40 runs (c), 500 runs (d)). Comparison of the solve time (e) achieved by Column Sum and Dual formulations as a function of the number of variables.*

inspired by the exploration bonuses of MBIE-EB (Strehl & Littman, 2008), which we refer to as *CountBased* in the following. The latter shares the same structure of our algorithm, but replaces the policy optimization sweeps with approximate value iterations (Bertsekas, 1995), where the reward for a given state is inversely proportional to the visit count of that state. It is worth noting that the results reported for the MaxEnt algorithm are related to the mixture policy $\pi_{\text{mix}} = (\mathcal{D}, \alpha)$, where $\mathcal{D} = (\pi_0, \dots, \pi_{k-1})$ is a set of k ϵ-deterministic policies, and $\alpha \in \Delta_k$ is a probability distribution over $\mathcal{D}$. For the sake of simplicity, we have equipped all the approaches with a little domain knowledge, i.e., the cardinality of $\mathcal{S}$ and $\mathcal{A}$. However, this can be avoided without a significant impact on the presented results. For every experiment, we will report the batch size N, and the parameters ξ, ζ of IDE^{3}AL. CountBased and MaxEnt employ ϵ-greedy policies having $\epsilon = \xi$ in all the experiments. In any plot, we will additionally provide the performance of a baseline policy, denoted as *Random*, that randomly selects an action in every state.

First, in Figure 7.4, we compare the Problems (7.3), (7.5), (7.6) on the Single Chain environment. On the one hand, we show the performance achieved by the exact solutions, i.e., computed with full knowledge of $\boldsymbol{P}$. While the plain formulations ($\xi = 0, \zeta = 1$) are remarkably similar, adding a constraint over the action entropy ($\xi = 0.1$) has a significantly different impact on their solutions. On the other hand, we illustrate the performance of IDE^{3}AL in learning a good exploratory policy from samples. In this case, the Frobenius formulation clearly achieves a better performance. In the following, we will report the results of IDE^{3}AL considering only the best-performing formulation, which, for all the presented experiments, corresponds to the Frobenius.

In Figure 7.5a, we show that IDE^{3}AL compares well against the other approaches in ex-

ploring the Double Chain domain. It achieves superior state entropy and state-action entropy, and it converges faster to the optimum. It displays also a higher probability of visiting the least favorable state, and it behaves positively in the estimation of $\widehat{P}$. Notably, the CountBased algorithm fails to reach high exploration due to a detachment problem (Ecoffet et al., 2021), since it fluctuates between two exploratory policies that are greedy towards the two directions of the chain. By contrast, in a domain having a clear direction for exploration, such as the simpler Single Chain domain, CountBased ties the explorative performances of IDE^{3}AL (Figure 7.5b). On the other hand, MaxEnt is effective in the exploration performance, but much slower to converge, both in the Double Chain and the Single Chain. Note that, in Figure 7.5a, the model estimation error of MaxEnt starts higher than the other, since it employs a different strategy to fill the transition probabilities of never reached states, inspired by (Brafman & Tennenholtz, 2002). In Figure 7.5c, we present an experiment on the higher-dimensional Knight Quest environment. IDE^{3}AL achieves a remarkable state entropy, while MaxEnt struggles to converge towards a satisfying exploratory policy. CountBased (not reported in Figure 7.5c), fails to explore the environment altogether, oscillating between policies with low entropy.

In Figure 7.5d, we illustrate how the policies learned in the Double Chain environment are effective to ease the learning of any supervised task downstream. To this end, the pretrained policies, learned by the three approaches through 3000 samples (Figure 7.5a), are employed to collect samples in a fixed horizon (within a range from 10 to 100 steps). Then, another policy is fine-tuned offline through approximate value iteration (Bertsekas, 1995) on this small amount of samples. The goal is to optimize a reward function that is 1 for the hardest state to reach (i.e., the state that is less frequently visited with a random policy), and 0 in all the other states. In this setting, all the methods prove to be rather successful w.r.t. the baseline, though IDE^{3}AL compares positively against the other strategies.

Finally, in Figure 7.5e, we provide a brief comparison of the empirical computational efficiency of optimizing the dual formulation of the state entropy objective (see Section 6.3) in comparison to our approach (we take the most efficient Column Sum as a reference here). Especially, we can see that the solve time of the dual formulation (Dual) is significantly higher than the solve time of Column Sum when the number of states and actions increases.

Part III

Practical Methods for State Entropy Maximization

CHAPTER 8

State Entropy Maximization in Continuous Domains

The content of this chapter is based on the paper "Task-Agnostic Exploration via Policy Gradient of a Non-Parametric State Entropy Estimate" co-authored with Lorenzo Pratissoli and Marcello Restelli, published at AAAI 2021.[1]

8.1 Introduction

As we mentioned in previous chapters, a key issue of state entropy maximization for unsupervised pre-training is that the feedback (i.e., the entropy induced by the current policy) is not directly available to the agent. Early approaches either relied on estimates of the environment's transition dynamics (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), from which to compute the state distribution and then the entropy, or estimates of the state distribution itself (Hazan et al., 2019; Lee et al., 2019), from which the entropy can be directly computed. However, both solutions hardly scale to the high-dimensional continuous domains that we often face in the real world, such as in robotic manipulation and autonomous driving. Especially, entropy estimates based on upstream density estimates, often referred to as *plug-in* estimates in the literature (Beirlant et al., 1997), are known to be brittle when variables are high dimensional (Hall & Morton, 1993).

[1] A complete reference can be found in the bibliography (Mutti et al., 2021).

In this chapter, we present the first state entropy maximization algorithm that is explicitly designed for high-dimensional continuous domains. The algorithm, which we call *Maximum Entropy POLicy optimization* (MEPOL),[2] is based on a policy-search procedure (Deisenroth et al., 2013) that learns a maximum-entropy policy by combining function approximation and non-parametric entropy estimation. The approach is completely *model-free* as it requires neither to model the environment transition dynamics nor to directly estimate the state distribution of any policy. Instead, it directly estimates the entropy from mere interactions with the environment. The entropy of continuous distributions can be speculated by looking at how random samples drawn from them lay out over the support surface (Beirlant et al., 1997). Intuitively, samples from a high entropy distribution would evenly cover the surface, while samples drawn from low entropy distributions would concentrate over narrow regions. Backed by this intuition, MEPOL relies on a k-nearest neighbors entropy estimator (Singh et al., 2003) to assess the quality of a given policy from a batch of interactions. Hence, it searches for a policy that maximizes this entropy estimate within a parametric policy space. To do so, it combines ideas from two successful state-of-the-art policy-search methods: TRPO (Schulman et al., 2015), as it performs iterative optimizations of the entropy index within trust regions around the current policies, and POIS (Metelli et al., 2018b), as these optimizations are performed offline via importance sampling. This recipe allows MEPOL to gracefully scale to continuous and high-dimensional domains while showing stable behavior during training.

Contents The chapter is organized as follows. First, we make a digression on non-parametric entropy estimators and their properties for later use (Section 8.2). Then, we formalize the learning problem that we aim to address with the MEPOL algorithm (Section 8.3). In Section 8.4, we report a detailed description of the MEPOL algorithm. In Section 8.5, we provide an extensive empirical analysis of the approach in continuous and high-dimensional domains. Finally, we report some further advancements that enhance the presented procedure to even greater efficiency and adapts it to visual domains (Section 8.6). The complete proofs of the reported results can be found in Appendix A.4.

8.2 Non-Parametric Entropy Estimation

Let $f(x)$ be a probability density function of a random vector $\boldsymbol{X}$ taking values in $\mathbb{R}^p$. We recall that its differential entropy (Shannon, 1948) is defined as:

$$H(f) = -\int f(x) \ln f(x) \, \mathrm{d}x.$$

When the distribution f is not available, this quantity can be estimated given a realization of $\boldsymbol{X} = \{x_i\}_{i=1}^N$ (Beirlant et al., 1997). In particular, to deal with high-dimensional data, we can turn to non-parametric k-Nearest Neighbors (k-NN) entropy estimators of the form (Singh et al., 2003)

$$\widehat{H}_k(f) = -\frac{1}{N} \sum_{i=1}^{N} \ln \frac{k}{N V_i^k} + \ln k - \Psi(k), \tag{8.1}$$

[2]The implementation of MEPOL can be found at `https://github.com/muttimirco/mepol`.

where Ψ is the digamma function, $\ln k - \Psi(k)$ is a bias correction term, V_i^k is the volume of the hyper-sphere of radius $R_i = |x_i - x_i^{k\text{-NN}}|$, which is the Euclidean distance between x_i an its k-nearest neighbor $x_i^{k\text{-NN}}$, so that

$$V_i^k = \frac{\left|x_i - x_i^{k\text{-NN}}\right|^p \cdot \pi^{p/2}}{\Gamma(\frac{p}{2}+1)},$$

where Γ is the gamma function, and p the dimensions of $\boldsymbol{X}$. The estimator (8.1) is known to be asymptotically unbiased and consistent (Singh et al., 2003).

When the target distribution f' differs from the sampling distribution f, we can provide an estimate of $H(f')$ by means of an Importance-Weighted (IW) k-NN estimator (Ajgl & Šimandl, 2011)

$$\widehat{H}_k(f'|f) = -\sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{W_i}{V_i^k} + \ln k - \Psi(k), \tag{8.2}$$

where $W_i = \sum_{j \in \mathcal{N}_i^k} w_j$, such that $\mathcal{N}_i^k$ is the set of indices of the k-NN of x_i, and w_j are the normalized importance weights of samples x_j, which are defined as

$$w_j = \frac{f'(x_j)/f(x_j)}{\sum_{n=1}^{N} f'(x_n)/f(x_n)}.$$

Since the latter IW estimator (8.2) will constitute a bedrock of the MEPOL algorithm that we are going to present in the next section, it is worth analyzing its statistical properties. Especially, in a similar flavor as in (Singh et al., 2003, Theorem 8) for the estimator (8.1), we can prove the following.[3]

Theorem 8.2.1. (Ajgl & Šimandl, 2011, Sec. 4.1) *Let f be a sampling distribution, f' a target distribution. The estimator $\widehat{H}_k(f'|f)$ is asymptotically unbiased for any $k \in \mathbb{N}$.*

Therefore, given a sufficiently large batch of samples from an unknown distribution f, we can get an unbiased estimate of the entropy of any distribution f', irrespective of the form of f and f'. However, if the distance between the two grows large, a high variance might negatively affect the estimation, as it is stated by the following result.

Theorem 8.2.2. *Let f be a sampling distribution, f' a target distribution. The asymptotic variance of the estimator $\widehat{H}_k(f'|f)$ is given by:*

$$\lim_{N\to\infty} \mathbb{V}\underset{x\sim f}{\text{ar}}\left[\widehat{H}_k(f'|f)\right] = \frac{1}{N}\left(\mathbb{V}\underset{x\sim f}{\text{ar}}\left[\overline{w}\ln\overline{w}\right] + \mathbb{V}\underset{x\sim f}{\text{ar}}\left[\overline{w}\ln R^p\right] + \left(\ln C\right)^2 \mathbb{V}\underset{x\sim f}{\text{ar}}\left[\overline{w}\right]\right),$$

where $\overline{w} = \frac{f'(x)}{f(x)}$, and $C = \frac{N\pi^{p/2}}{k\Gamma(p/2+1)}$ is a constant.

Finally, as a by-product of (8.2), we have access to a non-parametric IW k-NN estimate of the Kullback-Leibler (KL) divergence, given by (Ajgl & Šimandl, 2011)

$$\widehat{D}_{KL}(f\|f') = \frac{1}{N}\sum_{i=1}^{N} \ln \frac{k/N}{\sum_{j\in\mathcal{N}_i^k} w_j}. \tag{8.3}$$

[3]Note that Theorem 8.2.1 has been informally stated by (Ajgl & Šimandl, 2011) prior to this work. Our contribution is a first detailed proof of the statement, which is reported in Appendix A.4.

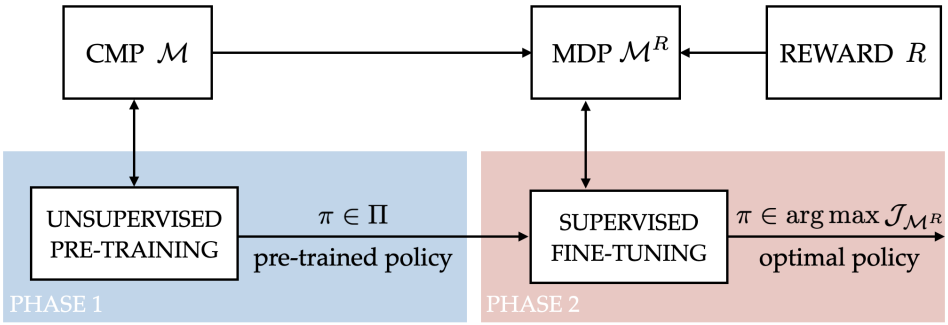


Figure 8.1: *Learning problem for unsupervised RL. Here we focus on the unsupervised pre-training phase.*

Note that, when $f' = f$, $w_j = 1/N$, the estimator (8.2) is equivalent to (8.1), while $\widehat{D}_{KL}(f||f')$ is zero.

8.3 Learning Problem

In this setting, the agent first interacts with a CMP $\mathcal{M}$ (without rewards) through episodes of length T. The agent's behavior, which is represented by a parametric policy $\pi_{\boldsymbol{\theta}}$ within a space of differentiable stochastic policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^q\}$, induces a specific distribution over the states of the CMP. Here we focus on the marginal state distribution over T-steps $d_T^{\pi_{\boldsymbol{\theta}}}$, which we will usually denote as $d_T^{\boldsymbol{\theta}}$ to ease the notation.[4] Then, we can define the unsupervised pre-training objective through state entropy maximization (see Section 3.3.5 and Section 6.2).

Unsupervised Pre-Training via State Entropy Maximization

$$\max_{\pi_{\boldsymbol{\theta}} \in \Pi_\Theta} \; H(d_T^{\boldsymbol{\theta}}) \tag{8.4}$$

Having pre-trained a policy $\pi_{\boldsymbol{\theta}} \in \arg\max_{\pi_{\boldsymbol{\theta}} \in \Pi_\Theta} H(d_T^{\boldsymbol{\theta}})$, the agent then faces a supervised fine-tuning task on the same CMP $\mathcal{M}$ and a given reward function R, which gives rise to the MDP $\mathcal{M}^R$. The objective of the supervised fine-tuning is defined through

$$\max_{\pi_{\boldsymbol{\theta}} \in \Pi_\Theta} \mathcal{J}_{\mathcal{M}^R}(\pi_{\boldsymbol{\theta}}) := \mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{T-1} R(s_t, a_t) \right]. \tag{8.5}$$

In this section, we will focus on the unsupervised pre-training phase (8.4), whereas we will consider the fine-tuning phase (8.5) only to (empirically) evaluate the pre-trained policy.

As the thoughtful reader might realize, optimizing (8.4) is not an easy task. Previous approaches would require either to estimate the transition model in order to obtain average

[4] Recall that the marginal state distribution is defined as $d_T^\pi(s) = \frac{1}{T} \sum_{t=0}^{T-1} d_t^\pi(s)$, where each term in the sum is $d_t^\pi(s) = Pr(s_t = s \mid \pi)$.

Algorithm 7 MEPOL

Input: exploration horizon T, sample-size N, trust region threshold δ, learning rate α, nearest neighbors k
initialize $\boldsymbol{\theta}$
for epoch = $1, 2, \dots,$ until convergence **do**
  draw a batch of $\lceil \frac{N}{T} \rceil$ trajectories of length T with $\pi_{\boldsymbol{\theta}}$
  build a dataset of particles $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$
  $\boldsymbol{\theta}' =$ *IS-Optimizer*$(\mathcal{D}_\tau, \boldsymbol{\theta})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$
end for
Output: maximum state entropy policy $\pi_{\boldsymbol{\theta}}$

IS-Optimizer

Input: dataset of particles $\mathcal{D}_\tau$, sampling parameters $\boldsymbol{\theta}$
initialize $h = 0$ and $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
while $\widehat{D}_{KL}(d_T^{\boldsymbol{\theta}_0} \| d_T^{\boldsymbol{\theta}_h}) \leq \delta$ **do**
  compute a gradient step:
    $\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \widehat{H}_k\big(d_T^{\boldsymbol{\theta}_h} | d_T^{\boldsymbol{\theta}_0}\big)$
  $h \leftarrow h + 1$
end while
Output: parameters $\boldsymbol{\theta}_h$

state distributions (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), or to directly estimate these distributions through a density model (Hazan et al., 2019; Lee et al., 2019). In contrast to the literature, we turn to non-parametric entropy estimation without explicit state distributions modeling, deriving a more practical policy-search approach that we present in the following section.

8.4 The MEPOL Algorithm

In this section, we present *Maximum Entropy POLicy optimization* (MEPOL), which tackles the problem (8.4) in continuous high-dimensional domains. MEPOL searches for a policy that maximizes the performance index $\widehat{H}_k(d_T^{\boldsymbol{\theta}})$ within a parametric space of stochastic differentiable policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^q\}$. The performance index is given by the non-parametric entropy estimator (8.1) where we replace f with the marginal state distribution $d_T^{\boldsymbol{\theta}}$. The approach combines ideas from two successful policy-search algorithms, TRPO (Schulman et al., 2015) and POIS (Metelli et al., 2018b), as it is reported in the following paragraphs. Algorithm 7 provides the pseudocode of MEPOL.

Trust-Region Entropy Maximization The algorithm is designed as a sequence of entropy estimate maximizations, called epochs, within a trust-region around the current policy $\pi_{\boldsymbol{\theta}}$ (Schulman et al., 2015). First, we select an exploration horizon T and an estimator parameter $k \in \mathbb{N}$. Then, at each epoch, a batch of trajectories of length T is sampled from the environment with $\pi_{\boldsymbol{\theta}}$, so as to take a total of N samples. By considering each state encountered in these trajectories as an unweighted particle, we have $\mathcal{D} = \{s_i\}_{i=1}^N$ where $s_i \sim d_T^{\boldsymbol{\theta}}$. Then, given a trust-region threshold δ, we aim to solve the following

optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}' \in \Theta}{\text{maximize}} \quad & \widehat{H}_k\big(d_T^{\boldsymbol{\theta}'}\big) \\ \text{subject to} \quad & \widehat{D}_{KL}\big(d_T^{\boldsymbol{\theta}} \big\| d_T^{\boldsymbol{\theta}'}\big) \leq \delta. \end{aligned} \tag{8.6}$$

The idea is to optimize Problem (8.6) via Importance Sampling (IS, Owen, 2013), in a fully off-policy manner partially inspired by (Metelli et al., 2018b), exploiting the IW entropy estimator (8.2) to calculate the objective, and the KL estimator (8.3) to compute the trust-region constraint. We detail the off-policy optimization in the following paragraph.

Importance Sampling Optimization We first expand the set of particles $\mathcal{D}$ by introducing $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$, where $\tau_i^t = (s_i^0, \dots, s_i^t = s_i)$ is the portion of the trajectory that leads to state s_i. In this way, for any policy $\pi_{\boldsymbol{\theta}'}$, we can associate each particle to its normalized importance weight:

$$\overline{w}_i = \frac{Pr(\tau_i^t|\pi_{\boldsymbol{\theta}'})}{Pr(\tau_i^t|\pi_{\boldsymbol{\theta}})} = \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_i^z|s_i^z)}{\pi_{\boldsymbol{\theta}}(a_i^z|s_i^z)}, \qquad\qquad w_i = \frac{\overline{w}_i}{\sum_{n=0}^N \overline{w}_n}.$$

Then, having set a constant learning rate α and the initial parameters $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, we consider a gradient ascent optimization of the IW entropy estimator (8.2),

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \widehat{H}_k\big(d_T^{\boldsymbol{\theta}_h}|d_T^{\boldsymbol{\theta}_0}\big), \tag{8.7}$$

until the trust-region boundary is reached, i.e., when it holds

$$\widehat{D}_{KL}\big(d_T^{\boldsymbol{\theta}_0} \big\| d_T^{\boldsymbol{\theta}_h}\big) > \delta.$$

The following theorem provides the expression for the gradient of the IW entropy estimator in Equation (8.7).

Theorem 8.4.1. *Let $\pi_{\boldsymbol{\theta}}$ be the current policy and $\pi_{\boldsymbol{\theta}'}$ a target policy. The gradient of the IW estimator $\widehat{H}_k(d_T^{\boldsymbol{\theta}'}|d_T^{\boldsymbol{\theta}})$ w.r.t. $\boldsymbol{\theta}'$ is given by*

$$\nabla_{\boldsymbol{\theta}'} \widehat{H}_k(d_T^{\boldsymbol{\theta}'}|d_T^{\boldsymbol{\theta}}) = -\sum_{i=0}^{N} \frac{\nabla_{\boldsymbol{\theta}'} W_i}{k} \left(V_i^k + \ln \frac{W_i}{V_i^k} \right),$$

where

$$\nabla_{\boldsymbol{\theta}'} W_i = \sum_{j \in \mathcal{N}_i^k} w_j \times \Bigg(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^N \prod_{z=0}^t \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)} \sum_{z=0}^t \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\sum_{n=1}^N \prod_{z=0}^t \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}} \Bigg).$$

8.5 Empirical Analysis

In this section, we present a comprehensive empirical analysis, which is organized as follows:

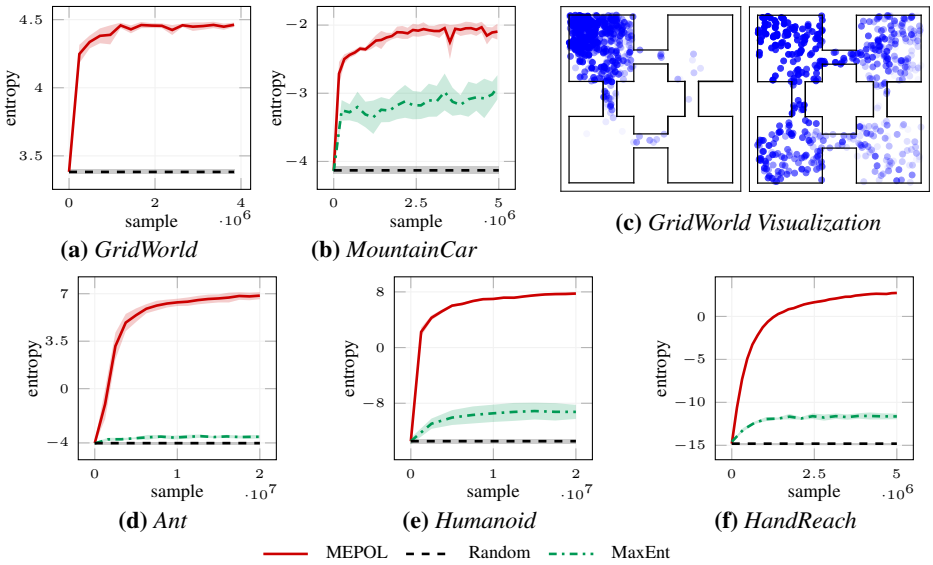


Figure 8.2: *Comparison of the entropy $H(d_T^{\boldsymbol{\theta}})$ as a function of training samples achieved by MEPOL, MaxEnt, and a random policy. (95% c.i. over 8 runs. MEPOL: k: 4 (c, d, e, f), 50 (b); T: 400 (c), 500 (d, e, f), 1200 (b). MaxEnt epochs: 20 (c), 30 (d, e, f)).*

8.5.1) We illustrate that MEPOL allows learning a maximum entropy policy in a variety of continuous domains, outperforming the previous state of the art (MaxEnt);

8.5.2) We illustrate how the exploration horizon T, over which the policy is optimized, maximally impacts the trade-off between state entropy and mixing time;

8.5.3) We reveal the significant benefit of initializing an RL algorithm (TRPO) with a MEPOL policy to solve numerous challenging continuous control tasks.

A thorough description of the experimental set-up, additional results, and visualizations are provided in Appendix B.1.

8.5.1 Unsupervised Pre-Training

In this section, we consider the ability of MEPOL to pre-train a state entropy maximizing policy according to the unsupervised pre-training objective (8.4). Such a policy is evaluated in terms of its induced entropy $H(d_T^{\boldsymbol{\theta}})$.[5] We chose k to optimize the performance of the estimator, albeit experiencing little to no sensitivity to this parameter (Appendix B.1.3). In any considered domain, we picked a specific T according to the time horizon we aimed to test in the subsequent supervised fine-tuning phase (Section 8.5.3). This choice is not relevant in the policy optimization process, while we discuss how it affects the properties of the optimal policy in the next section. Note that, in all the experiments, we adopted a neural network to represent the parametric policy $\pi_{\boldsymbol{\theta}}$ (see Appendix B.1 for details). We

[5] Whereas the MEPOL algorithm optimizes $\widehat{H}_k(d_T^{\boldsymbol{\theta}})$, we evaluate the actual entropy value, which is obtained through the k-NN entropy estimator over a large batch of samples.

compare our algorithm with MaxEnt (Hazan et al., 2019). To this end, we considered their practical implementation[6] of the algorithm to deal with continuous, non-discretized domains. Note that MaxEnt learns a mixture of policies rather than a single policy. To measure its state entropy, we stick with the original implementation by generating a batch as follows: For each step of a trajectory, we sample a policy from the mixture and we take an action with it. This is not our design choice, while we found that using the mixture in the usual way leads to inferior performance anyway. We also investigated SMM (Lee et al., 2019) as a potential comparison. We do not report its results here for two reasons: We cannot achieve significant performance w.r.t. the random baseline, and the difference with MaxEnt is merely in the implementation.

First, we evaluate unsupervised pre-training over two continuous illustrative domains: GridWorld (2D states, 2D actions) and MountainCar (2D, 1D). In these two domains, MEPOL successfully learns a policy that evenly covers the state space in a single batch of trajectories (state-visitation heatmaps are reported in Appendix B.1), while showcasing minimal variance across different runs (Figure 8.2a, 8.2b). Notably, it significantly outperforms MaxEnt in the MountainCar domain.[7] Additionally, in Figure 8.2c we show how a batch of samples drawn with a random policy (left) compares to one drawn with an optimal policy (right, the color fades with the time step). Then, we consider a set of high-dimensional continuous control environments from the Mujoco suite (Todorov et al., 2012): Ant (29D, 8D), Humanoid (47D, 20D), HandReach (63D, 20D). While we learn a policy that maps full state representations to actions, we maximize the state entropy over a subset of the state space dimensions: 7D for Ant (3D position and 4D torso orientation), 24D for Humanoid (3D position, 4D body orientation, and all the joint angles), 24D for HandReach (full set of joint angles). As we report in Figure 8.2d, 8.2e, 8.2f, MEPOL is able to learn policies with striking entropy in all the environments. As a by-product, it unlocks several meaningful high-level skills during the process, such as jumping, rotating, navigation (Ant), crawling, standing up (Humanoid), and basic coordination (Humanoid, HandReach). Crucially, the learning process is not negatively affected by the increasing number of dimensions, which is, instead, a well-known weakness of approaches based on explicit density estimation to compute the entropy (Beirlant et al., 1997). This issue is documented by the poor results of MaxEnt, which struggles to match the performance of MEPOL in the considered domains, as it prematurely converges to a low-entropy mixture.

Scalability As we detail above, in the experiments over continuous control domains we do not maximize the entropy over the full state representation. Note that this selection of features is not dictated by the inability of MEPOL to cope with even more dimensions, but to obtain reliable and visually interpretable behaviors (see Appendix B.1 for further details). To prove this point we conduct an additional experiment over a massively high-dimensional GirdWorld domain (200D, 200D). As we report in Figure 8.3b, even in this setting MEPOL handily learns a policy to maximize the state entropy.

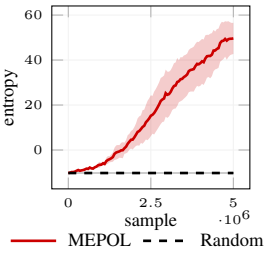
On MaxEnt Results One might realize that the performance reported for MaxEnt appears to be much lower than the one presented in (Hazan et al., 2019). Some aspects need

[6] `https://github.com/abbyvansoest/maxent/tree/master/humanoid`

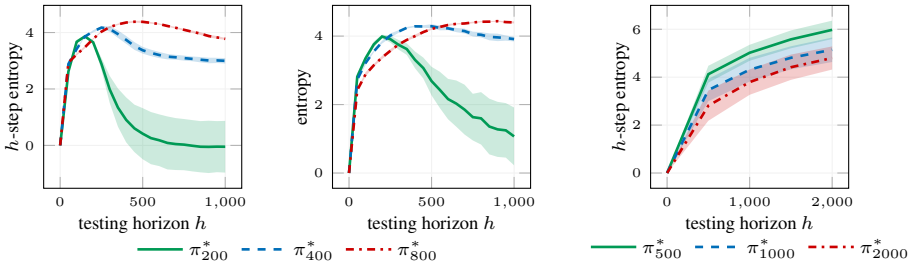
[7] We avoid the comparison in GridWorld since the environment resulted particularly averse to MaxEnt.

	MountainCar	Ant	Humanoid
samples	$5 \cdot 10^6$	$2 \cdot 10^7$	$2 \cdot 10^7$
MEPOL	**4.31 ± 0.04**	**3.67 ± 0.05**	**1.92 ± 0.08**
MaxEnt	3.36 ± 0.4	1.92 ± 0.05	0.96 ± 0.06
Random	1.98 ± 0.05	1.86 ± 0.06	0.84 ± 0.04

(a) *Comparison of the entropy over the 2D-discretized states achieved by MEPOL, MaxEnt, and a random policy (95% c.i. over 8 runs).*



(b) *200D-GridWorld*



(c) *GridWorld* **(d)** *Humanoid*

Figure 8.3: *Comparison of the entropy $H(d^\pi_T)$ over an extended (200D, 200D) GridWorld domain (b). Comparison of the h-step state entropy $H(d^\pi_{t=h})$ and the marginal state entropy $H(d^\pi_h)$ achieved by a set of policies trained over different horizons T as a function of the testing horizon h (c, d). (95% c.i. over 8 runs).*

to be considered. First, their objective is different, as they focus on the entropy of discounted stationary distributions instead of $d^{\boldsymbol{\theta}}_T$. However, in the practical implementation, they consider undiscounted, finite-horizon trajectories as we do. Secondly, their results are computed over all samples collected during the learning process, while we measure the entropy over a single batch. Lastly, one could argue that an evaluation over the same measure (k-NN entropy estimate) that our method explicitly optimize is unfair. Nevertheless, even evaluating the entropy of the 2D-discretized state space, which is the measure considered in (Hazan et al., 2019), leads to similar results (as reported in Figure 8.3a).

8.5.2 Impact of the Exploration Horizon Parameter

In this section, we discuss how choosing an exploration horizon T affects the properties of the learned policy. First, it is useful to distinguish between a *training horizon* T, which is an input parameter to MEPOL, and a *testing horizon* h on which the policy is evaluated. Especially, it is of particular interest to consider how an exploratory policy trained over T steps fares in exploring the environment for a mismatching number of steps h.

To this end, we carried out a set of experiments in the aforementioned GridWorld and Humanoid domains. We denote by π^*_T a policy obtained by executing MEPOL with a training horizon T and we consider the entropy of the h-step state distribution $H(d^{\pi^*_T}_{t=h})$ induced by $\boldsymbol{\theta}^*_T$. Figure 8.3c (left), referring to the GridWorld experiment, shows that a policy trained

over a shorter T might hit a peak in the entropy measure earlier (fast mixing), but other policies achieve higher entropy values at their optimum (highly exploring).[8] It is worth noting that the policy trained over 200 steps becomes overzealous when the testing horizon extends to higher values, while derailing towards a poor h-step entropy. In such a short horizon, the learned policy cannot evenly cover the four rooms and it overfits easy-to-reach locations. Unsurprisingly, also the marginal state entropy over h steps $H(d_h^{\pi_T^*})$, which is the actual objective we aim to maximize in unsupervised pre-training, is negatively affected, as we report in Figure 8.3c (right). This result points out the importance of properly choosing the training horizon in accordance with the downstream-task horizon the policy will eventually face. However, in other cases a policy learned over T steps might gracefully generalize to longer horizons, as confirmed by the Humanoid experiment (Figure 8.3d). The environment is free of obstacles that can limit the agent's motion, so there is no incentive to overfit an exploration behavior over a shorter T.

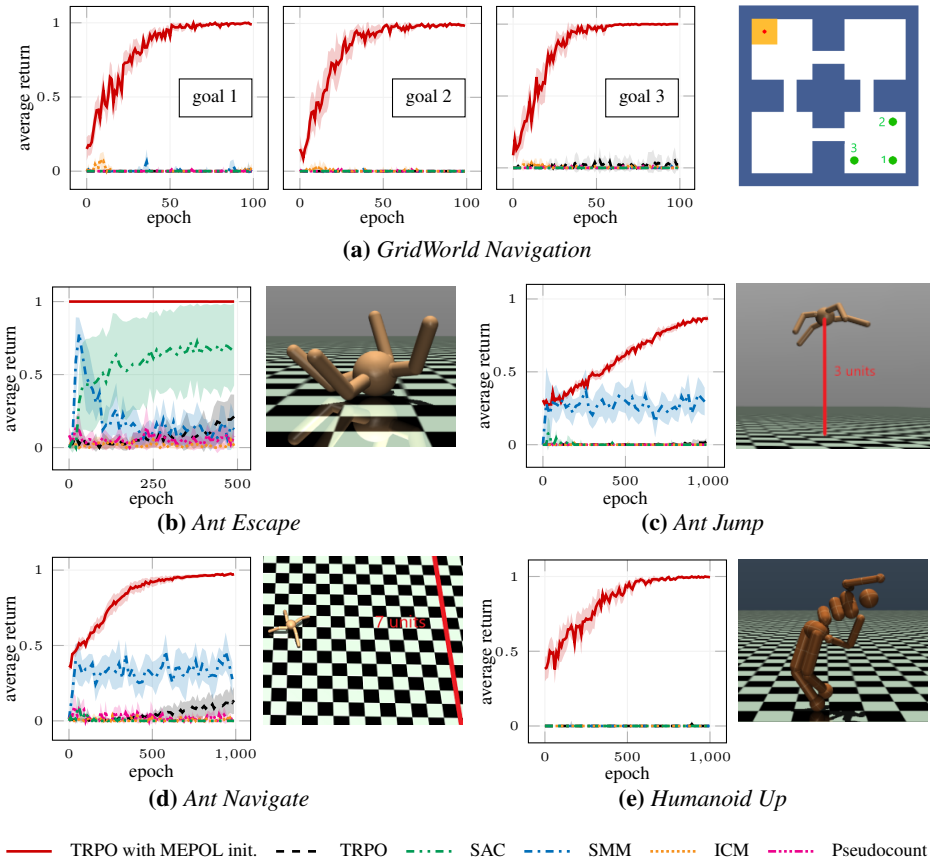
8.5.3 Supervised Fine-Tuning

In this section, we illustrate how a learning agent can benefit from an exploration policy learned by MEPOL when dealing with a variety of supervised RL tasks. Especially, we compare the performance achieved by TRPO (Schulman et al., 2015) initialized with a MEPOL policy (the one we learned in Section 8.5.1) w.r.t. a set of significant baselines that learn from scratch, i.e., starting from a randomly initialized policy. These baselines are: TRPO, SAC (Haarnoja et al., 2018), which promotes exploration over actions, SMM (Lee et al., 2019), which has an intrinsic reward related to the state-space entropy, ICM (Pathak et al., 2017), which favors exploration by fostering prediction errors, and Pseudocount (Bellemare et al., 2016), which assigns high rewards to rarely visited states. The algorithms are evaluated in their fine-tuning performance (8.5) on a series of sparse-reward RL tasks defined over the environments we considered in the previous sections.

Note that we purposefully chose an algorithm without a smart exploration mechanism, i.e., TRPO, to employ the MEPOL initialization. In this way, we can clearly show the merits of the initial policy in providing the necessary exploration. However, the MEPOL initialization can be combined with any other RL algorithm, potentially improving the reported results. In view of previous results in unsupervised pre-training (Section 8.5.1), where MaxEnt is plainly dominated by our approach, we do not compare with TRPO initialized with a MaxEnt policy, as it would not be a challenging baseline in this setting.

In GridWorld, we test three navigation tasks with different goal locations (see Figure 8.4a). The reward is 1 in the states having Euclidean distance to the goal lower than 0.1. For the Ant environment, we define three incrementally challenging tasks: Escape, Jump, Navigate. In the first, the Ant starts from an upside-down position and it receives a reward of 1 whenever it rotates to a straight position (Figure 8.4b). In Jump, the agent gets a reward of 1 whenever it jumps higher than three units from the ground (Figure 8.4c). In Navigate, the reward is 1 in all the states further than 7 units from the initial location (Figure 8.4d). Finally, in Humanoid Up, the agent initially lies on the ground and it receives a reward of 1 when it is able to stand up (Figure 8.4e). In all the considered tasks, the reward is zero

[8] The trade-off between entropy and mixing time has been substantiated for steady-state distributions in (Mutti & Restelli, 2020).



(a) *GridWorld Navigation*

(b) *Ant Escape*

(c) *Ant Jump*

(d) *Ant Navigate*

(e) *Humanoid Up*

Figure 8.4: *Comparison of the average return as a function of learning epochs achieved by TRPO with MEPOL initialization, TRPO, SAC, SMM, ICM, and Pseudocount over a set of sparse-reward RL tasks. For each task, we report a visual representation and learning curves. (95% c.i. over 8 runs).*

anywhere except for the goal states, an episode ends when the goal is reached.

As we show in Figure 8.4, the MEPOL initialization leads to a striking performance across the board, while the tasks resulted extremely hard to learn from scratch. In some cases (Figure 8.4b), MEPOL allows for zero-shot policy optimization, as the optimal behavior has been already learned in the unsupervised exploration stage. In other tasks (e.g., Figure 8.4a), the MEPOL-initialized policy has lower return, but it permits for lighting fast adaptation w.r.t. random initialization. Note that, to match the tasks' higher level of abstraction, in Ant Navigate and Humanoid Up we employed MEPOL initializations learned by maximizing the entropy over mere spatial coordinates (x-y in Ant, x-y-z in Humanoid). However, also the exact policies learned in Section 8.5.1 fare remarkably well in those scenarios (see Appendix B.1.4), albeit experiencing slower convergence.

8.6 Extensions to MEPOL

As a testament to the value of the MEPOL recipe, some notable subsequent works (Liu & Abbeel, 2021b; Seo et al., 2021; Yarats et al., 2021) have adopted a similar combination of non-parametric entropy estimation and policy optimization to achieve even superior results than what we have shown in this chapter. Here we will briefly comment on the innovations introduced by these works, and how they unlocked state entropy maximization in visual-based domains.

One of the clear weaknesses of MEPOL is that it requires a convenient metric space in order to get robust state entropy estimations. In the continuous control domains that we presented, the Euclidean metric is quite natural and it leads to great empirical results (Section 8.5.1). However, it was unclear whether the same recipe could have worked with more complicated metric spaces, such as those required for learning with visual inputs.

In the first extension to MEPOL, Liu & Abbeel (2021b) masterfully answer this question by proposing a method, called Active Pre-Training (APT), which can perform state entropy maximization over visual inputs. To achieve this feat, they introduce a representation learning module on top of the state entropy optimization, in which state representations are trained through a contrastive loss (Chen et al., 2020). Then, the non-parametric entropy estimate is computed through an Euclidean metric in the space of the representations. As a second innovation, Liu & Abbeel (2021b) note that the non-parametric entropy estimate computed over a batch of samples can be decoupled into a sort of intrinsic reward that assigns a scalar value to each state-action pair. This intrinsic reward is designed as follows[9]

$$R(s) \propto \log\left(\left|s - s^{k\text{-NN}}\right|^p\right).$$

Another key difference between APT and MEPOL is that APT keeps a replay buffer of the states visited across the learning process instead of computing the entropy estimates on the last batch of samples alone, which slightly changes the unsupervised pre-training objective. With these innovations, APT achieves outstanding results in the Atari games (Bellemare et al., 2013) and DeepMind Control (Tassa et al., 2018) benchmarks, showing that state entropy maximization is viable in visual-based domains as well.

In a subsequent work, Seo et al. (2021) show that comparable results on visual-based domains can be obtained avoiding a, sometimes cumbersome, representation learning process on top of the state entropy optimization. Especially, they present an algorithm, called Random Encoders for Efficient Exploration (RE3), which is similar to APT without the contrastive learning module, and where the states are instead processed through a randomly initialized encoder before the state entropy estimation is computed. They show that this remarkably simple metric space is enough to allow for state entropy maximization in domains with visual inputs. RE3 is implemented with the per-state entropy rewards presented in (Liu & Abbeel, 2021b) and it is evaluated on the DeepMind Control suite (Tassa et al., 2018).

Finally, Yarats et al. (2021) note that sampling from a replay buffer can lead to unstable entropy estimates from one iteration to the other. To overcome this issues, they propose

[9]Note that this is not a reward function in its usual sense, as it varies over time and depends on the policy being deployed.

an algorithm, called Proto-RL, which learns the embedding of states together with prototypical state representations. The latter are obtained through a clustering procedure over the collected states. The prototypical representations guide the sampling from the replay buffer to the benefit of the stability of the entropy estimations. This additional ingredient improves the results of APT over the DeepMind Control suite (Tassa et al., 2018).

CHAPTER 9

State Entropy Maximization in Multiple Environments

The content of this chapter is based on the paper "Unsupervised Reinforcement Learning in Multiple Environments" co-authored with Mattia Mancassola and Marcello Restelli, published at AAAI 2022.[1]

9.1 Introduction

In this chapter, we aim to push the generality of the unsupervised RL framework even further, by addressing the problem of *state entropy maximization in multiple environments*. In this setting, during the pre-training phase, the agent faces a class of reward-free environments that belong to the same domain but differ in their transition dynamics. At each turn of the learning process, the agent is drawn into an environment within the class, where it can interact for a finite number of steps before facing another turn. The ultimate goal of the agent is to pre-train a maximum entropy policy that helps to solve *any* subsequent fine-tuning task that can be specified over *any* environment of the class.

Specifically, we extend the usual state entropy maximization objective to the multiple-environments setting. Notably, when dealing with multiple environments, the pre-training

[1] A complete reference can be found in the bibliography (Mutti et al., 2022e).

becomes a *multi-objective* problem, as one can establish any combination of preferences over the environments. Previous unsupervised RL methods would blindly optimize the average of the pre-training objective across the class, implicitly establishing a uniform preference. Instead, in this work, we consider the mean of a critical percentile of the objective function, i.e., its Conditional Value-at-Risk (CVaR, Rockafellar et al., 2000) at level α, to prioritize the performance in particularly rare or adverse environments.

To provide intuition on the risk-sensitive strategy we advocate for, let us consider an illustrative example in which the agent interacts with a set of labyrinths without supervision. If the agent pre-trains its policy to maximize the state entropy on average, then there might be some labyrinths where the policy induces high entropy and some others where the entropy is low. At the fine-tuning phase, the agent will be tasked with reaching a specific goal state in one of such labyrinths: If the entropy induced by the pre-trained policy is low, the agent might fail to explore the labyrinth altogether, derailing the learning process. Instead, pre-training the policy to maximize the CVaR of the entropy over the multiple labyrinths lifts the floor of the entropy that the policy will induce in the labyrinth selected for fine-tuning, thus reducing the chance to derail the subsequent learning process due to insufficient exploration.

With this intuition, we propose a policy gradient algorithm (Deisenroth et al., 2013), α-*sensitive Maximum Entropy POLicy optimization* (αMEPOL),[2] to optimize the CVaR of the state entropy via mere interactions with the class of environments. As the name suggests, the algorithm is inspired by MEPOL (Mutti et al., 2021) and subsequent developments (Liu & Abbeel, 2021b; Seo et al., 2021; Yarats et al., 2021). In the footsteps of its progenitors, αMEPOL employs non-parametric methods to deal with state entropy estimation in continuous and high-dimensional environments. Then, it leverages these estimated values to optimize the CVaR of the entropy by following its policy gradient (Tamar et al., 2015b). The percentile of interest is determined by α, which basically controls how much we account for the tail behavior in the pre-training objective.

Finally, we provide an extensive experimental analysis of the proposed method in both the unsupervised pre-training over classes of multiple environments and the supervised fine-tuning over several tasks defined over the class. The analysis spans continuous control settings as well as visual domains. The exploration policy pre-trained with αMEPOL allows solving sparse-rewards tasks that are impractical to learn from scratch, while consistently improving the performance of a pre-training that is blind to the tail behavior of the entropy (e.g., MEPOL). We also include a comparison between state entropy maximization over multiple environments and classical meta-RL baselines (Finn et al., 2017a).

Contents The chapter is organized as follows. First, we discuss a few works that are particularly related to the problem of unsupervised RL in multiple environments (Section 9.2). Then, we propose a tractable formulation of the unsupervised pre-training objective (Section 9.4). In Section 9.5, we provide a preliminary theoretical analysis of the introduced problem formulation, which highlights the specific challenges of the multiple-environments variation of unsupervised RL. In Section 9.6, we describe the αMEPOL algorithm to deal with the unsupervised pre-training with percentile sensitivity. Finally, we provide an extensive empirical evaluation of αMEPOL (Section 9.7). The proofs of the theorems can be found in Appendix A.5.

[2]The implementation of αMEPOL can be found at `https://github.com/muttimirco/alphamepol`

9.2 Related Work

In this section, we revise the works that relate the most to the setting of unsupervised RL in multiple environments.

In previous work, Rajendran et al. (2020) considered a learning process composed of agnostic pre-training (called a *practice*) and supervised fine-tuning (a *match*) in a class of environments. However, in their setting the two phases are alternated, and the supervision signal of the matches allows to learn the reward for the practice through a meta-gradient.

Concurrently to us, Parisi et al. (2021) addressed the unsupervised RL in multiple environments. Whereas their setting is akin to ours, they come up with an essentially orthogonal solution. Especially, they consider a pre-training objective inspired by count-based methods (Bellemare et al., 2016) in place of our entropy objective. Whereas they design a specific bonus for the multiple-environments setting, they essentially establish a uniform preference over the class instead of prioritizing the worst-case environment as we do.

Finally, our framework resembles the *meta-RL* setting (Finn et al., 2017b), in which we would call *meta-training* the unsupervised pre-training, and *meta-testing* the supervised fine-tuning. However, none of the existing works combine unsupervised meta-training (Gupta et al., 2018) with a multiple-environments setting.

9.3 Definitions

Here we recall some useful definitions and we report the slight notation changes that we will adopt across the chapter. A vector $\boldsymbol{v}$ is denoted in bold, and v_i stands for its i-th entry.

Probability and Percentiles Let X be a random variable distributed according to a cumulative density function (cdf) $F_X(x) = Pr(X \leq x)$. We denote with $\mathbb{E}[X]$, $\mathbb{V}\mathrm{ar}[X]$ the expected value and the variance of X respectively. Let $\alpha \in (0, 1)$ be a confidence level, we call the α-percentile (shortened to $\alpha\%$) of the variable X its Value-at-Risk (VaR), which is defined as

$$\mathrm{VaR}_\alpha(X) = \inf \big\{ x \mid F_X(x) \geq \alpha \big\}.$$

Analogously, we call the mean of this same α-percentile the Conditional Value-at-Risk (CVaR) of X,

$$\mathrm{CVaR}_\alpha(X) = \mathbb{E}\left[X \mid X \leq \mathrm{VaR}_\alpha(X) \right].$$

Entropy and State Visitations We denote the state-visitation frequencies induced by a trajectory τ with $d_\tau(s) = \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{1}(s_{t,\tau} = s)$, and we denote as $d_\pi^{\mathcal{M}} = \mathbb{E}_{\tau \sim p_{\pi,\mathcal{M}}}[d_\tau]$ the marginal state distribution on the CMP $\mathcal{M}$. For simplicity, we will write the entropy $H(d_\tau)$ as a random variable $H_\tau \sim \delta(h - H(d_\tau)) p_{\pi,\mathcal{M}}(\tau)$, where $\delta(h)$ is a Dirac delta.

MDPs and RL By coupling a CMP $\mathcal{M}$ with a reward function R we obtain a Markov Decision Process (MDP, Puterman, 2014) $\mathcal{M}^R := \mathcal{M} \cup R$. Let $R(s, a)$ be the expected immediate reward when taking $a \in \mathcal{A}$ in $s \in \mathcal{S}$. The *performance* of a policy π over the

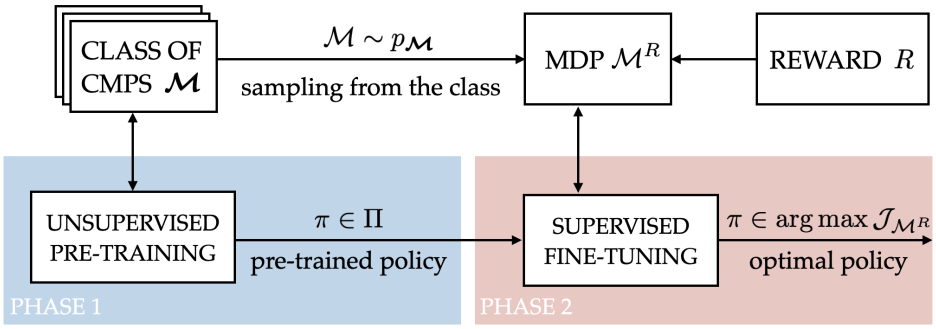


Figure 9.1: *Learning problem for unsupervised RL in multiple CMPs. Here we focus on the unsupervised pre-training phase.*

MDP $\mathcal{M}^R$ is defined as

$$\mathcal{J}_{\mathcal{M}^R}(\pi) = \mathop{\mathbb{E}}_{\pi, \mathcal{M}^R}\left[\sum_{t=0}^{T-1} R(s_t, a_t)\right]. \tag{9.1}$$

9.4 Unsupervised RL in Multiple Environments

Let $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, \dots, \mathcal{M}_I\}$ be a class of unknown CMPs, in which every element $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, P_i, D)$ has a specific transition model P_i, while $\mathcal{S}, \mathcal{A}, D$ are homogeneous across the class. At each turn, the agent is able to interact with a single environment $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. The selection of the environment to interact with is mediated by a distribution $p_{\boldsymbol{\mathcal{M}}}$ over $\boldsymbol{\mathcal{M}}$, outside the control of the agent. The aim of the agent is to pre-train a maximum state entropy policy that is general across all the MDPs $\mathcal{M}^R$ one can build upon $\boldsymbol{\mathcal{M}}$. One can straightforwardly extend the maximum state entropy objective to multiple environments by considering the expectation over the class of CMPs,

$$\mathcal{E}_{\boldsymbol{\mathcal{M}}}(\pi) = \mathop{\mathbb{E}}_{\substack{\mathcal{M} \sim p_{\boldsymbol{\mathcal{M}}} \\ \tau \sim p_{\pi, \mathcal{M}}}} \big[H_\tau\big],$$

where the usual entropy objective over the single environment $\mathcal{M}_i$ can be easily recovered by setting $p_{\mathcal{M}_i} = 1$. However, this objective function does not account for the tail behavior of H_τ, i.e., for the performance in environments of $\boldsymbol{\mathcal{M}}$ that are rare or particularly unfavorable to the current policy. This is decidedly undesirable as the agent may be tasked with an MDP built upon one of these adverse environments in the subsequent supervised fine-tuning, where even an optimal strategy w.r.t. $\mathcal{E}_{\boldsymbol{\mathcal{M}}}(\pi)$ may fail to provide sufficient exploration. To overcome this limitation, we look for a more nuanced exploration objective that balances the expected performance with the sensitivity to the tail behavior. By taking inspiration from the risk-averse optimization literature (Rockafellar et al., 2000), we consider the CVaR of the state visitation entropy induced by π over $\boldsymbol{\mathcal{M}}$,

$$\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi) = \mathrm{CVaR}_\alpha(H_\tau) = \mathop{\mathbb{E}}_{\substack{\mathcal{M} \sim p_{\boldsymbol{\mathcal{M}}} \\ \tau \sim p_{\pi, \mathcal{M}}}} \big[H_\tau \mid H_\tau \leq \mathrm{VaR}_\alpha(H_\tau)\big], \tag{9.2}$$

where α is a confidence level and $\mathcal{E}^1_{\boldsymbol{\mathcal{M}}}(\pi) := \mathcal{E}_{\boldsymbol{\mathcal{M}}}(\pi)$. The lower we set the value of α, the more we hedge against the possibility of a bad exploration outcome in some $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. In the following sections, we propose a method to effectively learn a policy $\pi^*_{\mathcal{E}} \in \arg\max \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi)$ through mere interactions with $\boldsymbol{\mathcal{M}}$, and we show how this serves as a pre-training for RL (the full process is depicted in Figure 9.1). A preliminary theoretical characterization of the problem of optimizing $\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi)$ is provided in the next section.

9.5 Preliminary Theoretical Analysis of the Problem

In this section, we aim to theoretically analyze the problem in (9.2), and especially, what makes a class of multiple CMPs hard to explore with a unique strategy. This has to be intended as a preliminary discussion on the problem, which could serve as a starting point for future works, rather than a thorough theoretical characterization. First, it is worth introducing some additional notation.

Lipschitz Continuity Let X, Y be two metric sets with metric functions d_X, d_Y. We say a function $f : X \to Y$ is L_f-Lipschitz continuous if it holds for some constant L_f

$$d_Y(f(x'), f(x)) \leq L_f d_X(x', x), \qquad \forall (x', x) \in X^2,$$

where the smallest L_f is the Lipschitz constant and the Lipschitz semi-norm is $\|f\|_L = \sup_{x', x \in X} \left\{ \frac{d_Y(f(x'), f(x))}{d_X(x', x)} : x' \neq x \right\}$. When dealing with probability distributions we need to introduce a proper metric. Let p, q be two probability measures, we will either consider the Wasserstein metric (Villani, 2009), defined as

$$d_{W_1}(p, q) = \sup_f \left\{ \left| \int_X f(x)(p(x) - q(x)) \, \mathrm{d}x \right| : \|f\|_L \leq 1 \right\},$$

or the Total Variation (TV) metric, defined as

$$d_{TV}(p, q) = \frac{1}{2} \int_X \left| p(x) - q(x) \right| \mathrm{d}x.$$

Intuitively, learning to explore a class $\boldsymbol{\mathcal{M}}$ with a policy π is challenging when the state distributions induced by π in different $\mathcal{M} \in \boldsymbol{\mathcal{M}}$ are diverse. The more diverse they are, the more their entropy can vary, and the harder is to get a π with a large entropy across the class. To measure this diversity, we are interested in the supremum over the distances between the state distributions $(d^{\mathcal{M}_1}_{\pi}, \dots, d^{\mathcal{M}_I}_{\pi})$ that a single policy $\pi \in \Pi$ realizes over the class $\boldsymbol{\mathcal{M}}$. We call this measure the *diameter* $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ of the class $\boldsymbol{\mathcal{M}}$. Since we have infinitely many policies in Π, computing $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ is particularly arduous. However, we are able to provide an upper bound to $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ defined through a Wasserstein metric.

Assumption 9.5.1. *Let $d_{\mathcal{S}}$ be a metric on $\mathcal{S}$. The class $\boldsymbol{\mathcal{M}}$ is L_{P^π}-Lipschitz continuous,*

$$d_{W_1}(P^\pi(\cdot|s'), P^\pi(\cdot|s)) \leq L_{P^\pi} d_{\mathcal{S}}(s', s), \qquad \forall (s', s) \in \mathcal{S}^2,$$

where $P^\pi(s|\overline{s}) = \int_{\mathcal{A}} \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a}) \, \mathrm{d}\overline{a}$ for $P \in \boldsymbol{\mathcal{M}}$, $\pi \in \Pi$. L_{P^π} is a constant $L_{P^\pi} < 1$.

Theorem 9.5.2. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs satisfying Ass. 9.5.1. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by the policy π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ as*

$$\mathcal{D}_{\boldsymbol{\mathcal{M}}} := \sup_{\substack{\pi \in \Pi \\ \mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}}} d_{W_1}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) \leq \sup_{P', P \in \boldsymbol{\mathcal{M}}} \frac{1 - L_{P^\pi}^T}{1 - L_{P^\pi}} \sup_{\substack{s \in \mathcal{S} \\ a \in \mathcal{A}}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)).$$

Theorem 9.5.2 provides a measure to quantify the hardness of the exploration problem in a specific class $\boldsymbol{\mathcal{M}}$, and to possibly compare one class with another. However, the value of $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ might result, due to the supremum over Π, from a policy that is far away from the policies we actually deploy while learning, say $(\pi_0, \dots, \pi_{\mathcal{E}}^*)$. To get a finer assessment of the hardness of $\boldsymbol{\mathcal{M}}$ we face in practice, it is worth considering a policy-specific measure to track during the optimization. We call this measure the *π-diameter* $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ of the class $\boldsymbol{\mathcal{M}}$. Theorem 9.5.3 provides an upper bound to $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ through a convenient TV metric.

Theorem 9.5.3. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy, and let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ as*

$$\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi) := \sup_{\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}} d_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) \leq \sup_{P', P \in \boldsymbol{\mathcal{M}}} T \mathop{\mathbb{E}}_{\substack{s \sim d_\pi^{\mathcal{M}} \\ a \sim \pi(\cdot|s)}} d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)).$$

The last missing piece we aim to derive is a result to relate the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ of the class $\boldsymbol{\mathcal{M}}$ (Theorem 9.5.3) with the actual exploration objective, i.e., the entropy of the state visitations induced by the policy π over the environments in the class. In the following theorem, we provide an upper bound to the *entropy gap* induced by the policy π within the class $\boldsymbol{\mathcal{M}}$.

Theorem 9.5.4. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy and $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ the corresponding π-diameter of $\boldsymbol{\mathcal{M}}$. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$, and let $\sigma_{\boldsymbol{\mathcal{M}}} \leq \sigma_{\mathcal{M}} := \inf_{s \in \mathcal{S}} d_\pi^{\mathcal{M}}(s), \forall \mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the entropy gap of the policy π within the model class $\boldsymbol{\mathcal{M}}$ as*

$$\sup_{\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}} \left| H(d_\pi^{\mathcal{M}'}) - H(d_\pi^{\mathcal{M}}) \right| \leq \left(\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\right)^2 / \sigma_{\boldsymbol{\mathcal{M}}} + \mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi) \log(1/\sigma_{\boldsymbol{\mathcal{M}}})$$

9.6 A Policy Gradient Approach

In this section, we present an algorithm, called *α-sensitive Maximum Entropy POLicy optimization* (αMEPOL), to optimize the exploration objective in (9.2) through mediated interactions with a class of continuous environments.

αMEPOL operates as a typical policy gradient approach (Deisenroth et al., 2013). It directly searches for an optimal policy by navigating a set of parametric differentiable policies $\Pi_\Theta := \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^n\}$. It does so by repeatedly updating the parameters $\boldsymbol{\theta}$ in

Algorithm 8 αMEPOL

Input: percentile α, learning rate β
Output: policy $\pi_{\boldsymbol{\theta}}$

initialize $\boldsymbol{\theta}$
for epoch $= 0, 1, \ldots,$ until convergence **do**
  for $i = 1, 2, \ldots, N$ **do**
    sample an environment $\mathcal{M}_i \sim p_{\boldsymbol{\mathcal{M}}}$
    sample a trajectory $\tau_i \sim p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}_i}$
    estimate H_{τ_i} with (9.3)
  end for
  estimate $\mathrm{VaR}_\alpha(H_\tau)$ with (9.4)
  estimate $\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ with (9.5)
  update parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\mathcal{M}}(\pi_{\boldsymbol{\theta}})$
end for

the gradient direction until a stationary point is reached. This update has the form

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}),$$

where β is a learning rate, and $\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ is the gradient of (9.2) w.r.t. $\boldsymbol{\theta}$. The following proposition provides the formula of $\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$. The derivation closely follows the one in (Tamar et al., 2015b, Proposition 1), which we have adapted to our objective function of interest (9.2).

Proposition 9.6.1. *The policy gradient of the exploration objective $\mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ w.r.t. $\boldsymbol{\theta}$ is given by*

$$\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \mathop{\mathbb{E}}_{\substack{\mathcal{M} \sim p_{\boldsymbol{\mathcal{M}}} \\ \tau \sim p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}}}} \Bigg[\Bigg(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau} | s_{t,\tau}) \Bigg) \times \Big(H_\tau - \mathrm{VaR}_\alpha(H_\tau) \Big) \Big| H_\tau \leq \mathrm{VaR}_\alpha(H_\tau) \Bigg].$$

However, in this work, we do not assume full knowledge of the class of CMPs $\boldsymbol{\mathcal{M}}$, and the expected value in Proposition 9.6.1 cannot be computed without having access to $p_{\boldsymbol{\mathcal{M}}}$ and $p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}}$. Instead, αMEPOL computes the policy update via a Monte Carlo estimation of $\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}$ from the sampled interactions $\{(\mathcal{M}_i, \tau_i)\}_{i=1}^N$ with the class of environments $\boldsymbol{\mathcal{M}}$. The policy gradient estimate itself relies on a Monte Carlo estimate of each entropy value H_{τ_i} from τ_i, and a Monte Carlo estimate of $\mathrm{VaR}_\alpha(H_\tau)$ given the estimated $\{H_{\tau_i}\}_{i=1}^N$. The following paragraphs describe how these estimates are carried out, while Algorithm 8 provides the pseudocode of αMEPOL.

Entropy Estimation We would like to compute the entropy H_{τ_i} of the state visitation frequencies d_{τ_i} from a single realization $\{s_{t,\tau_i}\}_{t=0}^{T-1} \subset \tau_i$. As in MEPOL (Mutti et al., 2021), we employ a k-Nearest Neighbors (k-NN) entropy estimator (Singh et al., 2003) of

the form

$$\widehat{H}_{\tau_i} \propto -\frac{1}{T}\sum_{t=0}^{T-1} \log \frac{k\,\Gamma(\frac{p}{2}+1)}{T\left\|s_{t,\tau_i} - s_{t,\tau_i}^{k\text{-NN}}\right\|^p \pi^{\frac{p}{2}}}, \tag{9.3}$$

where we recall that Γ is the Gamma function, $\|\cdot\|$ is the Euclidean distance, and $s_{t,\tau_i}^{k\text{-NN}} \in \tau_i$ is the k-nearest neighbor of s_{t,τ_i}. Differently from the previous Chapter 8.2, the entropy estimate is now computed on a single trajectory or a small batch of trajectories.

VaR Estimation The last missing piece to get a Monte Carlo estimate of the policy gradient $\nabla_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ is the value of $\mathrm{VaR}_\alpha(H_\tau)$. Being $H_{[1]},\dots,H_{[N]}$ the order statistics out of the estimated values $\{\widehat{H}_{\tau_i}\}_{i=1}^N$, we can naïvely estimate the VaR as

$$\widehat{\mathrm{VaR}}_\alpha(H_\tau) = H_{[\lceil \alpha N \rceil]}. \tag{9.4}$$

Albeit asymptotically unbiased, the VaR estimator in (9.4) is known to suffer from a large variance in finite sample regimes (Kolla et al., 2019), which is aggravated by the error in the upstream entropy estimates, which provide the order statistics. This variance is mostly harmless when we use the estimate to filter out entropy values beyond the $\alpha\%$, i.e., the condition $H_\tau \leq \mathrm{VaR}_\alpha(H_\tau)$ in Proposition 9.6.1. Instead, its impact is significant when we subtract it from the values within the $\alpha\%$, i.e., the term $H_\tau - \mathrm{VaR}_\alpha(H_\tau)$ in Proposition 9.6.1. To mitigate this issue, we consider a convenient baseline $b = -\mathrm{VaR}_\alpha(H_\tau)$ to be subtracted from the latter, which gives the Monte Carlo policy gradient estimator

$$\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \sum_{i=1}^{N} f_{\tau_i}\, \widehat{H}_{\tau_i}\, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{\mathrm{VaR}}_\alpha(H_\tau)), \tag{9.5}$$

where $f_{\tau_i} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau_i}|s_{t,\tau_i})$. Notably, the baseline b trades off a lower estimation error for a slight additional bias in the estimation (9.5). We found that this baseline leads to empirically good results and we provide some theoretical corroboration on its benefits in the next paragraph.

Baseline To corroborate the use of the baseline $b = -\mathrm{VaR}_\alpha(H_\tau)$, we compare the properties of two alternatives policy gradient estimators, with and without a baseline, i.e.,

$$\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \frac{1}{\alpha N}\sum_{i=1}^{N} f_{\tau_i}\, \left(\widehat{H}_{\tau_i} - \widehat{\mathrm{VaR}}_\alpha(H_{\tau_i})\right)\, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{\mathrm{VaR}}_\alpha(H_\tau)),$$

$$\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \frac{1}{\alpha N}\sum_{i=1}^{N} f_{\tau_i}\, \left(\widehat{H}_{\tau_i} - \mathrm{VaR}_\alpha(H_{\tau_i}) - b\right)\, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{\mathrm{VaR}}_\alpha(H_\tau)).$$

where $f_{\tau_i} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau_i}|s_{t,\tau_i})$. The former ($\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$) is known to be asymptotically unbiased (Tamar et al., 2015b), but it is hampered by the estimation error of the VaR term to be subtracted to each $\widehat{H}_{\tau_i}$ in finite sample regimes (Kolla et al., 2019). The latter ($\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$) introduces some bias in the estimate, but it crucially avoids the estimation error of the VaR term to be subtracted, as it cancels out with the baseline b. The following proposition, along with related lemmas, assesses the critical number of samples (n^*) for which an upper bound to the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ is lower to the estimation error of $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$.

Proposition 9.6.2. *Let f_H be the pdf of H_τ, for which there exist $\eta, \Delta > 0$ such that $f_H(H_\tau) > \eta$ for all $H_\tau \in \left[\mathrm{VaR}_\alpha(H_\tau) - \frac{\Delta}{2}, \mathrm{VaR}_\alpha(H_\tau) + \frac{\Delta}{2}\right]$. Let $\mathcal{U}$ be a large constant such that $f_{\tau_i} \leq \mathcal{U}$ for all τ_i. The number of samples n^* for which the estimation error ϵ of $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ is lower than the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ with probability at least $\delta \in (0, 1)$ is given by*

$$n^* = \frac{\log 2/\delta}{2\eta^2 \min\{\mathcal{U}^2 \alpha^2 b^2, \Delta^2\}}.$$

The Proposition 9.6.2 proves that there is little incentive to choose the policy gradient estimator $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ when the number of trajectories is lower than n^*, as its estimation error would exceed the bias introduced by the alternative estimator $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$. Unfortunately, it is not easy to compute n^* in our setting, as we do not assume to know the distribution of H_τ, but the requirement is arguably seldom matched in practice.

9.7 Empirical Evaluation

We provide an extensive empirical evaluation of the proposed methodology over the two-phase learning process described in Figure 9.1, which is organized as follows:

9.7.1) We show the ability of our method in pre-training an exploration policy in a class of continuous gridworlds, emphasizing the importance of percentile sensitivity;

9.7.2) We discuss how the choice of the percentile of interest affects the learned exploration strategy;

9.7.3) We highlight the benefit that the pre-trained strategy provides to the supervised fine-tuning in the same class;

9.7.4) We verify the scalability of our method with the size of the class, by considering a class of 10 continuous gridworlds;

9.7.5) We verify the scalability of our method with the dimensionality of the environments, by considering a class of 29D continuous control Ant domains;

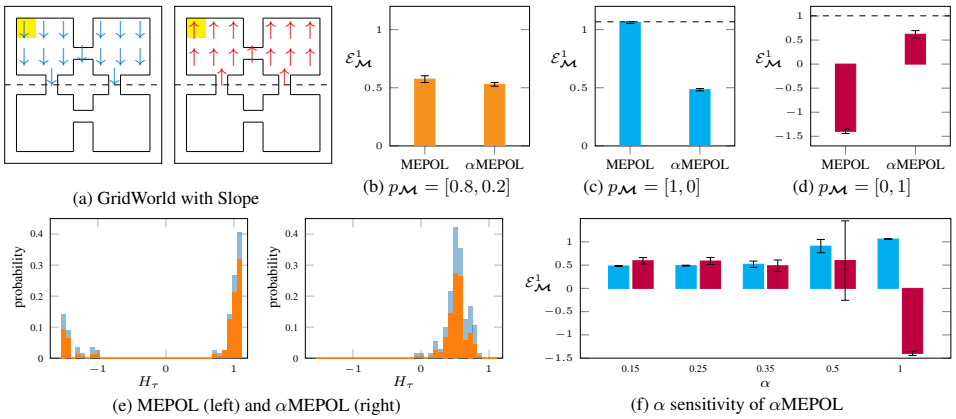
9.7.6) We verify the scalability of our method with visual inputs, by considering a class of 147D MiniGrid domains;

9.7.7) We show that the pre-trained strategy outperforms a policy meta-trained with MAML (Finn et al., 2017b; Gupta et al., 2018) on the same class.

A thorough description of the experimental setting is provided in Appendix B.2.

9.7.1 Unsupervised Pre-Training with Percentile Sensitivity

We consider a class $\boldsymbol{\mathcal{M}}$ composed of two different configurations of a continuous gridworld domain with 2D states and 2D actions, which we call the *GridWorld with Slope*.



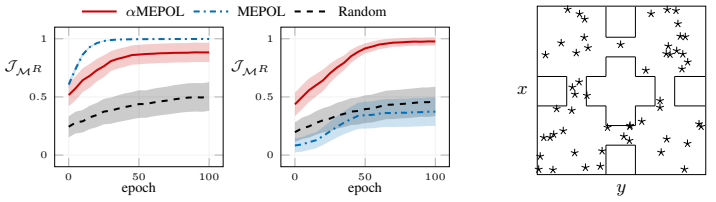
(a) GridWorld with Slope
(b) $p_{\mathcal{M}} = [0.8, 0.2]$
(c) $p_{\mathcal{M}} = [1, 0]$
(d) $p_{\mathcal{M}} = [0, 1]$
(e) MEPOL (left) and αMEPOL (right)
(f) α sensitivity of αMEPOL

Figure 9.2: *Pre-training performance $\mathcal{E}^1_{\mathcal{M}}$ obtained by αMEPOL ($\alpha = 0.2$) and MEPOL in the* GridWorld with Slope *domain **(a)**. The policies are trained on **(b)** and tested on **(b, c, d)**. The dashed lines in **(c, d)** represent the optimal performance. The empirical distribution having mean in **(b)** is reported in **(e)**. The behaviour of αMEPOL with different α is reported in **(f)**. For every plot, we provide 95% c.i. over 10 runs.*

In each configuration, the agent navigates through four rooms connected by narrow hallways, by choosing a (bounded) increment along the coordinate directions. A visual representation of the setting can be found in Figure 9.2a, where the shaded areas denote the initial state distribution and the arrows render a slope that favors or contrasts the agent's movement. The configuration on the left has a south-facing slope, and thus it is called GridWorld with South slope (GWS). Instead, the one on the right is called GridWorld with North slope (GWN) as it has a north-facing slope. This class of environments is unbalanced (and thus interesting to our purpose) for two reasons: First, the GWN configuration is more challenging from a pure exploration standpoint, since the slope prevents the agent from easily reaching the two bottom rooms; secondly, the distribution over the class is also unbalanced, as it is $p_{\mathcal{M}} = [Pr(\text{GWS}), Pr(\text{GWN})] = [0.8, 0.2]$. In this setting, we compare αMEPOL against MEPOL (Mutti et al., 2021), which is akin to αMEPOL with $\alpha = 1$,[3] to highlight the importance of percentile sensitivity w.r.t. a naïve approach to the multiple-environments scenario. The methods are evaluated in terms of the state visitation entropy $\mathcal{E}^1_{\mathcal{M}}$ induced by the exploration strategies they learn.

In Figure 9.2, we compare the performance of the optimal exploration strategy obtained by running αMEPOL ($\alpha = 0.2$) and MEPOL for 150 epochs on the GridWorld with Slope class ($p_{\mathcal{M}} = [0.8, 0.2]$). We show that the two methods achieve a very similar expected performance over the class (Figure 9.2b). However, this expected performance is the result of a (weighted) average of very different contributions. As anticipated, MEPOL has a strong performance in GWS ($p_{\mathcal{M}} = [1, 0]$, Figure 9.2c), which is close to the configuration-specific optimum (dashed line), but it displays a bad showing in the adverse GWN ($p_{\mathcal{M}} = [0, 1]$, Figure 9.2d). Conversely, αMEPOL learns a strategy that is much more robust to the configuration, showing a similar performance in GWS and GWN, as

[3] The pseudocode of MEPOL (see Chapter 8) is identical to Algorithm 8 except that all trajectories affect the gradient estimate in (9.5).



(a) Fine-tuning on GWS (left) and GWN (right)

(b) Sampled goals

Figure 9.3: *Fine-tuning performance $\mathcal{J}_{\mathcal{M}^R}$ as a function of learning epochs achieved by TRPO initialized with αMEPOL ($\alpha = 0.2$), MEPOL, and random exploration strategies, when dealing with a set of RL tasks specified on the* GridWorld with Slope *domain **(a)**. We provide 95% c.i. over 50 randomly sampled goal locations **(b)**.*

the percentile sensitivity prioritizes the worst case during training. To confirm this conclusion, we look at the actual distribution that is generating the expected performance in Figure 9.2b. In Figure 9.2e, we provide the empirical distribution of the trajectory-wise performance (H_τ), considering a batch of 200 trajectories with $p_{\mathcal{M}} = [0.8, 0.2]$. It clearly shows that MEPOL is heavy-tailed towards lower outcomes, whereas αMEPOL concentrates around the mean. *This suggests that with a conservative choice of α we can induce a good exploration outcome for every trajectory (and any configuration), while without percentile sensitivity we cannot hedge against the risk of particularly bad outcomes.* However, let us point out that not all classes of environments would expose such an issue for a naïve, risk-neutral approach (see Appendix B.2.5 for a counterexample), but it is fair to assume that this would arguably generalize to any setting where there is an imbalance (either in the hardness of the configurations or in their sampling probability) in the class. These are the settings we care about, as they require nuanced solutions (e.g., αMEPOL) for scenarios with multiple environments.

9.7.2 On the Value of the Percentile

In this section, we consider repeatedly training αMEPOL with different values of α in the GridWorld with Slope domain, and we compare the resulting exploration performance $\mathcal{E}^1_{\mathcal{M}}$ as before. In Figure 9.2f, we can see that the lower α we choose, the more we prioritize GWN (right bar for every α) at the expense of GWS (left bar). Note that this trend carries on with increasing α, ending in the values of Figures 9.2c, 9.2d. The reason for this behavior is quite straightforward, the smaller is α, the larger is the share of trajectories from the adverse configuration (GWN) ending up in the percentile at first, and thus the more GWN affects the policy update (see the gradient in (9.5)). *Note that the value of the percentile α should not be intended as a hyper-parameter to tune via trial and error, but rather as a parameter to select the desired risk profile of the algorithm.* Indeed, there is no way to say which of the outcomes in Figure 9.2f is preferable, as they are all reasonable trade-offs between the average and worst-case performance, which might be suited for specific applications. For the sake of consistency, in every experiment of our analysis we report results with a value of α that matches the sampling probability of the worst-case configuration, but similar arguments could be made for different choices of α.

9.7.3 Supervised Fine-Tuning

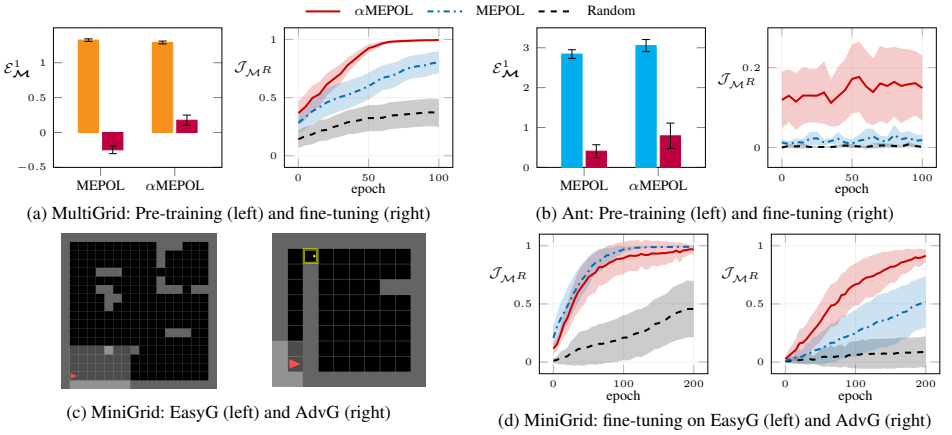
To assess the benefit of the pre-trained strategy, we design a family of MDPs $\mathcal{M}^R$, where $\mathcal{M} \in \{\text{GWS}, \text{GWN}\}$, and R is any sparse reward function that gives 1 when the agent reaches the area nearby a random goal location and 0 otherwise. On this family, we compare the performance achieved by TRPO (Schulman et al., 2015) with different initializations: The exploration strategies learned (as in Section 9.7.1) by αMEPOL ($\alpha = 0.2$) and MEPOL, or a randomly initialized policy (Random). These three variations are evaluated in terms of their average return $\mathcal{J}_{\mathcal{M}^R}$, which is defined in (9.1), over 50 randomly generated goal locations (Figure 9.3b). As expected, the performance of TRPO with MEPOL is competitive in the GWS configuration (Figure 9.3), but it falls sharply in the GWN configuration, where it is not significantly better than TRPO with Random. Instead, the performance of TRPO with αMEPOL is strong on both GWS and GWN. Despite the simplicity of the domain, solving an RL problem in GWN with an adverse goal location is far-fetched for both a random initialization and a naïve solution to the problem of unsupervised RL in multiple environments.

9.7.4 Scaling to Larger Classes of Environments

In this section, we consider a class $\boldsymbol{\mathcal{M}}$ composed of 10 different configurations of the continuous gridworlds presented in Section 9.7.1 (including the GWN as the worst-case configuration) which we call the *MultiGrid* domain. As before, we compare αMEPOL ($\alpha = 0.1$) and MEPOL on the exploration performance $\mathcal{E}^1_{\boldsymbol{\mathcal{M}}}$ achieved by the optimal strategy, in this case considering a uniformly distributed $p_{\boldsymbol{\mathcal{M}}}$. While the average performance of MEPOL is slightly higher across the class (Figure 9.4a left, left bar), αMEPOL still has a decisive advantage in the worst-case configuration (Figure 9.4a left, right bar). Just as in Section 9.7.3, this advantage transfer to the fine-tuning, where we compare the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO with αMEPOL, MEPOL, and Random initializations over 50 random goal locations in the GWN configuration (Figure 9.4a right). *Whereas in the following sections we will only consider classes of two environments, this experiment shows that the arguments made for small classes of environments can easily generalize to larger classes.*

9.7.5 Scaling to Increasing Dimensions

In this section, we consider a class $\boldsymbol{\mathcal{M}}$ consisting of two Ant environments, with 29D states and 8D actions. In the first, sampled with probability $p_{\mathcal{M}_1} = 0.8$, the Ant faces a wide descending staircase (*Ant Stairs Down*). In the second, the Ant faces a narrow ascending staircase (*Ant Stairs Up*, sampled with probability $p_{\mathcal{M}_2} = 0.2$), which is significantly harder to explore than the former. In the mold of the gridworlds in Section 9.7.1, these two configurations are specifically designed to create an imbalance in the class. As in Section 9.7.1, we compare αMEPOL ($\alpha = 0.2$) against MEPOL on the exploration performance $\mathcal{E}^1_{\boldsymbol{\mathcal{M}}}$ achieved after 500 epochs. αMEPOL fares slightly better than MEPOL both in the worst-case configuration (Figure 9.4b left, right bar) and, surprisingly, in the



(a) MultiGrid: Pre-training (left) and fine-tuning (right)

(b) Ant: Pre-training (left) and fine-tuning (right)

(c) MiniGrid: EasyG (left) and AdvG (right)

(d) MiniGrid: fine-tuning on EasyG (left) and AdvG (right)

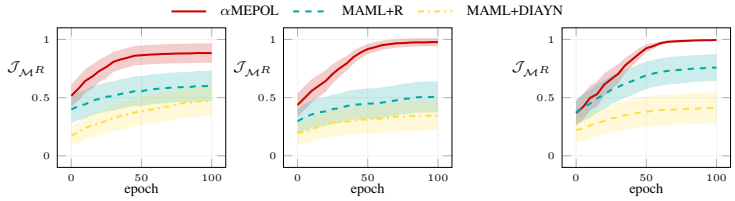
Figure 9.4: *Pre-training performance $\mathcal{E}^1_{\mathcal{M}}$ (95% c.i. over 10 runs) achieved by αMEPOL ($\alpha = 0.1$ **(a)**, $\alpha = 0.2$ **(b)**) and MEPOL in the in the* MultiGrid ***(a)*** *and* Ant ***(b)*** *domains. Fine-tuning performance $\mathcal{J}_{\mathcal{M}^R}$ (95% c.i. over 50 tasks **(a)**, 8 tasks **(b)**, 13 tasks **(d)**) obtained by TRPO with corresponding initialization (αMEPOL, MEPOL, Random), in the* MultiGrid ***(a)***, Ant ***(b)***, *and* MiniGrid ***(d)*** *domains.* MiniGrid *domains are illustrated in **(c)**.*

easier one (Figure 9.4b left, left bar).[4] Then, we design a set of incrementally challenging fine-tuning tasks in the *Ant Stairs Up*, which give reward 1 upon reaching a certain step of the staircase. Also in this setting, TRPO with αMEPOL initialization outperforms TRPO with MEPOL and Random in terms of the average return $\mathcal{J}_{\mathcal{M}^R}$ (Figure 9.4b right). Note that these sparse-reward continuous control tasks are particularly arduous: TRPO with MEPOL and Random barely learns anything, while even TRPO with αMEPOL does not handily reach the optimal average return (1).

9.7.6 Scaling to Visual Inputs

In this section, we consider a class $\boldsymbol{\mathcal{M}}$ of two partially-observable MiniGrid (Chevalier-Boisvert et al., 2018) environments, in which the observation is a 147D image of the agent's field of view. In Figure 9.4c, we provide a visualization of the domain: The easier configuration (EasyG, left) is sampled with probability $p_{\mathcal{M}_1} = 0.8$, the adverse configuration (AdvG, right) is sampled with probability $p_{\mathcal{M}_2} = 0.2$. Two factors make the AdvG more challenging to explore, which are the presence of a door at the top-left of the grid, and reversing the effect of the agent's movements (e.g., the agent goes backward when it tries to go forward). Whereas in all the previous experiments we estimated the entropy on the raw input features, visual inputs require a wiser choice of a metric. As proposed in (Seo et al., 2021), we process the observations through a random encoder before computing the entropy estimate in (9.3), while keeping everything else as in Algorithm 8. We run this

[4]Note that this would not happen in general, as we expect αMEPOL to be better in the worst-case but worse on average. In this setting, the percentile sensitivity positively biases the average performance due to the peculiar structure of the environments.



(a) GridWorld with Slope: GWS (left) and GWN (right) (b) MultiGrid

Figure 9.5: *Fine-tuning performance $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO initialized with αMEPOL ($\alpha = 0.2$ **(a)**, $\alpha = 0.1$ **(b)**), a MAML+R meta-policy, and a MAML+DIAYN meta-policy, when dealing with a set of RL tasks in the* GridWorld with Slope ***(a)*** *and the* MultiGrid ***(b)*** *domains. We provide 95% c.i. over 50 tasks.*

slightly modified version of αMEPOL ($\alpha = 0.2$) and MEPOL for 300 epochs. Then, we compare TRPO with the learned initializations (as well as Random) on sparse-reward fine-tuning tasks defined upon the class. As in previous settings, TRPO with αMEPOL results slightly worse than TRPO with MEPOL in the easier configuration (Figure 9.4d, left), but significantly better in the worst-case (Figure 9.4d, right). Notably, TRPO from scratch struggles to learn the tasks, especially in the AdvG (Figure 9.4d, right). *Although the MiniGrid domain is extremely simple from a vision standpoint, we note that the same architecture can be employed in more challenging scenarios (Seo et al., 2021), while the focus of this experiment is the combination between visual inputs and multiple environments.*

9.7.7 Comparison with Meta-RL

In this section, we compare our approach against meta-training a policy with MAML (Finn et al., 2017b) on the same *GridWorld with Slope* ($p_{\boldsymbol{\mathcal{M}}} = [0.8, 0.2]$) and *MultiGrid* (uniformly distributed $p_{\boldsymbol{\mathcal{M}}}$) domains that we have previously presented. Especially, we consider two relevant baselines. The first is MAML+R, to which we provide full access to the tasks (i.e., rewards) during meta-training. Note that this gives MAML+R an edge over αMEPOL, which operates reward-free training. The second is MAML+DIAYN (Gupta et al., 2018), which operates unsupervised meta-training through an intrinsic reward function learned with DIAYN (Eysenbach et al., 2019). As in previous sections, we consider the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO initialized with the exploration strategy learned by αMEPOL or the meta-policy learned by MAML+R and MAML+DIAYN. TRPO with αMEPOL fares clearly better than TRPO with the meta-policies in all the configurations (Figures 9.5a, 9.5b). Even if it works fine in fast adaptation (Appendix B.2.6), *MAML struggles to encode the diversity of task distribution into a single meta-policy and to deal with the most adverse tasks in the long run.* Moreover, DIAYN does not specifically handle multiple environments, and it fails to cope with the larger *MultiGrid* class.

CHAPTER 10

Conclusion and Future Directions

In this thesis, we have unveiled the potential of unsupervised reinforcement learning via state entropy maximization as a way to improve the generalization of the reinforcement learning approach to sequential decision-making problems.

Especially, we have provided practical pre-training methodologies that significantly improve the fine-tuning performance w.r.t. learning from scratch, both when the pre-training takes place in a single environment or in a set of multiple environments. Notably, this achievement was not obvious from a prior characterization of the computational and statistical complexity of the problem that we provided in this thesis, in which we revealed that an exact optimization of the state entropy maximization objective is intractable and that state entropy maximization is harder than standard reinforcement learning in general.

Despite the positive results that we presented, this thesis leaves open some important research questions, which might be a matter of interesting future directions. We report below a few of these open problems.

When is convex RL tractable? In Chapter 4, we have highlighted a fundamental mismatch between the infinite trials convex RL formulation and its finite trials formulation, which makes convex RL strictly harder than standard RL in practice. However, it would be interesting to understand under which assumptions on the underlying Markov process, the objective function, or the policy class, the convex RL problem can be solved efficiently either from a computational or statistical standpoint.

Can we optimize non-Markovian policies for state entropy maximization in practice? In Chapter 5, we have proved the importance of non-Markovianity to optimize the finite trials formulation of the state entropy maximization problem, but we also noted that the corresponding optimization problem is NP-hard in general. However, it would be interesting to consider function approximation to optimize non-Markovian policies in practice, and to see whether they can provide the crucial benefits they promise in theory.

Can we fully address the multi-objective nature of state entropy maximization in multiple environments? In Chapter 9, we have presented the state entropy maximization problem in multiple environments. Noting that one can establish every preference over the environments, this becomes an essentially multi-objective problem. While we provided a conservative solution to it, it would be interesting to learn a full approximation of the Pareto frontier of the pre-training performance over the environments (Parisi et al., 2016; Hayes et al., 2022).

Can we optimally address the exploration-exploitation trade-off in the supervised fine-tuning phase? In this thesis, we have focused on the unsupervised pre-training phase, while we reported fine-tuning results mainly for the evaluation of the pre-trained policies. However, we have only considered naïve fine-tuning methods, in which the pre-trained policy is used as an initial policy for a standard RL algorithm (TRPO (Schulman et al., 2015) most of the time). It would be interesting to develop fine-tuning algorithms that are aware of the pre-training, so that they can manage the exploration-exploitation trade-off more carefully (Campos et al., 2021; Pislar et al., 2021).

To conclude, we hope our dissertation has been kind to the reader, and the contributions we provided can inspire future advancements in unsupervised reinforcement learning.

Bibliography

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, 2004.

Abel, D., Dabney, W., Harutyunyan, A., Ho, M. K., Littman, M., Precup, D., and Singh, S. On the expressivity of markov reward. In *Advances in Neural Information Processing Systems*, 2021.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, 2017.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. In *Advances in Neural Information Processing Systems*, 2020.

Ajgl, J. and Šimandl, M. Differential entropy estimation by particles. *IFAC*, 2011.

Akshay, S., Bertrand, N., Haddad, S., and Helouet, L. The steady-state control problem for Markov decision processes. In *International Conference on Quantitative Evaluation of Systems*, 2013.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1): 89–129, 2008.

Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Åström, K. J. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

Bai, Q., Bedi, A. S., Agarwal, M., Koppel, A., and Aggarwal, V. Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach. In *AAAI Conference on Artificial Intelligence*, 2022.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Beirlant, J., Dudewicz, E. J., Györfi, L., and Van der Meulen, E. C. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 1997.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellman, R. Dynamic programming. *Princeton University Press*, 1957.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bertsekas, D. P. *Dynamic programming and optimal control*. Athena Scientific, 1995.

Bertsekas, D. P. and Tsitsiklis, J. N. *Introduction to probability*. Athena Scientific Belmont, MA, 2002.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., and Restelli, M. Risk-averse trust region optimization for reward-volatility reduction. In *International Joint Conference on Artificial Intelligence*, 2020.

Boyd, S., Diaconis, P., and Xiao, L. Fastest mixing markov chain on a graph. *SIAM review*, 2004.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 2002.

Brantley, K., Dudik, M., Lykouris, T., Miryoosefi, S., Simchowitz, M., Slivkins, A., and Sun, W. Constrained episodic reinforcement learning in concave-convex and knapsack settings. In *Advances in Neural Information Processing Systems*, 2020.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019b.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, 2020.

Campos, V., Sprechmann, P., Hansen, S., Barreto, A., Kapturowski, S., Vitvitskyi, A., Badia, A. P., and Blundell, C. Coverage as a principle for discovering transferable behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

Chatterji, N., Pacchiano, A., Bartlett, P., and Jordan, M. On the theory of reinforcement learning with once-per-episode feedback. In *Advances in Neural Information Processing Systems*, 2021.

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

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607. PMLR, 2020.

Cheung, W. C. Exploration-exploitation trade-off in reinforcement learning on online markov decision processes with global concave rewards. *arXiv preprint arXiv:1905.06466*, 2019a.

Cheung, W. C. Regret minimization for reinforcement learning with vectorial feedback and complex objectives. In *Advances in Neural Information Processing Systems*, 2019b.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. *GitHub repository*, 2018.

Chow, Y., Tamar, A., Mannor, S., and Pavone, M. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems*, 2015.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, 2019.

Csiszár, I. and Talata, Z. Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE Transactions on Information Theory*, 2006.

Dadashi, R., Hussenot, L., Geist, M., and Pietquin, O. Primal Wasserstein imitation learning. In *International Conference on Learning Representations*, 2020.

De Farias, D. P. and Van Roy, B. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

Deisenroth, M. P., Neumann, G., Peters, J., et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 2016.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Efroni, Y., Merlis, N., and Mannor, S. Reinforcement learning with trajectory feedback. In *AAAI Conference on Artificial Intelligence*, 2021.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017a.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017b.

Foster, D. J., Krishnamurthy, A., Simchi-Levi, D., and Xu, Y. Offline reinforcement learning: Fundamental barriers for value function approximation. In *Conference on Learning Theory*, 2021.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Fruit, R., Pirotta, M., Lazaric, A., and Ortner, R. Efficient bias-span-constrained exploration-exploitation in reinforcement learning. In *International Conference on Machine Learning*, 2018.

Furmston, T. and Barber, D. Variational methods for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Geist, M., Pérolat, J., Laurière, M., Elie, R., Perrin, S., Bachem, O., Munos, R., and Pietquin, O. Concave utility reinforcement learning: The mean-field game viewpoint. In *International Conference on Autonomous Agents and Multiagent Systems*, 2022.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, 2020.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

Grötschel, M., Lovász, L., and Schrijver, A. The ellipsoid method. In *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993.

Guo, Z. D., Azar, M. G., Saade, A., Thakoor, S., Piot, B., Pires, B. A., Valko, M., Mesnard, T., Lattimore, T., and Munos, R. Geometric entropic exploration. *arXiv preprint arXiv:2101.02055*, 2021.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Hall, P. and Morton, S. C. On the estimation of entropy. *Annals of the Institute of Statistical Mathematics*, 45(1):69–88, 1993.

Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., Van de Wiele, T., and Mnih, V. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2019.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, 2019.

He, S., Jiang, Y., Zhang, H., Shao, J., and Ji, X. Wasserstein unsupervised reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2022.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 1997.

Hodges, J. L. and Le Cam, L. The poisson approximation to the poisson binomial distribution. *The Annals of Mathematical Statistics*, 1960.

Hunter, J. J. Some stochastic properties of "semi-magic" and "magic" Markov chains. *Linear Algebra and its Applications*, 2010.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

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

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, 2021.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University College London (United Kingdom), 2003.

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

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

Kirkland, S. Column sums and the conditioning of the stationary distribution for a stochastic matrix. *Operators and Matrices*, 2010.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, 2006.

Kolla, R. K., Prashanth, L., Bhat, S. P., and Jagannathan, K. Concentration bounds for empirical conditional value-at-risk: The unbounded case. *Operations Research Letters*, 2019.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*, 2019.

Lambert, N., Wulfmeier, M., Whitney, W., Byravan, A., Bloesch, M., Dasagi, V., Hertweck, T., and Riedmiller, M. The challenges of exploration for offline reinforcement learning. *arXiv preprint arXiv:2201.11861*, 2022.

Laroche, R., Combes, R. T. d., and Buckman, J. Non-Markovian policies occupancy measures. *arXiv preprint arXiv:2205.13950*, 2022.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*. American Mathematical Soc., 2017.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, 2021a.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems*, 2021b.

Lusena, C., Goldsmith, J., and Mundhenk, M. Nonapproximability results for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 2001.

Ménard, P., Domingues, O. D., Jonsson, A., Kaufmann, E., Leurent, E., and Valko, M. Fast active learning for pure exploration in reinforcement learning. In *International Conference on Machine Learning*, 2021.

Metelli, A. M., Mutti, M., and Restelli, M. Configurable Markov decision processes. In *International Conference on Machine Learning*, 2018a.

Metelli, A. M., Papini, M., Faccio, F., and Restelli, M. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, 2018b.

Metelli, A. M., Papini, M., D'Oro, P., and Restelli, M. Policy optimization as online learning with mediator feedback. In *AAAI Conference on Artificial Intelligence*, 2021.

Miryoosefi, S., Brantley, K., Daume III, H., Dudik, M., and Schapire, R. E. Reinforcement learning with convex constraints. In *Advances in Neural Information Processing Systems*, 2019.

Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International Conference on Machine Learning*, 2020.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Modi, A., Chen, J., Krishnamurthy, A., Jiang, N., and Agarwal, A. Model-free representation learning and exploration in low-rank mdps. *arXiv preprint arXiv:2102.07035*, 2021.

Mundhenk, M., Goldsmith, J., Lusena, C., and Allender, E. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM (JACM)*, 2000.

Mutti, M. and Restelli, M. An intrinsically-motivated approach for learning highly exploring and fast mixing policies. In *AAAI Conference on Artificial Intelligence*, 2020.

Mutti, M., Pratissoli, L., and Restelli, M. Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *AAAI Conference on Artificial Intelligence*, 2021.

Mutti, M., De Santi, R., De Bartolomeis, P., and Restelli, M. Challenging common assumptions in convex reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022a.

Mutti, M., De Santi, R., and Restelli, M. The importance of non-Markovianity in maximum state entropy exploration. In *International Conference on Machine Learning*, 2022b.

Mutti, M., De Santi, R., Rossi, E., Calderon, J. F., Bronstein, M., and Restelli, M. Provably efficient causal model-based reinforcement learning for systematic generalization. In *AAAI Conference on Artificial Intelligence*, 2022c.

Mutti, M., Del Col, S., and Restelli, M. Reward-free policy space compression for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2022d.

Mutti, M., Mancassola, M., and Restelli, M. Unsupervised reinforcement learning in multiple environments. In *AAAI Conference on Artificial Intelligence*, 2022e.

Nedergaard, A. and Cook, M. k-means maximum entropy exploration. *arXiv preprint arXiv:2205.15623*, 2022.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., Peters, J., et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

Owen, A. B. Monte carlo theory, methods and examples. *Monte Carlo Theory, Methods and Examples*, 2013.

Papadimitriou, C. H. and Tsitsiklis, J. N. The complexity of Markov decision processes. *Mathematics of Operations Research*, 1987.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. Stochastic variance-reduced policy gradient. In *International Conference on Machine Learning*, 2018.

Papini, M., Metelli, A. M., Lupo, L., and Restelli, M. Optimistic policy optimization via multiple importance sampling. In *International Conference on Machine Learning*, 2019.

Parisi, S., Pirotta, M., and Restelli, M. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016.

Parisi, S., Dean, V., Pathak, D., and Gupta, A. Interesting object, curious agent: Learning task-agnostic exploration. In *Advances in Neural Information Processing Systems*, 2021.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 2008.

Peters, J., Mulling, K., and Altun, Y. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

Petersen, K. B., Pedersen, M. S., et al. The matrix cookbook. *Technical University of Denmark*, 2008.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. Safe policy iteration. In *International Conference on Machine Learning*, pp. 307–315. PMLR, 2013.

Pirotta, M., Restelli, M., and Bascetta, L. Policy gradient in lipschitz Markov decision processes. *Machine Learning*, 100(2):255–283, 2015.

Pislar, M., Szepesvari, D., Ostrovski, G., Borsa, D. L., and Schaul, T. When should agents explore? In *International Conference on Learning Representations*, 2021.

Prashanth, L. and Ghavamzadeh, M. Actor-critic algorithms for risk-sensitive mdps. In *Advances in Neural Information Processing Systems*, 2013.

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

Qin, Z., Chen, Y., and Fan, C. Density constrained reinforcement learning. In *International Conference on Machine Learning*, 2021.

Rajendran, J., Lewis, R., Veeriah, V., Lee, H., and Singh, S. How should an agent practice? In *AAAI Conference on Artificial Intelligence*, 2020.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rockafellar, R. T., Uryasev, S., et al. Optimization of conditional value-at-risk. *Journal of Risk*, 2000.

Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, 1991.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

Schweitzer, P. J. Perturbation theory and finite Markov chains. *Journal of Applied Probability*, 1968.

Schweitzer, P. J. and Seidmann, A. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

Seo, Y., Chen, L., Shin, J., Lee, H., Abbeel, P., and Lee, K. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, 2021.

Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal*, pp. 379–423, 1948.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

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

Silver, D., Singh, S., Precup, D., and Sutton, R. S. Reward is enough. *Artificial Intelligence*, 2021.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Singh, H., Misra, N., Hnizdo, V., Fedorowicz, A., and Demchuk, E. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 2003.

Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 2008.

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

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

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

Tamar, A. and Mannor, S. Variance adjusted actor critic algorithms. *arXiv preprint arXiv:1310.3697*, 2013.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. Policy gradient for coherent risk measures. In *Advances in Neural Information Processing Systems*, 2015a.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the cvar via sampling. In *AAAI Conference on Artificial Intelligence*, 2015b.

Tarbouriech, J. and Lazaric, A. Active exploration in markov decision processes. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Tarbouriech, J., Shekhar, S., Pirotta, M., Ghavamzadeh, M., and Lazaric, A. Active model estimation in markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2020.

Tarbouriech, J., Pirotta, M., Valko, M., and Lazaric, A. A provably efficient sample collection strategy for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, 2012.

Touati, A. and Ollivier, Y. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems*, 2021.

Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

Villani, C. *Optimal transport: Old and new*, volume 338. Springer, 2009.

Wang, R., Du, S. S., Yang, L., and Salakhutdinov, R. R. On reward-free reinforcement learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2020.

Watkins, C. J. and Dayan, P. Q-learning. *Machine Learning*, 1992.

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., and Weinberger, M. J. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989.

Xie, T., Jiang, N., Wang, H., Xiong, C., and Bai, Y. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Xie, T., Foster, D. J., Bai, Y., Jiang, N., and Kakade, S. M. The role of coverage in online reinforcement learning. *arXiv preprint arXiv:2210.04157*, 2022.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.

Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Ye, H., Chen, X., Wang, L., and Du, S. S. On the power of pre-training for generalization in rl: Provable benefits and hardness. *arXiv preprint arXiv:2210.10464*, 2022.

Yu, T., Tian, Y., Zhang, J., and Sra, S. Provably efficient algorithms for multi-objective competitive rl. In *International Conference on Machine Learning*, 2021.

Zahavy, T., O'Donoghue, B., Desjardins, G., and Singh, S. Reward is enough for convex mdps. In *Advances in Neural Information Processing Systems*, 2021.

Zahavy, T., Schroecker, Y., Behbahani, F., Baumli, K., Flennerhag, S., Hou, S., and Singh, S. Discovering policies with domino: Diversity optimization maintaining near optimality. *arXiv preprint arXiv:2205.13521*, 2022.

Zanette, A., Lazaric, A., Kochenderfer, M. J., and Brunskill, E. Provably efficient reward-agnostic navigation with linear value iteration. In *Advances in Neural Information Processing Systems*, 2020.

Zhan, W., Huang, B., Huang, A., Jiang, N., and Lee, J. Offline reinforcement learning with realizability and single-policy concentrability. In *Conference on Learning Theory*, 2022.

Zhang, C., Cai, Y., Huang, L., and Li, J. Exploration by maximizing Rényi entropy for reward-free rl framework. In *AAAI Conference on Artificial Intelligence*, 2021a.

Zhang, J., Koppel, A., Bedi, A. S., Szepesvari, C., and Wang, M. Variational policy gradient method for reinforcement learning with general utilities. In *Advances in Neural Information Processing Systems*, 2020a.

Zhang, S., Liu, B., and Whiteson, S. Mean-variance policy iteration for risk-averse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2021b.

Zhang, X., Ma, Y., and Singla, A. Task-agnostic exploration in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020b.

Zhang, Z., Du, S. S., and Ji, X. Nearly minimax optimal reward-free reinforcement learning. In *International Conference on Machine Learning*, 2021c.

APPENDIX $\mathcal{A}$

Missing Proofs

A.1 Proofs of Chapter 4

Proposition A.1.1 (Finite Trials vs Infinite Trials)**.** *We provide here some results on the objectives discussed in Table 4.1.*

(i) Let $\mathcal{F}(d) = R \cdot d$ *then* $\min_{\pi \in \Pi} \zeta_\infty(\pi) = \min_{\pi \in \Pi} \zeta_n(\pi),\ \forall n \in \mathbb{N}$

(ii) Let $\mathcal{F}(d) = R \cdot d$ *s.t.* $\lambda \cdot d \leq c$ *then* $\min_{\pi \in \Pi} \zeta_\infty(\pi) = \min_{\pi \in \Pi} \zeta_n(\pi),\ \forall n \in \mathbb{N}$

(iii) Let $\mathcal{F}(d) = \|d - d_E\|_2^2$ *then* $\min_{\pi \in \Pi} \zeta_\infty(\pi) < \min_{\pi \in \Pi} \zeta_n(\pi),\ \forall n \in \mathbb{N}$

(iv) Let $\mathcal{F}(d) = -d \cdot \log(d) = H(d)$ *then* $\min_{\pi \in \Pi} \zeta_\infty(\pi) < \min_{\pi \in \Pi} \zeta_n(\pi),\ \forall n \in \mathbb{N}$

(v) Let $\mathcal{F}(d) = \mathrm{KL}(d||d_E)$ *then* $\min_{\pi \in \Pi} \zeta_\infty(\pi) < \min_{\pi \in \Pi} \zeta_n(\pi),\ \forall n \in \mathbb{N}$

Proof. We report below the corresponding derivations.

(i) $\min_{\pi \in \Pi} \zeta_\infty(\pi) = \min_{\pi \in \Pi} R \cdot d^\pi = \min_{\pi \in \Pi} R \cdot \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[d_n] = \min_{\pi \in \Pi} \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[R \cdot d_n] = \min_{\pi \in \Pi} \zeta_n(\pi)$

(ii) $\min_{\pi \in \Pi} \zeta_\infty(\pi) = \min_{\pi \in \Pi, \lambda \cdot d^\pi \leq c} R \cdot d^\pi = \min_{\pi \in \Pi, \lambda \cdot d^\pi \leq c} R \cdot \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[d_n] = \min_{\pi \in \Pi, R \cdot d^\pi \leq c} \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[R \cdot d_n]$
$= \min_{\pi \in \Pi} \zeta_n(\pi)$

(iii) $\min_{\pi \in \Pi} \zeta_\infty(\pi) = \min_{\pi \in \Pi} \| \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[d_n] - d_E\|_2^2 < \min_{\pi \in \Pi} \mathop{\mathbb{E}}_{d_n \sim p_n^\pi}[\|d_n - d_E\|_2^2] = \min_{\pi \in \Pi} \zeta_n(\pi)$

(iv) $\min_{\pi\in\Pi}\zeta_\infty(\pi) = \min_{\pi\in\Pi}\mathbb{E}_{d_n\sim p_n^\pi}[d_n]\cdot\log\mathbb{E}_{d_n\sim p_n^\pi}[d_n] < \min_{\pi\in\Pi}\mathbb{E}_{d_n\sim p_n^\pi}[d_n\cdot\log d_n] = \min_{\pi\in\Pi}\zeta_n(\pi)$

(v) $\min_{\pi\in\Pi}\zeta_\infty(\pi) = \min_{\pi\in\Pi}\mathrm{KL}(\mathbb{E}_{d_n\sim p_n^\pi}[d_n]||d_E) < \min_{\pi\in\Pi}\mathbb{E}_{d_n\sim p_n^\pi}[\mathrm{KL}(d_n\,||\,d_E)] = \min_{\pi\in\Pi}\zeta_n(\pi)$

□

Theorem 4.4.2 (Approximation Error)**.** *Let $n\in\mathbb{N}$ be a number of trials, let $\delta\in(0,1]$ be a confidence level, let $\pi^\dagger\in\arg\max_{\pi\in\Pi}\zeta_n(\pi)$ and $\pi^\star\in\arg\max_{\pi\in\Pi}\zeta_\infty(\pi)$. Then, it holds with probability at least $1-\delta$*

$$err := \left|\zeta_n(\pi^\dagger)-\zeta_n(\pi^\star)\right| \le 4LT\sqrt{\frac{2S\log(4T/\delta)}{n}}$$

Proof. Let us first upper bound the approximation error as

$$\begin{aligned}
err := \left|\zeta_n(\pi^\dagger)-\zeta_n(\pi^\star)\right| &\le \left|\zeta_n(\pi^\dagger)-\zeta_\infty(\pi^\dagger)\right| + \left|\zeta_\infty(\pi^\dagger)-\zeta_n(\pi^\star)\right| && \text{(A.1)}\\
&\le \left|\zeta_n(\pi^\dagger)-\zeta_\infty(\pi^\dagger)\right| + \left|\zeta_\infty(\pi^\star)-\zeta_n(\pi^\star)\right| && \text{(A.2)}\\
&\le \left|\mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\dagger}}[\mathcal{F}(d_n)] - \mathcal{F}(d^{\pi^\dagger})\right| + \left|\mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\star}}[\mathcal{F}(d_n)] - \mathcal{F}(d^{\pi^\star})\right| && \text{(A.3)}\\
&\le \mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\dagger}}\left[\left|\mathcal{F}(d_n)-\mathcal{F}(d^{\pi^\dagger})\right|\right] + \mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\star}}\left[\left|\mathcal{F}(d_n)-\mathcal{F}(d^{\pi^\star})\right|\right] && \text{(A.4)}\\
&\le \mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\dagger}}\left[L\left\|d_n-d^{\pi^\dagger}\right\|_1\right] + \mathop{\mathbb{E}}_{d_n\sim p_n^{\pi^\star}}\left[L\left\|d_n-d^{\pi^\star}\right\|_1\right] && \text{(A.5)}\\
&\le 2L\max_{\pi\in\{\pi^\dagger,\pi^\star\}}\mathop{\mathbb{E}}_{d_n\sim p_n^\pi}\left[\|d_n-d^\pi\|_1\right] && \text{(A.6)}\\
&\le 2L\max_{\pi\in\{\pi^\dagger,\pi^\star\}}\mathop{\mathbb{E}}_{d_n\sim p_n^\pi}\left[\max_{t\in[T]}\|d_{n,t}-d_t^\pi\|_1\right], && \text{(A.7)}
\end{aligned}$$

where (A.1) is obtained by adding $\pm\zeta_\infty(\pi^\dagger)$ and then applying the triangle inequality, (A.2) follows by noting that $\zeta_\infty(\pi^\star)\ge\zeta_\infty(\pi^\dagger)$, we derive (A.3) by plugging the definitions of ζ_n,ζ_∞ in (A.2), then we obtain (A.4) from $|\,\mathbb{E}[X]|\le\mathbb{E}[|X|]$, we apply the Lipschitz assumption on $\mathcal{F}$ to write (A.5) from (A.4), we maximize over the policies to write (A.6), and we finally obtain (A.7) through a maximization over the episode's step by noting that $d_n=\frac{1}{T}\sum_{t\in[T]}d_{n,t}$ and $d^\pi=\frac{1}{T}\sum_{t\in[T]}d_t^\pi$. Then, we seek to bound with a high probability

$$\begin{aligned}
Pr\Big(\max_{\pi\in\{\pi^\dagger,\pi^\star\}}\max_{t\in[T]}\|d_{n,t}-d_t^\pi\|_1\ge\epsilon\Big) &\le Pr\Big(\bigcup_{\pi,t}\|d_{n,t}-d_t^\pi\|_1\ge\epsilon\Big) && \text{(A.8)}\\
&\le \sum_{\pi,t}Pr\Big(\|d_{n,t}-d_t^\pi\|_1\ge\epsilon\Big) && \text{(A.9)}\\
&\le 2T\;Pr\Big(\|d_{n,t}-d_t^\pi\|_1\ge\epsilon\Big), && \text{(A.10)}
\end{aligned}$$

where $\epsilon>0$ is a positive constant, and we applied a union bound to get (A.9) from (A.8). From concentration inequalities for empirical distributions (see Theorem 2.1 in (Weissman et al., 2003) and Lemma 16 in (Efroni et al., 2021)) we have

$$Pr\left(\|d_{n,t}-d_t^\pi\|_1\ge\sqrt{\frac{2S\log(2/\delta')}{n}}\right)\le\delta'. \tag{A.11}$$

By setting $\delta' = \delta/2T$ in (A.11), and then plugging (A.11) in (A.10), and again (A.10) in (A.7), we have that with probability at least $1-\delta$

$$\left|\zeta_n(\pi^\dagger) - \zeta_n(\pi^\star)\right| \leq 4LT\sqrt{\frac{2S\log(4T/\delta)}{n}},$$

which concludes the proof. □

Theorem 4.5.2 (Regret)**.** *For any confidence $\delta \in (0,1]$ and unknown convex MDP $\mathcal{CM}$, the regret of the* OPE-UCBVI *algorithm is upper bounded as*

$$\mathcal{R}(N) \leq O\Big(\Big[d_{\mathbf{w}}^{7/2}B^{3/2}T^2SA^{1/2}\Big]\sqrt{N}\Big)$$

with probability $1-\delta$.

Proof. To prove the result, we show that the described online learning setting can be translated into the once-per-episode framework (Chatterji et al., 2021). The main difference between the setting in (Chatterji et al., 2021) and ours is that they assume a binary feedback $y \in \{0,1\}$ coming from a logistic model

$$y|d = \begin{cases} 1 & \text{with prob.} \quad \sigma(\mathbf{w}_*^\top \phi(d)) \\ 0 & \text{with prob.} \quad 1-\sigma(\mathbf{w}_*^\top \phi(d)), \end{cases} \qquad \sigma(x) = \frac{1}{1+\exp(-x)}, \forall x \in \mathbb{R},$$

instead of our richer $\mathcal{F}(d)$. To transform the latter in the binary reward y, we note that $\mathcal{F}(d) = \mathbf{w}_*^\top \phi(d)$ through linear realizability (Assumption 4.5.1), then we filter $\mathcal{F}(d)$ through a logistic model to obtain $y = \sigma\big(\mathcal{F}(d)\big)$, which is then used as feedback for OPE-UCBVI. In this way, we can call Theorem 3.2 of (Chatterji et al., 2021) to obtain the same regret rate up to a constant factor, which is caused by the different range of per-episode contributions in the regret. For detailed derivations and the complete regret upper bound see (Chatterji et al., 2021). □

A.2 Proofs of Chapter 5

Theorem 5.3.1. *Let $x \in \{\infty, \gamma, T\}$, and let $\mathcal{D}_{\mathrm{NM}}^x = \{d_x^\pi(\cdot) : \pi \in \Pi_{\mathrm{NM}}\}$, $\mathcal{D}_{\mathrm{M}}^x = \{d_x^\pi(\cdot) : \pi \in \Pi_{\mathrm{M}}\}$ the corresponding sets of state distributions over a MDP. We can prove that:*

(i) The sets of stationary state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^\infty \equiv \mathcal{D}_{\mathrm{M}}^\infty$;

(ii) The sets of discounted state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^\gamma \equiv \mathcal{D}_{\mathrm{M}}^\gamma$ for any γ;

(iii) The sets of marginal state distributions are equivalent $\mathcal{D}_{\mathrm{NM}}^T \equiv \mathcal{D}_{\mathrm{M}}^T$ for any T.

Proof. First, note that a non-Markovian policy $\pi \in \Pi_{\mathrm{NM}}$ can always reduce to a Markovian policy $\pi \in \Pi_{\mathrm{M}}$ by conditioning the decision rules on the history length. Thus, $\mathcal{D}_{\mathrm{NM}}^x \supseteq \mathcal{D}_{\mathrm{M}}^x$ is straightforward for any $x \in \{\infty, \gamma, T\}$. From the derivations in (Puterman, 2014, Theorem 5.5.1), we have that $\mathcal{D}_{\mathrm{M}}^x \supseteq \mathcal{D}_{\mathrm{NM}}^x$ as well. Indeed, for any non-Markovian policy $\pi \in \Pi_{\mathrm{NM}}$, we can build a (non-stationary) Markovian policy $\pi' \in \Pi_{\mathrm{M}}$ as

$$\pi' = \big(\pi'_1, \pi'_2, \dots, \pi'_t, \dots\big), \quad \text{such that } \pi'_t(a|s) = \frac{d_t^\pi(s,a)}{d_t^\pi(s)}, \ \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}.$$

For $t = 0$, we have that $d_0^{\pi}(\cdot) = d_0^{\pi'}(\cdot) = \mu(\cdot)$, which is the initial state distribution. We proceed by induction to show that if $d_{t-1}^{\pi}(\cdot) = d_{t-1}^{\pi'}(\cdot)$, then we have

$$\begin{aligned} d_t^{\pi'}(s) &= \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} d_{t-1}^{\pi'}(s') \pi'_{t-1}(a|s') P(s|s', a) \\ &= \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{d_{t-1}^{\pi'}(s')}{d_{t-1}^{\pi}(s')} d_{t-1}^{\pi}(s', a) P(s|s', a) \\ &= \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} d_{t-1}^{\pi}(s', a) P(s|s', a) \\ &= d_t^{\pi}(s). \end{aligned}$$

Since $d_t^{\pi}(s) = d_t^{\pi'}(s)$ holds for any $t \geq 0$ and $\forall s \in \mathcal{S}$, we have $d_{\infty}^{\pi}(\cdot) = d_{\infty}^{\pi'}(\cdot)$, $d_{\gamma}^{\pi}(\cdot) = d_{\gamma}^{\pi'}(\cdot)$, $d_T^{\pi}(\cdot) = d_T^{\pi'}(\cdot)$, and thus $\mathcal{D}_{\mathrm{M}}^{x} \supseteq \mathcal{D}_{\mathrm{NM}}^{x}$. Then, $\mathcal{D}_{\mathrm{NM}}^{x} \equiv \mathcal{D}_{\mathrm{M}}^{x}$ follows. □

Corollary 5.3.2. *For every MDP, there exists a Markovian policy $\pi^* \in \Pi_{\mathrm{M}}$ such that $\pi^* \in \arg\max_{\pi \in \Pi} \zeta_{\infty}(\pi)$.*

Proof. The result is straightforward from Theorem 5.3.1 and noting that the set of non-Markovian policies Π_{NM} with arbitrary history-length is as powerful as the general set of policies Π. Thus, for every policy $\pi \in \Pi$ there exists a (possibly randomized) policy $\pi' \in \Pi_{\mathrm{M}}$ inducing the same (stationary, discounted or marginal) state distribution of π, i.e., $d^{\pi}(\cdot) = d^{\pi'}(\cdot)$, which implies $\mathcal{F}\big(d^{\pi}(\cdot)\big) = \mathcal{F}\big(d^{\pi'}(\cdot)\big)$. If it holds for any $\pi \in \Pi$, then it holds for $\pi^* \in \arg\max_{\pi \in \Pi} \mathcal{F}\big(d^{\pi}(\cdot)\big)$. □

Corollary 5.4.3 (Sufficient Condition)**.** *For every MDP $\mathcal{M}$ and trajectory $h_t \in \mathcal{H}_{[T]}$ for which any optimal Markovian policy $\pi_{\mathrm{M}} \in \Pi_{\mathrm{M}}$ is randomized (i.e., stochastic) in s_t, we have strictly positive gap $\mathcal{V}_{T-t}(\pi_{\mathrm{M}}, h_t) > 0$.*

Proof. This result is a direct consequence of the combination of Lemma 5.4.6 and Lemma 5.4.7. Indeed, if the policy $\pi_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \mathcal{E}(\pi)$ is randomized in s_t we have

$$0 < \mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s_t))\right] = \underset{hs_t \sim p_t^{\pi_{\mathrm{NM}}}}{\mathbb{V}\mathrm{ar}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right],$$

from Lemma 5.4.6, which gives a lower bound to the convex value gap $\underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}, h_t) > 0$ through Lemma 5.4.7. □

Lemma 5.4.6. *Let $\pi_{\mathrm{NM}} \in \Pi_{\mathrm{NM}}^{\mathrm{D}}$ be a deterministic non-Markovian policy such that $\pi_{\mathrm{NM}} \in \arg\max_{\pi \in \Pi} \zeta_1(\pi)$ on a CMP $\mathcal{M}$. For a fixed history $h_t \in \mathcal{H}_t$ ending in state s, the variance of the event of an optimal Markovian policy $\pi_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \zeta_1(\pi)$ taking $a^* = \pi_{\mathrm{NM}}(h_t)$ in s is given by*

$$\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s,t))\right] = \underset{hs \sim p_t^{\pi_{\mathrm{NM}}}}{\mathbb{V}\mathrm{ar}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right],$$

where $hs \in \mathcal{H}_t$ is any history of length t such that the final state is s, i.e., $hs := (h_{t-1} \in \mathcal{H}_{t-1}) \oplus s$, and $\mathcal{B}(x)$ is a Bernoulli with parameter x.

Proof. Let us consider the random variable $A \sim \mathcal{P}$ denoting the event "the agent takes action $a^* \in \mathcal{A}$". Through the law of total variance (Bertsekas & Tsitsiklis, 2002), we can write the variance

of A given $s \in \mathcal{S}$ and $t \geq 0$ as

$$\begin{aligned}
\mathbb{V}\mathrm{ar}\left[A|s,t\right] &= \mathbb{E}\left[A^2|s,t\right] - \mathbb{E}\left[A|s,t\right]^2 \\
&= \mathop{\mathbb{E}}_{h}\Big[\mathbb{E}\left[A^2|s,t,h\right]\Big] - \mathop{\mathbb{E}}_{h}\Big[\mathbb{E}\left[A|s,t,h\right]\Big]^2 \\
&= \mathop{\mathbb{E}}_{h}\Big[\mathbb{V}\mathrm{ar}\left[A|s,t,h\right] + \mathbb{E}\left[A|s,t,h\right]^2\Big] - \mathop{\mathbb{E}}_{h}\Big[\mathop{\mathbb{E}}_{\pi}\left[A|s,t,h\right]\Big]^2 \\
&= \mathop{\mathbb{E}}_{h}\Big[\mathbb{V}\mathrm{ar}\left[A|s,t,h\right]\Big] + \mathop{\mathbb{E}}_{h}\Big[\mathbb{E}\left[A|s,t,h\right]^2\Big] - \mathop{\mathbb{E}}_{h}\Big[\mathbb{E}\left[A|s,t,h\right]\Big]^2 \\
&= \mathop{\mathbb{E}}_{h}\Big[\mathbb{V}\mathrm{ar}\left[A|s,t,h\right]\Big] + \mathop{\mathbb{V}\mathrm{ar}}_{h}\Big[\mathbb{E}\left[A|s,t,h\right]\Big]. \qquad \text{(A.12)}
\end{aligned}$$

Now let the conditioning event h be distributed as $h \sim p_{t-1}^{\pi_{\mathrm{NM}}}$, so that the condition s,t,h becomes hs where $hs = (s_0, a_0, s_1, \ldots, s_t = s) \in \mathcal{H}_t$, and let the variable A be distributed according to $\mathcal{P}$ that maximizes the objective (5.2) given the conditioning. Hence, we have that the variable A on the left hand side of (A.12) is distributed as a Bernoulli $\mathcal{B}(\pi_{\mathrm{M}}(a^*|s,t))$, where $\pi_{\mathrm{M}} \in \arg\max_{\pi\in\Pi_{\mathrm{M}}} \mathcal{E}(\pi)$, and the variable A on the right hand side of (A.13) is distributed as a Bernoulli $\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))$, where $\pi_{\mathrm{NM}} \in \arg\max_{\pi\in\Pi_{\mathrm{NM}}} \mathcal{E}(\pi)$. Thus, we obtain

$$\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s,t))\right] = \mathop{\mathbb{E}}_{hs\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right] + \mathop{\mathbb{V}\mathrm{ar}}_{hs\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right]\right]. \qquad \text{(A.13)}$$

Under Assumption 5.4.4, we know from Lemma 5.4.5 that the policy π_{NM} is deterministic, i.e., $\pi_{\mathrm{NM}} \in \Pi_{\mathrm{D}}^{\mathrm{NM}}$, so that $\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs))\right] = 0$ for every hs, which concludes the proof. □

Lemma 5.4.7. *Let π_{M} be an optimal Markovian policy $\pi_{\mathrm{M}} \in \arg\max_{\pi\in\Pi_{\mathrm{M}}} \zeta_1(\pi)$ on a MDP $\mathcal{M}$. For any $h_t \in \mathcal{H}_{[T]}$, it holds $\underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) \leq \mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \leq \overline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}})$ such that*

$$\begin{aligned}
\underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) &= \frac{\mathcal{F}^* - \mathcal{F}_2^*}{\pi_{\mathrm{M}}(a^*|s_t)} \mathop{\mathbb{V}\mathrm{ar}}_{hs_t\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right], \\
\overline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}}) &= \frac{\mathcal{F}^* - \mathcal{F}_*}{\pi_{\mathrm{M}}(a^*|s_t)} \mathop{\mathbb{V}\mathrm{ar}}_{hs_t\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right],
\end{aligned}$$

where $\pi_{\mathrm{NM}} \in \arg\max_{\pi\in\Pi_{\mathrm{NM}}^{\mathrm{D}}} \zeta_1(\pi)$, and $\mathcal{F}_*, \mathcal{F}_2^*$ are given by

$$\begin{aligned}
\mathcal{F}_* &= \min_{h\in\mathcal{H}_{T-t}} \mathcal{F}(d_{h_t\oplus h}(\cdot)), \\
\mathcal{F}_2^* &= \max_{h\in\mathcal{H}_{T-t}\setminus\mathcal{H}_{T-t}^*} \mathcal{F}(d_{h_t\oplus h}(\cdot)) \qquad s.t.\ \mathcal{H}_{T-t}^* = \arg\max_{h\in\mathcal{H}_{T-t}} \mathcal{F}(d_{h_t\oplus h}(\cdot)).
\end{aligned}$$

Proof. From the definition of the convex value gap (Definition 5.4.1), we have that

$$\mathcal{V}_{T-t}(\pi_{\mathrm{M}}, h_t) = \mathcal{F}^* - \mathop{\mathbb{E}}_{h_{T-t}\sim p_{T-t}^{\pi_{\mathrm{M}}}}\left[\mathcal{F}\big(d_{h_t\oplus h_{T-t}}(\cdot)\big)\right],$$

in which we will omit h_t in the gap $\mathcal{V}_{T-t}(\pi_{\mathrm{M}}, h_t) = \mathcal{V}_{T-t}(\pi_{\mathrm{M}})$ as h_t is fixed by the statement. To derive a lower bound and an upper bound to $\mathcal{V}_{T-t}(\pi_{\mathrm{M}})$ we consider the impact that taking a sub-optimal action $a \in \mathcal{A}\setminus\{a^*\}$ in state s_t would have in a best-case and a worst-case CMP respectively. Especially, we can write

$$\begin{aligned}
\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) &= \mathcal{F}^* - \mathop{\mathbb{E}}_{h_{T-t}\sim p_{T-t}^{\pi_{\mathrm{M}}}}\left[\mathcal{F}\big(d_{h_t\oplus h_{T-t}}(\cdot)\big)\right] \\
&\geq \mathcal{F}^* - \pi_{\mathrm{M}}(a^*|s_t)\mathcal{F}^* - \big(1 - \pi_{\mathrm{M}}(a^*|s_t)\big)\mathcal{F}_2^* \\
&= (\mathcal{F}^* - \mathcal{F}_2^*)\big(1 - \pi_{\mathrm{M}}(a^*|s_t)\big)
\end{aligned}$$

and

$$\begin{aligned}\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) &= \mathcal{F}^* - \mathop{\mathbb{E}}_{h_{T-t}\sim p_{T-t}^{\pi_{\mathrm{M}}}}\left[\mathcal{F}\big(d_{h_t\oplus h_{T-t}}(\cdot)\big)\right] \\ &\leq \mathcal{F}^* - \pi_{\mathrm{M}}(a^*|s_t)\mathcal{F}^* - \big(1-\pi_{\mathrm{M}}(a^*|s_t)\big)\mathcal{F}_* \\ &= (\mathcal{F}^* - \mathcal{F}_*)\big(1-\pi_{\mathrm{M}}(a^*|s_t)\big).\end{aligned}$$

Then, we note that the event of taking a sub-optimal action $a \in \mathcal{A}\setminus\{a^*\}$ with a policy π_{M} can be modelled by a Bernoulli distribution $\mathcal{B}$ with parameter $\big(1-\pi_{\mathrm{M}}(a^*|s_t)\big)$. By combining the equation of the variance of a Bernoulli random variable with Lemma 5.4.6 we obtain

$$\mathbb{V}\mathrm{ar}\left[\mathcal{B}(\pi_{\mathrm{M}}(a^*|s_t))\right] = \pi_{\mathrm{M}}(a^*|s_t)\big(1-\pi_{\mathrm{M}}(a^*|s_t)\big) = \mathop{\mathbb{V}\mathrm{ar}}_{hs\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right]$$

which gives

$$\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \geq \frac{\mathcal{F}^*-\mathcal{F}_2^*}{\pi_{\mathrm{M}}(a^*|s_t)} \mathop{\mathbb{V}\mathrm{ar}}_{hs\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right] := \underline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}})$$

$$\mathcal{V}_{T-t}(\pi_{\mathrm{M}}) \leq \frac{\mathcal{F}^*-\mathcal{F}_*}{\pi_{\mathrm{M}}(a^*|s_t)} \mathop{\mathbb{V}\mathrm{ar}}_{hs\sim p_t^{\pi_{\mathrm{NM}}}}\left[\mathbb{E}\left[\mathcal{B}(\pi_{\mathrm{NM}}(a^*|hs_t))\right]\right] := \overline{\mathcal{V}}_{T-t}(\pi_{\mathrm{M}})$$

□

Theorem 5.5.4. Ψ_0 *is* NP*-hard.*

Proof. To prove the theorem, it is sufficient to show that there exists a problem $\Psi_c \in$ NP-hard so that $\Psi_c \leq_p \Psi_0$. We show this by reducing 3SAT, a well-known NP-complete problem, to Ψ_0. To derive the reduction we consider two intermediate problems, namely Ψ_1 and Ψ_2. Especially, we aim to show that the following chain of reductions holds

$$\Psi_0 \geq_m \Psi_1 \geq_p \Psi_2 \geq_p \text{3SAT}.$$

First, we define Ψ_1 and we prove that $\Psi_0 \geq_m \Psi_1$. Informally, Ψ_1 is the problem of finding a reward-maximizing Markovian policy $\pi_{\mathrm{M}} \in \Pi_{\mathrm{M}}$ w.r.t. the convex objective (5.2) encoded through a reward function in a convenient POMDP $\mathcal{M}_\Omega^R$. We can build $\mathcal{M}_\Omega^R$ from the MDP $\mathcal{M}$ similarly as the extended MDP $\widetilde{\mathcal{M}}_T^R$ (see Section 4.5.2 and the proof of Lemma 5.4.5 for details), except that the agent only access the observation space Ω instead of the extended state space $\mathcal{S}_\ell$. In particular, we define $\Omega = \mathcal{S}$ (note that $\mathcal{S}$ is the state space of the original MDP $\mathcal{M}$), and $O(s_\ell) = o$ is a deterministic function such that the given observation o is the last state in the history s_ℓ.

Then, the reduction $\Psi_0 \geq_m \Psi_1$ works as follows. We denote as $\mathcal{I}_{\Psi_i}$ the set of possible instances of Ψ_i. We show that Ψ_0 is harder than Ψ_1 by defining the polynomial-time functions ψ and ϕ such that any instance of Ψ_1 can be rewritten through ψ as an instance of Ψ_0, and a solution $\pi_{\mathrm{NM}}^* \in \Pi_{\mathrm{NM}}$ for Ψ_0 can be converted through ϕ into a solution $\pi_{\mathrm{M}}^* \in \Pi_{\mathrm{M}}$ for the original instance of Ψ_1.

$$\begin{array}{ccc} \mathcal{I}_{\Psi_1} & \xrightarrow{\ \psi\ } & \mathcal{I}_{\Psi_0} \\ & & \downarrow \\ \pi_{\mathrm{M}}^* & \xleftarrow[\ \phi\ ]{} & \pi_{\mathrm{NM}}^* \end{array}$$

The function ψ sets $\mathcal{S} = \Omega$ and derives the transition model of $\mathcal{M}$ from the one of $\mathcal{M}_\Omega^R$, while ϕ converts the optimal solution of Ψ_0 by computing

$$\pi_{\mathrm{M}}^*(a|o,t) = \sum_{ho\in\mathcal{H}_o} p_T^{\pi_{\mathrm{NM}}^*}(ho)\pi_{\mathrm{NM}}^*(a|ho),$$

where $\mathcal{H}_o$ stands for the set of histories $h \in \mathcal{H}_t$ ending in the observation $o \in \Omega$. Thus, we have that $\Psi_0 \geq_m \Psi_1$. We now define Ψ_2 as the policy existence problem w.r.t. the problem statement of Ψ_1. Hence, Ψ_2 is the problem of stating whether the value of a reward-maximizing Markovian policy $\pi^*_{\mathrm{M}} \in \arg\max_{\pi \in \Pi_{\mathrm{M}}} \mathcal{J}^R_{\mathcal{M}_\Omega}(\pi)$ is greater than 0. Since computing an optimal policy in POMDPs is in general harder than the relative policy existence problem (Lusena et al., 2001, Section 3), we have that $\Psi_1 \geq_p \Psi_2$.

For the last reduction, i.e., $\Psi_2 \geq_p$ 3SAT, we extend the proof of Theorem 4.13 in (Mundhenk et al., 2000), which states that the policy existence problem for POMDPs is NP-complete. In particular, we show that this holds within the restricted class of POMDPs defined in Ψ_1.

The restrictions on the POMDPs class are the following:

1. The reward function $R(s) \geq 0$ only in the subset of states reachable in T steps, otherwise $R(s) = 0$;
2. $S_\ell := |\mathcal{S}_\ell| = |\Omega|^T$.

Both limitations can be overcome in the following ways:

1. It suffices to add states with deterministic transitions so that $T = m \cdot n$ can be defined a priori, where T is the number of steps needed to reach the state with positive reward through every possible path. Here m is the number of clauses, and n is the number of variables in the 3SAT instance, as defined in (Mundhenk et al., 2000);
2. The POMDPs class defined by Ψ_1 is such that $S_\ell = |\Omega|^T$. Noticing that the set of observations corresponds with the set of variables and that from the previous point $T = m \cdot n$, we have that $|\Omega|^T = n^{m \cdot n}$, while the POMDPs class used by the proof above has $\widetilde{S} = m \cdot n^2$. Notice that $n \geq 2$ and $m \geq 1$ implies that $n^{m \cdot n} \geq m \cdot n^2$. Moreover, notice that every instance of 3SAT has $m \geq 1$ and $n \geq 3$. Hence, to extend the proof to the POMDPs class defined by Ψ_1 it suffices to add a set of states $\mathcal{S}^p_\ell$ such that $R(s) = 0, \forall s \in \mathcal{S}^p_\ell$.

Since the chain $\Psi_0 \geq_m \Psi_1 \geq_p \Psi_2 \geq_p$ 3SAT holds, we have that $\Psi_0 \geq_p$ 3SAT. Moreover, since 3SAT $\in$ NP-complete, we can conclude that Ψ_0 is NP-hard. □

A.3 Proofs of Chapter 7

Theorem 7.3.1. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP. The steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is uniform over $\mathcal{S}$ iff the matrix $\boldsymbol{P}^\pi = \boldsymbol{\Pi}\boldsymbol{P}$ is doubly stochastic.*

Proof. Let us recall the definition of the steady-state distribution of the MC induced by the policy π over the MDP:

$$d^\pi(s) = \sum\nolimits_{s' \in \mathcal{S}} P^\pi(s|s') d^\pi(s'), \quad \forall s \in \mathcal{S}.$$

If $\boldsymbol{d}^\pi$ is a uniform distribution we have:

$$\sum\nolimits_{s' \in \mathcal{S}} P^\pi(s|s') = 1, \quad \forall s \in \mathcal{S}, \tag{A.14}$$

then, the state transition matrix $\boldsymbol{P}^\pi$ is column stochastic, while it is also row stochastic by definition. Conversely, if the matrix $\boldsymbol{P}^\pi$ is doubly stochastic, we aim to prove that a $\boldsymbol{d}^\pi$ that is not uniform cause an inconsistency in the stationary condition $\boldsymbol{d}^\pi = (\boldsymbol{P}^\pi)^T \boldsymbol{d}^\pi$. Let us consider a perturbation of the uniform $\boldsymbol{d}^\pi$, such that $d^\pi(s) = \frac{1}{|\mathcal{S}|}$ for all the states in $\mathcal{S}$ outside of:

$$d^\pi(s_h) = \frac{1}{|\mathcal{S}|} + \alpha, \quad d^\pi(s_l) = \frac{1}{|\mathcal{S}|} - \alpha, \tag{A.15}$$

where α is a, sufficiently small, positive constant. Since $\boldsymbol{P}^\pi$ is doubly stochastic, the sum:

$$d^\pi(s_h) = \sum\nolimits_{s' \in \mathcal{S}} P^\pi(s_h|s') d^\pi(s'), \tag{A.16}$$

is a convex combination of the elements in $\boldsymbol{d}^\pi$. Hence, for the stationary condition to hold, we must have $P^\pi(s_h|s_h) = 1$ and $P^\pi(s_h|s) = 0$ for all s different from s_h. Nevertheless, a state with probability one on the self-loop cannot have a stationary distribution different from 0 or 1. □

Theorem 7.3.2. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP and $\mathbb{P}$ the space of doubly stochastic matrices. The entropy of the steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^\pi) \geq \log|\mathcal{S}| - |\mathcal{S}| \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_\infty^2 .$$

Proof. We start with rewriting the entropy of $\boldsymbol{d}^\pi$ as follows:

$$H(\boldsymbol{d}^\pi) = -\sum_{s \in \mathcal{S}} d^\pi(s) \log d^\pi(s) = -\sum_{s \in \mathcal{S}} d^\pi(s) \log\left(\frac{d^\pi(s)}{|\mathcal{S}|}|\mathcal{S}|\right) = \log|\mathcal{S}| - D_{KL}(\boldsymbol{d}^\pi||\boldsymbol{d}^u), \tag{A.17}$$

where $\boldsymbol{d}^u$ is the uniform distribution over the state space (all the entries equal to $\frac{1}{|\mathcal{S}|}$) and $D_{KL}(p||q)$ is the Kullback-Leibler (KL) divergence between distribution p and q.
Using the reverse Pinsker inequality (Csiszár & Talata, 2006, p. 1012 and Lemma 6.3), we can upper bound the KL divergence between $\boldsymbol{d}^\pi$ and $\boldsymbol{d}^u$:

$$D_{KL}(\boldsymbol{d}^\pi||\boldsymbol{d}^u) \leq \frac{\|\boldsymbol{d}^u - \boldsymbol{d}^\pi\|_1^2}{\min_{s \in \mathcal{S}} d^u(s)} = |\mathcal{S}| \cdot \|\boldsymbol{d}^u - \boldsymbol{d}^\pi\|_1^2 . \tag{A.18}$$

The total variation between the two steady-state distributions $\boldsymbol{d}^\pi$ and $\boldsymbol{d}^u$ can in turn be upper bounded by (see Schweitzer, 1968):

$$\|\boldsymbol{d}^u - \boldsymbol{d}^\pi\|_1 \leq \|\boldsymbol{Z}\|_\infty \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_\infty , \tag{A.19}$$

where $\boldsymbol{Z} = \left(\boldsymbol{I} - \boldsymbol{P^u} + \mathbf{1}_{|\mathcal{S}|}\frac{1}{|\mathcal{S}|}\right)^{-1}$ is the fundamental matrix and $\boldsymbol{P}^u$ is any doubly-stochastic matrix ($\boldsymbol{P}^u \in \mathbb{P}$). Since the fundamental matrix associated to any doubly-stochastic matrix is row stochastic (Hunter, 2010), then $\|\boldsymbol{Z}\|_\infty = 1$. Furthermore, since the bound in Equation (A.19) holds for any $\boldsymbol{P}^u \in \mathbb{P}$, we can rewrite the bound as follows:

$$\|\boldsymbol{d}^u - \boldsymbol{d}^\pi\|_1 \leq \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_\infty . \tag{A.20}$$

Combining Equations (A.18) and (A.20) we get an upper bound to the KL divergence, which, once replaced in Equation (A.17), provides the lower bound in the statement and concludes the proof.

□

Theorem 7.3.3. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP and $\mathbb{P}$ the space of doubly stochastic matrices. The entropy of the steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^\pi) \geq \log|\mathcal{S}| - |\mathcal{S}|^2 \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi}\boldsymbol{P}\|_F^2 .$$

Proof. From the properties of the matrix norms (Petersen et al., 2008), we have that for any $n \times n$ matrix $\boldsymbol{M}$ it holds:

$$\|\boldsymbol{M}\|_F \leq \frac{1}{\sqrt{n}} \|\boldsymbol{M}\|_\infty .$$

As a consequence:

$$\inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_F^2 \geq \frac{1}{|\mathcal{S}|} \left\|\overline{\boldsymbol{P}^u} - \boldsymbol{\Pi P}\right\|_\infty^2 \geq \frac{1}{|\mathcal{S}|} \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_\infty^2 ,$$

where $\overline{\boldsymbol{P}^u} = \arg\inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_F^2$. Combining this inequality with the result in Theorem 7.3.2 concludes the proof. □

Corollary 7.3.4. *The bound in Theorem 7.3.2 is never less than the bound in Theorem 7.3.3.*

Proof. From the properties of the matrix norms (Petersen et al., 2008), we have that for any $n \times n$ matrix $\boldsymbol{M}$ it holds:

$$\frac{\|\boldsymbol{M}\|_\infty}{\sqrt{n}} \leq \|\boldsymbol{M}\|_F \leq \sqrt{n} \|\boldsymbol{M}\|_\infty . \tag{A.21}$$

As a consequence:

$$|\mathcal{S}|^2 \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_F^2 \geq |\mathcal{S}| \left\|\overline{\boldsymbol{P}^u} - \boldsymbol{\Pi P}\right\|_\infty^2 \geq |\mathcal{S}| \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_\infty^2 ,$$

where $\overline{\boldsymbol{P}^u} = \arg\inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_F$. It follows that

$$\log|\mathcal{S}| - |\mathcal{S}|^2 \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_F^2 \leq \log|\mathcal{S}| - |\mathcal{S}| \inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{\Pi P}\|_\infty^2 .$$

□

Theorem 7.3.5. *Let $\boldsymbol{P}$ be the transition matrix of a given MDP. The entropy of the steady-state distribution $\boldsymbol{d}^\pi$ induced by a policy π is lower bounded by:*

$$H(\boldsymbol{d}^\pi) \geq \log|\mathcal{S}| - |\mathcal{S}| \left\| \left(\boldsymbol{I} - (\boldsymbol{\Pi P})^{\boldsymbol{T}} \right) \cdot \mathbf{1}_{|\mathcal{S}|} \right\|_1^2 .$$

Proof. We start with defining the vector $\boldsymbol{c}$ that results from the difference between the vector of ones $\mathbf{1}_{|\mathcal{S}|}$ and the vector of the column sums: $\boldsymbol{c} = \left(\boldsymbol{I} - (\boldsymbol{\Pi P})^{\boldsymbol{T}} \right) \cdot \mathbf{1}_{|\mathcal{S}|}$. We denote with $\widehat{\boldsymbol{P}_x}$ the matrix obtained from $\boldsymbol{P}^\pi$ by adding $\boldsymbol{c}^T$ to the row corresponding to state x:

$$\widehat{\boldsymbol{P}_x}(s, s') = \begin{cases} \boldsymbol{P}^\pi(s, s') + c(s'), & \text{if } s = x \\ \boldsymbol{P}^\pi(s, s'), & \text{otherwise} \end{cases}$$

It is worth noting that, since $\sum_{s \in \mathcal{S}} c(s) = 0$, the column sums and the row sums of matrix $\widehat{\boldsymbol{P}_x}$ are all equal to 1. Nonetheless, $\widehat{\boldsymbol{P}_x}$ is not guaranteed to be doubly stochastic since its entries can be lower than 0. However, it is possible to show that

$$\inf_{\boldsymbol{P}^u \in \mathbb{P}} \|\boldsymbol{P}^u - \boldsymbol{P}^\pi\|_\infty \leq \left\| \widehat{\boldsymbol{P}_x} - \boldsymbol{P}^\pi \right\|_\infty = \|\boldsymbol{c}\|_1 .$$

When $\widehat{\boldsymbol{P}_x}$ is doubly stochastic, the above inequality holds by definition. When $\widehat{\boldsymbol{P}_x}$ has negative entries, it is always possible to transform it to a doubly stochastic matrix without increasing the L_∞ distance from $\boldsymbol{P}^\pi$. In order to remove the negative entries of $\widehat{\boldsymbol{P}_x}$, we need to trade probability with the other states, so as to preserve the row sum. Each state that gives probability to state x, will receive the same amount of probability taken by the columns corresponding to positive values of the

vector $\boldsymbol{c}$. In order to illustrate this procedure, we consider a four-state MDP and a policy π that leads to the following state transition matrix:

$$\boldsymbol{P}^{\pi} = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 \\ 0.3 & 0.5 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.1 & 0 \end{bmatrix}.$$

The corresponding vector $\boldsymbol{c}$ is

$$\boldsymbol{c} = \begin{bmatrix} -0.9 & -0.7 & 0.7 & 0.9 \end{bmatrix}^T.$$

Summing $\boldsymbol{c}^T$ to the first row of $\boldsymbol{P}^{\pi}$ we get:

$$\widehat{\boldsymbol{P}_{s_1}} = \begin{bmatrix} -0.1 & -0.5 & 0.7 & 0.9 \\ 0 & 0.9 & 0.1 & 0 \\ 0.3 & 0.5 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.1 & 0 \end{bmatrix}.$$

Since we have two negative elements, to get a doubly stochastic matrix we can modify the matrix as follows:

- move 0.1 from element $(3,1)$ to $(1,1)$ and (to keep the row sum equal to 1) move 0.1 from $(1,3)$ to $(3,3)$
- move 0.5 from element $(2,2)$ to $(1,2)$ and (to keep the row sum equal to 1) move 0.5 from $(1,3)$ to $(2,3)$

The resulting matrix is:

$$\widehat{\boldsymbol{P}} = \begin{bmatrix} 0 & 0 & 0.1 & 0.9 \\ 0 & 0.4 & 0.6 & 0 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.1 & 0 \end{bmatrix} \in \mathbb{P}.$$

The described procedure yields a doubly stochastic matrix $\widehat{\boldsymbol{P}}$ such that $\left\| \widehat{\boldsymbol{P}} - \boldsymbol{P}^{\pi} \right\|_{\infty} \leq \|\boldsymbol{c}\|_1$. Combining this upper bound with the result in Theorem 7.3.2 concludes the proof. □

A.4 Proofs of Chapter 8

Theorem 8.2.1. (Ajgl & Šimandl, 2011, Sec. 4.1) *Let f be a sampling distribution, f' a target distribution. The estimator $\widehat{H}_k(f'|f)$ is asymptotically unbiased for any $k \in \mathbb{N}$.*

Proof. The proof follow the sketch reported in (Ajgl & Šimandl, 2011, Section 4.1). First, We consider the estimator $\widehat{G}_k(f'|f) = \widehat{H}_k(f'|f) - \ln k + \Psi(k)$, that is:

$$\widehat{G}_k(f'|f) = \sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{V_i^k}{W_i}. \tag{A.22}$$

By considering its expectation w.r.t. the sampling distribution we get:

$$\mathop{\mathbb{E}}_{x \sim f}[\widehat{G}_k(f'|f)] = \mathop{\mathbb{E}}_{x \sim f}\left[\sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{1}{W_i} \frac{R_i^p \pi^{p/2}}{\Gamma(\frac{p}{2}+1)} \right],$$

where, for the sake of clarity, we will replace each logarithmic term as:

$$T_i = \ln \frac{1}{\sum_{j \in \mathcal{N}_i^k} w_j} \frac{R_i^p \pi^{p/2}}{\Gamma(\frac{p}{2}+1)}. \tag{A.23}$$

Since we are interested in the asymptotic mean, we can notice that for $N \to \infty$ we have ${}^{W_i}/_{k} \to {}^{\overline{w}_i}/_{N}$, where $\overline{w}_i = {}^{f'(x)}/_{f(x)}$ are the unnormalized importance weights (Ajgl & Šimandl, 2011, Section 4.1). Thus, for $N \to \infty$, we can see that:

$$\mathop{\mathbb{E}}_{x \sim f}\big[\widehat{G}_k(f'|f)\big] = \mathop{\mathbb{E}}_{x \sim f}\left[\sum_{i=1}^{N} \frac{\overline{w}_i}{N} T_i\right] = \mathop{\mathbb{E}}_{x \sim f'}\left[\frac{1}{N}\sum_{i=1}^{N} T_i\right],$$

where the random variables $T_1, T_2, \ldots, T_N$ are identically distributed, so that:

$$\mathop{\mathbb{E}}_{x \sim f}\big[\widehat{G}_k(f'|f)\big] = \mathop{\mathbb{E}}_{x \sim f'}\big[T_1\big].$$

Thus, to compute the expectation, we have to characterize the following probability for any real number r and any $x \sim f'$:

$$Pr\big[T_1 > r | X_1 = x\big] = Pr\big[R_1 > \rho_r | X_1 = x\big],$$

where we have:

$$\rho_r = \left[\frac{W_i \cdot \Gamma(\frac{p}{2}+1) \cdot e^r}{\pi^{p/2}}\right]^{\frac{1}{p}}.$$

We can rewrite the probability as a binomial:

$$Pr\big[R_1 > \rho_r | X_1 = x\big] = \sum_{i=0}^{k-1} \binom{N-1}{i} \big[Pr(S_{\rho_r,x})\big]^i \big[1 - Pr(S_{\rho_r,x})\big]^{N-1-i},$$

where $P(S_{\rho_r,x})$ is the probability of x lying into the sphere of radius ρ_r, denoted as $S_{\rho_r,x}$. Then, we employ the Poisson Approximation (Hodges & Le Cam, 1960) to this binomial distribution, reducing it to a Poisson distribution having parameter:

$$\begin{aligned}\lim_{N\to\infty}\big[NPr(S_{\rho_r,x})\big] &= \lim_{N\to\infty}\left[Nf(x)\frac{\pi^{p/2}\cdot\rho_r^p}{\Gamma(\frac{p}{2}+1)}\right] \\ &= \lim_{N\to\infty}\big[NW_i f(x)e^r\big] = \overline{w}_i f(x) k e^r = f'(x) k e^r.\end{aligned}$$

Therefore, we get:

$$\lim_{N\to\infty} Pr\big[T_1 > r | X_1 = x\big] = \sum_{i=0}^{k-1} \frac{\big[kf'(x)e^r\big]^i}{i!} e^{-kf'(x)e^r} = Pr\big[T_x > r\big],$$

such that the random variable T_x has the pdf:

$$h_{T_x}(y) = \frac{\big[kf'(x)e^r\big]^k}{(k-1)!} e^{-kf'(x)e^r}, -\infty < y < \infty.$$

Finally, we can compute the expectation following the same steps reported in (Singh et al., 2003, Theorem 8):

$$\begin{aligned}\lim_{N\to\infty}\mathbb{E}\left[T_1|X_1=x\right] &= \int_{-\infty}^{\infty} y\frac{\left[kf'(x)e^r\right]^k}{(k-1)!}e^{-kf'(x)e^r}\,\mathrm{d}y\\ &= \int_0^{\infty}\left[\ln z-\ln k-\ln f'(x)\right]\frac{\left[z^{k-1}\right]}{(k-1)!}e^{-z}\,\mathrm{d}z\\ &= \frac{1}{\Gamma(k)}\int_0^{\infty}\left[\ln(z)z^{k-1}e^{-z}\right]\mathrm{d}z-\ln k-\ln f'(x)\\ &= \Psi(k)-\ln k-\ln f'(x),\end{aligned}$$

which for a generic x it yields:

$$\lim_{N\to\infty}\mathop{\mathbb{E}}_{x\sim f'}\left[T_1\right]=H(f')-\ln k+\Psi(k)=\lim_{N\to\infty}\mathop{\mathbb{E}}_{x\sim f}\left[\widehat{G}_k(f'|f)\right].$$

□

Theorem 8.2.2. *Let f be a sampling distribution, f' a target distribution. The asymptotic variance of the estimator $\widehat{H}_k(f'|f)$ is given by:*

$$\lim_{N\to\infty}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\widehat{H}_k(f'|f)\right]=\frac{1}{N}\left(\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}\ln\overline{w}\right]+\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}\ln R^p\right]+\left(\ln C\right)^2\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}\right]\right),$$

where $\overline{w}=\frac{f'(x)}{f(x)}$, and $C=\frac{N\pi^{p/2}}{k\Gamma(p/2+1)}$ is a constant.

Proof. We consider the limit of the variance of $\widehat{H}_k(f'|f)$, we have:

$$\begin{aligned}\lim_{N\to\infty}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\widehat{H}_k(f'|f)\right] &= \lim_{N\to\infty}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\widehat{G}_k(f'|f)\right]\\ &= \lim_{N\to\infty}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\sum_{i=1}^{N}\frac{W_i}{k}T_i\right]=\lim_{N\to\infty}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\frac{1}{N}\sum_{i=1}^{N}\overline{w}_iT_i\right],\end{aligned}$$

where $\widehat{G}_k(f'|f)$ is the estimator without the bias correcting term (A.22), and T_i are the logarithmic terms (A.23). Then, since the distribution of the random vector $\left(\overline{w}_1T_1,\overline{w}_2T_2,\ldots,\overline{w}_NT_N\right)$ is the same as any permutation of it (Singh et al., 2003):

$$\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\frac{1}{N}\sum_{i=1}^{N}\overline{w}_iT_i\right]=\frac{\mathbb{V}\mathrm{ar}_{x\sim f}\left[\overline{w}_1T_1\right]}{N}+\frac{N(N-1)}{N^2}\,\mathrm{Cov}\left(\overline{w}_1T_1,\overline{w}_2T_2\right).$$

Assuming that, for $N\to\infty$, the term $\mathrm{Cov}(\overline{w}_1T_1,\overline{w}_2T_2)\to 0$ as its non-IW counterpart (Singh et al., 2003, Theorem 11), we are interested on the first term $\mathbb{V}\mathrm{ar}_{x\sim f}\left[\overline{w}_1T_1\right]$. Especially, we can derive:

$$\begin{aligned}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1T_1\right] &= \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1\ln\frac{1}{\overline{w}_1}\frac{N}{k}\frac{R_1^p\pi^{p/2}}{\Gamma(\frac{p}{2}+1)}\right]\\ &= \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[-\overline{w}_1\ln\overline{w}_1+\overline{w}_1\ln R_1^p+\overline{w}_1\ln\frac{N}{k}\frac{\pi^{p/2}}{\Gamma(\frac{p}{2}+1)}\right],\end{aligned}$$

where in the following, we will substitute $C = \frac{N\pi^{p/2}}{k\Gamma(p/2+1)}$. Then, we can write the second momentum as:

$$\mathop{\mathbb{E}}_{x\sim f}\left[\left(\overline{w}_1 T_1\right)^2\right] = \mathop{\mathbb{E}}_{x\sim f}\Big[\left(\overline{w}_1 \ln \overline{w}_1\right)^2 + \left(\overline{w}_1 \ln R_1^p\right)^2 + \left(\overline{w}_1 \ln C\right)^2 - 2\overline{w}_1^2 \ln \overline{w}_1 \ln R_1^p - 2\overline{w}_1^2 \ln \overline{w}_1 \ln C + 2\overline{w}_1^2 \ln R_1^p \ln C\Big],$$

while the squared expected value is:

$$\begin{aligned}\left(\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 T_1\right]\right)^2 &= \left(-\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 \ln \overline{w}_1\right] + \mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 \ln R_1^p\right] + \mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 \ln C\right]\right)^2 \\ &= \left(\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 \ln \overline{w}_1\right]\right)^2 + \left(\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 R_1^p\right]\right)^2 + \left(\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 \ln C\right]\right)^2 \\ &\quad - 2\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1^2 \ln \overline{w}_1 \ln R_1^p\right] - 2\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1^2 \ln \overline{w}_1 \ln C\right] + 2\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1^2 \ln C \ln R_1^p\right].\end{aligned}$$

Thus, we have:

$$\begin{aligned}\mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 T_1\right] &= \mathop{\mathbb{E}}_{x\sim f}\left[\left(\overline{w}_1 T_1\right)^2\right] - \left(\mathop{\mathbb{E}}_{x\sim f}\left[\overline{w}_1 T_1\right]\right)^2 \\ &= \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 \ln \overline{w}_1\right] + \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 \ln R_1^p\right] + \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 \ln C\right] \\ &= \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 \ln \overline{w}_1\right] + \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1 \ln R_1^p\right] + \left(\ln C\right)^2 \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\left[\overline{w}_1\right].\end{aligned}$$

Summing it up, we can write the asymptotic order of the variance as:

$$\lim_{N\to\infty} \mathop{\mathbb{V}\mathrm{ar}}_{x\sim f}\big[\widehat{H}(f'|f)\big] = \frac{\mathbb{V}\mathrm{ar}_{x\sim f}\left[\overline{w}(x)\ln \overline{w}(x)\right] + \mathbb{V}\mathrm{ar}_{x\sim f}\left[\overline{w}(x)\ln R(x)^p\right] + \left(\ln C\right)^2 \mathbb{V}\mathrm{ar}_{x\sim f}\left[\overline{w}(x)\right]}{N}$$

□

Theorem 8.4.1. *Let $\pi_{\boldsymbol{\theta}}$ be the current policy and $\pi_{\boldsymbol{\theta}'}$ a target policy. The gradient of the IW estimator $\widehat{H}_k(d_T^{\boldsymbol{\theta}'}|d_T^{\boldsymbol{\theta}})$ w.r.t. $\boldsymbol{\theta}'$ is given by*

$$\nabla_{\boldsymbol{\theta}'} \widehat{H}_k(d_T^{\boldsymbol{\theta}'}|d_T^{\boldsymbol{\theta}}) = -\sum_{i=0}^{N} \frac{\nabla_{\boldsymbol{\theta}'} W_i}{k}\left(V_i^k + \ln \frac{W_i}{V_i^k}\right),$$

where

$$\nabla_{\boldsymbol{\theta}'} W_i = \sum_{j\in\mathcal{N}_i^k} w_j \times \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N}\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}\sum_{z=0}^{t}\nabla_{\boldsymbol{\theta}'}\ln\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\sum_{n=1}^{N}\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}\right).$$

Proof. We consider the IW entropy estimator (8.2), that is

$$\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) = -\sum_{i=1}^{N}\frac{W_i}{k}\ln\frac{W_i}{V_i^k} + \ln k - \Psi(k), \tag{A.24}$$

where:

$$W_i = \sum_{j \in \mathcal{N}_i^k} w_j = \sum_{j \in \mathcal{N}_i^k} \frac{\prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z)}{\pi_{\boldsymbol{\theta}}(a_j^z|s_j^z)}}{\sum_{n=1}^{N} \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}.$$

Then, by differentiating Equation (A.24) w.r.t. $\boldsymbol{\theta}'$, we have:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}'} \widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) \\
&\quad = -\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}'} \left(\frac{\sum_{j \in \mathcal{N}_i^k} w_j}{k} \ln \frac{\sum_{j \in \mathcal{N}_i^k} w_j}{V_i^k} + \ln k - \psi(k) \right) \\
&\quad = -\sum_{i=1}^{N} \left(\frac{\sum_{j \in \mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j}{k} \ln \frac{\sum_{j \in \mathcal{N}_i^k} w_j}{V_i^k} + \frac{\sum_{j \in \mathcal{N}_i^k} w_j}{k} \frac{V_i^k}{\sum_{j \in \mathcal{N}_i^k} w_j} \sum_{j \in \mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j \right) \\
&\quad = -\sum_{i=1}^{N} \frac{\sum_{j \in \mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j}{k} \left(V_i^k + \ln \frac{\sum_{j \in \mathcal{N}_i^k} w_j}{V_i^k} \right). \qquad \text{(A.25)}
\end{aligned}$$

Finally, we consider the expression of $\nabla_{\boldsymbol{\theta}'} w_j$ in Equation (A.25) to conclude the proof:

$$\begin{aligned}
\nabla_{\theta} w_j &= w_j \nabla_{\boldsymbol{\theta}'} \ln w_j \\
&= w_j \nabla_{\boldsymbol{\theta}'} \left(\ln \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z)}{\pi_{\boldsymbol{\theta}}(a_j^z|s_j^z)} - \ln \sum_{n=1}^{N} \overbrace{\prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}^{\boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}'} \boldsymbol{Prod}_n}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \boldsymbol{Prod}_n \nabla_{\boldsymbol{\theta}'} \ln \left(\boldsymbol{Prod}_n \right)}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \left(\boldsymbol{Prod}_n \sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z) \right)}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right).
\end{aligned}$$

□

A.5 Proofs of Chapter 9

Theorem 9.5.2. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs satisfying Ass. 9.5.1. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by the policy π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ as*

$$\mathcal{D}_{\boldsymbol{\mathcal{M}}} := \sup_{\substack{\pi \in \Pi \\ \mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}}} d_{W_1}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) \leq \sup_{P', P \in \boldsymbol{\mathcal{M}}} \frac{1 - L_{P^\pi}^T}{1 - L_{P^\pi}} \sup_{\substack{s \in \mathcal{S} \\ a \in \mathcal{A}}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)).$$

Proof. The proof follows techniques from (Pirotta et al., 2015). Let us report a preliminary result which states that the function $h_f(\overline{s}) = \int_{\mathcal{A}} \pi(\overline{a}|\overline{s}) \int_{\mathcal{S}} P(s|\overline{s}, \overline{a}) \, \mathrm{d}s \, \mathrm{d}\overline{a}$ has a Lipschitz constant equal

to L_{P^π} (Pirotta et al., 2015, Lemma 3):

$$\begin{aligned}\left|h_f(\overline{s}') - h_f(\overline{s})\right| &= \left| \int_{\mathcal{S}} f(s) \int_{\mathcal{A}} \pi(a|\overline{s}') P(s|\overline{s}', a)\, \mathrm{d}a\, \mathrm{d}s - \int_{\mathcal{S}} f(s) \int_{\mathcal{A}} \pi(a|\overline{s}) P(s|\overline{s}, a)\, \mathrm{d}a\, \mathrm{d}s \right| \\ &= \left| \int_{\mathcal{S}} f(s) \Big(P^\pi(s|\overline{s}') - P^\pi(s|\overline{s}) \Big)\, \mathrm{d}s \right| \leq L_{P^\pi} d_{\mathcal{S}}(\overline{s}', \overline{s}), \end{aligned} \tag{A.26}$$

where $d_{\mathcal{S}}$ is a metric over $\mathcal{S}$ and $P^\pi(s|\overline{s}) = \int_{\mathcal{A}} \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a})\, \mathrm{d}\overline{a}$. Then, we note that the marginal state distribution over T steps $d_\pi^{\mathcal{M}}$ can be written as a sum of the contributions $d_{\pi,t}^{\mathcal{M}}$ related to any time step $t \in [T]$, which is

$$d_\pi^{\mathcal{M}}(s) = \frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}}(s). \tag{A.27}$$

Hence, we can look at the Wasserstein distance of the state distributions for some $t \in [T]$ and $\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}$. We obtain

$$\begin{aligned}
&d_{W_1}(d_{\pi,t}^{\mathcal{M}'}, d_{\pi,t}^{\mathcal{M}}) \\
&= \sup_f \left\{ \left| \int_{\mathcal{S}} \Big(d_{\pi,t}^{\mathcal{M}'}(s) - d_{\pi,t}^{\mathcal{M}}(s) \Big) f(s)\, \mathrm{d}s \right| : \|f\|_L \leq 1 \right\} && \text{(A.28)} \\
&= \sup_f \left\{ \left| \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \Big(d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) \pi(\overline{a}|\overline{s}) P'(s|\overline{s}, \overline{a}) \right.\right. \\
&\quad \left.\left. - d_{\pi,t-1}^{\mathcal{M}}(\overline{s}) \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right| : \|f\|_L \leq 1 \right\} \\
&= \sup_f \left\{ \left| \int_{\mathcal{S}} d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\overline{a}|\overline{s}) \Big(P'(s|\overline{s}, \overline{a}) - P(s|\overline{s}, \overline{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right.\right. && \text{(A.29)} \\
&\quad \left.\left. + \int_{\mathcal{S}} \Big(d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) - d_{\pi,t-1}^{\mathcal{M}}(\overline{s}) \Big) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a}) f(s)\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right| : \|f\|_L \leq 1 \right\} && \text{(A.30)} \\
&\leq \sup_f \left\{ \left| \int_{\mathcal{S}} d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\overline{a}|\overline{s}) \Big(P'(s|\overline{s}, \overline{a}) - P(s|\overline{s}, \overline{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right| : \|f\|_L \leq 1 \right\} \\
&\quad + \sup_f \left\{ \left| \int_{\mathcal{S}} \Big(d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) - d_{\pi,t-1}^{\mathcal{M}}(\overline{s}) \Big) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a}) f(s)\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right| : \|f\|_L \leq 1 \right\} \\
&\leq \sup_f \left\{ \int_{\mathcal{S}} d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) \int_{\mathcal{A}} \pi(\overline{a}|\overline{s})\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s} \right. \\
&\quad \times \sup_{\overline{s} \in \mathcal{S}, \overline{a} \in \mathcal{A}} \left\{ \left| \int_{\mathcal{S}} \Big(P'(s|\overline{s}, \overline{a}) - P(s|\overline{s}, \overline{a}) \Big) f(s)\, \mathrm{d}s \right| \right\} : \|f\|_L \leq 1 \Big\} \\
&\quad + L_{P^\pi} \sup_f \left\{ \left| \int_{\mathcal{S}} \Big(d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) - d_{\pi,t-1}^{\mathcal{M}}(\overline{s}) \Big) \frac{h_f(\overline{s})}{L_{P^\pi}}\, \mathrm{d}\overline{s} \right| : \|f\|_L \leq 1 \right\} && \text{(A.31)} \\
&= \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L_{P^\pi} d_{W_1}(d_{\pi,t-1}^{\mathcal{M}'}, d_{\pi,t-1}^{\mathcal{M}}), && \text{(A.32)}
\end{aligned}$$

where we plugged the common temporal relation $d_{\pi,t}^{\mathcal{M}}(s') = \int_{\mathcal{S}} \int_{\mathcal{A}} d_{\pi,t-1}^{\mathcal{M}}(s) \pi(a|s) P(s'|s,a)\, \mathrm{d}s\, \mathrm{d}a$ into (A.28), we sum and subtract $\int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d_{\pi,t-1}^{\mathcal{M}'}(\overline{s}) \pi(\overline{a}|\overline{s}) P(s|\overline{s}, \overline{a})\, \mathrm{d}s\, \mathrm{d}\overline{a}\, \mathrm{d}\overline{s}$ to get (A.29), (A.30), and we apply the inequality in (A.26) to obtain (A.31) and then (A.32). To get rid of the dependence

to the state distributions $d^{\mathcal{M}'}_{\pi,t-1}$ and $d^{\mathcal{M}}_{\pi,t-1}$, we repeatedly unroll (A.32) to get

$$d_{W_1}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) \leq \left(\sum_{j=0}^{t} L^j_{P^\pi}\right) \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^t_{P^\pi} d_{W_1}(D', D) \quad \text{(A.33)}$$

$$= \left(\frac{1-L^t_{P^\pi}}{1-L_{P^\pi}}\right) \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^t_{P^\pi} d_{W_1}(D', D), \quad \text{(A.34)}$$

where we note that $d_{W_1}(d^{\mathcal{M}'}_{\pi,0}, d^{\mathcal{M}}_{\pi,0}) = d_{W_1}(D', D)$ to derive (A.33), and we assume $L_{P^\pi} < 1$ (Assumption 9.5.1) to get (A.34) from (A.33). As a side note, when the state and action spaces are discrete, a natural choice of a metric is $d_{\mathcal{S}}(s', s) = \mathbb{1}(s' \neq s)$ and $d_{\mathcal{A}} = \mathbb{1}(a' \neq a)$, which results in the Wasserstein distance being equivalent to the total variation, the constant $L_{P^\pi} = 1$, and $\sum_{j=0}^{t} L^j_{P^\pi} = t$. More details over the Lipschitz constant L_{P^π} can be found in (Pirotta et al., 2015). Finally, we can exploit the result in (A.34) to write

$$d_{W_1}(d^{\mathcal{M}'}_{\pi}, d^{\mathcal{M}}_{\pi}) = \sup_f \left\{ \left| \int_{\mathcal{S}} \left(\frac{1}{T}\sum_{t=0}^{T-1} d^{\mathcal{M}'}_{\pi,t}(s) - \frac{1}{T}\sum_{t=0}^{T-1} d^{\mathcal{M}}_{\pi,t}(s) \right) f(s)\,\mathrm{d}s \right| : \|f\|_L \leq 1 \right\} \quad \text{(A.35)}$$

$$\leq \frac{1}{T}\sum_{t=0}^{T-1} \sup_f \left\{ \left| \int_{\mathcal{S}} \left(d^{\mathcal{M}'}_{\pi,t}(s) - d^{\mathcal{M}}_{\pi,t}(s) \right) f(s)\,\mathrm{d}s \right| : \|f\|_L \leq 1 \right\}$$

$$\leq \frac{1}{T}\sum_{t=0}^{T-1} \frac{1-L^t_{P^\pi}}{1-L_{P^\pi}} \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^t_{P^\pi} d_{W_1}(D', D)$$

$$\leq \frac{1-L^T_{P^\pi}}{1-L_{P^\pi}} \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^T_{P^\pi} d_{W_1}(D', D), \quad \text{(A.36)}$$

in which we use (A.27) to get (A.35). The result follows from (A.36) by assuming the initial state distribution D to be shared across all the CMPs in $\boldsymbol{\mathcal{M}}$, and taking the supremum over $P', P \in \boldsymbol{\mathcal{M}}$. □

Theorem 9.5.3. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy, and let $d^{\mathcal{M}}_{\pi}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ as*

$$\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi) := \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} d_{TV}(d^{\mathcal{M}'}_{\pi}, d^{\mathcal{M}}_{\pi}) \leq \sup_{P',P\in\boldsymbol{\mathcal{M}}} T \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi} \\ a\sim\pi(\cdot|s)}} d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)).$$

Proof. The proof follows techniques from (Metelli et al., 2018a), especially Proposition 3.1. Without loss of generality, we take $\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}$. With some overloading of notation, we will alternatively identify a CMP with the tuple $\mathcal{M}$ or its transition model P. Let us start considering the TV between the marginal state distributions induced by π over $\mathcal{M}', \mathcal{M}$, we can write

$$d_{TV}(d^{\mathcal{M}'}_{\pi}, d^{\mathcal{M}}_{\pi})$$

$$= \frac{1}{2}\int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi}(s) - d^{\mathcal{M}}_{\pi}(s) \right| \mathrm{d}s = \frac{1}{2}\int_{\mathcal{S}} \left| \frac{1}{T}\sum_{t=0}^{T-1} d^{\mathcal{M}'}_{\pi,t}(s) - \frac{1}{T}\sum_{t=0}^{T-1} d^{\mathcal{M}}_{\pi,t}(s) \right| \mathrm{d}s \quad \text{(A.37)}$$

$$\leq \frac{1}{2T}\sum_{t=0}^{T-1} \int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi,t}(s) - d^{\mathcal{M}}_{\pi,t}(s) \right| \mathrm{d}s = \frac{1}{T}\sum_{t=0}^{T-1} d_{TV}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}), \quad \text{(A.38)}$$

where we use (A.27) to get (A.37). Then, we provide an upper bound to each term of the final sum in (A.38), i.e.,

$$
\begin{aligned}
& d_{TV}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) \\
&\quad = \frac{1}{2}\int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi,t}(s) - d^{\mathcal{M}}_{\pi,t}(s) \right| \mathrm{d}s \\
&\quad = \frac{1}{2}\int_{\mathcal{S}} \left| \int_{\mathcal{A}}\int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\overline{s})\pi(\overline{a}|\overline{s})P'(s|\overline{s},\overline{a}) - d^{\mathcal{M}}_{\pi,t-1}(\overline{s})\pi(\overline{a}|\overline{s})P(s|\overline{s},\overline{a}) \right| \mathrm{d}\overline{s}\,\mathrm{d}\overline{a}\,\mathrm{d}s && \text{(A.39)} \\
&\quad \leq \frac{1}{2}\int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi,t-1}(\overline{s}) - d^{\mathcal{M}}_{\pi,t-1}(\overline{s}) \right| \int_{\mathcal{A}}\int_{\mathcal{S}} \pi(\overline{a}|\overline{s})P'(s|\overline{s},\overline{a})\,\mathrm{d}\overline{s}\,\mathrm{d}\overline{a}\,\mathrm{d}s && \text{(A.40)} \\
&\qquad + \frac{1}{2}\int_{\mathcal{S}}\int_{\mathcal{A}} d^{\mathcal{M}}_{\pi,t-1}(\overline{s})\pi(\overline{a}|\overline{s}) \int_{\mathcal{S}} \left| P'(s|\overline{s},\overline{a}) - P(s|\overline{s},\overline{a}) \right| \mathrm{d}\overline{s}\,\mathrm{d}\overline{a}\,\mathrm{d}s && \text{(A.41)} \\
&\quad = d_{TV}(d^{\mathcal{M}'}_{\pi,t-1}, d^{\mathcal{M}}_{\pi,t-1}) + \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,t-1} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] && \text{(A.42)} \\
&\quad = \sum_{j=1}^{t-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,j} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D), && \text{(A.43)}
\end{aligned}
$$

where we use the temporal relation $d^{\mathcal{M}}_{\pi,t}(s') = \int_{\mathcal{S}}\int_{\mathcal{A}} d^{\mathcal{M}}_{\pi,t-1}(s)\pi(a|s)P(s'|s,a)\,\mathrm{d}s\,\mathrm{d}a$ to get (A.39), in which we sum and subtract $\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d^{\mathcal{M}}_{\pi,t-1}(\overline{s})\pi(\overline{a}|\overline{s})P(s|\overline{s},\overline{a})\,\mathrm{d}s\,\mathrm{d}\overline{a}\,\mathrm{d}\overline{s}$ to obtain (A.40) and (A.41), and we repeatedly unroll (A.42) to write (A.43), noting that $d_{TV}(d^{\mathcal{M}'}_{\pi,0}, d^{\mathcal{M}}_{\pi,0}) = d_{TV}(D', D)$. Finally, we can plug (A.43) in (A.38) to get

$$
\begin{aligned}
& d_{TV}(d^{\mathcal{M}'}_{\pi}, d^{\mathcal{M}}_{\pi}) \\
&\quad \leq \frac{1}{T}\sum_{t=0}^{T-1} d_{TV}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) \\
&\quad \leq \frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{t-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,j} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D) \\
&\quad \leq \sum_{t=0}^{T-1}\int_{\mathcal{S}} \frac{1}{T}\sum_{j=0}^{T-1} d^{\mathcal{M}}_{\pi,j}(s) \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] \mathrm{d}s + d_{TV}(D', D) && \text{(A.44)} \\
&\quad = \sum_{t=0}^{T-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D) && \text{(A.45)} \\
&\quad = T \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D), && \text{(A.46)}
\end{aligned}
$$

in which we have used (A.27) to obtain (A.45) from (A.44). The final result is straightforward from (A.45) by assuming the initial state distribution D to be shared across all the CMPs in $\boldsymbol{\mathcal{M}}$, and taking the supremum over $P', P \in \boldsymbol{\mathcal{M}}$. □

Theorem 9.5.4. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy and $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ the corresponding π-diameter of $\boldsymbol{\mathcal{M}}$. Let $d^{\mathcal{M}}_{\pi}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$, and let $\sigma_{\boldsymbol{\mathcal{M}}} \leq \sigma_{\mathcal{M}} := \inf_{s\in\mathcal{S}} d^{\mathcal{M}}_{\pi}(s), \forall \mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the entropy gap of*

the policy π within the model class $\boldsymbol{\mathcal{M}}$ as

$$\sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left|H(d_\pi^{\mathcal{M}'}) - H(d_\pi^{\mathcal{M}})\right| \leq \left(\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\right)^2/\sigma_{\boldsymbol{\mathcal{M}}} + \mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\log(1/\sigma_{\boldsymbol{\mathcal{M}}})$$

Proof. Let us expand the entropy gap of the policy π as

$$\begin{aligned}
&\sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left|H(d_\pi^{\mathcal{M}'}) - H(d_\pi^{\mathcal{M}})\right| \\
&\quad= \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{\left| -\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\log d_\pi^{\mathcal{M}'}(s)\,\mathrm{d}s + \int_{\mathcal{S}} d_\pi^{\mathcal{M}}(s)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s\right|\right\} && \text{(A.47)}\\
&\quad\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{\left| \int_{\mathcal{S}} \Big(d_\pi^{\mathcal{M}}(s) - d_\pi^{\mathcal{M}'}(s)\Big)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s\right|\right. \\
&\qquad\left. + \left|\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\Big(\log d_\pi^{\mathcal{M}'}(s) - \log d_\pi^{\mathcal{M}}(s)\Big)\,\mathrm{d}s\right|\right\} && \text{(A.48)}\\
&\quad\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{ -\log\sigma_{\mathcal{M}}\int_{\mathcal{S}}\left|d_\pi^{\mathcal{M}'}(s) - d_\pi^{\mathcal{M}'}(s)\right|\mathrm{d}s + D_{KL}\big(d_\pi^{\mathcal{M}'}||d_\pi^{\mathcal{M}}\big)\right\} && \text{(A.49)}\\
&\quad\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{ -\log\sigma_{\mathcal{M}} D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) + \big(D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}})\big)^2/\sigma_{\mathcal{M}}\right\} && \text{(A.50)}\\
&\quad\leq \left(\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\right)^2/\sigma_{\boldsymbol{\mathcal{M}}} - \mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\log\sigma_{\boldsymbol{\mathcal{M}}} && \text{(A.51)}
\end{aligned}$$

in which we sum and subtract $\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s$ to obtain (A.48) from (A.47), $\log d_\pi^{\mathcal{M}}(s)$ is upper bounded with $\log\sigma_{\mathcal{M}}$ to get (A.49), and we use the reverse Pinsker's inequality $D_{KL}(p||q) \leq (D_{TV}(p,q))^2/\inf_{x\in\mathcal{X}} q(x)$ (Csiszár & Talata, 2006, p. 1012 and Lemma 6.3) to obtain (A.56). Finally, we get the result by upper bounding $D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}})$ with the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ and $\sigma_{\mathcal{M}}$ with $\sigma_{\boldsymbol{\mathcal{M}}}$ in (A.50). □

Proposition 9.6.1. *The policy gradient of the exploration objective $\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ w.r.t. $\boldsymbol{\theta}$ is given by*

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim p_{\boldsymbol{\mathcal{M}}}\\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}}\Bigg[&\Bigg(\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{t,\tau}|s_{t,\tau})\Bigg)\\
&\times\Big(H_\tau - \mathrm{VaR}_\alpha(H_\tau)\Big)\Big|H_\tau \leq \mathrm{VaR}_\alpha(H_\tau)\Bigg].
\end{aligned}$$

Proof. Let us start from expanding the exploration objective (9.2) to write

$$\begin{aligned}
\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi) &= \mathrm{CVaR}_\alpha(H_\tau)\\
&= \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim p_{\boldsymbol{\mathcal{M}}}\\ \tau\sim p_{\pi,\mathcal{M}}}}\big[H_\tau \mid H_\tau \leq \mathrm{VaR}_\alpha(H_\tau)\big] = \frac{1}{\alpha}\int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)h\,\mathrm{d}h, && \text{(A.52)}
\end{aligned}$$

where $p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}$ is the probability density function (pdf) of the random variable H_τ when the policy $\pi_{\boldsymbol{\theta}}$ is deployed on the class of environments $\boldsymbol{\mathcal{M}}$, and the last equality comes from the definition of CVaR (Rockafellar et al., 2000). Before computing the gradient of (A.52), we derive a preliminary result for later use, i.e.,

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}}\int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)\,\mathrm{d}h\\
&\quad= \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)\,\mathrm{d}h + \nabla_{\boldsymbol{\theta}}\,\mathrm{VaR}_\alpha(H_\tau)p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(\mathrm{VaR}_\alpha(H_\tau)) = 0, && \text{(A.53)}
\end{aligned}$$

which follows directly from the Leibniz integral rule, noting that $\mathrm{VaR}_\alpha(H_\tau)$ depends on $\boldsymbol{\theta}$ through the pdf of H_τ. We now take the gradient of (A.52) to get

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi) \\
&= \nabla_{\boldsymbol{\theta}} \frac{1}{\alpha} \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) h \,\mathrm{d}h \\
&= \frac{1}{\alpha} \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) h \,\mathrm{d}h + \frac{1}{\alpha} \nabla_{\boldsymbol{\theta}} \,\mathrm{VaR}_\alpha(H_\tau)\, \mathrm{VaR}_\alpha(H_\tau) p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(\mathrm{VaR}_\alpha(H_\tau)) \qquad \text{(A.54)} \\
&= \frac{1}{\alpha} \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) \Big(h - \mathrm{VaR}_\alpha(H_\tau) \Big) \,\mathrm{d}h, \qquad \text{(A.55)}
\end{aligned}$$

where (A.54) follows from the Leibniz integral rule, and (A.55) is obtained from (A.54) through (A.53), which we can write as $p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(\mathrm{VaR}_\alpha(H_\tau)) = \frac{1}{\nabla_{\boldsymbol{\theta}} \mathrm{VaR}_\alpha(H_\tau)} \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) \,\mathrm{d}h$. All of the steps above are straightforward replications of the derivations by Tamar et al. (2015b), Proposition 1. To conclude the proof we just have to compute the term $\nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)$, which is specific to our setting. Especially, we note that

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) \\
&\quad = \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \delta(h - H_\tau) \,\mathrm{d}\tau \,\mathrm{d}\mathcal{M} \qquad \text{(A.56)} \\
&\quad = \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \nabla_{\boldsymbol{\theta}} \log p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \delta(h - H_\tau) \,\mathrm{d}\tau \,\mathrm{d}\mathcal{M} \\
&\quad = \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau}|s_{t,\tau}) \Big) \delta(h - H_\tau) \,\mathrm{d}\tau \,\mathrm{d}\mathcal{M}, \qquad \text{(A.57)}
\end{aligned}$$

where (A.56) and (A.57) are straightforward from the definitions in Section 9.3, and $\mathcal{T}$ is the set of feasible trajectories of length T. Finally, the result follows by plugging (A.57) into (A.55), which gives

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi) = \frac{1}{\alpha} \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \\
\times \int_{-\infty}^{\mathrm{VaR}_\alpha(H_\tau)} \delta(h - H_\tau) \Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau}|s_{t,\tau}) \Big) \Big(h - \mathrm{VaR}_\alpha(H_\tau) \Big) \,\mathrm{d}h \,\mathrm{d}\tau \,\mathrm{d}\mathcal{M}.
\end{aligned}$$

□

Proposition 9.6.2. *Let f_H be the pdf of H_τ, for which there exist $\eta, \Delta > 0$ such that $f_H(H_\tau) > \eta$ for all $H_\tau \in \big[\mathrm{VaR}_\alpha(H_\tau) - \frac{\Delta}{2}, \mathrm{VaR}_\alpha(H_\tau) + \frac{\Delta}{2}\big]$. Let $\mathcal{U}$ be a large constant such that $f_{\tau_i} \leq \mathcal{U}$ for all τ_i. The number of samples n^* for which the estimation error ϵ of $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ is lower than the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ with probability at least $\delta \in (0,1)$ is given by*

$$n^* = \frac{\log 2/\delta}{2\eta^2 \min\{\mathcal{U}^2 \alpha^2 b^2, \Delta^2\}}.$$

Proof. The proof is straightforward by considering the estimation error ϵ of $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ equal to the upper bound of the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ from Lemma A.5.1 (see below), i.e., $\epsilon = \mathcal{U}\alpha b$. Then, we set $\delta = 2\exp\big(-2n^*\eta^2 \min(\mathcal{U}^2\alpha^2 b^2, \Delta^2)\big)$ from Lemma A.5.2 (see below), which gives the result through simple calculations. □

Lemma A.5.1. *The expected bias of the policy gradient estimate $\widehat{\nabla}^b_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ can be upper bounded as*

$$\mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \big[\text{bias}\big] = \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \big[\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \widehat{\nabla}^b_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})\big] \leq \mathcal{U}\alpha b,$$

where $\mathcal{U}$ is a constant such that $f_{\tau_i} \leq \mathcal{U}$ for all τ_i.

Proof. This Lemma can be easily derived by means of

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \big[\text{bias}\big] \\
&= \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \Big[\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \widehat{\nabla}^b_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})\Big] \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} f_{\tau_i} \left(\widehat{H}_{\tau_i} - \text{VaR}_{\alpha}(H_{\tau_i}) - b\right) \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{\text{VaR}}_{\alpha}(H_{\tau_i}))\right] \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau} \left(\widehat{H}_{\tau} - \text{VaR}_{\alpha}(H_{\tau}) - b\right) \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{\text{VaR}}_{\alpha}(H_{\tau}))\right] \qquad \text{(A.58)} \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) + \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau}\, b\, \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{\text{VaR}}_{\alpha}(H_{\tau}))\right] \qquad \text{(A.59)} \\
&= \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau}\, b\, \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{\text{VaR}}_{\alpha}(H_{\tau}))\right] \leq \mathcal{U}\alpha b, \qquad \text{(A.60)}
\end{aligned}$$

where (A.59) follows from (A.58) by noting that the estimator without the baseline term is unbiased (Tamar et al., 2015b), and (A.60) is obtained by upper bounding f_{τ} with $\mathcal{U}$ and noting that $\mathbb{E}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \big[\mathbb{1}(\widehat{H}_{\tau} \leq \widehat{\text{VaR}}_{\alpha}(H_{\tau}))\big] = \alpha.$ □

Lemma A.5.2 (VaR concentration bound from (Kolla et al., 2019))**.** *Let X be a continuous random variable with a pdf f_X for which there exist $\eta, \Delta > 0$ such that $f_X(x) > \eta$ for all $x \in \big[\text{VaR}_{\alpha}(X) - \frac{\Delta}{2}, \text{VaR}_{\alpha}(X) + \frac{\Delta}{2}\big]$. Then, for any $\epsilon > 0$ we have*

$$Pr\big[|\widehat{\text{VaR}}_{\alpha}(X)_{\alpha} - \text{VaR}_{\alpha}(X)| \geq \epsilon\big] \leq 2\exp\big(-2n\eta^2 \min(\epsilon^2, \Delta^2)\big),$$

where $n \in \mathbb{N}$ is the number of samples employed to estimate $\widehat{\text{VaR}}_{\alpha}(X)$.

APPENDIX B

Experimental Details

B.1 Experimental Details of Chapter 8

B.1.1 Environments

For all the environments presented in Chapter 8, we use off-the-shelf implementations from the OpenAI gym library (Brockman et al., 2016) with the exception of GridWorld, which we coded from scratch as described in the next paragraph. We also slightly modified the MountainCar environment (see Figure B.1) by adding a wall on top of the right mountain to make the environment non-episodic.

In GridWorld, the agent can navigate a map composed of four rooms connected by four hallways, as represented in Figure B.1. At each step the agent can choose how much to move on the x and y axes. The maximum continuous absolute change in position along any of the axes is 0.2. Each room is a space of 5 by 5 units, thus, the agent needs around 50 steps to move from one side of the room to the other along a straight line. Any action that leads the agent to collide with a wall is ignored and the agent remains in the previous state (position).

B.1.2 Policies

In all the reported experiments, the policy is encoded through a Gaussian distribution with diagonal covariance matrix. It takes as input the environment state features and outputs an action vector $a \sim \mathcal{N}(\mu, \sigma^2)$. The mean μ is state-dependent and it is the downstream output of a densely connected neural network. The standard deviation is state-independent and it is represented by a separated

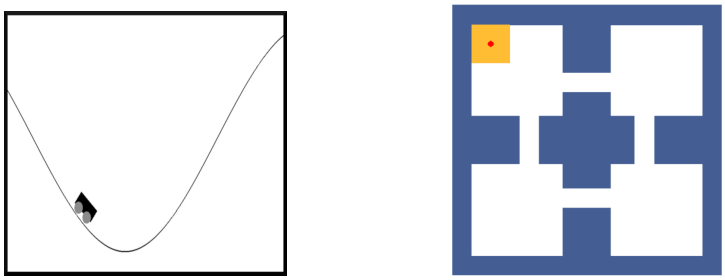


Figure B.1: *Visual representation of the MountainCar (left) and GridWorld (right) environments. In Gridworld, the agent is represented with the red circle, and it starts each episode in a random position inside the yellow area.*

trainable vector. The dimension of μ, σ and a vectors is equal to the action-space dimensions of the specific environment.

B.1.3 Unsupervised Pre-Training

Continuous Control Set-Up

Here we further comment the experimental set-up for continuous control domains that we have briefly described in Section 8.5.1, especially concerning how we select the set of features on which the entropy index is maximized. First, in the Ant domain, we maximize the entropy over a 7D space of spatial coordinates (3D) and torso orientation (4D), excluding joint angles and velocities. This is to obtain an intermediate hurdle, in terms of dimensionality, w.r.t. the smaller GridWorld and MountainCar, and the most complex Humanoid and HandReach. Instead, in the latter two domains, we essentially maximize the entropy over the full state space excluding velocities and external forces, so that we have a 24D space both in Humanoid (3D position, 4D body orientation, and all the joint angles) and HandReach (all the joint angles). We noted that including external forces does not affect the entropy maximization in a meaningful way, as they resulted always zero during training. The reason why we also discarded velocities from the entropy computation is twofold. First, we noted that velocity-related features are quite unstable and spiky, so that they can be harmful without normalization, which we avoided. Secondly, we think that maximizing the entropy over the largest set of features is not necessarily a good idea when targeting generalization (see B.1.4), especially if improving the entropy over some features reduces the entropy over some other (as in the case of positions/angles and velocities).

MEPOL

As outlined by the MEPOL pseudocode in Algorithm 7, in each epoch a dataset of particles $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$ is gathered for the given time horizon T. We call N_{traj} the number of trajectories, or batch size, used to build the dataset, so that $N = N_{traj} \cdot T$. Before starting the main loop of the algorithm we perform some training steps to force the policy to output a zero μ vector. This is instrumental to obtain a common starting point across all the seeds and can be removed without affecting the algorithm behavior.

For what concerns the k-nearest neighbors computation we use the *neighbors* package from the *scikit-learn* library (Pedregosa et al., 2011), which provides efficient algorithms and data structures to first compute and then query nearest neighbors (KD-tree in our case). Note that the computational complexity of each epoch is due to the cost of computing the k-NN entropy estimation, which is $\mathcal{O}(pN \log N)$ to build the tree (p is the number of dimensions), and $\mathcal{O}(kN \log N)$ to search the neighbors for every sample. Summing it up we get a complexity in the order of $\mathcal{O}(N \log N(p + k))$ for each epoch. In the table below, we report all the relevant MEPOL parameters used in the experiments.

Table B.1: *MEPOL Parameters*

	MountainCar	GridWorld	Ant	Humanoid	HandReach
Number of epochs	650	200	2000	2000	2000
Horizon (T)	400	1200	500	500	50
Batch Size (N_{traj})	20	20	20	20	50
Kl threshold (δ)	15.0	15.0	15.0	15.0	15.0
Learning rate (α)	10^{-4}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Max iters	30	30	30	30	30
Number of neighbors (k)	4	50	4	4	4
Policy hidden layer sizes	(300,300)	(300,300)	(400,300)	(400,300)	(400,300)
Policy hidden layer act. function	ReLU	ReLU	ReLU	ReLU	ReLU
Number of seeds	8	8	8	8	8

MaxEnt

As already outlined in Section 8.5.1, we use the original MaxEnt implementation to deal with continuous domains.[1] We adopt this implementation also for the Ant and MountainCar environments, which were originally presented only as discretized domains. [2] [3] This allowed us not only to work in a truly continuous setting, as we do in MEPOL, but also to obtain better results than the discretized version. The only significant change we made is employing TRPO instead of SAC for the RL component. This is because, having tested MaxEnt with both the configurations, we were able to get slightly superior performance, and a more stable behavior, with TRPO.

In the tables below, we report all the relevant MaxEnt parameters used in the experiments, as well as the corresponding ranges (or sets) of values over which we searched for their optimal values. The TRPO parameters are reported employing the same notation as in (Duan et al., 2016). In the Ant and Humanoid experiments, we use a higher time horizon (T_d) than the optimized one (T) to estimate the state density induced by the mixture of policies. The reason why we do this is that, otherwise, we were not able to obtain reliable density estimations. The batch size used for the density estimation is denoted as N_{traj_d}. We also report the number of neighbors (k) that we used to calculate the entropy of the mixture in the plots of Section 8.5.1.[4] The entropy is computed over the same time horizon T and the same batch size N_{traj} used in MEPOL. Note that the horizon reported for TRPO is the same as the objective horizon T. The neural policy architectures are not reported as they are the same as in Table B.1.

[1] `https://github.com/abbyvansoest/maxent/tree/master/humanoid`
[2] `https://github.com/abbyvansoest/maxent_base`
[3] `https://github.com/abbyvansoest/maxent/tree/master/ant`
[4] Note that this value does not affect the learning process of MaxEnt, as it is only sued for evaluation.

Table B.2: *MaxEnt Parameters - MountainCar*

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	60	-
Density Horizon (T_d)	400	-
Density Batch Size (N_{traj_d})	100	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	0.1	[0.1, 2.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	50	40, 50, 60
Horizon	400	-
Sim. steps per Iter.	4000	-
δ_{KL}	0.1	0.1, 0.01, 0.001
Discount (γ)	0.99	-
Number of seeds	8	-

Table B.3: *MaxEnt Parameters - Ant*

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	10000	-
Density Batch Size (N_{traj_d})	10	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	0.2	[0.1, 2.0]
PCA	Yes (3 components)	Yes, No
	TRPO	
Num. Iter.	300	300, 500
Horizon	500	-
Sim. steps per Iter.	5000	-
δ_{KL}	0.1	0.1, 0.008
Discount (γ)	0.99	-
Number of seeds	8	-

Table B.4: *MaxEnt Parameters - Humanoid*

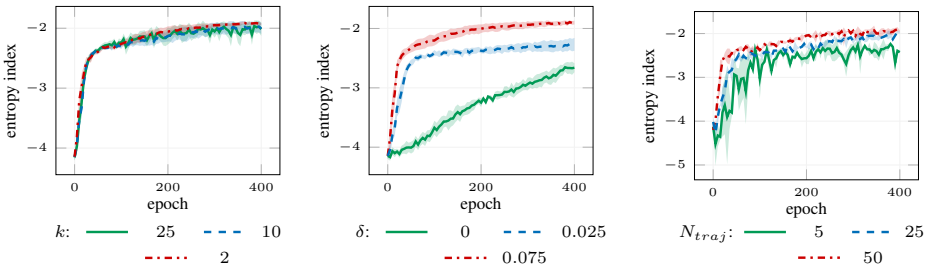
	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	50000	-
Density Batch Size (N_{traj_d})	20	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	1.0	[0.1, 2.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	300	200, 300
Horizon	500	-
Sim. steps per Iter.	5000	-
δ_{KL}	0.1	0.1, 0.008
Discount (γ)	0.99	-
Number of seeds	8	-

Table B.5: *MaxEnt Parameters - HandReach*

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	10000	-
Density Batch Size (N_{traj_d})	20	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	1.0	[0.1, 5.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	300	200, 300
Horizon	50	-
Sim. steps per Iter.	500	-
δ_{KL}	0.1	-
Discount (γ)	0.99	-
Number of seeds	8	-

Parameters Sensitivity

In Figure B.2, we show how the selection of the main parameters of MEPOL impacts on the learning process. To this end, we consider a set of experiments in the illustrative MountainCar domain, where we vary one parameter at a time to inspect the change in the obtained entropy. As we can notice, the algorithm shows little sensitivity to the number of neighbors (k) considered in the entropy estimation. Allowing off-policy updates through an higher KL threshold δ positively impacts the learning efficiency. Furthermore, MEPOL displays a good behavior even when we limit the batch-size.



(a) *k-NN* **(b)** *KL threshold* **(c)** *Batch-size*

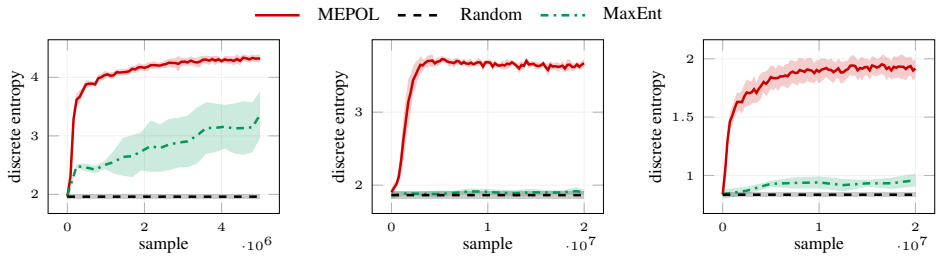
Figure B.2: *Comparison of the entropy index as a function of the learning epochs for MEPOL with different set of parameters on the MountainCar domain. (95% c.i. over 8 runs, $T = 400$ (a,b,c), $k = 4$ (b,c), $\delta = 0.05$ (a,c), $N_{traj} = 100$ (a,b)).*

Discrete Entropy

In Figure B.3, we report the plots for the evaluation of the entropy on the 2D-discretized state-space from which we have taken the values reported in Figure 8.3a. In Ant and Humanoid experiments, the considered state-space is the 2D, discretized, agent's coordinates (x, y). These plots were created while running the experiments in Section 8.5.1.

State-Visitation Heatmaps

In Figure B.4, and Figure B.5, we report the MEPOL state-coverage heatmaps obtained in the GridWorld and MountainCar experiments. In Figure B.6, you can see the state-coverage heatmap in the Ant domain over a 12 by 12 units space, which is centered in the Ant starting position. The thick borders in the heatmap are due to trajectories going out of bounds. These heatmaps were created while running the experiments presented in Section 8.5.1 by discretizing the continuous state-space.



(a) *MountainCar* **(b)** *Ant* **(c)** *Humanoid*

Figure B.3: *Comparison of the entropy computed on the 2D-discretized state-space as a function of training samples achieved by MEPOL, MaxEnt, and a random policy in the MountainCar, Ant and Humanoid domains in the setting presented in Section 8.5.1 (95% c.i. over 8 runs).*

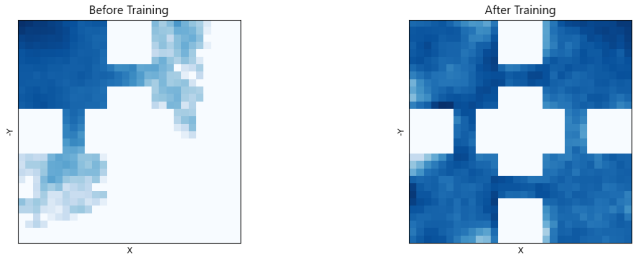


Figure B.4: *MEPOL log-probability state visitation heatmaps in the GridWorld domain created by running the policy for* $N_{traj} = 100$, $T = 1200$.

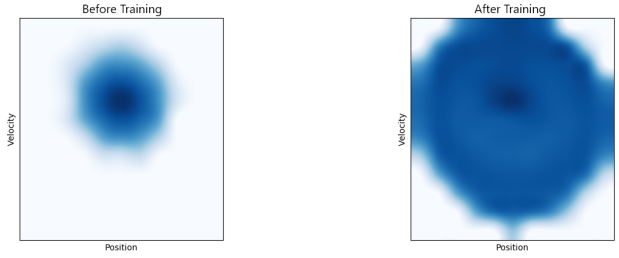


Figure B.5: *MEPOL log-probability state visitation heatmaps in the MountainCar domain created by running the policy for* $N_{traj} = 100$, $T = 400$.

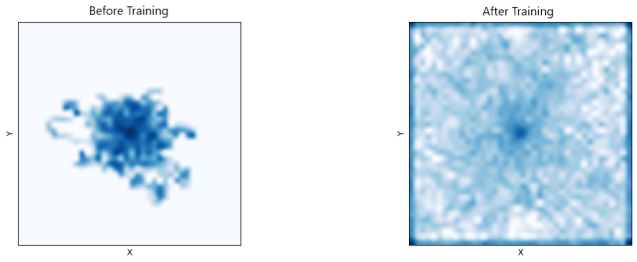


Figure B.6: *MEPOL log-probability (x, y) state visitation heatmaps in the Ant domain created by running the policy for* $N_{traj} = 100$, $T = 500$.

B.1.4 Supervised Fine-Tuning

Algorithms

We use the TRPO implementation from OpenAI's SpinningUp library.[5] For SAC, we adopt an open-source codebase from (Ciosek et al., 2019).[6] For SMM, we use the original codebase, which provides also an implementation of ICM, and Pseudocount.[7]

Parameters Detail

In Table B.6, we report the TRPO parameters used in the experiments, following the same notation as in (Duan et al., 2016). Both the basic agent and the MEPOL agent use the same parameters. In Table B.7, we report the SAC parameters following the notation as in (Haarnoja et al., 2018). For the SMM baseline, we use 4 skills. In SMM, ICM, and Pseudocount, we equally weight the extrinsic and the intrinsic components, as we have not seen any improvement doing otherwise. Note that SMM, ICM, and Pseudocount are built on top of SAC for which we adopt the same parameters as in Table B.7. The neural policy architectures are not reported as they are the same as in Table B.1.

Table B.6: *TRPO Parameters for Goal-Based Reinforcement Learning*

	GridWorld	AntEscape	AntJump	AntNavigate	HumanoidUp
Num. Iter.	100	500	1000	1000	2000
Horizon	1200	500	500	500	2000
Sim. steps per Iter.	12000	5000	50000	50000	20000
δ_{KL}	10^{-4}	10^{-2}	10^{-2}	10^{-2}	10^{-2}
Discount (γ)	0.99	0.99	0.99	0.99	0.99
Number of seeds	8	8	8	8	8

Table B.7: *SAC Parameters for Goal-Based Reinforcement Learning*

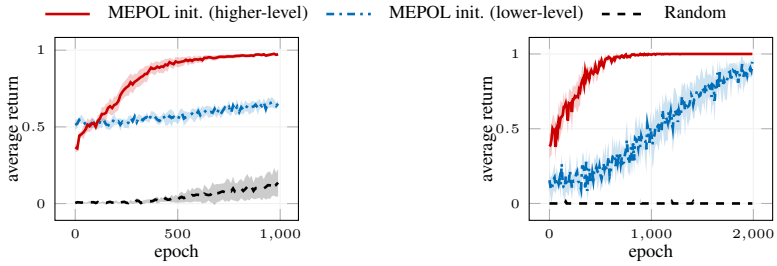
	GridWorld	AntEscape	AntJump	AntNavigate	HumanoidUp
Epoch	100	500	1000	1000	2000
Num. Updates	12000	5000	5000	5000	6000
Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Discount (γ)	0.99	0.99	0.99	0.99	0.99
Replay buffer size (γ)	10^6	10^6	10^6	10^6	10^6
Number of samples per mini batch	256	256	256	256	256
Number of seeds	8	8	8	8	8

Higher-Level and Lower-Level Policies

In this section, we discuss the performance achieved by MEPOL policies when facing higher-level tasks, such as 2D navigation (Figure B.7a) and standing-up (Figure B.7b). Especially, we can see that higher-level MEPOL policies, which are trained to maximize the entropy over spatial coordinates

[5] https://github.com/openai/spinningup
[6] https://github.com/microsoft/oac-explore
[7] https://github.com/RLAgent/state-marginal-matching

(x-y in Ant, x-y-z in Humanoid), outperform lower-level MEPOL policies, which are trained to maximize entropy over spatial coordinates, orientation, and joint angles (as reported in Section 8.5.1). This is not surprising, since higher-level policies better match the level of abstraction of the considered tasks. However, it is worth noting that also lower-level policies achieve a remarkable initial performance, and a positive learning trend, albeit experiencing slower convergence.



(a) *Ant Navigate* **(b)** *Humanoid Up*

Figure B.7: *Comparison of the average return as a function of learning epochs achieved by TRPO with MEPOL initialization (higher-level, lower-level) and a Random initialization (95% c.i. over 8 runs).*

B.2 Experimental Details of Chapter 9

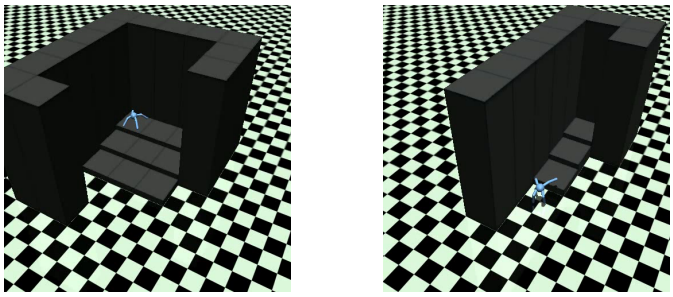
In this section, we report an extensive description of the conducted experiments, including details about the considered environments, policies, algorithms, and values of the hyper-parameters.

B.2.1 Environments

We use three different environments in our experiments. The first one is a custom implementation of a gridworld, coded from scratch. The second one is an adapted version of the rllab Ant-Maze environment (Duan et al., 2016).

GridWorld with Slope

In *GridWorld with Slope* (2D states, 2D actions), the agent can move inside a map composed of four rooms connected by four narrow hallways, by choosing at each step how much to move on the x and y axes. The side of the environment measures 2 units and the maximum viable space of the agent at each step is 0.2. Thus, the agent needs around 10 steps to go from one side to the other on a straight line. When the agent collides with the external borders or with the internal walls, it is re-positioned according to a custom function. This is done not only to make the interaction more realistic, but also to limit the possibility to have a negative infinite entropy resulting from the k-NN computation, which can occur when the samples are too close and the value of the parameter k is not high enough. This precaution is particularly useful in our scenario, due to the presence of a slope, and especially in the *adversarial* configuration GWN, because of the initial position of the agent, which is sampled in a small square in the top-right corner. It is easy to see that in the first epochs in the GWN environment, the agent would repeatedly collide with the top-border, leading in general to a much more lower entropy w.r.t. to GWS.



(a) *Ant Stairs Down* **(b)** *Ant Stairs Up*

Figure B.8: *Illustration of the* Ant Stairs *domain. We show a render of the* Ant Stairs Down *environment* ***(a)*** *and of the adverse* Ant Stairs Up *environment* ***(b)***.

The slope is applied only in the upper half of the environment, since we found this to be a good trade-off between the intention of maintaining a difference in terms of risk among the two configurations and the overall complexity of the exploration. Indeed, we noted that by applying the slope to the whole GridWorld, the advantage in terms of exploration entailed by the risk-averse approach is even higher, but it struggles to explore the bottom states of the environment with a reasonable number of samples. The slope is computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{2}, \frac{\Delta_{max}}{20})$, where $\Delta_{max} = 0.2$ is the maximum step that the agent can perform.

MultiGrid

In *MultiGrid*, everything works as in *GridWorld with Slope*, but we indeed have 10 configurations. These environments differ for both the shape and the type of slope to which they are subject to. The *adversarial* configuration is still GWN, but the slope is computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{2.6}, \frac{\Delta_{max}}{20})$, where $\Delta_{max} = 0.2$. The other 9 gridworlds have instead a different arrangement of the walls (see the heatmaps in Figure B.12) and the slope, computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{3.2}, \frac{\Delta_{max}}{20})$ with $\Delta_{max} = 0.2$, is applied over the entire environment. Two configurations are subject to south-facing slope, three to east-facing slope, one to south-east-facing slope and three to no slope at all.

AntStairs

We adopt the Ant-Maze environment (29D states, 8D actions) of rllab (Duan et al., 2016) and we exploit its malleability to build two custom configurations which could fit our purposes. The adverse configuration consists of a narrow ascending staircase (*Ant Stairs Up*) made up of an initial square (the initial position of the Ant), followed by three blocks of increasing height. The simpler configuration consists of a wide descending staircase (*Ant Stairs Down*), made up of 3×3 blocks of decreasing height and a final 1×3 flat area. Each block has a side length slightly greater than the Ant size. A visual representation of such settings is provided in Figure B.8. During the *Unsupervised Pre-Training* phase, $\mathcal{E}^{\alpha}_{\mathcal{M}}$ is maximized over the x,y spatial coordinates of the ant's torso.

MiniGrid

We use the MiniGrid suite (Chevalier-Boisvert et al., 2018), which consists of a set of fast and light-weighted gridworld environments. The environments are partially observable, with the dimension

of the agent's field of view having size $7 \times 7 \times 3$. Both the observation space $\mathcal{S}$ and the action space $\mathcal{A}$ are discrete, and in each tile of the environment there can be only one object at the same time. The set of objects is $O = \{wall, floor, lava, door, key, ball, box, goal\}$. The agent can move inside the grid and interact with these objects according to their properties. In particular, the actions comprise turning left, turning right, moving forward, picking up an object, dropping an object and toggling, i.e., interacting with the objects (e.g., to open a door). We exploit the suite's malleability to build two custom environments. The simpler one has a size of 18×18, and it simply contains some sparse walls. The adverse configuration is smaller, 10×10, and is characterized by the presence of a door at the top of a narrow hallway. The door is closed but not locked, meaning that the agent can open it without using a key. Moreover, we modify the movement of the agent so that the direction is given by the bottom of the triangle instead of the top. The intuition is that by doing this we are essentially changing the shape of the agent, causing an additional hurdle for the exploration.

As regards the training procedure, everything remains the same, except for two differences. The first difference is that the k-NN computation is performed on the representation space generated by a fixed random encoder. Note that this random encoder is not part of the policy. It is randomly initialized and not updated during the training in order to produce a more stable entropy estimate. In addition, before computing the distances, we apply to its output a random Gaussian noise $\epsilon \sim \mathcal{N}(0.001, 0.001)$ truncated in $[0, 0.001]$. We do this to avoid the aliasing problem, which occurs when we have many samples (more than k) in the same position, thus having zero distance and producing a negative infinite entropy estimate. The homogeneity of the MiniGrid environments in terms of features make this problem more frequent. The second difference is the addition of a bootstrapping procedure for the easy configuration, meaning that we use only a subset of the mini-batches of the easy configuration to update the policy. Especially, we randomly sample a number of mini-batches that is equal to the dimension of the $\mathcal{D}_\alpha$ dataset so that MEPOL uses the same number of samples of αMEPOL. The reason why we avail this method is to avoid a clear advantage for MEPOL in learning effective representations, since it usually access more samples than αMEPOL. Note that it is not a stretch, since we are essentially balancing the information available to the two algorithms.

B.2.2 Policies

In all the experiments but one the policy is a Gaussian distribution with diagonal covariance matrix. It takes as input the environment state features and outputs an action vector $a \sim \mathcal{N}(\mu, \sigma^2)$. The mean μ is state-dependent and is the downstream output of a densely connected neural network. The standard deviation is state-independent and it is represented by a separated trainable vector. The dimension of μ, σ, and a vectors is equal to the action-space dimension of the environment. The only experiment with a different policy is the MiniGrid one, for which we adopt the architecture recently proposed by (Seo et al., 2021). Thus, we use a random encoder made up of 3 convolutional layers with kernel 2, stride 1, and padding 0, each activated by a ReLU function, and with 16, 32 and 64 filters respectively. The first ReLU is followed by a 2D max pooling layer with kernel 2. The output of the encoder is a 64 dimensional tensor, which is then fed to a feed-forward neural network with two fully-connected layers with hidden dimension 64 and a Tanh activation function.

B.2.3 Algorithms

In this section, we provide an extended pseudocode (Algorithm 9) of the αMEPOL algorithm we presented in Section 9.6, along with some additional comments.

Given a probability distribution $p_{\boldsymbol{\mathcal{M}}}$, the algorithm operates by iteratively sampling an environment $\mathcal{M}_i \in \boldsymbol{\mathcal{M}}$ drawn according to $p_{\boldsymbol{\mathcal{M}}}$ and then sampling B trajectories of length T from it using

Algorithm 9 αMEPOL

Input: initial policy $\pi_{\boldsymbol{\theta}_0}$, exploration horizon T, number of trajectories N, batch-size B, percentile α, learning rate β, trust-region threshold δ, sampling distribution $p_{\boldsymbol{\mathcal{M}}}$
Output: exploration policy $\pi_{\boldsymbol{\theta}_h}$

for epoch $= 0, 1, \ldots,$ until convergence **do**
  for $i = 1, 2, \ldots, N$ **do**
    sample an environment $\mathcal{M}_i \sim p_{\boldsymbol{\mathcal{M}}}$
    for $j = 1, 2, \ldots, B$ **do**
      sample a trajectory $\tau_j \sim p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}_i}$ of length T
    end for
  end for
  initialize dataset $\mathcal{D} = \emptyset$, off-policy step $h = 0$ and $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
  while $\widehat{D}_{KL}(\pi_{\boldsymbol{\theta}_0} || \pi_{\boldsymbol{\theta}_h}) \leq \delta$ **do**
    for $j = 1, 2, \ldots, B$ **do**
      estimate H_{τ_j} with (8.2)
      append $\widehat{H}_{\tau_j}$ to $\mathcal{D}$
    end for
    sort $\mathcal{D}$ and split it in $\mathcal{D}_\alpha$ and $\mathcal{D}_{1-\alpha}$
    compute a gradient step $\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \beta \widehat{\nabla}_{\boldsymbol{\theta}_h} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}_h})$
    $h \leftarrow h + 1$
  end while
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_h$
end for

$\pi_{\boldsymbol{\theta}}$, where B is the dimension of each mini-batch. Then, the estimate of the entropy of each mini-batch $\widehat{H}_{\tau_j}$ is computed by means of the estimator in (8.2) and appended to the dataset $\mathcal{D}$. Once the dataset $\mathcal{D}$ is obtained, we can straightforwardly derive a risk-sensitive policy update by just subsampling from it, so that to keep only the realizations below the α-percentile. This can be easily done by sorting $\mathcal{D}$ in ascending order and considering only the αN first mini-batches. Then, we can compute the gradient as follows:

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \frac{1}{\alpha N} \sum_{i=1}^{N} f_{\tau_i} \, \widehat{H}_{\tau_i} \, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{\mathrm{VaR}}_{\alpha}(H_\tau)).$$

The operations carried out once all the trajectories have been sampled are executed in a fully off-policy manner, in which we repeat the same steps until the trust-region boundary is reached or until the number of off-policy iterations exceeds a specified limit. The reason why we introduce an additional parameter B, instead of considering one trajectory at a time, is due to the fact that a significant amount of samples (see the parameters in Table B.8) is needed to obtain a reliable estimate of the entropy, noting that the entropy estimator is only asymptotically unbiased.

B.2.4 Hyperparameter Values

To choose the values of the hyper-parameters of αMEPOL, MEPOL, and TRPO we mostly relied on the values reported in B.1, which have been optimized for a the risk-neutral approach. By avoiding to specifically fine-tune the hyper-parameters for the αMEPOL algorithm (and TRPO with αMEPOL initialization), we obtain conservative comparisons between αMEPOL and the MEPOL baseline.

Unsupervised Pre-Training

In Table B.8, we report the parameters of αMEPOL and MEPOL that are used in the experiments described in Section 9.7.1, Section 9.7.2, Section 9.7.4 and Section 9.7.5.

Table B.8: *αMEPOL and MEPOL Parameters for the Unsupervised Pre-Training*

	GRIDWORLD WITH SLOPE	MULTIGRID	ANT	MINIGRID
NUMBER OF EPOCHS	150	50	400	300
HORIZON (T)	400	400	400	150
NUMBER OF TRAJ. (N)	200	500	150	100
MINI-BATCH DIMENSION (B)	5	5	5	5
α-PERCENTILE	0.2	0.1	0.2	0.2
SAMPLING DIST. ($p_{\boldsymbol{\mathcal{M}}}$)	[0.8,0.2]	[0.1,...,0.1]	[0.8,0.2]	[0.8,0.2]
KL THRESHOLD (δ)	15	15	15	15
LEARNING RATE (β)	10^{-5}	10^{-5}	10^{-5}	10^{-5}
NUMBER OF NEIGHBORS (k)	30	30	500	50
POLICY HIDDEN LAYER SIZES	(300,300)	(300,300)	(400,300)	*
POLICY HIDDEN LAYER ACT. FUNCT.	RELU	RELU	RELU	*
NUMBER OF SEEDS	10	10	10	10

* See Section B.2.2 for full details on the architecture.

Supervised Fine-Tuning

In Table B.9, we report the TRPO parameters that are used in the experiments described in Section 9.7.3, Section 9.7.4, Section 9.7.5 and Section 9.7.7.

Table B.9: *TRPO Parameters for the Supervised Fine-Tuning*

	GRIDWORLD WITH SLOPE	MULTIGRID	ANT	MINIGRID
NUMBER OF ITER.	100	100	100	200
HORIZON	400	400	400	150
SIM. STEPS PER ITER.	1.2×10^4	1.2×10^4	4×10^4	7.5×10^3
δ_{KL}	10^{-4}	10^{-4}	10^{-2}	10^{-4}
DISCOUNT (γ)	0.99	0.99	0.99	0.99
NUMBER OF SEEDS	50	50	8	13
NUMBER OF GOALS	50	50	8	13

Meta-RL

In Table B.10 and Table B.11, we report the MAML and DIAYN parameters that are used in the experiments described in Section 9.7.7, in order to meta-train a policy on the *GridWorld with Slope* and *MultiGrid* domains. For MAML, we adopted an open-source codebase,[8] while for DIAYN we used the original implementation.

[8] `https://github.com/tristandeleu/pytorch-maml-rl`

Table B.10: *MAML Parameters for the Meta-Training*

	GridWorld with Slope	MultiGrid
Number of batches	200	200
Meta batch size	20	20
Fast batch size	30	30
Num. of Grad. Step	1	1
Horizon	400	400
Fast learning rate	0.1	0.1
Policy hidden layer sizes	(300,300)	(300,300)
Policy hidden layer act. function	ReLU	ReLU
Number of seeds	8	8

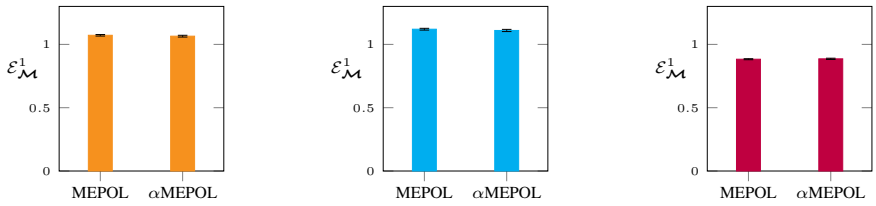
Table B.11: *DIAYN Parameters*

	GridWorld with Slope	MultiGrid
Number of epochs	1000	1000
Horizon	400	400
Number of skills	20	20
Learning rate	3×10^{-4}	3×10^{-4}
Discount (γ)	0.99	0.99
Policy hidden layer sizes	(300,300)	(300,300)
Policy hidden layer act. function	ReLU	ReLU
Number of seeds	8	8

B.2.5 Counterexamples

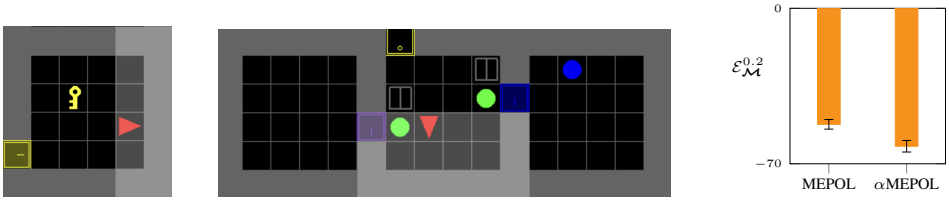
In this section, we provide a couple of convenient example to confirm the fact that there are classes of environments in which we would not need any particularly smart solution for the multiple environments problem, beyond a naïve, risk-neutral approach. First, we consider two GridWorld environments that differ for the shape of the traversable area, sampled according to $p_{\boldsymbol{\mathcal{M}}} = [0.8, 0.2]$, and we run αMEPOL with $\alpha = 0.35$ and MEPOL, obtaining the two corresponding exploration policies. In Figure B.9 we show the performance (measured by $\mathcal{E}^1_{\boldsymbol{\mathcal{M}}}$) obtained by executing those policies on each setting. Clearly, regardless of what configuration we consider, there is no advantage deriving from the use of a risk-averse approach as αMEPOL, meaning that the class of environments $\boldsymbol{\mathcal{M}}$ is balanced in terms of hardness of exploration.

In the second counterexample, we consider different configurations of the MiniGrid (Chevalier-Boisvert et al., 2018) domain that we considered in Section 9.7.6. Especially, we aim to show that configurations that are visibly different from an human perspective are sometimes not really challenging from a multiple environments standpoint. Indeed, this setting would be challenging only if the policy that the agent should deploy to explore one configuration is significantly different to the one needed to explore another configuration. In this case, the agent should trade-off the performance in one configuration and the other. As we show in Figure B.10, the combination of Unlock-v0 and ObstructedMaze-2Dlhb-v0 does not have this feature, and the MEPOL baseline is able to find a policy that works well in both the configurations.



(a) $p_{\mathcal{M}} = [0.8, 0.2]$ **(b)** $p_{\mathcal{M}} = [1, 0]$ **(c)** $p_{\mathcal{M}} = [0, 1]$

Figure B.9: *Comparison of the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ obtained by αMEPOL ($\alpha = 0.35$) and MEPOL in the* GridWorld Counterexample *domain. The polices are trained (50 epochs, 8×10^4 samples per epoch) on the configuration **(a)** and tested on **(a, b, c)**. We provide 95% c.i. over 4 runs.*



(a) *Unlock-v0* **(b)** *ObstructedMaze-2Dlhb-v0* **(c)** $p_{\mathcal{M}} = [0.8, 0.2]$

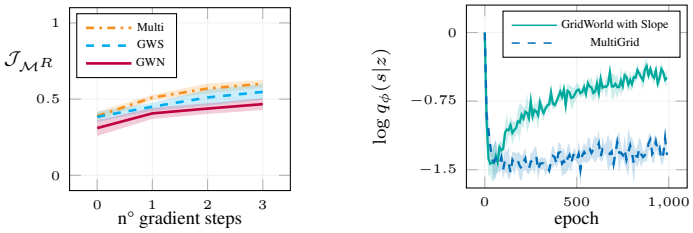
Figure B.10: *Comparison of the exploration performance $\mathcal{E}^{0.2}_{\mathcal{M}}$ obtained by αMEPOL ($\alpha = 0.2$) and MEPOL **(c)** in a* MiniGrid *domain with the configurations* Unlock-v0 ***(a)*** *and* ObstructedMaze-2Dlhb-v0 ***(b)***. *We provide 95% c.i. over 8 runs.*

B.2.6 Meta-RL

In this section, we provide additional details on the experiments of Section 9.7.7. Especially, we show that MAML does perform well on its own objective, which is to learn a fast-adapting policy during meta-training (Figure B.11a). Instead, in Figure B.11b we highlight the performance measure of DIAYN (Eysenbach et al., 2019). In particular, the more $\log q_\phi(s|z)$ grows with the learning epochs, the better is the intrinsic reward we feed to MAML+DIAYN. Clearly, DIAYN struggles to deal with the larger *MultiGrid* class of environments, which explains the inferior performance of MAML+DIAYN in this domain.

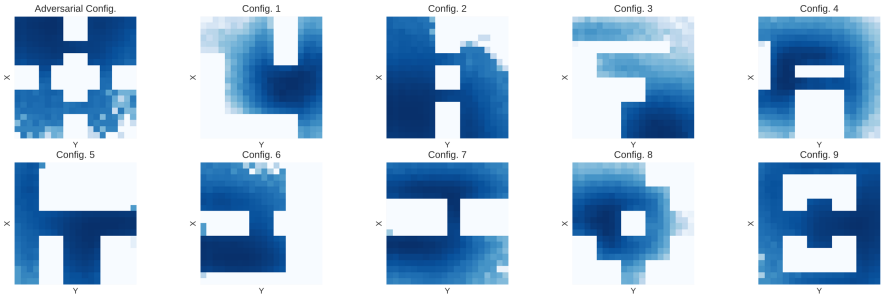
B.2.7 Additional Visualizations

In this section, we provide some additional visualizations, which are useful to better understand some of the domains used in the experiments of Section 9.7. In Figure B.12 we report the state-visitation frequencies achieved by αMEPOL (Figure B.12a) and MEPOL (Figure B.12b) in each configuration of the *MultiGrid* domain. Clearly, αMEPOL manages to obtain a better exploration in the adversarial configuration w.r.t. MEPOL, especially in the bottom part of the environment, which is indeed the most difficult part to visit. On the other environments, the performance is overall comparable. In Figure B.8 we show a render of the *Ant Stairs* domain, illustrating both the environments used in the experiments of Section 9.7.5. Note that the front walls are hidden to allow for a better visualization.

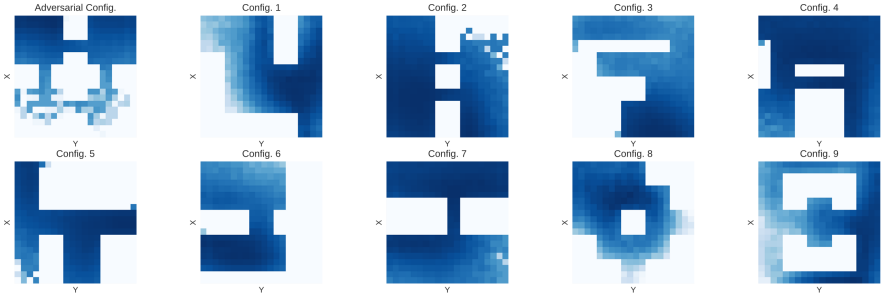


(a) *MAML: Fast Adaptation* **(b)** *DIAYN: Discriminability*

Figure B.11: *We illustrate the fast-adapting behavior of MAML in the* GridWorld with Slope ***(a)****, and the skills discriminability of DIAYN as a function of learning epochs* ***(b)****. We provide 95% c.i. over 8 runs.*



(a) α*MEPOL*



(b) *MEPOL*

Figure B.12: *Heatmaps of the state visitations (200 trajectories) induced by the exploration policies trained with* α*MEPOL* ($\alpha = 0.1$) ***(a)*** *and MEPOL* ***(b)*** *in the* MultiGrid *domain.*