Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AI	Artificial Intelligence.
BH	Balance Heuristic Estimator.
GP	Gaussian Process.
GPOMDP	Gradient of the average reward in Partially Observable MDPs.
GPUCB	Gaussian Process Upper Confidence Bound.
HOO	Hierarchical Optimistic Optimization.
KL	Kullback-Leibler.
LQG	Linear Quadratic Gaussian regulation.
MAB	Multi Armed Bandit.
MDP	Markov Decision Process.
MIS	Multiple Importance Sampling.
OFU	Optimism in the Face of Uncertainty.
OPTIMIST	Optimistic Policy opTImization via Multiple Importance Sampling with Truncation.
PBPOIS	Parameter-Based Policy Optimization via Importance Sampling.
PG	Policy Gradient.
PGPE	Policy Gradient with Parameter-Based Exploration.

PGT	Policy Gradient Theorem estimator.
PS	Policy Search.
RL	Reinforcement Learning.
std	standard deviation.
TS	Thompson Sampling.
UCB	Upper Confidence Bound.
VIME	Variational Information Maximizing Exploration.

Chapter 1

Introduction

With the advent of the 21st century, worldwide economies have been profoundly shaped by an ever increasing access to large amounts of data and faster computers. This two elements entailed a new dawn of Artificial Intelligence (AI), which represents today a dominant field in the technological and scientific landscape, with inevitable repercussions on whole society. At the heart of this new dawn is Machine Learning (ML) [36], a branch of AI that aims at building computers capable of learning from experience, i.e., the data. Since computers gather knowledge from experience, there is no need for a human computer operator to formally specify all the knowledge that the computer needs to solve complex, intelligent tasks.

Many real-world problems require the solver (the agent) to make a sequence of decisions without knowing the dynamics of the environment in which it is operating and, possibly, without receiving an immediate feedback on the goodness of its actions. This type of problems arises in many fields such as automation control, robotics, finance, operations management and games. Reinforcement Learning (RL) is a branch of ML that deals with such problems, and which has seen important advances in the in the last few decades [86]. In RL, an autonomous agent learns by repeated interaction with an unknown environment, which is partially observed by the agent. The unique information available to the agent is its internal representation of the environment, and a feedback (a numerical reward) that serves as an evaluation of its actions. Hence, the agent learns by trial and error, as humans would do when learning a new skill without the presence of a teacher. We refer to the strategy adopted by the agent when interacting with the environment as its *policy*: a, possibly stochastic, mapping that instructs the agent about what action to perform in each state. The goal is to learn a suitable *policy* to maximize the cumulative sum of rewards collected until the end of the task. Since the reward received by the agent can be infrequent or even very sparse, the agent needs sufficient exploration in order to improve the knowledge of the unknown environment. Hence, the agent has to *exploit* the most rewarding actions learned, but also has to *explore* in order to make better action selection in the future. This trade-off, called *exploration-exploitation trade-off*, arises at every decision step of the learning process. Traditional exploration strategies, such as random walks, aim to generate heterogeneous experiences by selecting actions randomly. This is obviously extremely inefficient and quickly leads to unfeasible learning times as the complexity of the problem grows. More advanced exploration techniques, referred to as *directed exploration* [90], leverage on the knowledge acquired while learning to explore more efficiently.

One of the current challenges of RL is to *continuous control* tasks, such as robotic locomotion, in which states and actions are naturally modelled as real numbers. Policy Search (PS) [28] is a family of RL algorithms that are particularly suited to this class of problems because they scale gracefully with dimensionality and offer appealing convergence guarantees. In PS, the behaviour of the agent, or *policy*, is explicitly modelled, typically as a stochastic parametric function from states to actions. Learning corresponds to the optimization of a performance measure, typically an estimation of the cumulative sum of rewards, w.r.t. the policy parameters.

1.1 Motivation and goal

The available literature on PS focuses mainly on the problem of *finding* the optimal policy with the minimum amount of interaction [87, 78, 80, 76, 58, 32]. This is well motivated, as interacting with some environments can be very expensive. However, in many cases, we are also interested in the performance of the agent *during* the learning process. This goal is particularly relevant in applications where an agent must be deployed in the real world to perfect its behaviour (e.g., robot learning) or to learn at all (e.g., recommender systems). In such cases, the *exploration-exploitation* dilemma arises naturally, as the agent must continually find the right trade-off between complying with its current expertise or widening it by trial and error. In the context of PS, exploration is carried out in the space of policy parameters, and this makes exploration especially needed. In fact, the parameter space is often characterized by multiple local optima, and common optimization strategies such as gradient ascent get stuck in sub-optimal solutions. To tackle this problem, the scientific community has proposed various strategies, mainly focused on random exploration through the maximization of the entropy of the policy. These solutions do not exploit the information gathered by the agent across iterations for guiding exploration. Instead, they only push the agent to consistently explore in an experience-agnostic fashion, often without any convergence guarantees [99, 38, 39]. At present, the PS literature lacks directed strategies for exploring the parameter space.

This lack of direct exploration solutions for PS, of paramount importance for nowadays challenges such as robotics, motivates the research direction of this thesis project.

Equivalently, it must minimize its total *regret* w.r.t. the optimal behaviour. This problem has been thoroughly studied in the field of Multi Armed Bandit (MAB) [6, 53]. In this simple framework, an agent has to repeatedly select an action, called an *arm* in this context, in order to maximize an unknown, stochastic reward. This can be seen as RL without states.

Chapter 2

Preliminaries

In this chapter we provide an introduction to the Multi-Armed Bandit framework and the Reinforcement Learning framework, which are the two frameworks of reference of this thesis project. The Reinforcement Learning section is opportunely preceded by an overview of Markov Decision Processes, which are the building blocks of Reinforcement Learning. Then, we present a particular class of strategies to solve the Reinforcement Learning problem: Policy Search methods. The aim is to introduce concepts that will be used extensively in the following chapters.

2.1 Multi Armed Bandits

Consider a situation in which a gambler (the *agent*) is sitting in a casino, staring at a number of slot machines from which he wants to pick the most profitable one and play with it all night long. Each slot machine is characterized by a certain *mean reward.* The objective of the gambler is to maximize the gains that he will have won by the end of the night, but, unfortunately, he can not know *a priori* the mean reward associated to each slot machine. Hence, the gambler must find a suitable strategy in order to effectively *explore* his possibilities, by testing the different slot machines, and, then, *exploit* the ones that prove to be more profitable. In fact, the gambler would face an equivalent problem if he was to play a single machine with as many *arms* (levers) as the number of slot machines present in the casino, each with its own specific mean reward. Hence the name MAB [51] given to this problem, in homage to the one-armed bandit, an old-fashioned name for a lever operated slot machine ("bandit" because it steals your money). In the artificial intelligence literature, MAB refers to a broad class of problems that can be formulated as this gambling learning problem. Of course, MABs have other, more valuable, applications other than gambling. Current applications span from web interfaces configuration, news recommendation, dynamic pricing, ad placement, networking routing and game design.

Definition 2.1.1. (Multi Armed Bandit) *A MAB is a tuple $\langle \mathcal{X}, \mathcal{R} \rangle$, where:*

(i) $\mathcal{X}$ is the $d_{\mathcal{X}}$-dimensional set of possible arms, or actions, that the agent can pick; $\mathcal{X}$ can be discrete or continuous, one dimensional or multi-dimensional, finite or infinite.

(ii) $\mathcal{R} : \mathcal{X} \to \Delta(\mathbb{R})$ is an unknown function of rewards such that for every $x \in \mathcal{X}$ it assigns a probability measure $\mathcal{R}(\cdot|x)$ over $\mathbb{R}$.

One can think to the set of possible arms and reward functions associated to them as the *environment* the learner interacts with. According to the specific characteristics of the environment, there exist a broad taxonomy of MABs. For our reader, it is sufficient to know that in this work we deal with *stochastic stationary bandits*. In this case, the environment is restricted to generate the reward in response to each action from a *probability distribution* $\mathcal{R}(\cdot|x)$ that is specific to that action, stationary along time and independent of the previous action choices and rewards. For example, the reward distribution could be Gaussian or Bernoulli. The expected reward of the stationary distribution associated to arm x is noted $\mu(x) = \mathop{\mathbb{E}}_{r \sim \mathcal{R}(\cdot|x)}[r]$.
The MAB game is played sequentially by a *learner* over multiple rounds $t = 1, 2, \ldots, T$, up to the *horizon* $T \in \mathbb{N}$, which depends on the problem at hand. Evidently, the agent can have memory but can not foresee the future. Thus, the current action x_t should only depend upon the sequence of previous actions and rewards, the *history* $\mathcal{I} = \{x_0, r_0, x_1, r_1, \ldots, x_{t-1}, r_{t-1}\}$. A *policy* is a mapping from histories to actions which represents the behaviour of the agent, its strategy. the first question which springs to mind is then: how can we evaluate the quality of a policy? To this aim, the literature extensively adopts a performance measure called the *regret*;

Definition 2.1.2. (Immediate Regret) *The immediate regret suffered by a learner at round t is:*

$$\Delta_t = \mu(x^*) - \mu(x_t) \tag{2.1}$$

where $x^* \in \arg\max_{x \in \mathcal{X}} \mu(x)$ is an arm with the highest expected reward i.e., an optimal arm.

Definition 2.1.3. (Regret) *The regret of a learner is the cumulated sum of the instantaneous regrets:*

$$Regret(T) = \sum_{t=0}^{T} \Delta_t \tag{2.2}$$

The quality of a certain policy is then given by the rate of growth of the regret as the horizon T grows. A good learner achieves sub-linear regret, which means that $Regret(T) = o(T)$ or, equivalently, that $\lim_{T\to\infty} Regret(T)/T = 0$. For example, some state of the art policies for different bandit settings have regrets close to $O(\sqrt{T})$ or $O(\log(n))$ [53]. The regret has two characteristics that make it a good performance metric. First, it supplies a degree of normalization because it is invariant under translation of rewards. Second, it represents the price paid by the learner for not knowing the true environment. Anyway, a crucial aspect for designing a sub-linear regret policy is to carefully balance the trade-off between exploration and exploitation that characterizes all bandits.

2.1.1 Exploration and exploitation

In order to maximize its cumulative reward, a bandit agent must prefer high-reward arms discovered in past rounds. But to discover such arms, it has to spend a certain amount of rounds trying arms that it has not selected before. In other words, the agent has to *exploit* the most profitable actions revealed in the past, but it also has to *explore* in order to make better action selections in the future. If the reward function is stochastic as in stationary stochastic bandits, even more resources should be dedicated to exploration, because one trial is not enough to have a good estimate of the expected reward of one arm. In fact, the exploration-exploitation trade-off is a well known mechanism that goes back to psychology and it affects us all in our daily lives. Take, for example, a typical lunch break on a working day. Say that your favourite restaurant is right around the corner. If you go there every day (*exploitation*), you will be confident of what you will get, but miss the chances of discovering an even better option. On the contrary, if you try new places all the time (*exploration*), you are very likely going to eat unpleasant food from time to time. The sweet spot is usually in between these two extreme options and a crucial challenge for any bandit algorithm is to properly balance between exploration and exploitation. As we will see, this dilemma arises in *reinforcement learning* too and will be at the very core of our discussion.
Now, we proceed by briefly sketching out two bandit settings that are of central

interest to this work.

2.1.2 Stochastic Bandits With Finitely Many Arms

An important declination of the bandit problem, much studied in the literature [53], is the stationary stochastic bandit problem with *finitely many arms*, i.e., stationary stochastic bandits whose action space consists of a discrete arm set $|\mathcal{X}| = K \in \mathbb{N}_+$. This setting is particularly important because of its semplicity, which makes it a perfect starting point for understanding the exploration-exploitaiton trade-off and for designing algorithms that can be extended to more complex settings afterwards. Also, many real world applications can be modeled as finitely-armed stationary stochastic bandits.
In order to deepen our understanding of this setting, and of the bandit problem in general, let's discuss the most simple policy that one can imagine: the *explore-then-commit* algorithm. Basically, this policy consists in choosing each arm a certain number of times m and subsequently exploit by playing the arm that appeared to be the best after exploration. Because there are K actions, the algorithm will explore for mK rounds before choosing a single action for the remaining rounds. The choice of the agent goes to the arm i with the highest average pay-off received up to round t:

$$\widehat{\mu}_i(t) = \frac{1}{T_i(t)} \sum_{s=1}^{t} \mathbb{1}_{\{x_s=i\}} r_s \tag{2.3}$$

where $T_i(t) = \sum_{s=1}^{t} \mathbb{1}_{\{x_s=i\}}$ is the number of times up to round t the agent picks action i. Algorithm 2.1 shows the pseudo-code of the explore-then-commit policy. Recall $\forall a, b \in \mathbb{N}^+, a \bmod b = a - b\lfloor a/b \rfloor$.
It suffices a quick glance to the algorithm to have a tangible demonstration of the importance of the trade-off between exploration and exploitation. This is confirmed by the analysis of the regret.

Theorem 2.1.1. *[53] The expected regret of the explore-then-commit policy is bounded by:*

$$\mathit{Regret}(T) \leqslant \min\left(m, \lceil \tfrac{T}{K} \rceil\right) \sum_{i=1}^{K} \Delta_i + \max\left(T - mK, 0\right) \sum_{i=1}^{K} \Delta_i \exp\left(-\frac{m\Delta_i^2}{4}\right) \tag{2.4}$$

where Δ_i is the immediate regret of arm i as defined in 2.1.2.

Algorithm 2.1 Explore-then-commit

```
1: Input: $m \in \mathbb{N}$
2: for $t = 1, \dots, T$ do
3:     Choose arm
           $$x_t = \begin{cases} i, & \text{if } (t \bmod K) + 1 = i \text{ and } t \leqslant mK \\ \arg\max \widehat{\mu}_i(mK), & \text{if } t > mK \end{cases}$$
4: end for
```

Evidently, if m is large the policy explores for too long and the first term will be eventually too large. On the other hand, if m is too small, then the probability that the algorithm exploits the wrong arm will grow and the second term will become too large. Explore-then-commit is an evident case of undirected exploration: during the exploration phase, all the arms are pulled uniformly without leveraging any information that the agent collect in the process.
In the next chapter, we will present various techniques that tackle the exploration problem in a more directed way.

2.1.3 $\mathcal{X}$-armed bandits

Another stationary stochastic bandit setting interesting to our scope is the *continuum-armed bandit* setting [1]. Here, the arms are chosen from a subset of the real numbers $\mathcal{X} \in \mathbb{R}^{d_{\mathcal{X}}}$, hence, the action space is infinite. Moreover, the mean rewards are assumed to be a continuous function of the arms. An example of such bandits are continuous *Lipschitz bandits* [55], where the expected reward is a Lipschitz function of the arm. This assumption is extremely useful because it implies that the information obtained by selecting one arm can be shared to its neighbouring arms. Thus, exploration can be made more efficient, and possibly directed, by exploiting the correlation between arms. Indeed, when the set of arms is infinite it is necessary to make some assumption on the correlation between arms. If arms can not share any information, learning become unfeasible. $\mathcal{X}$*-armed bandits* [20] are a further generalization of this setting where the set of arms, $\mathcal{X}$, is allowed to be a generic measurable space and the mean-payoff function is *weak Lipschitz* with respect to a dissimilarity function that is known to the decision maker.

By now, the reader should have noticed that a distinguishing feature of bandit problems is that the learner never needs to plan for the future. More precisely, in bandits we invariably make the assumption that the learner's choices and rewards tomorrow are not affected by its decision today. In the next chapter, we discuss a more general framework that includes this kind of long-term planning.

2.2 Markov Decision Processes

A Markov Decision Process (MDP) is a more powerful way to model the interaction between an agent, which is the learner and the decision maker, and an environment. When the agent performs an action a, the environment responds presenting a new state s to the agent and rewarding the agent with a certain scalar signal called *immediate reward.* Importantly, the environment dynamics of a MDP are *stationary*, i.e., they do not depend upon time. Moreover, the state s_{h+1} at time step $h+1$ must depend only on the previous state s_h and action a_h. This property, called *Markovian Property*, entails that the agent can forget the states and actions of the past. Still, differently from what happened in bandits, the current state and set of possible actions and subsequent rewards the agent faces, are determined by his previous choice. This is a strong mechanism that encourages the agent to take into account the consequences of its choices over time. On top of this, there is another mechanism that makes the agent even more concerned by long term consequences. In fact, the objective of the agent is to maximize over time the *cumulative reward* or *return*, which is the cumulated sum of the immediate rewards obtained after each action undertaken by the agent. Evidently, sometimes it is more profitable to sacrifice immediate reward in order to reach a higher cumulative reward in the long term. This mechanism pushes the agent to take into account the future in its current decisions.
Hence, MDPs are a powerful tool capable of modelling many challenges that we encounter in life, science and engineering. Take, for example, a hungry newborn infant in his mother's arms. Through various attempts, the infant moves around his arms, head and mouth looking for food. In this way, he changes its state until the moment he eventually receives a positive reward, the mother's milk. Little by little, the baby will learn the precise sequence of actions and states that rewards him with milk. Having understood the general principles underlying MDPs, we can now move to a formal definition.

Definition 2.2.1. (Markov Decision Process) *A continuous MDP [69, 86] is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$, where:*

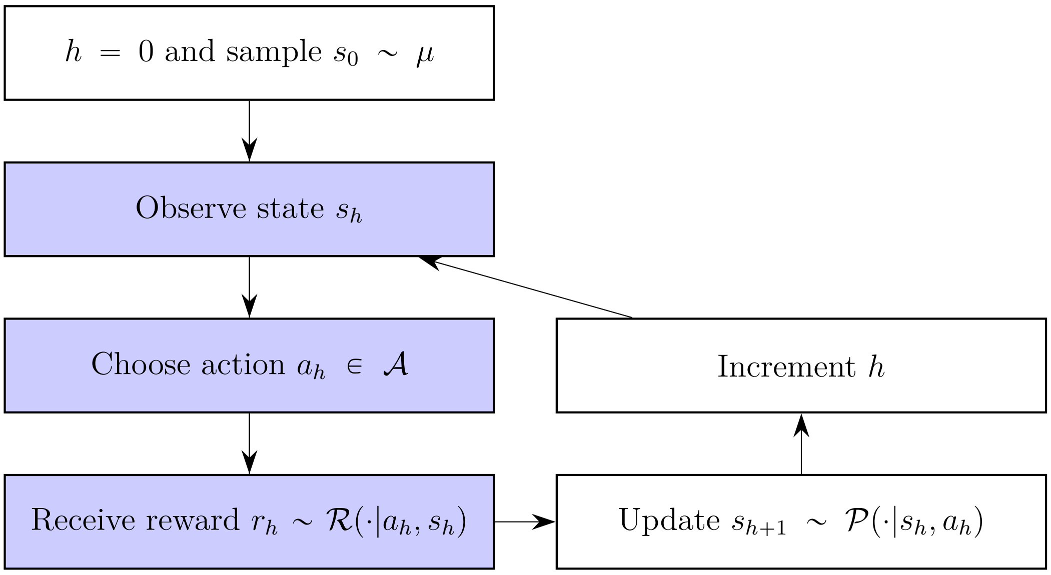
$h = 0$ and sample $s_0 \sim \mu$
Observe state s_h
Choose action $a_h \in \mathcal{A}$
Increment h
Receive reward $r_h \sim \mathcal{R}(\cdot|a_h, s_h)$
Update $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$

Figure 2.1: Interaction protocol for Markov decision processes

(i) $\mathcal{S} \in \mathbb{R}^{d_\mathcal{S}}$ *is the $d_\mathcal{S}$-dimensional* state space*, i.e., the set of all possible observable states of the environment.*

(ii) $\mathcal{A} \in \mathbb{R}^{d_\mathcal{A}}$ *is the $d_\mathcal{A}$-dimensional* action space*, i.e., the set of all possible actions that the agent can perform. Sometimes, not all the actions are performable in all states. In such cases, we can define the set $\mathcal{A}(s)$ for all $s \in \mathcal{S}$, such as $\mathcal{A} = \bigcup_{s\in\mathcal{S}} \mathcal{A}(s)$.*

(iii) $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ *is a function called* transition model *such that, for every $s \in \mathcal{S}$ and $a \in \mathcal{A}$, it assigns a probability measure $\mathcal{P}(\cdot|s, a)$ over $\mathcal{S}$. In general, we denote with $\Delta(\mathcal{X})$ the set of probability measures over a measurable space $\mathcal{X}$; The corresponding probability density function is $P(\cdot|s, a)$.*

(iv) $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta[-R_{\max}, R_{\max}]$ *is a bounded* reward *function such that, every time the agent chooses action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, it assigns a probability measure $\mathcal{R}(\cdot|s, a)$ over $[-R_{\max}, R_{\max}]$. With little abuse of notation, we will denote $\mathcal{R}(s, a)$ its expected return too. $R_{\max}$ is the maximum absolute reward that the agent can receive, i.e., $\sup_{s,a}|\mathcal{R}(s, a)|$.*

(v) $\gamma \in (0, 1]$ *is a discount factor to apply to future rewards. From an economical point of view, it accounts for the fact that an agent might be more interested in a pay-off obtained in the near future rather than a pay-off obtained far in the future. From a statistical point of view the discount factor is related to the probability that the process continues for another decision epoch.*

(vi) $\mu \in \Delta(\mathcal{S})$ *is the initial state distribution, such that the initial state is drawn as* $s_0 \sim \mu$*.*

In the more general case, the state space $\mathcal{S}$ and the action space $\mathcal{A}$, which are the sensor and actuator possibilities of the agent, respectively, can be both continuous and discrete, finite or infinite. In what follows, we will focus on the continuous case because it is the most relevant to our work. The time is typically modelled as a discrete sequence of decision steps represented by the natural numbers: $\mathcal{H} = \{0, 1, \dots, H\}$, where $H \in \mathbb{N}$ is the *horizon* of a given task, which can be either infinite ($H = \infty$) or finite. The agent-environment interaction ends when either the horizon is reached or a *terminal state* is reached, i.e., a state from which no other state can be reached. Tasks are called *episodic* when there exists a terminal state. After having reached a terminal state, usually all the rewards are considered to be zero. A *trajectory* is the sequence of states, actions and rewards $r_{h+1} \sim \mathcal{R}(\cdot|a_h, s_h)$ up to the last time step of the episode: $\tau = \{s_h, a_h, r_{h+1}\}_{h=0}^{H-1}$.

2.2.1 Policies

The core of a MDP agent is the *policy* π, a mapping from perceived states of the environment to possible actions.

Definition 2.2.2. (Policy) *A policy is a stochastic function* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$*, such that the current action is drawn as* $a \sim \pi(\cdot|s)$

Hence, deterministic policies $\pi : \mathcal{S} \to \mathcal{A}$, such that the current action is prescribed as $a = \pi(s)$, are a particular case. As for bandits, in the most general case a policy takes as input the past history of states and actions, but in our work we only consider *memoryless policies*. In fact, we can always find an optimal policy that depends only on the current state. Moreover, we assumed that the agent's policy is stationary.

Remark 2.2.1. Note that we are adopting different notations for time steps in bandits and MDPs, t and h respectively. The reason is that we can think of a full MDP episode performed with policy π as a unique decision epoch t of a bandit. In other words, one could consider a bandit in which, at iteration t, the *bandit agent* pulls a policy π. The *MDP agent* perform a trajectory by interacting with its environment according to policy π. Then, the bandit agent obtains a reward

$r_t = \sum_{h=1}^{H} r_h$. This is the modelling approach that we will adopt in our work. For all details see Section (4.2).

2.2.2 Performance

A we stated above, the goal of an MDP agent is usually to maximize the total discounted reward, also called *return*:

$$\nu = \sum_{h=0}^{\infty} \gamma^h r_{h+1}, \tag{2.5}$$

where $\gamma \in (0, 1]$ is the discount rate. Given this objective, how can we evaluate the agent's policy ? We need to design a *utility function* which represents the performance of the agent's policy with respect to its objective. There are two common formulations of the *performance* $J(\pi)$ in the reinforcement learning literature [87]. In the *start state formulation* the performance is calculated as the long-term discounted reward obtained by starting from a designated start state s_0:

$$J(\pi) = \mathbb{E}[\nu \mid s_0 \sim \mu, \pi]. \tag{2.6}$$

Note that $\gamma = 1$ is allowed in episodic tasks only, otherwise the convergence of the performance is not guaranteed. We can now conveniently introduce the *stationary distribution of states under* π:

$$d^\pi(s) = (1 - \gamma) \sum_{h=0}^{\infty} \gamma^h Pr(s_h = s \mid s_0 \sim \mu, \pi), \tag{2.7}$$

which is the probability induced by the policy π of having $s_h = s$, discounted by γ^h, when $t \to \infty$. This allows us to rewrite the performance measure as:

$$J(\pi) = \int_{\mathcal{S}} d^\pi(s) \int_{\mathcal{A}} \pi(a|s)\mathcal{R}(s, a)\mathrm{d}a\mathrm{d}s. \tag{2.8}$$

In the *average reward formulation* policies are ranked according to their long-term expected reward per step:

$$\begin{aligned} J(\pi) &= \lim_{n\to\inf} \frac{1}{n}\mathbb{E}\left[\sum_{h=1}^{n} r_h \mid \pi\right] \\ &= \int_{\mathcal{S}} d^{\pi}(s)\int_{\mathcal{A}} \pi(a|s)\mathcal{R}(s,a)\mathrm{d}a\mathrm{d}s, \end{aligned} \tag{2.9}$$

where $d^{\pi}(s) = \lim_{n\to\infty} Pr(s_h = s \mid s_0 \sim \mu, \pi)$. In this work, we mostly adopt the first formulation of the performance utility function on episodic tasks, i.e., tasks with an absorbing (terminal) state.

2.2.3 Value functions

From the start state formulation of performance we can derive the value associated to a state s by following policy π, i.e., the *state-value function* $V^{\pi}(s) : \mathcal{S} \to \mathbb{R}$ defined recursively as:

$$V^{\pi}(s) = \mathbb{E}[\nu \mid s_0 = s, \pi] \tag{2.10}$$

$$= \int_{\mathcal{A}} \pi(a|s)\left(\mathcal{R}(s,a) + \gamma\int_{\mathcal{S}} P(s'|s,a)V^{\pi}(s')ds'\right)da, \tag{2.11}$$

which allows to re write the performance as:

$$J(\pi) = \int_{\mathcal{S}} \mu(s)V^{\pi}(s)ds. \tag{2.12}$$

Equation 2.11 is known as *Bellman's equation* of the state-value function. Thanks to another Bellman's equation we can define another value function, namely $Q^{\pi} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$. This value function is more practical for control problems, in which we prefer to reason in terms of which actions are more valuable in a given state. Thus, we define the *action-value function* $Q^{\pi}(s,a)$ as:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}[\nu \mid s_0 = s, a_0 = a, \pi] \\ &= \mathcal{R}(s,a) + \gamma\int_{\mathcal{S}} P(s'|s,a)\int_{\mathcal{A}} \pi(a'|s')Q^{\pi}(s',a')da'ds', \end{aligned} \tag{2.13}$$

The difference between the two value functions is known as advantage function [9]:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{2.14}$$

Intuitively, the advantage function represents how much an action a is convenient is a state s w.r.t. the average utility of the actions.
Value functions are useful for evaluating the quality of a given policy or in order to define policies themselves. In fact, having an estimation of the value function for every state (and possibly action) one can build a policy following a greedy scheme. At each step we may choose the state-action pair that maximize the action-value function, or an action that brings to the maximally valuable state among those that are reachable from the current state.

2.3 Reinforcement Learning

The most basic way to solve MDPs is *dynamic programming*, which simply consists in solving the Bellman equations presented in the previous chapter through a fixed-point iteration scheme. *Value iteration* and *Policy Iteration* are the most popular dynamic programming algorithms [86], which provide the basic structure for most of the value-based reinforcement learning methods. However, these algorithms suffer of two major problems. First, they require knowledge of the MDP transition kernel and of the reward function, which is not available in many real world problems. Second, the update involves an optimization with respect to all possible actions, which can be carried out efficiently only in the case of finite (small) action spaces. RL is a subfield of Machine Learning which aims at solving large-scale problems where the dynamics of the MDP are not know, and thus dynamic programming algorithms can not be employed.

2.3.1 Problem formulation

RL is a branch of machine learning that aims at building learning agents capable to behave in a stochastic and possibly unknown environment, where the only feedback consists of a scalar reward signal. In order to maximize the long-run reward, the agent must learn which state-action pairs are the most profitable by trial-and-error. Therefore, RL algorithms can be seen as computational methods to solve MDPs by directly interacting with the environment, for which a model may or may not be available. The RL problem can be formulated as a stochastic optimization problem aiming at finding the optimal policy π^*:

$$\pi^* = \arg\max_{\pi} J(\pi) \tag{2.15}$$

Remarkably, it is guaranteed that every MDP admits an optimal policy when the space of possible policies is unconstrained [69].

Trial-and-error search and a *delayed reward signal* can be seen as the most characteristic features of reinforcement learning. Compared to supervised learning, one of the main branches of machine learning, the feedback the learner receives is usually less frequent and can be very sparse. In supervised learning, the agent is provided with examples of the correct or expected behaviour by a knowledgeable external supervisor and the agent's goal is to learn how to replicate these examples as well as possible, and possibly generalize this knowledge to new examples. In reinforcement learning, the agent receives a numerical reward that only represents a partial feedback about the goodness of actions taken. A feedback system of this sort is said to be *evaluative* rather than *instructive* and it makes it much more difficult for the agent to learn how to behave in uncharted territory. Evidently, the exploration-exploitation trade-off is a substantial problem in RL as much as for bandits. An agent must exploit what is known of the environment and the reward function in order to obtain rewards, but it also needs to explore to select better actions in the future. On top of this, since rewards might be delayed in time, it will be difficult for the agent to understand which actions are mostly responsible for the received outcome. This represents another characteristic problem of reinforcement learning known as *credit assignement problem* [86]. As we will see, RL algorithms are based on two key ideas: the first is to use samples to derive an approximate representation of the unknown dynamics of the controlled system. The second idea is to use function approximation methods to estimate value functions and policies in high-dimensional state and action spaces.

2.3.2 Taxonomy

Along the years, many algorithms have been proposed to solve the RL problem as presented above. The solutions proposed often display similar approaches that can be used to outline a rough taxonomy of the RL algorithms. In our taxonomy we consider the algorithms along four dimensions: *model requirements*, *policy-based sampling strategy*, *solution search*, *sample usage*. The first dimension refers to the requirements of the algorithm w.r.t. the model of the MDP. On the two sides of this dimension we have:

- *Model-based* algorithms, which require an explicit approximation of the model of the MDP. Not every model-based algorithm is equally demanding on this matter: some of them might need information about every element of the MDP (transition model, reward model, initial state distribution, etc.) while others might require approximations only for a fraction of them, e.g., the reward model only. Examples: DYNA [85], contextual policy-search[50].
- *Model-free* algorithms, on the other hand, do not require any explicit approximation of the model. Model-free algorithms usually have low computational demands but require lots of data in order to get good results; they are well-suited for problems in which computation is costly but sampling is cheap. Examples: Q-learning [93], SARSA [71], REINFORCE [97].

The *policy-based sampling strategy* dimension refers to the relation between the policy that is being used to interact with the environment (*behavioural policy*) and the policy that is being learned by the agent (*target policy*);

- in *off-policy* algorithms there is a clear distinction between the behavioural policy, the one which collects samples and explore the environment, and target policy, which is learned independently. This approach is often useful to safely improve exploration but it is usually more cumbersome to analyse theoretically. Examples: Q-learning [93], off-policy actor-critic [27].
- in *on-policy* algorithms there is no distinction between target and behavioural policy. Examples: SARSA [71], TD{λ} [84].

The *solution search-space* dimension refers to the space where the optimal solution is searched:

- in direct *policy search* methods the optimal solution is searched in the space of policies, e.g., the space in which their parameters lie. These methods are well-suited for large state or action spaces, produce smooth changes in the policy during the learning process, and allow to introduce prior expert knowledge in the policy; unfortunately, they often suffer from a high variance and have optimization issues. Examples: REINFORCE [97], G(PO)MDP [10].

- *Value-based* methods rely instead on the computation of the value function to learn the optimal policy indirectly. Value-based techniques have good convergence properties under small state and action spaces and turn out very suitable when there is insufficient or none prior knowledge about the problem. However, changes to the underlying policy during the learning process can be unstable, and the convergence properties disappear when large (e.g., continuous) state and action spaces demand the use of function approximation. Examples: value-iteration [86], FQI [5].

Differently from the other dimensions which are populated by sharp dichotomies, policy and value-based methods can be combined, often with very successful results. For example, *actor-critic* methods [37] look for the optimal policy in the policy parameter space while leveraging on the predictive power of value-functions in order to guide the parameters update.
The fourth dimension of the taxonomy is the *update frequency*, which refers to the degree of interleaving between learning and experience. Along this dimension, we distinguish:

- *continuing* algorithms, which update the policy (directly or indirectly through the value function) every time new information is available, possibly at each time step;
- *episodic* algorithms, which divides experience into finite segments, called episodes, and update the policy in between each episode;
- *batch* algorithms, which divide learning and experience in two distinct phases. Typically, these are off-policy methods in which a set of behavioural policies collects the data and the target policy is updated upon completion of the batch.

A finer classification of this dimension is presented in [52], where the authors remark that the distinctions between the categories does not depend on the actual formulation: almost any algorithm can fall in any of these three categories depending on the implementation.

2.4 Policy Search

As mentioned in the introduction, this work focuses on improving the exploration task in continuous state-action domains, possibly characterized by some noise in the state as it happens for control tasks with noisy sensors. In such conditions, value-based methods present numerous difficulties. First, they have no guarantees of convergence to an optimal policy due to the need of function approximation for large, continuous MDPs; second, value based methods are highly sensitive to small perturbations in the state, which makes them useless in noisy control tasks. Hence, direct PS methods represent the best alternative in such settings. Theoretical guarantees are often available [59], environment noise has less impactful effects, and prior domain knowledge can be exploited to design ad-hoc policies for specific tasks. Moreover, approximate value functions can still be used to guide the search for the optimal policy, such as in actor-critic methods. Indeed, PS has been successfully applied to complex control tasks, as reported in [28]: from robotic arms playing baseball to simulated table tennis, from dart throwing to pan-cake flipping.

2.4.1 Overview

Given a predetermined class of policies $\hat{\Pi}$, PS aims at finding the policy whose performance $J(\pi)$ is as close as possible to the performance of the optimal policy $J(\pi^*) = J^*$:

$$\hat{\pi} = \arg\min_{\pi \in \hat{\Pi}} \left\| J^* - J(\pi) \right\|_p . \tag{2.16}$$

When $p = 2$, this optimization problem can be interpreted as finding the policy $\hat{\pi} \in \hat{\Pi}$ whose orthogonal projection in the space of Markovian stationary policies Π coincides with the optimal policy π^*.
A Variety of technique have been proposed in literature to solve the problem defined by Equation (2.16). As well as for value-based methods, we can distinguish between model-free, which learn policies directly based on sampled trajectories, and model-based approaches, which use the sampled trajectories to first build a model of the state dynamics. A part from this macro distinction, a further classification can be based on whether we predict trajectories deterministically or generate them stochastically by sampling. In the case of *stochastic trajectory*

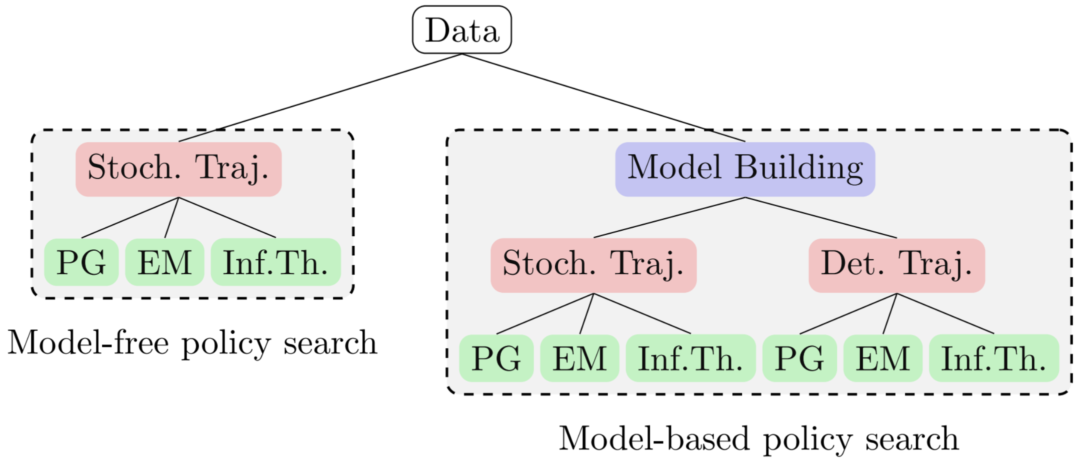

Figure 2.2: A taxonomy of PS methods [28]

generation, trajectories are sampled from the interaction of the agent with the environment, or a simulation of it. *Deterministic trajectory prediction* does not sample trajectories, but analytically predicts the trajectory distribution $p_{\boldsymbol{\theta}}(\tau)$. Typically, deterministic trajectory prediction is computationally more involved than sampling trajectories from the system. However, for the subsequent policy update, deterministic trajectory prediction can allow for an analytic computation of gradients, which can be advantageous over stochastic trajectory generation, where these gradients can only be approximated. Moreover, we can classify policy search methods according to their policy update strategies, as depicted in Figure (2.2). The policy updates in both model-free and model-based PS are based on either policy gradients Policy Gradient (PG) [87], expectation-maximization-based updates [26, 48], or information-theoretic insights [82]. In the followings, we will focus on model-free PS with PG updates.

2.4.2 Policy gradient

In the context of model-free PG, the agent's behaviour is modelled as a differentiable parametric policy $\pi_{\boldsymbol{\theta}} : \mathcal{S} \to \Delta(\mathcal{A})$, independent of time (*stationary*), such that the current action is drawn as $a_h \sim \pi_{\boldsymbol{\theta}}(\cdot|s_h) = \pi(\cdot|s_h, \boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m$ are the policy parameters. The most simple parametrization one could think of is the linear one, where the policy only depends linearly on the policy parameters and the action is drawn deterministically:

$$a = \pi_{\boldsymbol{\theta}}(s) = \boldsymbol{\theta}^T \phi(s), \tag{2.17}$$

where $\phi(s)$ can be any state feature function. In stochastic formulations, typically, a zero-mean Gaussian noise vector is added to $\pi_{\boldsymbol{\theta}}(s)$, so that the policy becomes:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2}\left(\frac{a - \boldsymbol{\theta}^T\phi(s)}{\sigma}\right)^2\right). \tag{2.18}$$

In some cases, the covariance matrix is learnable too, and the parameter set becomes $\{\boldsymbol{\theta}; \boldsymbol{\Sigma}\}$. In most cases, $\boldsymbol{\Sigma}$ is diagonal, thus easing the complexity of the learning task along with the theoretical analysis. Hence, Problem (2.16) translates into finding the optimal policy parameters:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Theta} J(\boldsymbol{\theta}). \tag{2.19}$$

This new optimization problem can be solved by resorting to gradient ascent on the policy parameters, which is guaranteed to converge to a local optimum at least:

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \alpha \boldsymbol{G}^{-1}(\boldsymbol{\theta}^t)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t), \tag{2.20}$$

where $\alpha \geqslant 0$ is the *learning rate*, or *step size*, and $\boldsymbol{G}(\boldsymbol{\theta})$ is a positive definite matrix, called *preconditioning matrix*. The quantity $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ denotes the *policy gradient* PG which gives the name to this method. Its expression is derived in [87].

Theorem 2.4.1. Policy Gradient Theorem *Given a MDP $\mathcal{M}$ and a Markovian stationary stochastic parametric policy $\pi_{\boldsymbol{\theta}}$ differentiable w.r.t. to $\boldsymbol{\theta}$ for all state-action pairs $(s, a) \in (\mathcal{S}, \mathcal{A})$, the gradient of the policy performance is given by:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathcal{S}} d^{\pi_{\boldsymbol{\theta}}}(s) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \mathrm{d}a\mathrm{d}s$$

$\pi_{\boldsymbol{\theta}}(s)$ is the stationary state distribution induced by policy $\pi_{\boldsymbol{\theta}}$, as defined in Equation (2.7). By applying a trivial differentiation rule, $\nabla \log f = \nabla f / f$, we can rewrite Equation (2.21) in a way that will turn out extremely useful:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathcal{S}} d^{\pi_{\boldsymbol{\theta}}}(s) \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s,a) \mathrm{d}a \mathrm{d}s \\ &= \mathop{\mathbb{E}}_{s \sim d^{\pi_{\boldsymbol{\theta}}}, a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s,a) \right]. \qquad (2.21)\end{aligned}$$

This rephrasing is known as the REINFORCE trick [97] and gives the name to an algorithm which is a milestone of RL. In practical applications it can be inconvenient or impossible to compute the Q-function. One solution is to leverage simple function approximators belonging to the class of *compatible basis functions*, i.e., functions of the form $f_{\boldsymbol{\omega}} = \boldsymbol{\omega}^T \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$. Such approximators can replace $Q^{\pi_{\boldsymbol{\theta}}}(s,a)$ in Equation (2.21) for the calculation of the policy gradient as proven in [87].
A second solution that prevent from calculating $Q^{\pi_{\boldsymbol{\theta}}}(s,a)$ explicitly is to resort to a trajectory-based reformulation of Equation (2.21). By rephrasing the expected performance under policy $\pi_{\boldsymbol{\theta}}$ defined in Equation (2.6) as:

$$J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} [\mathcal{R}(\tau)], \qquad (2.22)$$

where $p_{\boldsymbol{\theta}}$ is the distribution over trajectories $\tau \in \mathcal{T}$ induced by the policy $\pi_{\boldsymbol{\theta}}$, and $\mathcal{R}(\tau) = \sum_{h=0}^{H-1} \gamma^h r_{h+1}$, we can write:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathcal{T}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\tau) \mathcal{R}(\tau) \mathrm{d}\tau \qquad (2.23) \\ &= \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau) \mathcal{R}(\tau) \right] \qquad (2.24)\end{aligned}$$

This formulation of the PG turns out to be extremely practical. Not only because it prevents from the calculation of the Q-function, but also because it does not require any knowledge about the transition kernel of the MDP. In fact, $\log p_{\boldsymbol{\theta}}(\tau)$ can be easily derived from the sheer knowledge of the trajectory and definition of the policy:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau) &= \nabla_{\boldsymbol{\theta}} \log \left(\mu(s_{\tau,0}) \prod_{t=0}^{H-1} \pi_{\boldsymbol{\theta}}(a_{\tau,h}|s_{\tau,h}) P(a_{\tau,h+1}|s_{\tau,h+1}) \right) \qquad (2.25) \\ &= \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau,h}|s_{\tau,h}), \ \forall \tau \in \mathcal{T} \qquad (2.26)\end{aligned}$$

At this point, the careful reader might have noticed that we have not yet discussed the meaning of the $\boldsymbol{G}(\boldsymbol{\theta})$ in the parameters update Equation (2.20). The choice of this matrix represent the direction we want to take when performing a gradient update. The two most common direction choices in the RL literature are *steepest gradient ascent* and *natural gradient ascent*. In the first case, it is sufficient to set $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{I}$. However, this option might be ineffective in many cases [4]:

- when there are large plateaus where the gradient is very small and does not point in the direction of the global optimum;
- when the performance function (or, in general, the loss function to optimize) is multimodal;
- when the gradient is not isotropic in magnitude w.r.t. any direction away from its maximum, creating troubles in the step size tuning.

In such cases, the *natural gradient* has been empirically proven to have faster convergence and to avoid premature convergence to local maxima. Natural gradient considers the parameter space as a Riemann manifold equipped with its own norm $\|\boldsymbol{\theta}\|^2_{\boldsymbol{G}(\boldsymbol{\theta})} = \boldsymbol{\theta}^T\boldsymbol{G}(\boldsymbol{\theta})\boldsymbol{\theta}$, which replaces the Euclidean norm, $\|\boldsymbol{\theta}\|^2_{\boldsymbol{I}} = \boldsymbol{\theta}^T\boldsymbol{\theta}$. Such parametric space is a Riemann manifold whose points are probability measures defined on a common probability space. Since $\boldsymbol{G}$ represents the local Riemann metric tensor, in this context it is given by the *Fisher Information Matrix*, i.e., $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{F}(\boldsymbol{\theta})$:

$$\boldsymbol{F}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau\sim p_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau)\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau)^T\right]. \tag{2.27}$$

For more details on the advantages of the natural gradient over the steepest ascent gradient we refer to [4, 3, 68].
We are now ready to present the general setup for policy gradient methods, showed in Algorithm (2.2).

2.4.3 Policy gradient estimation

As shown in the generic policy gradient algorithm, at each iteration an estimate of the policy gradient is required. Indeed, finding a good estimator is one of the main

Algorithm 2.2 Generic PG algorithm

```
1: **Input**: initial policy parameters $\boldsymbol{\theta}^0$, learning rate $\alpha$
2: **Initialize** $t = 0$
3: **while** not converged **do**
4:     Estimate $\boldsymbol{G}(\boldsymbol{\theta}^t)$ with an estimator $\widehat{\boldsymbol{G}}(\boldsymbol{\theta}^t)$
5:     Estimate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t)$ with an estimator $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t)$
6:     Update the policy parameters $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \alpha \widehat{\boldsymbol{G}}^{-1}(\boldsymbol{\theta}^t)\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t)$
7:     $t = t + 1$
8: **end while**
9: **return** an approximation of the optimal policy parameters $\boldsymbol{\theta}^*$
```

challenges of policy gradient methods. Since *RL* aims at solving MDPs whose model is unknown to the agent, the sole mean to estimate the policy gradient is by leveraging on the experience collected, i.e., the samples. Among several techniques proposed in the literature over the last years, the most prominent approaches to estimate the policy gradient from samples are *Finite-Difference* and *Likelihood Ratio* methods [35].

Finite differences

Finite difference methods aim at approximating the policy gradient as a quotient of finite increments. The idea is to perform multiple perturbations $\Delta\boldsymbol{\theta}_1, \Delta\boldsymbol{\theta}_2, ..., \Delta\boldsymbol{\theta}_N$ of the policy parameters. Then, with each set of perturbed parameters collect a number of sample trajectories in order to calculate $\Delta\hat{J}_i = J(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_i) - J(\boldsymbol{\theta})$. At this point, the policy gradient can be estimated by regression as:

$$\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = (\boldsymbol{\Delta\Theta}^T \boldsymbol{\Delta\Theta})^{-1} \boldsymbol{\Delta\hat{J}}, \tag{2.28}$$

where $\boldsymbol{\Delta\Theta} = [\Delta\boldsymbol{\theta}_1, \Delta\boldsymbol{\theta}_2, \dots, \Delta\boldsymbol{\theta}_N]$ and $\boldsymbol{\Delta\hat{J}} = [\Delta\hat{J}_1, \Delta\hat{J}_2, \dots, \Delta\hat{J}_N]$. This method is easy to implement and very efficient when applied to deterministic tasks or pseudo-random number simulations. However, it becomes useless in real control tasks, where a large number of trajectories is required and noise slows down convergence. Moreover, the choice of the perturbation of the parameters is a very difficult problem which may cause instability. For these reasons, the likelihood-ratio method is largely preferred in real control tasks.

Likelihood ratio

Likelihood ratio methods were among the first policy search methods introduced in the early 1990s by Williams [97], and include the famous REINFORCE algorithm. These methods build upon the policy gradient formulation given by Equation (2.24), which, in the same way of Equation (2.21), has been rewritten by applying the so called *REINFORCE trick*, also called *likelihood ratio trick*. As we have seen in Equation (2.26), the policy gradient (Equation (2.24)) can be approximated by using a sum over the sampled trajectories. In a similar way, we can estimate $\mathcal{R}(\tau)$ in a Monte-Carlo fashion. However, such estimates suffer of very large variance. The variance can be reduced by introducing a baseline b for the trajectory reward $\mathcal{R}(\tau)$, i.e.,

$$\hat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, b) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau) \left(\mathcal{R}(\tau) - b \right) \right] \tag{2.29}$$

Remarkably, adding the baseline keep the estimator unbiased. The baseline can be chosen arbitrarily in order to minimize the variance, with the sole condition of being action-independent. The variance-minimizing baseline is usually referred to as *optimal baseline*. Clearly, the optimal baseline depends on the specific likelihood gradient estimation adopted.

2.4.4 Algorithms

The most popular sample-based estimate of the gradient is undoubtedly REINFORCE [97], which uses the total return directly:

$$\hat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, b)_{RF} = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h} | s_{\tau_i,h}) \right) \left(\sum_{h=0}^{H-1} \gamma^h r_{\tau_i}^{h+1} - b \right), \tag{2.30}$$

based on N independent trajectories. The variance of such estimator is represented by a $m \times m$ covariance matrix, with m equals the number of parameters of the policy. We can obtain an optimal baseline in two ways. Either we minimize each element of the diagonal, obtaining a component-wise baseline made of m elements [68]. Or we can minimize the trace of the covariance matrix, obtaining a scalar baseline[98]:

$$(b_{RF}^*)_j = \frac{\mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau,h} | s_{\tau,h}) \right)^2 \mathcal{R}(\tau) \right]}{\mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau,h} | s_{\tau,h}) \right)^2 \right]}, \; j = 1, 2, ..., m; \tag{2.31}$$

Algorithm 2.3 Episodic REINFORCE with component-wise optimal baseline.

```
1: Input: policy parametrization $\boldsymbol{\theta}$
2: Initialize: $n = 0$
3: while not converged do
4:     collect a trajectory $\tau_n$
5:     for every gradient component $j = 1, 2, \dots, m$ do
6:         Estimate the optimal baseline
               $$b_j^n = \frac{\sum_{i=1}^{n} \left( \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h}|s_{\tau_i,h}) \right)^2 \left( \sum_{h=0}^{H-1} \gamma^h r_{\tau_i}^{h+1} \right)}{\sum_{i=1}^{n} \left( \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h}|s_{\tau_i,h}) \right)^2}$$
7:         Estimate the gradient element
               $$\hat{\nabla}_{\boldsymbol{\theta}_j} J(\boldsymbol{\theta})^n = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_j} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h}|s_{\tau_i,h}) \right) \left( \sum_{h=0}^{H-1} \gamma^h r_{\tau_i}^{h+1} - b_j^n \right)$$
8:     end for
9:     $n = n + 1$
10: end while
11: return gradient estimate $\hat{\nabla}_{\boldsymbol{\theta}_j} J(\boldsymbol{\theta})^N$
```

$$b_{RF}^* = \frac{\underset{\tau \sim p_{\boldsymbol{\theta}}}{\mathbb{E}} \left[\left\| \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau,h}|s_{\tau,h}) \right\|_2^2 \mathcal{R}(\tau) \right]}{\underset{\tau \sim p_{\boldsymbol{\theta}}}{\mathbb{E}} \left[\left\| \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau,h}|s_{\tau,h}) \right\|_2^2 \right]}. \tag{2.32}$$

As for the REINFORCE estimator (2.30), these optimal baselines are estimated simply by replacing the expectation with the empirical average. Algorithm (2.3) shows the full REINFORCE procedure with component-based optimal baseline estimation. For sake of clarity, we underline that this algorithm, as well as the others that we discuss in this section, is meant to carry out step (5) of the generic policy gradient algorithm, i.e., Algorithm (2.2).

Another solution tackling the high-variance problem of REINFORCE is to leverage on the intuitive observation that future actions do not depend on past rewards (unless policy changes take place continuously along the trajectory). Hence, we can derive two other well-know gradient estimators, Gradient of the average reward in Partially Observable MDPs (GPOMDP) [10] and PGT [87]:

$$\hat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, b)_{GPOMDP} = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{h=0}^{H-1} \left(\sum_{h'=0}^{h} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h'} | s_{\tau_i,h'}) \right) \left(\gamma^h r_{\tau_i}^{h+1} - b_h \right) \right), \tag{2.33}$$

$$\hat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, b)_{PGT} = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{h=0}^{H-1} \gamma^h \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{\tau_i,h} | s_{\tau_i,h}) \left(\sum_{h'=h}^{H-1} \gamma^{h'-h} r_{\tau_i}^{h'+1} - b_h \right) \right). \tag{2.34}$$

In [68] the authors show that these two estimators are equivalent, despite their apparent difference. In the same paper, the authors show how to derive the optimal baseline for GPOMDP, which, differently from the REINFORCE case, can be time dependent. In the case b = 0 and under mild assumptions, this gradient estimation has been proven to suffer from less variance than REINFORCE [98]. However, large variance remains a common trait of these Monte Carlo PG techniques. In fact, being trajectories generated by sampling at each time step according to a stochastic policy $\pi_{\boldsymbol{\theta}}$, the estimators inherit the variance of the policy. To address this issue, in [78] the authors propose the Policy Gradient with Parameter-Based Exploration (PGPE) method, in which the search in the policy space is replaced with a direct search in the model parameter space. This allows the use of a deterministic controller $\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m$ for sampling the trajectories, i.e., $\pi_{\boldsymbol{\theta}}(a|s) = \delta(a - v_{\boldsymbol{\theta}}(s))$, where $v_{\boldsymbol{\theta}}$ is a deterministic function of the state s, e.g., [79]. The policy parameters are sampled from a distribution $\nu_{\boldsymbol{\xi}} \in \delta(\Theta)$, called *hyper-policy*, where $\boldsymbol{\xi} \in \Xi \subseteq \mathbb{R}^d$ are the hyper-policy parameters, or *hyper-parameters*. Thus, in episodic PGPE it is sufficient to sample the parameters $\boldsymbol{\theta} \sim \nu_{\boldsymbol{\xi}}$ once at the beginning of the episode and then generate an entire trajectory following the deterministic policy $\pi_{\boldsymbol{\theta}}$, with a consequent reduction of the gradient estimate variance. As an additional benefit, the parameter gradient is estimated by direct parameter perturbations, without having to back-propagate any derivatives, which allows to use non-differentiable controllers. For what concern the hyper-policy parametrization choice, the most typical one is Gaussian with diagonal covariance matrix, as it happens in classical PG. Although the policy is deterministic, the exploration is guaranteed by the stochasticity of the hyper-policy. The hyper-parameters $\boldsymbol{\xi}$ are updated by following the gradient ascent direction of the gradient of the expected reward, which can be rewritten as:

Algorithm 2.4 Episodic PGPE

```
1: **Input**: initial hyper-parameters $\boldsymbol{\xi}^0$, learning rate $\alpha$
2: **Initialize** $t = 0$
3: **while** not converged **do**
4:     **for** $n = 1, 2, \dots, N$ **do**
5:         Sample controller parameters $\boldsymbol{\theta}^{(n)} \sim \nu_{\boldsymbol{\xi}_t}$
6:         Sample trajectory $\tau \sim p_{\boldsymbol{\theta}}$ under policy $\pi_{\boldsymbol{\theta}}$
7:     **end for**
8:     Estimate optimal baseline $b$
9:     Estimate the hyper-policy gradient:
           $$\hat{\nabla}_{\boldsymbol{\xi}} J(\boldsymbol{\xi}_t) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}^{(i)})(\mathcal{R}(\tau^{(i)}) - b)$$
10:    Update the hyper-parameters $\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t + \alpha \hat{\nabla}_{\boldsymbol{\xi}} J(\boldsymbol{\xi}_t)$
11:    $t = t + 1$
12: **end while**
13: **return** an approximation of the optimal policy parameters $\boldsymbol{\theta}^* \sim \nu_{\boldsymbol{\xi}^*}$
```

$$J(\boldsymbol{\xi}) = \int_{\Theta} \int_{\mathcal{T}} \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}) p_{\boldsymbol{\theta}}(\tau) \mathcal{R}(\tau) \mathrm{d}\tau \mathrm{d}\boldsymbol{\theta}, \tag{2.35}$$

where $\nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}) p_{\boldsymbol{\theta}}(\tau) = p_{\boldsymbol{\xi}}(\boldsymbol{\theta}, \tau)$ because τ is conditionally independent from $\boldsymbol{\xi}$ given $\boldsymbol{\theta}$. Applying the likelihood ratio technique, we obtain:

$$\nabla_{\boldsymbol{\xi}} J(\boldsymbol{\xi}) = \mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim \nu_{\boldsymbol{\xi}}, \tau \sim p_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}) \mathcal{R}(\tau) \right] \tag{2.36}$$

In order to further reduce the estimate variance, we can adopt an estimator with baseline and compute the optimal one similarly to the REINFORCE case. The episodic PGPE algorithm is reported in Algorithm 2.4.

2.5 Multiple Importance Sampling

In the last section of this chapter we present a statistical tool which is an extension of the well known *importance sampling*. Although this is not strictly related to learning itself, we introduce this tool to the reader because our work leverages it to enable off-line learning for the agent, in a way that is described thoroughly in Chapter (4). We conclude this section with a lemma which represents the first little contribution of this work to the literature and an essential building block for further developments.

Importance sampling [24, 67] is a technique that allows estimating the expectation of a function under some *target* or *proposal* distribution with samples drawn from a different distribution, called *behavioral.* Let P and Q be probability measures on a measurable space $(\mathcal{Z}, \mathcal{F})$, such that $P \ll Q$ (i.e., P is absolutely continuous w.r.t. Q). The importance weight $w_{P/Q}$ is the Radon-Nikodym derivative of P w.r.t. Q, i.e., $w_{P/Q} \equiv \frac{\mathrm{d}P}{\mathrm{d}Q}$. Let p and q be the densities of P and Q, respectively, w.r.t. a reference measure. From the chain rule, $w_{P/Q} = \frac{p}{q}$. In the continuous case, p and q are probability density functions of absolutely continuous random variables having laws P and Q, respectively, and $w_{P/Q}$ is a likelihood ratio. Given a bounded function $f : \mathcal{Z} \to \mathbb{R}$, and a set of i.i.d. outcomes $z_1, \dots, z_N$ sampled from Q, the importance sampling estimator of $\mu := \mathbb{E}_{z \sim P}\left[f(z)\right]$ is:

$$\widehat{\mu}_{\mathrm{IS}} = \frac{1}{N} \sum_{i=1}^{N} f(z_i) w_{P/Q}(z_i), \tag{2.37}$$

which is an *unbiased* estimator [67], i.e., $\mathbb{E}_{z_i \overset{\text{iid}}{\sim} Q}\left[\widehat{\mu}_{IS}\right] = \mu$.

Multiple Importance Sampling (MIS) [92] is a generalization of the importance sampling technique which allows samples drawn from several different behavioral distributions to be used for the same estimate. Let $Q_1, \dots, Q_K$ be all probability measures over the same probability space as P, and $P \ll Q_k$ for $k = 1, \dots, K$. Let $\beta_1(z), \dots, \beta_K(z)$ be mixture weights, i.e., for all $z \in \mathcal{Z}$, $\beta_1(z) + \cdots + \beta_K(z) = 1$ and $\beta_k(z) \geqslant 0$ for $k = 1, \dots, K$. Let z_{ik} denote the i-th sample drawn from Q_k. Given N_k i.i.d. samples from each Q_k, the MIS estimator is:

$$\widehat{\mu}_{\mathrm{MIS}} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k} \beta_k(z_{ik}) w_{P/Q_k}(z_{ik}) f(z_{ik}), \tag{2.38}$$

which is also an unbiased estimator of μ for any valid choice of the mixture weights. A common choice of the mixture weights having desirable variance properties is the balance heuristic [92]:

$$\beta_k(z) = \frac{N_k q_k(z)}{\sum_{j=1}^{K} N_j q_j(z)}, \tag{2.39}$$

which yields the Balance Heuristic Estimator (BH):

$$\widehat{\mu}_{\mathrm{BH}} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \frac{p(z_{ik})}{\sum_{j=1}^{K} N_j q_j(z_{ik})} f(z_{ik}). \tag{2.40}$$

Since (2.39) are valid mixture weights, $\widehat{\mu}_{\mathrm{BH}}$ is an unbiased estimator of μ. Moreover, its variance is not significantly larger than any other choice of the mixture weights [92]. An appealing interpretation of the balance heuristic is that we can look at $\widehat{\mu}_{\mathrm{BH}}$ as an importance sampling estimation in which samples are drawn from the mixture $\Phi = \frac{p(z_{ik})}{\sum_{j=1}^{K} \frac{N_j}{N} q_j(z_{ik})}$.

To further characterize the variance of this estimator, we introduce the concept of *Rényi divergence*. Given probability measures P and Q on $(\mathcal{Z}, \mathcal{F})$, where $P \ll Q$ and Q is σ-finite, the α-Rényi divergence is defined as [70]:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \int_{\mathcal{Z}} \left(w_{P/Q}\right)^{\alpha} \mathrm{d}Q, \tag{2.41}$$

for $\alpha \in [0, \infty]$[1]. We denote the *exponentiated α-Rényi divergence* with:

$$d_\alpha(P\|Q) = \exp\{D_\alpha(P\|Q)\}. \tag{2.42}$$

Of particular interest is d_2, as the variance of the importance weight is $\mathbb{V}\mathrm{ar}_{z\sim Q}\left[w_{P/Q}(z)\right] = d_2(P\|Q) - 1$, which is a divergence itself [25]. For this reason, we always mean the 2-Rényi divergence when omitting the order α. The Rényi divergence was used by [57] to upper bound the variance of the importance sampling estimator as:

$$Var_{z_i \overset{\mathrm{iid}}{\sim} q} [\widehat{\mu}_{\mathrm{IS}}] \leqslant \|f\|_\infty^2 \, d_2(P\|Q)/N. \tag{2.43}$$

A similar result can be derived for the BH estimator:

Lemma 2.5.1. *Let P and $\{Q_k\}_{k=1}^K$ be probability measures on the measurable space $(\mathcal{Z}, \mathcal{F})$ such that $P \ll Q_k$ and $d_2(P\|Q_k) < \infty$ for $k = 1, \dots, K$. Let $f : \mathcal{Z} \to \mathbb{R}$ be a bounded function, i.e., $\|f\|_\infty < \infty$. Let $\widehat{\mu}_{BH}$ be the balance heuristic estimator of f, as defined in (2.40), using N_k i.i.d. samples from each Q_k. Then, the variance of $\widehat{\mu}_{BH}$ can be upper bounded as:*

$$\mathbb{V}\mathrm{ar}_{z_{ik} \overset{iid}{\sim} Q_k} [\widehat{\mu}_{BH}] \leqslant \|f\|_\infty^2 \frac{d_2(P\|\Phi)}{N}, \tag{2.44}$$

[1]The special cases $\alpha = 0, 1$ and ∞ are defined as limits.

where $N = \sum_{k=1}^{K} N_k$ is the total number of samples and $\Phi = \sum_{k=1}^{K} \frac{N_k}{N} Q_k$ is a finite mixture.

Chapter 3

Exploration Techniques

In this chapter we present the state of the art of MABs and RL techniques that focus on the exploration challenge arising in these learning frameworks. While sketching a sort of taxonomy of the exploration techniques, we describe some emblematic algorithms for each category, with particular attention to those that inspired our work. Indeed, the exploration problem in the PS setting is the main target of our thesis project. Our main contributions, described in Chapters (4-5), have been designed after a careful literature review. In this chapter we try to report the most relevant insights collected along this review. The reader may notice that, despite this thesis project aims at solving the exploration problem in the framework of PS, few references to PS algorithms will be given. In fact, to the best of our knowledge, little research has been done so far on the exploration problem in PS.

Following the taxonomy outlined by Sebatian Thrun in 1992 for the RL framework [90], we divide both MABs and RL exploration techniques in two families of exploration schemes: *undirected* and *directed* exploration. While the former family is closely related to random walk exploration, directed exploration techniques memorize exploration-specific knowledge which is used for guiding the exploration search. In many finite deterministic domains, any learning technique based on undirected exploration is inefficient in terms of learning time, i.e. learning time is expected to scale exponentially with the size of the state space [95]. For all these domains, reinforcement learning using a directed technique can always be performed in polynomial time, demonstrating the important role of exploration in reinforcement learning [90]. However, an important remark is that we do not address the problem of pure exploration, in which there is no price to be paid for exploration and the only objective is to find the reward-maximizing actions. Instead, we investigate effective exploration in the more general context of learning described in the previous chapter (2.1.1), where the agent faces the exploration-exploitation dilemma all along its journey. How can the agent achieve a better balance between exploration and exploitation? How can it explore more effectively? Can it improve the effectiveness of its exploration while going through the learning process? We will try to answer these questions in the two sections of this chapter. Each of the sections will start with an overview on undirected exploration techniques, then directed strategies will be investigated more thoroughly.

3.1 Exploration in Multi Armed Bandits

3.1.1 Undirected Exploration

The most uninformed and basic way of exploring an unknown environment is to generate actions randomly with uniform probability. This method is often applied if exploration costs do not matter during learning. Sometimes tasks are divided into two phases, namely an exploration and an exploitation phase, and costs are not considered during the exploration phase. This is the case of the *explore-then-commit* algorithm (Algorithm (2.1)) presented in the previous chapter, which explores uniformly the finite number of arms m before starting the exploitation phase, after mK steps. Another common algorithm with undirected uniform exploration is the ϵ-*greedy* algorithm [53]. One can think of it as an extension of the *explore-then-commit* algorithm which allows to continue exploring with probability ϵ even during the performance phase (i.e., the exploitation phase). The idea is, after a pure exploration phase of mK steps, to exploit the estimated best arm $x_t = \arg\max \widehat{\mu}_i(t)$ with probability $(1 - \epsilon_t)$, and explore uniformly the remaining set of arms with probability ϵ_t, at each step. In 2002, Auer et al. [6] analyzed the regret of ϵ-*greedy* with slowly decreasing ϵ and showed its asymptotic convergence to the optimal solution. These are just asymptotic guarantees, however, and say little about the practical effectiveness of the methods.

Another approach to undirected exploration which is more effective when costs are relevant during learning is *non-uniform* exploration utilizing the current utility estimates to influence action-selection. The higher the expected utility of action i, the more likely i gets selected. This ensures that the learning system explores and exploits simultaneously, as it happens for the ϵ-greedy algorithm, but in an intuitively more effective way since actions are not chosen uniformly during exploration. An example of utility-driven non-uniform undirected exploration is the Boltzmann-distributed exploration [21]. By defining the average reward of an arm as in Equation (2.3):

$$\hat{\mu}_i(t) = \frac{1}{T_i(t)} \sum_{s=1}^{t} \mathbb{1}_{\{x_s=i\}} r_s, \tag{3.1}$$

the probability for an action i to get selected is a non-linear function of $\hat{\mu}_i$:

$$Pr(i) = \frac{e^{\hat{\mu}_i \tau^{-1}}}{\sum_{a\in\mathcal{X}} e^{\hat{\mu}_a \tau^{-1}}}. \tag{3.2}$$

Here τ is a gain factor, often called *temperature*, which determines the amount of randomness in the action-selection procedure. With $\tau \to 0$, pure exploitation is approached, and with $\tau \to \infty$ the resulting distribution approaches the uniform distribution.
In the rest of this section, we present some techniques of *directed exploration*, which memorize knowledge about the learning process itself and utilize this knowledge to guide the exploration.

3.1.2 Count-based exploration and Upper Confidence Bound

Count-based exploration memorizes knowledge about the number of times $T_i(t)$ action i has been visited up to time-step t, for all $i \in \mathcal{X}$. Strategies based on counting usually evaluate actions by a linear combination of an exploitation term and an exploration term, called the *exploration bonus*, which is a (usually inverse) function of $T_i(t)$. Hence, action x_t is selected in a deterministic fashion as:

$$x_t = \arg\max_{x\in\mathcal{X}} \{\widehat{\mu}_x + bonus(T_x(t-1))\}. \tag{3.3}$$

Such strategy is at the core of the celebrated Upper Confidence Bound (UCB) algorithm [51, 1, 6], that overcomes many of the limitations of strategies based on exploration followed by commitment. The algorithm has many different forms, depending on the distributional assumptions on the noise. Here, we assume the noise is 1-subgaussian, i.e.,:

Definition 3.1.1. Subgaussianity *A random variable X is σ-subgaussian if, for all $\lambda \in \mathbb{R}$, it holds that $\mathbb{E}\left[\exp(\lambda X)\right] \leqslant \exp(\lambda^2\sigma^2/2)$.*

Coupling this property with *Markov's inequality*, we can prove that:

Theorem 3.1.1. *[53] If X is σ-subgaussian, then, for any $\epsilon \geqslant 0$,*

$$P(X \geqslant \epsilon) \leqslant \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

From this theorem, we can derive a key corollary which gives us precious information about the concentration of the sample mean around the true mean of independent, σ-subgaussian random variables:

Corollary 3.1.1. *Assume that $X_i - \mu$, for $i = 1, 2, \ldots, n$ are independent, σ-subgaussian random variables. Then, for any $\epsilon \geqslant 0$, with probability at least $1 - \delta$, $\delta \in [0, 1]$:*

$$\mu \leqslant \widehat{\mu} + \sqrt{\frac{2\sigma^2\log(1/\delta)}{n}} \quad \textit{and} \quad \mu \geqslant \widehat{\mu} - \sqrt{\frac{2\sigma^2\log(1/\delta)}{n}},$$

where $\widehat{\mu} = \frac{1}{n}\sum_{t=1}^{n} X_t$.

When considering its options in round t, the learner has observed $T_i(t-1)$ samples from arm i and received rewards from that arm with an empirical mean of $\widehat{\mu}_i$. Then an *optimist* candidate for the unknown mean of the ith arm is:

$$UCB_i(t-1, \delta) = \widehat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}}. \tag{3.4}$$

Great care is required when comparing (3.1.1) and (3.4) because in the former the number of samples is the constant n, but in the latter it is a random variable $T_i(t-1)$. Apart from this technicality, the intuition remains that δ is approximately an upper bound on the probability of the event that the above quantity is an underestimate of the true mean. In fact, by Corollary (3.1.1), for any $\delta \in [0, 1]$:

$$Pr\left(\mu \geqslant \widehat{\mu} + \sqrt{\frac{2\sigma^2\log(1/\delta)}{n}}\right) \leqslant \delta. \tag{3.5}$$

Algorithm 3.1 UCB(δ) algorithm

```
1: Input: $K$ arms and $\delta$
2: Choose each action once
3: for $t = 1, 2, \dots, T$ do
4:     if $t \leqslant K$ then
5:         $x_t = t$
6:     else
7:         $x_t = \arg\max_i UCB_i(t-1, \delta)$
8:     end if
9: end for
```

After having picked all actions once, the UCB policy simply picks at each time step t the action i whose $UCB_i(t-1, \delta)$ index is maximum, as shown in Algorithm (3.1).

The value of $1-\delta$ is called the *confidence level*, and different choices lead to different algorithms, each with their pros and cons, and sometimes different analysis. For example, Auer et al. [6], present an algorithm originating in [1] called *UCB1* which achieves asymptotic logarithmic regret $\mathcal{O}(\log T)$, for all reward distributions, with no prior knowledge of the reward distribution. Equivalently, we say that UCB1 achieves asymptotic sub-linear pseudo-regret $\widetilde{\mathcal{O}}(1)$. *UCB1* takes $\delta = 1/t$, where t is the current time step, resulting in:

$$UCB1_i(t, \delta) = \widehat{\mu}_i(t) + \sqrt{\frac{2\log(t)}{T_i(t)}} \tag{3.6}$$

However, rather than considering 1-subgaussian noise, Auer et al. [6] consider bandits where the payoffs are confined to the $[0, 1]$ interval, which are ensured to be 1/2-subgaussian. Better bounds have been proven for modifications of the UCB1 algorithm, for example Improved UCB [8] and Kullback-Leibler Upper confidence Bound (KL-UCB) [33].

3.1.3 Optimism in the Face of Uncertainty

The family of UCB algorithms is of particular relevance because it represents a successful implementation of the principle of Optimism in the Face of Uncertainty (OFU), which is applicable to various exploration problems, not only finite-armed

stochastic bandits. The OFU principle states that one should choose their actions as if the environment is as nice as plausibly possible.
To illustrate the intuition imagine of being in the same situation depicted in (2.1.1), i.e., a standard lunch break from work. You can choose between your good old-fashioned favourite restaurant or sampling a restaurant that you never visited before. Taking an optimistic view of the unknown restaurant leads to exploration because without data it *could* be amazing. Then, after trying the new option a few times you can update your statistics about each choice and make a more informed decision. On the other hand, taking a pessimistic view of the new option discourages exploration and you may suffer significant regret if the local options are delicious. The UCB strategy assigns to each arm an upper confidence bound that with high probability is an overestimate of the unknown mean reward. The intuitive reason why this leads to sublinear regret is simple. Assuming the upper confidence bound assigned to the optimal arm is indeed an overestimation, then another arm can only be played if its upper confidence bound is larger than that of the optimal arm, which in turn is larger than the mean of the optimal arm. And yet this cannot happen too often because the additional data provided by playing a suboptimal arm means that the upper confidence bound for this arm will eventually fall below that of the optimal arm. The next algorithm that we discuss will be useful to better understand the OFU principle and its exploration power, with application to a wider set of arms.

3.1.4 Hierarchical Optimistic Optimization

In 2011, Bubeck et al. [20] proposed a novel optimistic arm selection strategy whose regret improved upon previous results on continuum and Lipschitz bandits (e.g., [47, 45]), extending and generalizing the environment class of application to arbitrary topological spaces. In particular, the setting is that of $\mathcal{X}$-armed bandits, introduced in (2.1.3). This algorithm is particularly interesting to us for two reasons:

(i) it applies the OFU principle to continuous action spaces, as we wish to do in the context of RL Policy Search;

(ii) it leverages the correlation between arms (expressed as a dissimilarity function) to explore the environment more effectively.

The functioning of the Hierarchical Optimistic Optimization (HOO) algorithm is very similar to the UCB strategy, with application to a more general set of arms. Consider for example a compact, two-dimensional continuous set of arms, like a square plane in the euclidean space. The strategy is to recursively cutting the square in rectangles, sub-rectangles and so on, building a binary graph of sets and subsets. We start from the whole square, pull a random arm from it, then divide it in two rectangles. These two rectangles become the new leaves of the graph. Subsequently, we pick one of the two leaves, pull a random arm from it, and spawn two new leaves by sectioning it. Then we repeat the procedure. At each time step, the choice of the leaf (rectangle) depends on an upper bound of the mean payoff of the arms laying within it. All the arms within the same rectangle are considered to have a similar mean payoff because of the structure imposed to the arm set.

Formally, the HOO strategy assumes that the decision maker is able to cover the space of arms in a recursive manner, successively refining the regions in the covering such that the diameters of these sets shrink at a known geometric rate when measured with the dissimilarity metric. In particular, the authors define a tree of coverings as:

Definition 3.1.2. *(*Tree of coverings*) A tree of coverings is a family of measurable subsets* $(\mathcal{P}_{(h,i)})_{1\leqslant i\leqslant 2^h, h\geqslant 0}$ *of* $\mathcal{X}$ *such that for all fixed integer* $h \geqslant 0$*, the covering* $\bigcup_{1\leqslant i\leqslant 2^h} P_{h,i} = \mathcal{X}$ *holds. Moreover, the elements of the covering are obtained recursively: each subset* $P_{h,i}$ *is covered by the two subsets* $P_{h+1,2i-1}$ *and* $P_{h+1,2i}$*.*

Remark 3.1.1. A typical choice for the coverings in a cubic domain is to let the domains be hyper-rectangles. They can be obtained, for example, in a dyadic manner, by splitting at each step hyper-rectangles in the middle along their longest side, in an axis parallel manner; if all sides are equal, we split them along the first axis.

The number of visits of a node (h, i) up to time-step T is given by the number of visits of every node belonging to $\mathcal{C}(h, i)$, i.e., the set of the node (h, i) and its descendants:

$$T_{h,i}(t) = \sum_{l=1}^{t} \mathbb{1}_{\{(h_l,i_l)\in\mathcal{C}(h,i)\}}, \tag{3.7}$$

where (h_l, i_l) are the coordinates of the node selected at time step l. Then, the empirical average of the rewards received for the time-points when the path followed by the algorithm has gone through (h, i) is:

$$\widehat{\mu}_{h,i}(t) = \frac{1}{T_{h,i}(t)} \sum_{l=1}^{t} r_l \mathbb{1}_{\{(h_l,i_l) \in \mathcal{C}(h,i)\}}. \tag{3.8}$$

Thus, similarly to the $UCB(\delta)$ algorithm, an upper confidence bound can be defined as:

$$U_{h,i}(t) = \widehat{\mu}_{h,i}(t) + \sqrt{\frac{2 \log t}{T_{h,i}(t)}} + \nu_1 \rho^h, \tag{3.9}$$

where $\rho \in (0,1)$ and $\nu_1 \in \mathbb{R}_0^+$ are parameters of the algorithm. Specifically, given depth h, $\nu_1 \rho^h$ represents an upper bound of the diameter of each node $\mathcal{P}_{(h,i)}$ of the three of coverings. The existence of such bound is a necessary assumption for the functioning of HOO. Along with the nodes the algorithm stores what the authors call B-values:

$$B_{h,i}(t) = \min\{U_{h,i}(t), \max\{B_{h+1,2i-1}(t), B_{h+1,2i}(t)\}\} \tag{3.10}$$

Finally, let us denote by $\mathcal{T}$ the infinite tree of coverings, by $\mathcal{T}_t$ the set of nodes of the tree that have been picked in previous rounds and by $\mathcal{S}_t$ the nodes which are not in $\mathcal{T}_t$. Now, for a node (h,i) in $\mathcal{S}_t$, we define its B-value to be $B_{h,i}(t) = +\infty$. We now have everything we need to present the full HOO strategy in Algorithm (3.2).

The definition and use of B-values, that puts in relationship each node (subset of $\mathcal{X}$) with all its children (subsets of the subset of $\mathcal{X}$), shows how the algorithm relies on the correlation between the reward of an arm and that of its neighbours. Indeed, the proof of the regret achieved by HOO relies on the following assumption:

Assumption 3.1.1. *The mean-payoff function μ is weakly Lipschitz w.r.t. a dissimilarity metric ℓ, i.e., $\forall x_1, x_2 \in \mathcal{X}$,*

$$\mu^* - \mu(x_2) \leqslant \mu^* - \mu(x_1) + \max\{\mu^* - \mu(x_1), \ell(x_1, x_2)\}. \tag{3.12}$$

Note that weak Lipschitzness is satisfied whenever μ is 1-Lipschitz, i.e., $\forall x_1, x_2 \in \mathcal{X}$, $|\mu(x_1) - \mu(x_2)| \leqslant \ell(x_1, x_2)$. On the other hand, weak Lipschitzness

Algorithm 3.2 HOO algorithm

```
1: **Input:** the infinite tree of coverings $\mathcal{T}$, $\rho \in (0,1)$ and $\nu_1 \in \mathbb{R}_0^+$
2: **Initialize:** $t = 0$, $\mathcal{T}_0 = \varnothing$, $\mathcal{S}_0 = \mathcal{T}$, $B_{h,i}(0) = +\infty\ \forall (h,i) \in \mathcal{S}_0$
3: Choose the root node $(h_0, i_0) = (0,1)$ and update $t = t + 1$
4: **for** $t = 1, 2, \dots, T$ **do**
5:     Choose a node according to the deterministic rule:
           $(h_t, i_t) = \arg\max_{(h,i)\in\mathcal{S}_t} B_{h,i}(t)$    (3.11)
6:     Choose, possibly at random, an arm $x_t \in \mathcal{P}_{(h_t,i_t)}$ and collect reward $r_t$
7:     Update $B_{h,i}(t)$ for $(h_t, i_t)$ and all its parent nodes
8:     $\mathcal{T}_{t+1} = \mathcal{T}_t \cup (h_t, i_t)$, $\mathcal{S}_{t+1} = \mathcal{S}_t / (h_t, i_t)$
9: **end for**
```

implies local (one-sided) 1-Lipschitzness at any maxima. Indeed, at an optimal arm x^*, Equation (3.12) rewrites as $\mu(x^*) - \mu(x_2) \leqslant \ell(x^*, x_2)$.
By carefully choosing the tree of coverings and the algorithm parameters, the authors show that the regret of the proposed approach is bounded by $\widetilde{\mathcal{O}}(\sqrt{T}) = \mathcal{O}(\log \sqrt{T})$. For more details, refer to [20].

3.1.5 Posterior Sampling

Another way to address the exploration-exploitation dilemma in MABs is by posterior sampling. Particularly relevant is an heuristic called Thompson Sampling (TS), firstly introduced by [89], which has been thoroughly studied in the literature for its good properties. This technique is considered to be close to UCB algorithms because it also allocates exploratory effort to actions that might be optimal i.e., it also builds upon the OFU principle [72]. Indeed, similarly to UCB, TS avoids the under-exploitation of actions that are potentially good and prevents from wasting time on actions that are already known to be bad. The Bayesian approach is what sets TS apart from the techniques previously mentioned. It requires a prior distribution for each arm as input, which encodes the initial belief that a learner has on each arm reward. Subsequently, at each round t, it samples from each arm distribution, chooses the action with the best reward and updates its posterior distribution by means of the Bayes'rule.
The elements of TS are the followings:

(i) a set Θ or parameters θ of the distribution of $r \sim \mathcal{R}_\theta(\cdot|x)$;

(ii) a prior distribution $P(\theta)$ on these parameters;

(iii) the history $\mathcal{I}_t = \{x_0, r_0, x_1, r_1, \ldots, x_{t-1}, r_{t-1}\}$ of past observations;

(iv) a likelihood function $P(\mathcal{I}_t|\theta)$;

(v) a posterior distribution $P(\theta|\mathcal{I}_t) \propto P(\mathcal{I}_t|\theta)P(\theta)$.

TS consists in playing the action $x \in \mathcal{X}$ according to the probability that it maximizes the expected reward according to your belief, i.e.,:

$$\arg\max_{x\in\mathcal{X}} \mathbb{E}[r|x] = \arg\max_{x\in\mathcal{X}} \int \mathbb{E}[r|\theta, x]P(\theta|\mathcal{I}_t)d\theta \tag{3.13}$$

Once having observed the reward from the chosen arm, the posterior distribution can be updated using the Bayes' rule. In practice, it is either impossible or computationally expensive to compute the integral above. Therefore, usually we resort to sampling $\hat{\theta}_t \sim P(\theta|\mathcal{I}_t)$ at time t, and then play the arm $x_t = \arg\max_{x\in\mathcal{X}} \mathbb{E}[r|\hat{\theta}_t, x]$. Conceptually, this means that the player instantiates their beliefs randomly in each round, and then acts optimally according to them. In [2] the authors prove that for the K-armed stochastic bandit problem, Thompson Sampling has expected regret of

$$\mathbb{E}[Regret(T)] \leqslant (\sqrt{KT\log T}), \tag{3.14}$$

which is identical to the best known problem-independent bound for the expected regret of UCB1 [18].

Example: Thompson Sampling for Bernoulli MAB Suppose that the reward function of each arm i, $i = 1, 2, \ldots, K$ is a Bernoulli of parameter p_i: $r_t, i \sim \mathcal{B}(p_i)$ i.e., the result of pulling an arm is either a *success* (with probability p_i) or a *failure* (with probability $1 - p_i$). We know that, given a uniform prior over the parameter $p \sim \mathcal{U}(0, 1) = Beta(1, 1)$, and having observed $\alpha - 1$ successes and $\beta - 1$ failures, the posterior distribution of the parameter p of a Bernoulli is a Beta distribution of parameters α and β (Beta distributions are frequently adopted as priors because of their conjugacy properties). Hence, TS can be performed as shown in Algorithm(3.3).

For a more extensive discussion on algorithms implementing TS refer to [73].

Algorithm 3.3 TS for Bernoulli MAB with Beta priors

```
1: Initialize: the prior distribution for each arm as $Beta(1,1)$
2: for $t = 1, 2, \ldots, T$ do
3:     for $i = 1, 2, \ldots, K$ do
4:         Sample $\hat{p}_i \sim Beta(\alpha_i, \beta_i)$
5:     end for
6:     Pull arm $x_t = \arg\max_i \hat{p}_i$ and observe $r_t \in \{0,1\}$
7:     Update distribution:
           $$(\alpha_i, \beta_i) = \begin{cases} (\alpha_i, \beta_i), & \text{if } x_t \neq i \\ (\alpha_i, \beta_i) + (r_t, 1 - r_t), & \text{if } x_t = i \end{cases} \quad (3.15)$$
8: end for
```

3.1.6 Gaussian Process Upper Confidence Bound

We conclude by presenting an algorithm, Gaussian Process Upper Confidence Bound (GPUCB) [81], of particular interest to us for three reasons:

(i) it serves as a further example of application of the OFU principle and of UCB variant;

(ii) it formalizes the broad problem of optimizing an unknown, noisy function, that is expensive to evaluate, as a multi-armed bandit problem;

(iii) it deals with a continuous set of arms.

Indeed, our main contribution builds upon a similar formalization of the RL continuous control problem. Moreover, it also represents a case of posterior sampling different from TS.
Consider the MAB problem presented in Chapter (2), where the reward distribution can be seen as the sum of a reward function (e.g. the expected reward function $\mu(\boldsymbol{x}_t)$, that here we note as $f(\boldsymbol{x}_t)$) and noise, i.e., $y_t = \mathcal{R}(\cdot|x_t) = f(\boldsymbol{x}_t) + \epsilon_t$. As we already know, the goal is to maximize the sum of the rewards collected by choosing a point of the domain $\boldsymbol{x}_t \in \mathcal{X}$ at each round t up to the horizon T. For example, we might want to find locations of highest temperature in a building by sequentially activating sensors in a spatial network and regressing on their measurements. Each activation draws battery power, so we want to sample from as few sensors as possible. In this context, as in many other MAB problems, a natural performance metric is the cumulative regret as defined in

Algorithm 3.4 GPUCB

```
1: Input: GP prior $\mu_0 = 0$, $\mu_0 = 0$, $\beta_t \in \mathbb{R}^+$
2: for $t = 1, 2, \dots, T$ do
3:     Choose $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mu_{t-1}(\boldsymbol{x}) + \sqrt{\beta_t}\sigma_{t-1}(\boldsymbol{x})$
4:     Sample $y_t = f(\boldsymbol{x}_t) + \epsilon_t$
5:     Perform Bayesian update to obtain $\mu_t$ and $\sigma_t$
6: end for
```

Definition (2.1.3). GPUCB explicitly encourages additional exploration in a UCB sense by choosing, at each step, the arm associated to the highest upper bound of the mean return. Given an arm x this bound is calculated by summing the mean and standard deviation of the estimator of f. In this case, f is sampled from a prior distribution with known mean and standard deviation, whose posterior is updated in a Bayesian fashion at every step. In order to enforce some implicit properties like smoothness without relying on any parametric assumption, the authors of [81] model f as a sample from a Gaussian Process (GP): a collection of dependent random variables, one for each $\boldsymbol{x} \in \mathcal{X}$, every finite subset of which is multivariate Gaussian distributed [96]. A GP is a stochastic process $GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, specified by its mean $\mu(\boldsymbol{x}) = \mathbb{E}[f(x)]$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))$. GPUCB generally adopts $GP(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ as prior distribution over f. A major advantage of working with GPs is the existence of simple analytic formulae for mean and co-variance of the posterior distribution, which allow easy implementation of algorithms. For a noisy sample $\boldsymbol{y}_T = [y_1, y_2, \dots, y_T]^T$ at points $A_T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T\}$ over f, the posterior of f is a GP distribution again specified by mean $\mu_T(\boldsymbol{x})$, covariance $k_T(\boldsymbol{x}, \boldsymbol{x}')$ and variance $\sigma_T^2(\boldsymbol{x})$:

$$\mu_T(\boldsymbol{x}) = \boldsymbol{k}_T(\boldsymbol{x})^T(\boldsymbol{K}_T + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_T, \tag{3.16}$$

$$k_T(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_T(\boldsymbol{x})^T(\boldsymbol{K}_T + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_T(\boldsymbol{x}'), \tag{3.17}$$

$$\sigma_T^2(\boldsymbol{x}) = k_T(\boldsymbol{x}, \boldsymbol{x}), \tag{3.18}$$

$$\tag{3.19}$$

where $\boldsymbol{k}_T(\boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), \dots, k(\boldsymbol{x}_T, \boldsymbol{x})]$ and $\boldsymbol{K}_T$ is the positive definite kernel matrix $[k(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x}, \boldsymbol{x}' \in A_T}$. The full GPUCB procedure, motivated by the UCB algorithm, is shown in Algorithm 3.4.

Despite being a successful and easy-to-implement algorithm in many cases, GPUCB suffers of two main drawbacks. First, the assumption of f being sampled from a GP prior is unrealistic in many cases. Furthermore, finding the upper confidence index in line 3 of Algorithm (3.4) may be hard if the arm set $\mathcal{X}$ is continuous (or generally infinite). Indeed, the bound is multimodal in general. Although global search heuristics might be effective, no successful practical applications have been registered in the continuous case to our knowledge.

3.2 Exploration in Reinforcement Learning

3.2.1 Undirected Exploration

The most classic undirected exploration strategies in RL are the same of those introduced earlier for MABs. Indeed, the ϵ-*greedy* policy is one of the most widely used and known algorithm in reinforcement learning [86]. In the same way, Boltzmann exploration, which uses the exponential of the standard Q-function as the probability of an action, has been extensively studied [90, 21]. Similar policy representations are energy-based models, with the Q-value obtained from an energy model such as a restricted Boltzmann machine [74]. An interesting family of extensions to these techniques goes under the name of *maximum entropy* RL.

Maximum Entropy Reinforcement Learning

As stated in Chapter (2), Equation (2.15), the goal of RL is to find the optimal policy π^* which maximizes the performance $J(\pi)$. Maximum entropy RL augments this objective by introducing an entropy term, such that the optimal policy also aims at maximizing its entropy at each visited state:

$$\pi^* = \arg\max_{\pi} \sum_{h} \mathop{\mathbb{E}}_{s_h, a_h \sim d^\pi} \left[\mathcal{R}(s_h, a_h) + \alpha \mathcal{H}(\pi(\cdot|s_h)) \right], \tag{3.20}$$

where α is a convenient hyperparameter that can be used to determine the relative importance of entropy and reward. This objective function entails a novel approach to undirected exploration w.r.t. Boltzmann exploration. While the latter greedily maximizes entropy at the current time step, maximum entropy RL explicitly optimizes for policies that aim to reach states where actions will have high entropy in the future. This distinction is crucial, since the maximum entropy objective can be shown to maximize the entropy of the entire trajectory distribution for the policy π, while the greedy Boltzmann exploration approach does not. Optimization problems of this type have been covered in a number of scientific papers, e.g., [99, 38, 39]. In [38], Haarnoja et al. extend the objective presented above to infinite time horizons with the introduction of a discount factor λ:

$$\pi^* = \arg\max_{\pi} \sum_{h} \mathop{\mathbb{E}}_{s_h, a_h \sim d^\pi} \left[\sum_{l} \gamma^{l-h} \mathop{\mathbb{E}}_{s_l, a_l \sim d^\pi} \left[\mathcal{R}(s_l, a_l) + \alpha \mathcal{H}(\pi(\cdot|s_l)) | s_h, a_h \right] \right]. \tag{3.21}$$

his objective corresponds to maximizing the discounted expected reward and entropy for future states originating from every state-action tuple (s_h, a_h) weighted by its probability d^π under the current policy. Then, they define a soft Q-value Q^π_{soft} for any policy π as the expectation under π of the discounted sum of rewards and entropy:

$$Q^\pi_{soft} = r_1 + \mathop{\mathbb{E}}_{s_0 = s, a_0 = a, \tau \sim \pi} \left[\sum_{h=1}^{\infty} r_{h+1} + \alpha \mathcal{H}(\pi(\cdot|s_h)) \right], \tag{3.22}$$

from which they derive the soft Bellman equation. The authors build upon this novel setting an algorithm called *Soft Q-learning*, which has similar mechanisms to traditional *Q-learning* and adopts deep function approximation to compute the Q-values over a continuous domain. Among the advantages presented over other deep RL approaches, Soft Q-learning turns out to be more effective for learning multi-modal policies for exploration. In fact, similarly to MABs, during the learning process it is often best to keep trying multiple available options until the agent is confident that one of them is the best. However, deep RL algorithms for continuous control typically use unimodal action distributions, which are not well suited to capture such multi-modality. As a consequence, such algorithms may prematurely commit to one mode and converge to suboptimal behaviour, as it happens in the practice. One year later, the same authors of *Soft Q-learning* [38] proposed a powerful extension named *Soft Actor-Critic* [39], which combined off-policy updates with a stable stochastic actor-critic formulation.

Following the same pattern adopted for MABs, we proceed now with presenting some techniques that deals with the exploration challange in a more directed way, i.e., by leveraging on relevant information collected during the learning process to lead exploration more effectively.

3.2.2 Count-based exploration and Intrinsic Motivation

The classic, theoretically-justified MABs exploration methods based on counting state-action visitations and turning this count into a bonus reward have been introduced in (**??**). Taking inspiration from the MAB literature, many similar count-based algorithms have been designed for tabular RL. It is the case of the notorious E^3 algorithm [43], and two other popular *optimist* algorithms: R_{max} [15] and *UCRL* [7]. In recent works [11, 88, 65, 23], they have also been adapted to large, non-tabular RL problems . A relevant example is that of Bellamare et al. [11], later developed in [65], which proposes a novel algorithm for deriving a *pseudo-count* from an arbitrary density model. Let ρ be a density model on a finite space $\mathcal{X}$ and $\rho_n(x)$ the probability assigned by the model to x after being trained on a sequence of stated $x_1, x_2, \dots, x_n \in \mathcal{X}$. Assume $\rho_n(x) > 0$ for all x, n. The *recording probability* $\rho'_n(x)$ is then the probability the model would assign to x one more time. The *prediciton gain* of ρ is:

$$PG_n(x) = \log \rho'_n(x) - \log \rho_n(x). \tag{3.23}$$

Note that $PG_n(x) \geqslant 0$ for all $x \in \mathcal{X}$ whenever the the density model ρ is *learning-positive*, i.e., if $\rho'_n(x) \geqslant \rho_n(x)$ for all $x_1, x_2, \dots, x_n$. For learning -positive ρ, the authors define the *pseudo-count* as:

$$\widehat{N}_n(x) = \frac{\rho_n(x)(1 - \rho'_n(x))}{\rho'_n(x) - \rho_n(x)}, \tag{3.24}$$

derived from postulating that a single observation of $x \in \mathcal{X}$ should lead to a unit increase in pseudo-count:

$$\rho_n(x) = \frac{\widehat{N}_n(x)}{\hat{n}}, \; \rho'_n(x) = \frac{\widehat{N}_n(x) + 1}{\hat{n} + 1}, \tag{3.25}$$

where $\hat{n}$ is the *pseudo-count total.* The pseudo-count generalizes the usual state visitation count function $N_n(x)$. Moreover, under certain assumptions on ρ_n, pseudo-counts grow approximately linearly with real counts. Crucially, the pseudo-count can be approximated using the prediction gain of the density model:

$$\widehat{N}_n(x) = \left(e^{PG_n(x)} - 1\right)^{-1} \tag{3.26}$$

Its main use is to define an *exploration bonus* that can be added to every extrinsic reward r_n received by the agent. For example, at step n, the agent would face a total reward R_n given by:

$$r'_n = r_n + (\widehat{N}_n(x))^{-\frac{1}{2}}, \tag{3.27}$$

which incentivize the agent to try to re-experience surprising situations. In [65] the authors adopt PixelCNN, an advanced neural density model for images first introduced in [60], to supply a pseudo-count. They manage to combine PixelCNN pseudo-counts with different agent architectures to dramatically improve the state of the art on several hard Atari games, a popular set of RL environments for benchmarking.

Interestingly, this notion of pseudo-counts and their application is closely related to the notion of *intrinsic motivation*. In fact, quantities related to prediction gain have been used for similar purposes in the intrinsic motivation literature [54], where they measure an agent's *learning progress* [66]. The concept of *intrinsic motivation* has been introduced in the RL literature by Singh et al. in [22], who borrowed it from psychology. Psychologists call behaviour intrinsically motivated when it is engaged for its own sake rather than as a step toward solving a specific problem of clear practical value. This concept builds on the one of *curiosity*, which has been also extensively investigated in the RL literature after being introduced in it by professor Jürgen Schmidhuber in 1991 [75]. The rationale behind it is that, in some RL scenarios, as well as in human life, rewards are supplied to the agent so sparsely that traditional techniques fails miserably in the learning challenge. This is a problem as the agent receives reinforcement for updating its policy only if it succeeds in reaching a pre-specified goal state. Hoping to stumble into a goal state by chance (i.e. random exploration) is likely to be futile for all but the simplest of environments. Intrinsic motivation / curiosity have been used to instil the need to explore the environment and discover novel states in the agent, regardless of the reward scheme. Most formulations of intrinsic motivation can be grouped into two broad classes:

(i) encourage the agent to explore *novel* states;

(ii) encourage the agent to perform actions that reduce the error/uncertainty in the agent's ability to predict the consequence of its own actions, i.e., its knowledge about the environment.

Measuring *novelty* requires a statistical model of the distribution of the environmental states, as the one described above, whereas measuring prediction error/uncertainty requires building a model of environmental dynamics that predicts the next state s_{h+1} given the current state s_h and the action a_h executed at time h. Evidently, these mechanisms, particularly the first one, are closely related to the concept of count-based exploration. Indeed, Bellamare et al. showed in [11] the close relationship between the two.

Variational Information Maximizing Exploration

To help the reader grasping these concepts, we proceed by describing a successful implementation of the principle of intrinsic motivation, the Variational Information Maximizing Exploration (VIME) algorithm [41]. VIME makes use of the information gain about the agent's internal belief of the dynamics model as a driving force, as described in the following. The approach taken is model-based, where the agent models the environment dynamics via a model $p(s_{h+1}|s_h, a_h, \theta)$, parametrized by the random variable Θ with values $\theta \in \mathbf{\Theta}$. Assuming a prior $p(\theta)$, it keeps a distribution over dynamic models through a distribution over θ, which is updated in a Bayesian fashion. The authors formalize the intrinsic goal of taking actions that maximize the reduction in uncertainty about the dynamics as taking actions that lead to states that are maximally informative about the dynamics model. In other terms, the agent is encouraged to maximize the mutual information between the next state distribution S_{h+1} and the model parameter Θ:

$$I(S_{h+1}; \Theta|\xi_h, a_h) = \underset{s_{h+1} \sim \mathcal{P}(\cdot|\xi_h, a_h)}{\mathbb{E}} \left[D_{KL}[p(\theta|\xi_h, a_h, s_{h+1})||p(\theta|\xi_h)] \right], \tag{3.28}$$

where $D_{KL}(P||Q)$ is the Kullback-Leibler (KL) divergence between probabilities P and Q, $\mathcal{P}$ are the true environment dynamics and $\xi_h = \{s_1, a_1, ..., s_h\}$ is the history of the agent up until time step t. This KL divergence can be interpreted as information gain. Therefore we can add each term $I(S_{h+1}; \Theta|\xi_h, a_h)$ as an *intrinsic reward* to the standard reward r_h obtained by the agent at time step t.

The trade-off between exploitation and exploration can now be realized explicitly as follows:

$$r'_h = r_h + \eta D_{KL}[p(\theta|\xi_h, a_h, s_{h+1})||p(\theta|\xi_h)], \tag{3.29}$$

with $\eta \in \mathbb{R}^+$ being a hyperparameter controlling the urge to explore. The biggest practical issue with maximizing information gain for exploration is that the computation of this equation requires calculating the posterior $p(\theta|\xi_h, a_h, s_{h+1})$, which is generally intractable. The practical solution adopted by the authors for calculating the posterior $p(\theta|\mathcal{D})$ for a dataset $\mathcal{D}$ is to approximate it through an alternative distribution $q(\theta;\phi)$, parametrized by ϕ, by minimizing $D_{KL}[q(\theta;\phi)||p(\theta|\mathcal{D})]$. This is done through maximization of the *variational lower bound* $L[q(\theta;\phi), \mathcal{D}]$, also called *evidence lower bound* [12]:

$$L[q(\theta;\phi), \mathcal{D}] = \mathop{\mathbb{E}}_{\theta\sim q(\cdot|\phi)}[\log p(\mathcal{D}|\theta)] - D_{KL}[q(\theta;\phi)||p(\theta)]. \tag{3.30}$$

Rather than computing information gain in Equation (3.29) explicitly, we compute an approximation to it, leading to the following total reward:

$$r'_h = r_h + \eta D_{KL}[q(\theta;\phi_{h+1})||q(\theta;\phi_h)], \tag{3.31}$$

with ϕ_{h+1} the updated and ϕ_t the old parameters representing the agent's belief. Natural candidates for parametrizing the agent's dynamics model $q(\theta;\phi)$ are Bayesian neural networks, whose weight distribution ($q(\theta;\phi)$ itself) is given by the fully factorized Gaussian distribution [13]. This is particularly convenient as it allows for a simple analytical formulation of the KL divergence, that can be easily approximated. In fact, VIME approximates the KL divergence between the former belief ϕ_t on the dynamics and the approximate updated belief ϕ'_t. Algorithm 3.5 outline the procedure. For more details about the implementation the reader should refer to [41].

3.2.3 Posterior Sampling

The TS technique described in the previous section has been adopted and investigated in the framework of RL as well [83, 62, 63, 61]. Here, TS involves

Algorithm 3.5 VIME

```
 1: for each epoch $t$ do
 2:     for for $h = 1, 2, \dots, H$ do
 3:         Generate action $a_h \sim \pi(s_h)$ and sample $s_{h+1} \sim \mathcal{P}(\cdot|\xi_h, a_h)$, get $r_h$
 4:         Add triplet $(s_h, a_h, s_{h+1})$ to FIFO replay pool R
 5:         Compute the approximated divergence $D_{KL}[q(\theta;\phi'_t||q(\theta;\phi_t)]$
 6:         Construct $r'_h = r_h + \eta D_{KL}[q(\theta;\phi'_t||q(\theta;\phi_t)]$
 7:     end for
 8:     Sample randomly $\mathcal{D}$ from R
 9:     Minimize $-L[q(\theta;\phi_t), \mathcal{D}]$ and obtain the updated posterior $q(\theta;\phi_{t+1})$
10:     Use rewards $r'_1, r'_h, \dots, r'_H$, to update policy $\pi$ using any standard RL
    method
11: end for
```

sampling a statistically plausibly set of action values (or policies) and selecting the maximizing action (or best policy). These values can be generated, for example, by sampling from the posterior distribution over MDPs and computing the state-action value function of the sampled MDP. This approach, originally proposed in [83], is called posterior sampling for reinforcement learning. In [64], the authors present theoretical and experimental arguments to show that posterior sampling is better than optimism in synthesizing efficient exploration with powerful generalization. Although we build our work leveraging on the principle of optimism in the face of uncertainty, posterior sampling appears to be an interesting option to further investigate. Indeed, as we will see, the formalization of the RL problem that we propose in this paper seems to be a good testing ground for TS.

Chapter 4

Optimistic Policy Search via Multiple Importance Sampling

In this chapter, we present the main theoretical contributions of this thesis project. In Section (4.1), we extend Section (2.5) to develop robust MIS estimators that will play an essential role in the algorithms proposed. In Section (4.2) we provide a formalization of the online policy optimization problem. The algorithms are presented in Section (4.3) and analyzed in Section (4.4). The proposed algorithms, called Optimistic Policy opTImization via Multiple Importance Sampling with Truncation (OPTIMIST) and OPTIMIST2, are based on the OFU principle and follows the UCB strategy, both introduced in Section (2.1) and covered with practical implementations in (3.1). The idea is to leverage on these techniques to deal with the problem of exploration in continuous-action RL, for which the solutions proposed so far have been largely heuristic, as we have seen in Chapter (3). However, the solutions proposed here should not be interpreted as an application of the MAB framework to PS but rather as a way to formulate exploration in policy search as a MAB-like problem

4.1 Robust Importance Sampling Estimation

In this section, we discuss how to perform a robust importance weighting estimation. Recently, it has been observed that, in many cases of interest, the plain estimator (2.37) presents problematic tail behaviors [57], preventing the use of exponential concentration inequalities. In fact, unless we require that $d_\infty(P\|\Phi)$ is finite, i.e., that the importance weight have finite supremum, there always exists a value $\alpha > 1$ such that $d_\alpha(P\|\Phi) = +\infty$. A common heuristic to address this problem consists in truncating the weight [42]:

$$\check{\mu}_{\text{IS}} = \frac{1}{N}\sum_{i=1}^{N} \min\left\{M, w_{P/Q}(z_i)\right\} f(z_i), \tag{4.1}$$

where $M > 0$ is a threshold to limit the magnitude of the importance weight. Similarly, for the multiple importance sampling case, restricting to the BH, we have:

$$\check{\mu}_{\text{BH}} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} \min\left\{M, \frac{p(z_{ik})}{\sum_{j=1}^{K}\frac{N_j}{N}q_j(z_{ik})}\right\} f(z_{ik}). \tag{4.2}$$

Clearly, since we are changing the importance weights, we introduce a bias term, but, by reducing the range of the estimation, we get a benefit in terms of variance. Below, we present the bias-variance analysis of the estimator $\check{\mu}_{\text{BH}}$ and we conclude by showing that we are able, using an adaptive truncation, to guarantee an exponential concentration (differently from the non-truncated case).

Lemma 4.1.1. *Let P and $\{Q_k\}_{k=1}^{N}$ be probability measures on the measurable space $(\mathcal{Z},\mathcal{F})$ such that $P \ll Q_k$ and there exists $\epsilon \in (0,1]$ s.t. $d_{1+\epsilon}(P\|Q_k) < \infty$ for $k = 1,\dots,K$. Let $f : \mathcal{Z} \to \mathbb{R}_+$ be a bounded non-negative function, i.e., $\|f\|_\infty < \infty$. Let $\check{\mu}_{BH}$ be the truncated balance heuristic estimator of f, as defined in (4.2), using N_k i.i.d. samples from each Q_k. Then, the bias of $\check{\mu}_{BH}$ can be bounded as:*

$$0 \leqslant \mu - \underset{z_{ik}\overset{iid}{\sim}Q_k}{\mathbb{E}}\left[\check{\mu}_{BH}\right] \leqslant \|f\|_\infty M^{-\epsilon} d_{1+\epsilon}\left(P\|\Phi\right)^{\epsilon}, \tag{4.3}$$

and the variance of $\check{\mu}_{BH}$ can be bounded as:

$$\underset{z_{ik}\overset{iid}{\sim}Q_k}{\mathbb{V}\mathrm{ar}}\left[\check{\mu}_{BH}\right] \leqslant \|f\|_\infty^2 M^{1-\epsilon}\frac{d_{1+\epsilon}\left(P\|\Phi\right)^{\epsilon}}{N}, \tag{4.4}$$

where $N = \sum_{k=1}^{K} N_k$ is the total number of samples and $\Phi = \sum_{k=1}^{K}\frac{N_k}{N}Q_k$ is a finite mixture.

It is worth noting that, by selecting $\epsilon = 1$, equation (4.4) reduces to Lemma 2.5.1, as the truncation operation can only reduce the variance. Clearly, the smaller we choose M, the larger the bias. Overall, we are interested in mining the joint contribution of bias and variance. Keeping P and Φ fixed we observe that the bias depends on M only, whereas the variance depends on M and on the number of samples N. Intuitively, we can allow larger truncation thresholds

M as the number of samples N increases. The following result states that, when using an *adaptive threshold* depending on N, we are able to reach exponential concentration.

Theorem 4.1.1. *Let P and $\{Q_k\}_{k=1}^N$ be probability measures on the measurable space $(\mathcal{Z}, \mathcal{F})$ such that $P \ll Q_k$ and there exists $\epsilon \in (0,1]$ s.t. $d_{1+\epsilon}(P\|Q_k) < \infty$ for $k = 1, \dots, K$. Let $f : \mathcal{Z} \to \mathbb{R}_+$ be a bounded non-negative function, i.e., $\|f\|_\infty < \infty$. Let $\check{\mu}_{BH}$ be the truncated balance heuristic estimator of f, as defined in (4.2), using N_k i.i.d. samples from each Q_k. Let $M_N = \left(\frac{N d_{1+\epsilon}(P\|\Phi)^\epsilon}{\log \frac{1}{\delta}}\right)^{\frac{1}{1+\epsilon}}$, then with probability at least $1-\delta$:*

$$\check{\mu}_{BH} \leqslant \mu + \|f\|_\infty \left(\sqrt{2} + \frac{1}{3}\right) \left(\frac{d_{1+\epsilon}(P\|\Phi) \log \frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}, \tag{4.5}$$

and also, with probability at least $1-\delta$:

$$\check{\mu}_{BH} \geqslant \mu - \|f\|_\infty \left(\sqrt{2} + \frac{4}{3}\right) \left(\frac{d_{1+\epsilon}(P\|\Phi) \log \frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}. \tag{4.6}$$

Our adaptive truncation approach and the consequent concentration results resemble the ones proposed in [19]. However, unlike [19] we do not remove samples with too high value, but we exploit the nature of the importance weighted estimator only to limit the weight magnitude. Indeed, this form of truncation turned out to be very effective in practice [42].

4.2 Problem Formalization

The online learning problem that we aim to solve does *not* fall within the traditional MAB framework (not in its basic version, anyway) and can benefit from an ad-hoc formalization, provided in this section.

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be our decision set, or *arm set* in MAB jargon. Let $(\Omega, \mathcal{F}, P)$ be a probability space. Let $\{Z_{\boldsymbol{x}} : \Omega \to \mathcal{Z} \mid \boldsymbol{x} \in \mathcal{X}\}$ be a set of continuous random vectors parametrized by $\mathcal{X}$, with common sample space $\mathcal{Z} \subseteq \mathbb{R}^m$. We denote with $p_{\boldsymbol{x}}$ the probability density function of $Z_{\boldsymbol{x}}$. Finally, let $f : \mathcal{Z} \to \mathbb{R}$ be a bounded *payoff function*, and $\mu(\boldsymbol{x}) = \mathbb{E}_{z \sim p_{\boldsymbol{x}}}[f(z)]$ its expectation under $p_{\boldsymbol{x}}$. For each iteration $t = 0, \dots, T$, we select an arm $\boldsymbol{x}_t$, draw a sample z_t from $p_{\boldsymbol{x}_t}$, and observe payoff $f(z_t)$, up to horizon T. The goal is to maximize the expected total payoff:

$$\max_{\boldsymbol{x}_0,\dots,\boldsymbol{x}_T \in \mathcal{X}} \sum_{t=0}^{T} \mathop{\mathbb{E}}_{z_t \sim p_{\boldsymbol{x}_t}} [f(z_t)] = \max_{\boldsymbol{x}_0,\dots,\boldsymbol{x}_T \in \mathcal{X}} \sum_{t=0}^{T} \mu(\boldsymbol{x}_t). \tag{4.7}$$

Although we can evaluate $p_{\boldsymbol{x}}$ for each $\boldsymbol{x} \in \mathcal{X}$, we can only observe $f(z_t)$ for the z_t that are actually sampled. This models precisely the online, episodic policy optimization problem. Within this problem formulation, we distinguish between two categories of policy optimization: *action-based* and *parameter-based* policy optimization.

- In *action-based* policy optimization, $\mathcal{X}$ corresponds to the parameter space Θ of a class of stochastic policies $\{\pi_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \Theta\}$; $\mathcal{Z}$ to the set of possible trajectories; $p_{\boldsymbol{x}}$ to the density $p_{\boldsymbol{\theta}}$ over trajectories induced by policy $\pi_{\boldsymbol{\theta}}$; and $f(z)$ to cumulated reward $\mathcal{R}(\tau)$.
- In *parameter-based* policy optimization, $\mathcal{X}$ corresponds to the hyperparameter space Ξ of a class of stochastic hyperpolicies $\{\nu_{\boldsymbol{\xi}} \mid \boldsymbol{\xi} \in \Xi\}$; $\mathcal{Z}$ to the set of possible trajectories collected with policy $\pi_{\boldsymbol{\theta}}$, with $\boldsymbol{\theta} \in \Theta \sim \nu_{\boldsymbol{\xi}}$; $p_{\boldsymbol{x}}$ to hyperpolicy $\nu_{\boldsymbol{\xi}}$; and $f(z)$ to performance $J(\boldsymbol{\theta})$.

In both cases, each iteration corresponds to a single episode, and horizon T is the total number of episodes (not to be confused with the trajectory horizon H). With reference to the standard PS framework, the first category is closely related to standard PG methods that perform a search in a parametric policy space by following the gradient of the utility function. The second is more related to methods searching directly in the space of parameters like PGPE. From now on, we will refer to (4.7) simply as the policy optimization problem.

Remark 4.2.1. In abstract terms, (4.7) is a sequential decision problem over a functional space of random variables, and may have applications beyond policy optimization.

The peculiarity of this framework, compared to the classic MAB one, is the special structure existing over the arms. In particular, the expected payoff μ of different arms is correlated thanks to the stochasticity of the $p_{\boldsymbol{x}}$'s on a common sample space $\mathcal{Z}$. We *could*, of course, frame policy optimization as a MAB problem, at the cost of ignoring some structure. It would be enough to regard $\mu(\boldsymbol{x})$ as the expectation of a totally unknown, stochastic reward function. This would put us in the continuous MAB framework [46], but would ignore the special arm correlation. In the following, we will show how this correlation can be exploited to guarantee efficient exploration. This insight stems from the literature on $\mathcal{X}$-armed bandits, as discussed in (2.1.3) and (3.1.4).

4.3 Algorithms

In this section, we use the mathematical tools presented so far to design a policy search algorithm that efficiently explores the space of solutions. The proposed algorithm, called OPTIMIST, is based on the OFU principle and follows the UCB strategy.

To apply the UCB strategy to the policy optimization problem described above, we need an estimate of the objective $\mu(\boldsymbol{x})$ and a confidence region. We use importance sampling to capture the correlation among the arms. In particular, to better use all the data that we collect, we would like to use a multiple importance sampling estimator like the one from (2.38). Unfortunately, the heavy-tailed behavior of this estimator would result in an inefficient exploration. Instead, we use the robust balance heuristic estimator $\check{\mu}_{\mathrm{BH}}$ from (4.2), which has better tail behavior. To simplify the notation, we treat each sample $\boldsymbol{x}$ as a distinct one. This is w.l.o.g. (as each sample is always multiplied by its number of occurrences anyway) and corresponds to the case $K = t-1$ and $N_k \equiv 1$. Hence, at each iteration t:

$$\check{\mu}_t(\boldsymbol{x}) = \sum_{k=0}^{t-1} \min\left\{ M_{t-1}, \frac{p_{\boldsymbol{x}}(z_k)}{\sum_{j=1}^{t-1} p_{\boldsymbol{x}_j}(z_k)} \right\} f(z_k), \tag{4.8}$$

where $M_t = \left(\frac{t d_{1+\epsilon}(p_{\boldsymbol{x}} \| \Phi_t)^{\epsilon}}{\log \frac{1}{\delta_t}} \right)^{\frac{1}{1+\epsilon}}$ and $\Phi_t = \frac{1}{t}\sum_{k=0}^{t-1} p_{\boldsymbol{x}_k}$. According to Theorem (4.1.1), the following *index*:

$$B_t^{\epsilon}(\boldsymbol{x}, \delta_t) := \check{\mu}_t(\boldsymbol{x}) + \|f\|_{\infty} \left(\sqrt{2} + \frac{4}{3} \right) \left(\frac{d_{1+\epsilon}(p_{\boldsymbol{x}_t} \| \Phi_t) \log \frac{1}{\delta_t}}{t} \right)^{\frac{\epsilon}{1+\epsilon}}, \tag{4.9}$$

is an upper bound on $\mu(\boldsymbol{x})$ with probability at least $1-\delta_t$, i.e., an upper confidence bound. The OPTIMIST algorithm simply selects, at each iteration t, the arm with the largest value of the index $B_t^{\epsilon}(\boldsymbol{x})$, breaking ties deterministically. The pseudocode is provided in Algorithm (4.1). The initial arm $\boldsymbol{x}_0$ is arbitrary, as no prior information is available. The regret analysis of Section (4.4) will provide a confidence schedule $(\delta_t)_{t=1}^{T}$. Knowledge of the actual horizon T is not needed. Although we can use any $\epsilon \in (0, 1]$, we suggest to use $\epsilon = 1$ in practice, as it yields the more common 2-Rényi divergence. To be able to compute the indexes (or to perform any kind of index maximization), the algorithm needs to store all the $\boldsymbol{x}_t$ together with the observed pay-offs $f(z_t)$, hence $\mathcal{O}(Td)$ space is required, where d is the dimensionality of the arm space $\mathcal{X}$ (not to be confused with cardinality $|\mathcal{X}|$, which may be infinite).

Algorithm 4.1 OPTIMIST

```
1: Input: initial arm $\boldsymbol{x}_0$, confidence schedule $(\delta_t)_{t=1}^T$, order $\epsilon \in (0, 1]$
2: Draw sample $z_0 \sim p_{\boldsymbol{x}_0}$ and observe payoff $f(z_0)$
3: for $t = 1, \dots, T$ do
4:     Select arm $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} B_t^\epsilon(\boldsymbol{x}, \delta_t)$
5:     Draw sample $z_t \sim p_{\boldsymbol{x}_t}$ and observe payoff $f(z_t)$
6: end for
```

The optimization step (line 4) may be very difficult when $\mathcal{X}$ is not discrete [81], as the index $B_t^\epsilon(\boldsymbol{x}, \delta_t)$ is non-convex and non-differentiable. Global optimization methods could be applied at the cost of giving up theoretical guarantees. In practice, this direction may be beneficial, but we leave it to future, more application-oriented work. Instead, we propose a general discretization method. The key intuition, common in the continuous MAB literature, is to make the discretization progressively finer. The pseudocode for this variant, called OPTIMIST2, is reported in Algorithm (4.2). Note that the arm space $\mathcal{X}$ itself is fixed (and infinite), as adaptive discretization is performed for optimization purposes only. Implementing any variant of OPTIMIST to solve a policy optimization problem, whether in the action-based or in the parameter-based formulation, requires some additional caveats, discussed in Section (5.1).

4.4 Regret Analysis

In this section, we provide high-probability guarantees on the quality of the solution provided by Algorithm (4.1). First, we rephrase the optimization problem (4.1) in terms of *regret minimization*. The instantaneous regret, as defined in (2.1.2), is:

$$\Delta_t = \mu(\boldsymbol{x}^*) - \mu(\boldsymbol{x}_t), \tag{4.10}$$

where $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mu(\boldsymbol{x})$. Let $Regret(T) = \sum_{t=0}^T \Delta_t$ be the total regret, as in (2.1.3). As $\mu(\boldsymbol{x}^*)$ is a constant, problem (4.7) is trivially equivalent to:

$$\min_{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T \in \mathcal{X}} Regret(T). \tag{4.11}$$

In the following, we will show that Algorithm (4.1) yields sublinear regret under some mild assumptions. The proofs combine techniques from [81] and [19] and are reported in Appendix (A). First, we need the following assumption on the Rényi divergence:

Assumption 4.4.1. *For all $t = 1, \dots, T$, the $(1+\epsilon)$-Rényi divergence is uniformly bounded as:*

$$\sup_{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T \in \mathcal{X}} d_{1+\epsilon}(p_{\boldsymbol{x}_t} \| \Phi_t) = v_\epsilon < \infty,$$

where $\Phi_t = \frac{1}{t} \sum_{k=0}^{t-1} p_{\boldsymbol{x}_k}$*,*

which can be easily enforced through careful policy (or hyperpolicy) design, as discussed in 5.1.2.

4.4.1 Discrete arm set

We start from the discrete case, where $|\mathcal{X}| = K \in \mathbb{N}_+$. This setting is particularly convenient, as the optimization step can be trivially solved in time $\mathcal{O}(Kt)$ per iteration,[1] where t is from evaluation of (4.8) via clever caching. This sums up to total time $\mathcal{O}(KT^2)$. This setting is also of practical interest: even in applications where $\mathcal{X}$ is naturally continuous (e.g., robotics), the set of solutions that can be actually tried in practice may sometimes be constrained to a discrete, reasonably small set. In this simple setting, OPTIMIST achieves $\widetilde{\mathcal{O}}(T^{\frac{1}{1+\epsilon}})$ regret:

Theorem 4.4.1. *Let $\mathcal{X}$ be a discrete arm set with $|\mathcal{X}| = K \in \mathbb{N}_+$. Under Assumption (4.4.1), Algorithm (4.1) with confidence schedule $\delta_t = \frac{3\delta}{t^2\pi^2 K}$ guarantees, with probability at least $1 - \delta$:*

$$\textit{Regret}(T) \leqslant \Delta_0 + CT^{\frac{1}{1+\epsilon}} \left[v_\epsilon \left(2 \log T + \log \frac{\pi^2 K}{3\delta} \right) \right]^{\frac{\epsilon}{1+\epsilon}},$$

where $C = (1+\epsilon)\left(2\sqrt{2} + \frac{5}{3}\right) \|f\|_\infty$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

This yields a $\widetilde{\mathcal{O}}(\sqrt{T})$ regret when $\epsilon = 1$.

4.4.2 Compact arm set

Now, we consider the more general case of a compact arm set $\mathcal{X} \in \mathbb{R}^d$. This case is also more interesting as it allows to tackle virtually any RL task. We can assume, w.l.o.g., that $\mathcal{X}$ is entirely contained in a box $[-D, D]^d$, with $D \in \mathbb{R}_+$. We also need the following assumption on the expected payoff:

[1]We consider the evaluation of pdf's, payoffs and Rényi divergences in (4.8) atomic, as its complexity is heavily problem-dependent.

Assumption 4.4.2. *The expected payoff μ is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that, for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:*

$$|\mu(\boldsymbol{x}') - \mu(\boldsymbol{x})| \leqslant L \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_1 .$$

This assumption is easily satisfied for policy optimization, as shown in the following:

Lemma 4.4.1. *In the policy optimization problem, Assumption 4.4.2 can be replaced by:*

$$\sup_{s \in \mathcal{S}, \boldsymbol{\theta} \in \Theta} \mathop{\mathbb{E}}_{a \sim \pi_{\boldsymbol{\theta}}} \left[|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)| \right] \leqslant \boldsymbol{u}_1, \tag{4.12}$$

in the action-based paradigm, and by:

$$\sup_{\boldsymbol{\xi} \in \Xi} \mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim \rho_{\boldsymbol{\xi}}} \left[|\nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta})| \right] \leqslant \boldsymbol{u}_2, \tag{4.13}$$

in the parameter-based paradigm, where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are d-dimensional vectors and the inequalities are component-wise.

In the proof (Appendix (A)), we show how to derive the corresponding Lipschitz constants, and show how (4.12) and (4.13) are satisfied by the commonly-used Gaussian policy and hyperpolicy, respectively. This is enough for OPTIMIST to achieve $\widetilde{\mathcal{O}}(d^{\frac{\epsilon}{1+\epsilon}} T^{\frac{1}{1+\epsilon}})$ regret:

Theorem 4.4.2. *Let $\mathcal{X}$ be a d-dimensional compact arm set with $\mathcal{X} \subseteq [-D, D]^d$. Under Assumptions (4.4.1) and (4.4.2), Algorithm (4.1) with confidence schedule $\delta_t = \frac{6\delta}{\pi^2 t^2 (1 + d^d t^{2d})}$ guarantees, with probability at least $1 - \delta$:*

$$\textit{Regret}(T) \leqslant \Delta_0 + C T^{\frac{1}{1+\epsilon}} \left[v_\epsilon \left(2(d+1) \log T + d \log d + \log \frac{\pi^2}{3\delta} \right) \right]^{\frac{\epsilon}{1+\epsilon}} + \frac{\pi^2 L D}{6},$$

where $C = (1+\epsilon) \left(2\sqrt{2} + \frac{5}{3} \right) \|f\|_\infty$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

This yields a $\widetilde{\mathcal{O}}(\sqrt{dT})$ regret when $\epsilon = 1$. Unfortunately, the optimization step may be very time-consuming. In some applications, we can assume the time required to draw samples to dominate the computational time. In fact, drawing a sample (Algorithm (4.1), line 2) corresponds to generating a whole trajectory of experience, which may take a long time, especially in real-world applications.

Algorithm 4.2 OPTIMIST2

```
1: Input: initial arm $\boldsymbol{x}_0$, confidence schedule $(\delta_t)_{t=1}^T$, discretization schedule $(\tau_t)_{t=1}^T$, order $\epsilon \in (0,1]$
2: Draw sample $z_0 \sim p_{\boldsymbol{x}_0}$ and observe payoff $f(z_0)$
3: for $t = 1, \dots, T$ do
4:     Discretize $\mathcal{X}$ with a uniform grid $\widetilde{\mathcal{X}}_t$ of $\tau_t^d$ points
5:     Select arm $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \widetilde{\mathcal{X}}_t} B_t^\epsilon(\boldsymbol{x}, \delta_t)$
6:     Draw sample $z_t \sim p_{\boldsymbol{x}_t}$ and observe payoff $f(z_t)$
7: end for
```

4.4.3 Discretization

When optimization over the infinite arm space $\mathcal{X}$ is not feasible, Algorithm (4.2) can be used instead. This variant restricts the optimization to a progressively finer grid $\widetilde{\mathcal{X}}_t$ of $(\tau_t)^d$ vertices. A reasonably coarse discretization schedule can be used at the price of a worse (but still sublinear) regret:

Theorem 4.4.3. *Let $\mathcal{X}$ be a d-dimensional compact arm set with $\mathcal{X} \subseteq [-D, D]^d$. For any $\kappa \geqslant 2$, under Assumptions (4.4.1) and (4.4.2), Algorithm (4.2) with confidence schedule $\delta_t = \frac{6\delta}{\pi^2 t^2 \left(1 + \lceil t^{1/\kappa} \rceil^d\right)}$ and discretization schedule $\tau_t = \lceil t^{\frac{1}{\kappa}} \rceil$ guarantees, with probability at least $1 - \delta$:*

$$Regret(T) \leqslant \Delta_0 + C_1 T^{\left(1-\frac{1}{\kappa}\right)} d + C_2 T^{\frac{1}{1+\epsilon}} \cdot \left[v_\epsilon \left((2 + d/\kappa) \log T + d \log 2 + \log \frac{\pi^2}{3\delta} \right) \right]^{\frac{\epsilon}{1+\epsilon}},$$

where $C_1 = \frac{\kappa}{\kappa - 1} LD$, $C_2 = (1+\epsilon)\left(2\sqrt{2} + \frac{5}{3}\right) \|f\|_\infty$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

Let us focus on the case $\epsilon = 1$, which is the only one of practical interest in the scope of this paper. For $\kappa = 2$, we obtain regret $\widetilde{\mathcal{O}}(d\sqrt{T})$. Unfortunately, the time required for optimization is exponential in arm space dimensionality d, i.e., $\mathcal{O}(\lceil t^{\frac{1}{\kappa}} \rceil^d)$. For $d \geqslant 2$, we can break the curse of dimensionality by taking $\kappa = d$. In this case, the regret is $\widetilde{\mathcal{O}}\left(dT^{\left(1-\frac{1}{d}\right)}\right)$. On the other hand, the time per iteration is only $\mathcal{O}(t^2)$. Note that the regret is sublinear for any choice of κ. Going further: for any $\zeta > 0$, $\kappa = \frac{d}{\zeta}$ grants $O(t^{1+\zeta})$ time per iteration at the cost of $\widetilde{\mathcal{O}}\left(dT^{\left(1-\frac{\zeta}{d}\right)}\right)$ regret.

Remark 4.4.1. The worse dependency $\widetilde{\mathcal{O}}(d)$ of the regret on the arm space dimensionality (w.r.t. $\widetilde{\mathcal{O}}(\sqrt{d})$ of Algorithm (4.1)) is also necessary to prevent the time per iteration from being exponential in d.

Chapter 5

Numerical Simulations

In this chapter, we start by discussing two practical aspects related to the experimental applications of the algorithms described in Chapter (4). Then, we proceed by showing the results of our numerical simulations on three RL benchmarking challenges: the Linear Quadratic Gaussian regulation (LQG), the Continuous Mountain Car problem and the Inverted Pendulum task.

The programming language chosen for implementing the numerical simulations discussed in this chapter is Python. The code is publicly available on GitHub[1], and it is built upon the OpenAI Baselines library [29].

5.1 Practical Aspects

In the numerical simulations that will be presented in the following sections we place ourselves in the *parameter-based* PS setting (see Section (4.2) for details). We adopt Gaussian distributions as target and behavioural hyperpolicies $\nu_{\boldsymbol{\xi}} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, from which the policy parameters $\boldsymbol{\theta}$ are drawn. In particular, we will adopt hyperpolicies $\nu_{\boldsymbol{\xi}}$, with hyperparameters $\boldsymbol{\xi} = \{\boldsymbol{\theta}, \boldsymbol{\Sigma}\}$, where $\boldsymbol{\Sigma}$ is diagonal:

$$\nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right) \tag{5.1}$$

$$= \frac{1}{\sqrt{(2\pi)^m \prod_{i=1}^m \sigma_i^2}} \exp\left(-\frac{1}{2}\sum_{i=1}^m \frac{(\theta_i-\mu_i)^2}{\sigma_i^2}\right). \tag{5.2}$$

[1]https://github.com/T3p/baselines/tree/exploration/baselines

This allows the use of a deterministic controller $\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m$ for sampling the trajectories, that we define as $\pi_{\boldsymbol{\theta}}(a|s) = \delta(a - \boldsymbol{\theta} s)$. This parameter-based setting follows the one adopted in PGPE, as described in (2.4.4).
This distribution choice for our target and behavioural hyperpolicies allows a comfortable computation of the robust balance heuristic estimator $\check{\mu}_t(\boldsymbol{x})$ defined in Equation (4.8). Unfortunately, that is not enough for the computation of the upper bound $B_t^{\epsilon}(\boldsymbol{x}, \delta_t)$ (4.9) that we need to optimize in each iteration t of OPTIMIST.

5.1.1 Divergence Between Gaussian Multivariate Distributions

Indeed, OPTIMIST also requires to compute the exponentiated Rényi divergence between the target hyperpolicy $p_{\boldsymbol{x}}$ and the mixture Φ_t, i.e., $d_{1+\epsilon}(p_{\boldsymbol{x}} \| \Phi_t) = d_{1+\epsilon}(\nu_{\boldsymbol{\xi}} \| \Phi_t)$, at each iteration. Even for Gaussian distributions, this quantity cannot be obtained in closed form, while the Rényi divergence between Gaussians can be computed exactly. In this section, we provide an upper bound for computing the exponentiated Rényi divergence between a generic distribution and a mixture.

Theorem 5.1.1. *Let P be a probability measure and $\Phi = \sum_{k=1}^{K} \beta_k Q_k$, with $\beta_k \in [0,1]$ and $\sum_{k=1}^{K} \beta_k = 1$, be a finite mixture of the probability measures $\{Q_k\}_{k=1}^{K}$. Then, for any $\alpha \geqslant 1$, the exponentiated α-Rényi divergence can be bounded as:*

$$d_\alpha(P \| \Phi) \leqslant \frac{1}{\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P \| Q_k)}}. \tag{5.3}$$

The proof can be found in Appendix A. We can easily compute this upper bound of the exponentiated Rényi divergence between the target distribution and the mixture of behavioural distributions. In fact, all the hyperpolicies employed are multivariate diagonal Gaussian distributions, and the Rényi divergence between multivariate Gaussian distributions is known [34]. Let $P \sim \mathcal{N}(\boldsymbol{\mu}_P, \boldsymbol{\Sigma}_P)$, $Q \sim \mathcal{N}(\boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_Q)$ and $\alpha \in [0, \infty]$:

$$D_\alpha(P||Q) = \frac{1}{\alpha}(\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q)^T \boldsymbol{\Sigma}_\alpha^{-1} (\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q) - \frac{1}{2(\alpha - 1)} \log \frac{\det(\boldsymbol{\Sigma}_\alpha)}{\det(\boldsymbol{\Sigma}_P)^{1-\alpha} \det(\boldsymbol{\Sigma}_Q)^{\alpha}}, \tag{5.4}$$

where $\mathbf{\Sigma}_\alpha = \alpha\mathbf{\Sigma}_Q + (1-\alpha)\mathbf{\Sigma}_P$ under the assumption that $\mathbf{\Sigma}_\alpha$ is positive-definite.

5.1.2 Uniformly Bounded Rényi divergence

The other concern about the exponentiated Rényi divergence between the target and the mixture of behavioural hyperpolicies is to make it compliant with Assumption (4.4.1), i.e., uniformly bounded. Without this assumption, the results on OPTIMIST regret (Theorems (4.4.1),(4.4.2) and (4.4.3)) are no more guaranteed. This assumption can be easily respected by careful hyperpolicy design. First, note that the results on the regret are provided for a compact continuous (or finite discrete) arm set, hence the maximum distance among the parameters is bounded. Additionally, we must ensure that the Rényi divergence is bounded. As showed in Theorem (5.1.1), it is enough that the divergence is finite between the target and one of the components of the mixture to guarantee a bound on the divergence between a target distribution and a mixture of behavioural distributions. Hence, we will focus on the constraints between behavioural/target pairs. As an example, for multivariate diagonal Gaussian distributions with fixed covariance Assumption (4.4.1) is easily guaranteed. In fact, the Rényi divergence is a continuous function of the mean parameter [34] and a continuous function on a compact set is bounded. If the standard deviation (or covariance matrix) is also part of the parameter set, additional constraints are needed, as one can understand by examining Equation (5.4). For $\epsilon = 1$, the standard deviation σ_P of the target distribution must not be larger than twice that of the behavioural (σ_Q) for the divergence to be finite. Hence, given a minimum $\sigma_0 > 0$, it is enough to constrain the search within $[\sigma_0, 2\cdot\sigma_0]$. We also suggest initializing the first behavioural distribution with $\sigma_Q = 2\cdot\sigma_0$, so that the algorithm will move towards smaller standard deviations. This will result in a less stochastic behaviour. Similar constraints can be defined for other kinds of policies [34].

5.2 Linear Quadratic Gaussian Regulator

The goal of our numerical simulations on LQG is twofold. First, we need a simple continuous control problem to understand the functioning of our algorithms. Second, we want to compare OPTIMIST (Algorithm (4.1)) with two classical MAB algorithms presented in Chapter (3): UCB1 (3.1.2) and GPUCB (3.4), in

the case of discrete parameter space Ξ.
The LQG problem [68] is a continuous state-action space MDP that represents a useful testing ground for control algorithms, mainly because of its simplicity. At each time-step h, the transition kernel and reward function are given by:

$$\boldsymbol{s}_{h+1} = \boldsymbol{A}\boldsymbol{s}_h + \boldsymbol{B}\boldsymbol{a}_h + \boldsymbol{\eta}_h \tag{5.5}$$

$$r_h = \boldsymbol{s}_h^T\boldsymbol{Q}\boldsymbol{s}_h + \boldsymbol{a}_h^T\boldsymbol{R}\boldsymbol{a}_h \tag{5.6}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are coefficient matrices and $\boldsymbol{\eta}_h$ is a noise process assumed to be a Gaussian white noise $\boldsymbol{\eta}_h \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{LQG})$ with uncorrelated components $\boldsymbol{\Sigma}_{LQG} = \sigma_{LQG}\boldsymbol{I}$. The reward r_h has to be intended as a cost for the agent, something it wants to avoid. Intuitively, in this problem the agent has to bring its state to zero, while facing a cost proportional to the magnitude of its state and action. The optimal control policy in steady state conditions is the linear controller $\boldsymbol{a}_h = \boldsymbol{K}\boldsymbol{s}_h$, where matrix $\boldsymbol{K}$ can be found by solving a Riccati equation [30]. We conducted our experiments on one-dimensional LQG. For implementation reasons, we consider the case in which the state space is limited to $\mathcal{S} = [-4, 4]$, the action space is $\mathcal{A} = [-4, 4]$ and the horizon is limited to $H = 20$. At the beginning of the episode, the start-state is initialized randomly $s_0 \sim \mathcal{U}([-4, 4])$. The agent samples $H = 20$ steps-long trajectories with a discount factor of $\gamma = 0.99$, for a total of $T = 5000$ iterations.

As mentioned in the previous section, in all our experiments we employ diagonal Gaussian hyperpolicies. In this case the hyperpolicy is univariate $\nu_{\boldsymbol{\xi}} = \mathcal{N}(\mu, \sigma^2)$, where μ is the mean parameter to be learned and σ can be either fixed or learnable as well. On the other hand, the policy π_θ adopted by the agent is a deterministic linear controller $a_h = \pi_\theta(s_h) = \theta \cdot s_h$, with $\theta \sim \nu_{\boldsymbol{\xi}}$. Since UCB1 requires a positive return $r_t \in [0, 1]$, we standardized the trajectory return $\mathcal{R}(\boldsymbol{\xi}_t) = \sum_{h=0}^{H-1} \gamma^h r_{h+1}$ associated to the arm ξ_t pulled at iteration t:

$$r_t = \mathcal{R}(\boldsymbol{\xi}_t)/\mathcal{R}_{max}, \tag{5.7}$$

where $\mathcal{R}_{max}$ is the maximum achievable cumulated return:

$$\mathcal{R}_{max} = \sum_{h=0}^{H-1} \gamma^h \cdot 1 = \frac{1-\gamma^H}{1-\gamma}. \tag{5.8}$$

a	b	q	r	γ	H	T	σ_{LQG}	δ
1.00	1.00	0.90	0.90	0.99	20	5000	0.10	0.20

Table 5.1: Environmental coefficients (left-side), task coefficients (center) and OPTIMIST input parameters (right-side) for the LQG experiments.

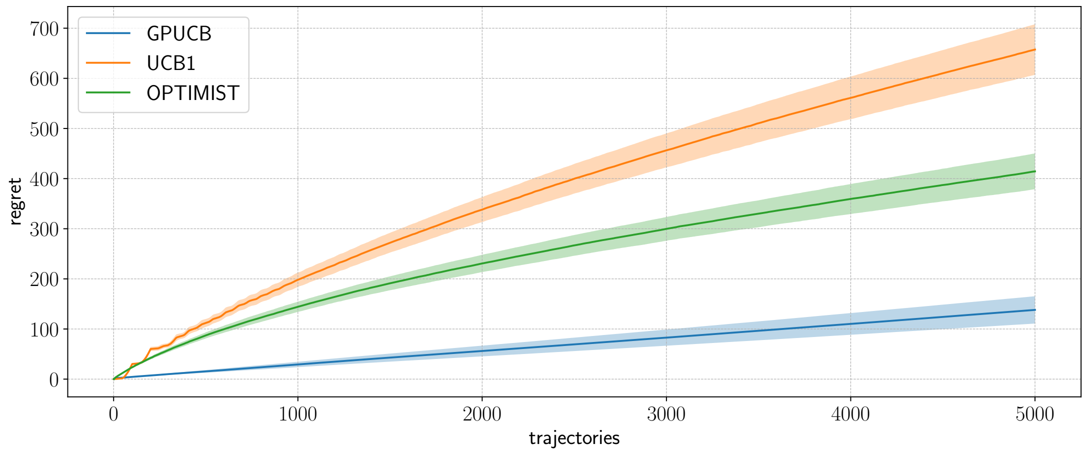

Figure 5.1: Cumulative regret in the LQG experiment. Comparison between OPTIMIST, UCB1 and GPUCB when learning the hyperpolicy mean. (30 runs, 95% c.i.)

All algorithms are run with the confidence schedule proposed in Theorem (4.4.1), i.e., $\delta_t = \frac{3\delta}{t^2\pi^2 K}$, with $\delta = 0.2$ (similar results have been obtained with different values of δ). The parameters used in the LQG experiments are summarized in Table (5.1).

5.2.1 Gain only

In order to benchmark OPTIMIST with both UCB1 and GPUCB on a discrete set, we first consider the case in which the only learnable parameter of $\nu_{\boldsymbol{\xi}}$ is the gain μ. To this end, we consider a uniform discretization of the interval $[-1, 1]$ made of 100 arms (every arm is a possible choice of μ). The hyperpolicy standard deviation is fixed to $\sigma = 0.15$. We experimented with different values of σ and this turned out to be a good choice for making the task noisy, but not too noisy to prevent learning.

In Figure (5.1), we show the cumulative regret of the three algorithms, averaged over 30 runs. We can see that our algorithm significantly outperforms UCB1.

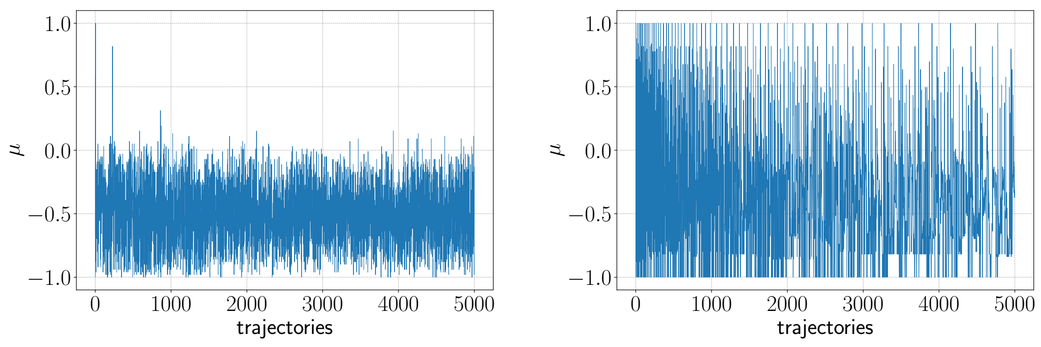

Figure 5.2: The gain parameter μ selected at each iteration of GPUCB (left) and OPTIMIST (right) in the LQG experiment.

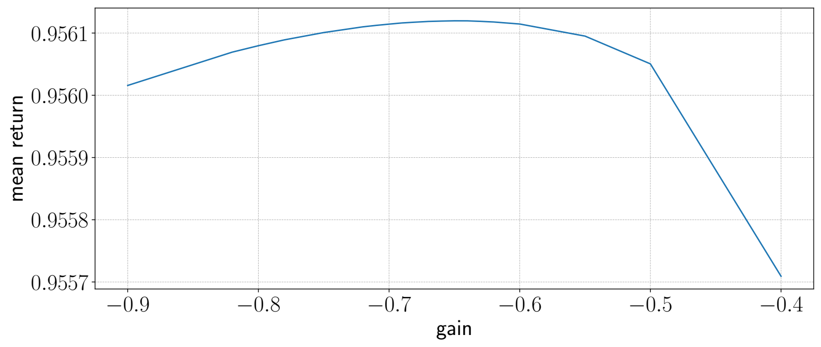

Figure 5.3: Mean return of arms $\mu \in [-0.9, -0, 5]$, calculated by averaging the return collected over 2000 trajectories in the LQG experiment.

Indeed, OPTIMIST is able to exploit the structure of arms, i.e., hyperpolicies, by means of the MIS estimation, whereas UCB1 does not make any assumption on arm correlation. In other words, OPTIMIST performs a more informed (directed) exploration, leveraging what the agent has experienced in past episodes more effectively. On the contrary, GPUCB shows a better performance w.r.t. to OPTIMIST. We point out that GPUCB requires to specify, at the beginning of learning, the kernel of the Gaussian process from which the payoff function is sampled. We employed the default scikit-learn[2] kernel, i.e., the radial basis function kernel:

$$k_T(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\lambda \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right), \tag{5.9}$$

where λ is a free parameter. However, as it often happens in control tasks,

[2]A popular library for data mining and data analysis. Available at: https://scikit-learn.org/stable/

γ	H	T	δ	k
1.00	500	5000	0.20	3

Table 5.2: Task parameters (left side) and OPTIMIST input parameters (right side) for the Continuous Mountain Car experiment.

our payoff is not actually sampled from a Gaussian process. This invalidates all theoretical guarantees of GPUCB and may be at the root of its strong commitment to exploitation, contrary to UCB1 and OPTIMIST. We can visualize the amount of exploration carried out by the three algorithms by looking at Figure (5.2), which depicts, the arm μ pulled by GPUCB and OPTIMIST at every iteration. Indeed, OPTIMIST explores the set of arms around the optimum much more extensively then GPUCB, which, in the very first steps, commits to a near-optimal set of arms (the neighbourhood of $\mu = -0.5$ interval) and sticks to it all along. This explains the lower regret of GPUCB w.r.t. OPTIMIST. In fact, the LQG task presents a pretty wide set of optimal or near-optimal arms spanning in $[-0.9, -0.5]$, as shown in Figure (5.3). Therefore, exploitation is a rewarding strategy in this setting.

5.2.2 Gain and standard deviation

In the second experiment on LQG, we learn both the mean and the variance parameter of the Gaussian hyperpolicy: $\nu_{\boldsymbol{\xi}} = \mathcal{N}(\mu, \exp(2\rho))$, where $\boldsymbol{\xi} = (\mu, \rho)^T = (\xi_1, \xi_2)^T$. The parameters used in the experiment are the same as before, reported in Table (5.1). In Figure (5.4), we show the cumulative regret averaged over 5 runs comparing OPTIMIST, UCB1 and GPUCB. We see a trend similar to the case in which we learn the mean parameter only. While OPTIMIST is able to exploit the structure of the arms induced by the fact that hyperpolicies share information, beating UCB1, GPUCB still displays a better performance. The wider confidence intervals are a direct consequence of the wider spectrum of variances adopted by the hyperpolicy, bringing to very different choices of policy parameters and, subsequently, very difference performance from one run to another.

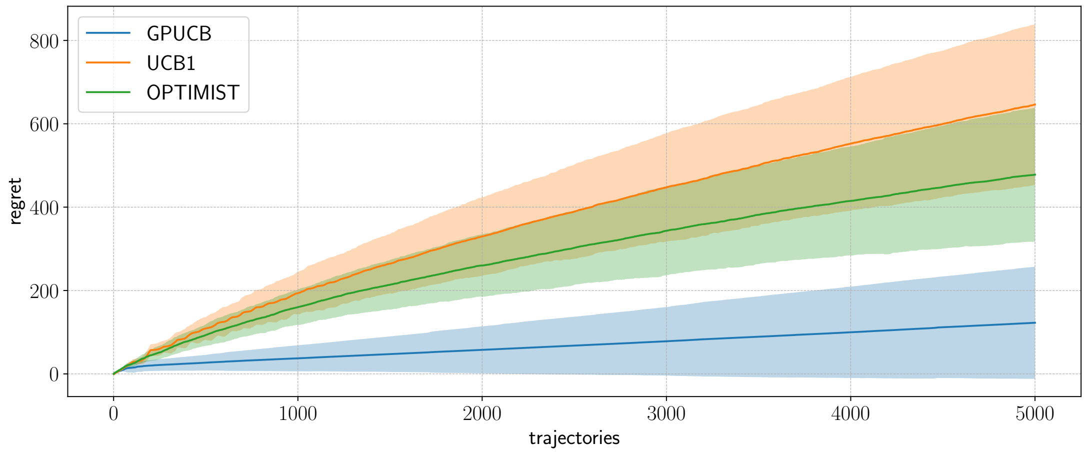

Figure 5.4: Cumulative regret in the LQG experiment, comparing OPTIMIST, UCB1 and GPUCB when learning both the mean and the standard deviation hyperparameters. (30 runs, 95% c.i.)

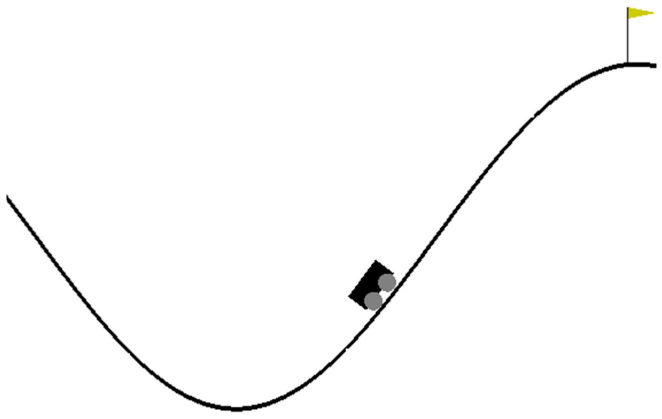

Figure 5.5: Graphical representation of the Mountain Car problem [16].

5.3 Continuous Mountain Car

The second experiment, illustrates the behaviour of OPTIMIST2 when the parameters of the hyperpolicy belong to a compact (continuous) space, on the Continuous Mountain Car task [17]. We chose the Continuous Mountain Car because it is a simple, well known, continuous problem and because it constitutes a relevant exploration challenge w.r.t. other simple tasks such as LQG.

In this problem, graphically represented in Figure (5.5), the agent has to control the engine of an under-powered car in order to reach a target. The target is on top of a hill on the right-hand side of the car. If the car reaches it or goes beyond, the episode terminates. On the left-hand side, there is another hill.

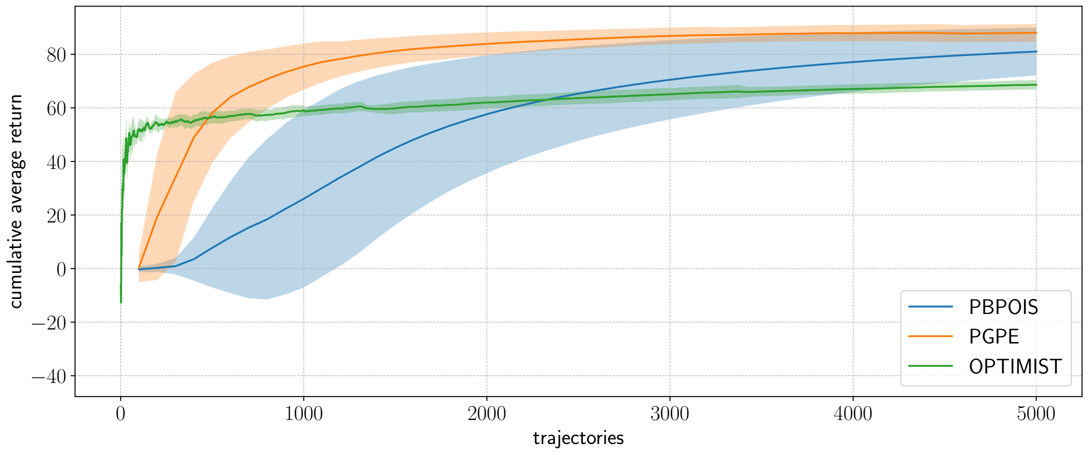

Figure 5.6: Cumulative regret in the Continuous Mountain Car experiment. Comparison between OPTIMIST, PGPE and Parameter-Based Policy Optimization via Importance Sampling (PBPOIS) when learning the two-dimensional hyperpolicy mean. (5 runs, 95% c.i.)

Climbing this hill can be used to gain potential energy and accelerate towards the target. On top of this second hill, the car cannot go further than a position equal to -1, as if there was a wall. Reward is 100 for reaching the target of the hill on the right hand side, minus the squared sum of actions from start to goal. This reward function raises an exploration challenge, because if the agent does not reach the target soon enough, it will figure out that it is better not to move, and won't find the target eventually. The state space $\mathcal{S} = [-1.20, 0.60] \times [-0.07, 0.07]$ is constituted by a bi-dimensional vector with current position and velocity of the car. The action space $\mathcal{A} \in \mathbb{R}$ is continuous: positive values correspond to a forward engine traction, negative values to backward engine traction. Every episode starts with the car in a random position between -0.6 and -0.4, with null velocity. The agent samples $H = 500$ steps-long trajectories with a discount factor of $\gamma = 1.00$ (the rationale of this choice is discussed in Remark (5.3.1)), for a total of $T = 5000$ iterations.

In our experiments we use a Gaussian hyperpolicy with a two-dimensional learnable mean $\boldsymbol{\mu} = \boldsymbol{\xi} = (\xi_1, \xi_2)^T$, within a box $\boldsymbol{\mu} \in [-1, -1] \times [0, 20]$, and a fixed covariance $\boldsymbol{\Sigma} = \mathrm{diag}(0.15^2, 3^2)$. Concerning the confidence and discretization schedules for OPTIMIST2, we adopted those suggested in Theorem (4.4.3), i.e., $\delta_t = \frac{6\delta}{\pi^2 t^2 \left(1 + \lceil t^{1/\kappa} \rceil^d\right)}$ and $\tau_t = \lceil t^{\frac{1}{\kappa}} \rceil$, with $\delta = 0.2$ and $k = 3$. The choice of k influences the granularity of the discretization. Since $\tau_t = \lceil t^{\frac{1}{\kappa}} \rceil$ the granularity is non-decreasing with time, but smaller values of k allow for a finer granularity at a given iteration t. For more information about the choice of k, refer to Remark (5.3.2). The parameters used in the Mountain Car experiments are summarized in Table (5.1).

We compare OPTIMIST2 (4.2) against parameter-based policy optimization algorithms PGPE (2.4) and PBPOIS [57]. The latter is a parameter-based off-policy policy search algorithm which optimizes a lower bound of the performance estimator. Similarly to OPTIMIST, this estimator is built upon single importance sampling. We chose to compare OPTIMIST with PBPOIS not only because of this similarity, but also because of their intrinsic difference. In fact, despite being both off-policy algorithms based on importance sampling, PBPOIS optimizes a lower bound on the performance, which implies a very limited exploration. By contrast, OPTIMIST optimizes an upper bound on performance, accordingly to the OFU principle. OPTIMIST and PBPOIS are also compared with PGPE, which laid the foundations for both of them. Moreover, PGPE too is known to lack exploration of the parameter space. Finally, note that, unlike OPTIMIST, both PGPE and PBPOIS adopt natural policy gradient ascent to optimize their hyperpolicies. The best learning rate α for PGPE was searched in the set $\{3, 2, 1, 0.1, 0.01, 0.001\}$. For PBPOIS, we used the suggested hyperparameters.

Results are shown in Figure (5.6). We can notice that OPTIMIST2 is able to learn a good policy in a very short time thanks to its better exploration capabilities. However, the policy gradient methods outperform it on the long run. In fact, the exploitative behaviour of the other two algorithms results in a better performance w.r.t. OPTIMIST, which continues to explore even after having found suitable parameters in few iterations. Even though these results might not be satisfactory from a performance point of view, our primary goal was to develop a PS algorithm that would explore effectively the space of parameters. From this point of view, the results turn out to be successful. To convince ourselves of this claim, we refer to Figure (5.7), in which we show the density estimation of the probability distribution induced over the arm set $\boldsymbol{\xi} = \boldsymbol{\mu}$ by PGPE (left) and OPTIMIST (right) during learning. The estimation has been carried out by means of the Gaussian kernel density estimator [77] implemented in the SciPy library.[3] Intuitively, this represents the likelihood of each two-dimensional arm to be pulled by either PGPE or OPTIMIST.

[3]https://docs.scipy.org/doc/

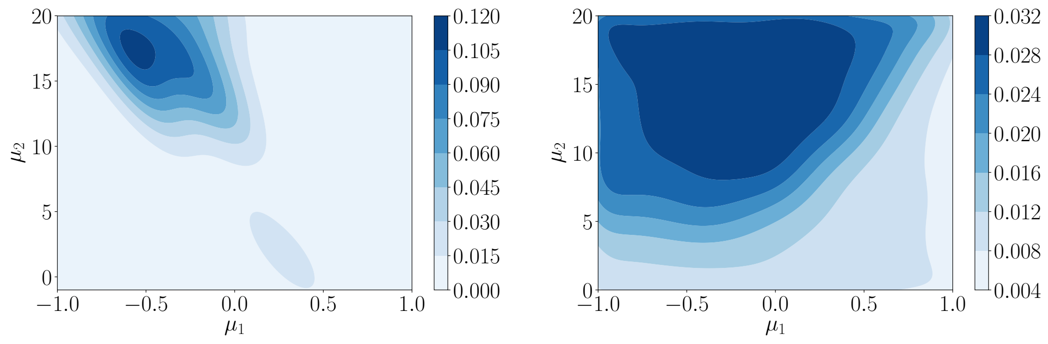

Figure 5.7: Gaussian kernel density estimation [77] of the probability distribution of the arm set $\boldsymbol{\xi} = \boldsymbol{\mu}$ induced by PGPE (left) and OPTIMIST (right).

State-space dimension	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
$d = 2$	5041	324	81	36	25
$d = 4$	2541168	104976	6561	1296	625

Table 5.3: Number of points in the discretized state space, following discretization schedule $\tau_t = \lceil t^{\frac{1}{\kappa}} \rceil$, where d refers to the number of space dimensions and k is a free-parameter.

Remark 5.3.1. The rationale behind the choice of undiscounted ($\gamma = 1.00$) rewards lies in the design of the reward system of the Continuous Mountain Car task. During learning, the car usually does not manage to reach the target before a few hundred iterations, while collecting rewards equal to minus the squared sum of actions in all previous steps. In such scenario, the agent would barely distinguish between a successful episode (that ended by reaching the target) and an unsuccessful one. For the sake of clarity, we will illustrate this situation with an example. Say that, after driving up and down the hills for 300 episodes, the agent finally manages to reach the target. Also, imagine that we are slightly discounting rewards with $\gamma = 0.99$. In this situation, the positive reward received would be $r_{301} = 0.99^{300} \cdot 100.00 = 4.90$, while the negative reward cumulated in previous steps ($h = 1, 2, \ldots, 299$) would at least be numbered in tens. This means that the return of the episode would still be negative and numbered in tens, because $|r_{301}| \ll |\sum_{h=0}^{299} \gamma^h r_{h+1}|$. Hence, the agent would not learn from this successful episode, because it could not tell it from an unsuccessful episode. This problem is solved by setting $\gamma = 1$.

Remark 5.3.2. Considering that OPTIMIST needs to evaluate bound (4.9) for every arm, at each iteration, the finer the discretization schedule the longer the

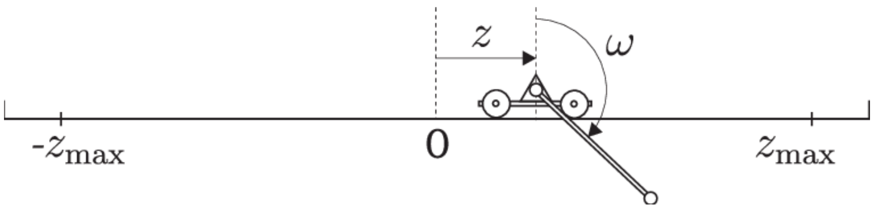

Figure 5.8: Graphical representation of the Inverted Pendulum task [94].

γ	H	T	δ	k
0.99	500	10000	0.20	5

Table 5.4: Task parameters (left side) and OPTIMIST input parameters (right side) for the Inverted Pendulum experiment.

time required by numerical simulation. Indeed, the optimization of the bound is much more computationally demanding than any other operation undertaken by OPTIMIST. In Table (5.3) we report the total number of arms resulting from different choices of k, according to the state-space dimension d, at final time step $T = 5000$. As a reference, with the computing capacity at our disposal during this thesis project [4], the time needed for optimizing the bound over 5000 iterations was:

- approximately 4 hours with $d = 2$ and $k = 3$;
- approximately 39 hours with $d = 4$ and $k = 4$.

Evidently, our limited time and computing capacity did not allow for the finest of the discretization schedules. However, we think that the discretization adopted in the Mountain Car and Inverted Pendulum (discussed next) experiments was appropriate to the task at hand.

5.4 Inverted Pendulum

As third comparative experiment, we attempted to run OPTIMIST on the Cart-Pole Swing-Up task [91], also referred to as Inverted Pendulum task. Unfortunately, in this case OPTIMIST was not able to learn any performant policy, neither optimal nor near-optimal. However, we present the experiment and some

[4]20 cores of an Intel Xeon Processor E7-8880 v4 (55M Cache, 2.20 GHz, 126 GB of memory).

considerations on what could have gone wrong as a starting point for future developments. This problem is a classic benchmark for non-linear control[5], which requires a thorough exploration for finding the optimal policy. The system consists of a pole attached to a cart, depicted in Figure 5.8). The force applied to the cart can be controlled, and the goal is to swing the pole to an upward position and stabilise it. This must be accomplished without the cart crashing into the walls of the track. The state space $\mathcal{S} \in \mathbb{R}^4$ consists of four observed variables. The position of the cart u (constrained in $[-3, 3]$), the angle of the pole measured from the upward position ϕ, and their first derivatives u' and ϕ'. Control input is the force applied to the cart. The reward system is designed in a way that the agent receives a non-positive reward except when it manages to achieve an upward position:

$$\mathcal{R}(\boldsymbol{a}_t|\boldsymbol{s}_t) = \begin{cases} -100, & \text{if } |u_t| > 3 \\ \cos(\phi) & \text{if } |u_t| \leqslant 3 \end{cases} \tag{5.10}$$

This task is known for requiring a thorough exploration of the parameter space because it is easy to encounter local optima which are far from the global optimum in terms of performance. An example would be a policy that continuously rotates the pole in order to receive a close-to-zero return, spending half of the time downwards and half upwards. The detailed dynamics and constraints for the simulated cart-pole system can be found in [44], while for the implementation details (including the bounds applied to the position and the force) the reader can refer to the *rllab* implementation[6]. Before every trajectory, the system is initialized to a random state taken from the uniform distributions $u \in [-1, 1]$, $u' \in [-2, 2]$, $\phi \in [\pi - 1, \pi + 1]$, $\phi' \in [-3, +3]$. The agent samples $H = 500$ steps-long trajectories with a discount factor of $\gamma = 0.99$, for a total of $T = 1000$ iterations.

In our experiments we used a Gaussian hyperpolicy with a four-dimensional learnable mean $\boldsymbol{\mu} = \{\mu_1, \mu_2, \mu_3, \mu_4\}$, within a box $\boldsymbol{\mu} \in [-0.2, 0.2]^4$, and a fixed covariance $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$, with $\sigma = 0.001$. Concerning the confidence and discretization schedules for OPTIMIST2, we adopted those suggested in Theorem (4.4.3), i.e., $\delta_t = \frac{6\delta}{\pi^2 t^2 \left(1 + \lceil t^{1/\kappa} \rceil^d\right)}$ and $\tau_t = \lceil t^{\frac{1}{\kappa}} \rceil$, with $\delta = 0.2$ and $k = 4$. The summary of the parameters used in the Inverted Pendulum experiments is reported in Table (5.4).

[5]A broad benchmarking study on Inverted Pendulum is discussed in [31].

[6]https://github.com/rll/rllab

Unfortunately, OPTIMIST2 failed to learn a good policy in this task. This is clearly visible by looking at the hyperparameters curve over decision epochs: even after 10000 iterations the algorithm is still exploring (more or less) uniformly the parameter space. We report the curve of μ_1 in Figure (5.9). The reason for this behaviour is likely to be σ. Indeed, for very small values of sigma, such as $\sigma = 0.001$, the behavioural hyperpolicies adopted during learning do not share information with the target hyperpolicy. In other words, the Rényi between the target and the mixture of behaviourals becomes very large and dominates the OPTIMIST bound (4.9). We report the bound here for the sake of clarity:

$$B_t^1(\boldsymbol{x}, \delta_t) := \check{\mu}_t(\boldsymbol{x}) + \|f\|_\infty \left(\sqrt{2} + \frac{4}{3}\right) \left(\frac{d_2(p_{\boldsymbol{x}_t} \| \Phi_t) \log \frac{1}{\delta_t}}{t}\right)^{\frac{1}{2}}. \tag{5.11}$$

Evidently, when the Rényi divergence is very high the second half (the *exploration bonus*) dominates the bound, and obliges the agent to explore disregarding the estimated performances of the arms. This intuition is confirmed by looking at the very irregular plots of the truncated MIS estimator $\check{\mu}_t(\boldsymbol{x}_t)$ and the exploration bonus $\|f\|_\infty \left(\sqrt{2} + \frac{4}{3}\right) \left(\frac{d_2(p_{\boldsymbol{x}_t} \| \Phi_t) \log \frac{1}{\delta_t}}{t}\right)^{\frac{1}{2}}$ for the arms $\boldsymbol{x}_t$ selected at iterations $t = 1, 2, \dots, T$, reported in Figure (5.10). The bonus (whose value is frequently over 10000) largely dominates the performance estimator (whose value is mostly negative). The intuitive counter-measure would be to adopt bigger values of σ. Unfortunately, the Inverted Pendulum has a very narrow set of near optimal arms and higher variances hinder learning, as confirmed by our numerical simulations. Indeed, this knot could represent another interesting starting point for further developments.

5.5 Action-based setting

The reader may wonder why we did not carry out numerical simulations in the *action-based* setting. Actually, we did make some experiments, but we briefly understood that there was a major obstacle represented by the computation of the Rényi divergence. In fact, although what has been discussed in Section (5.1) also applies to policies (not only hyperpolicies), the loss function cannot be directly optimized since computing $d_{1+\epsilon}(p_{\boldsymbol{x}} \| \Phi_t)$ requires the approximation of an integral over the trajectory space and, for stochastic environments, to know

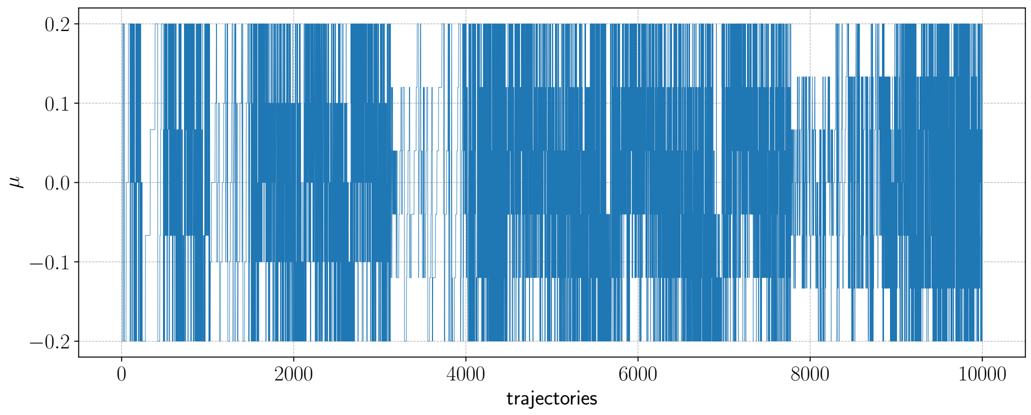

Figure 5.9: The hyperpolicy mean parameter selected at each iteration of OPTIMIST in the Inverted Pendulum experiment.

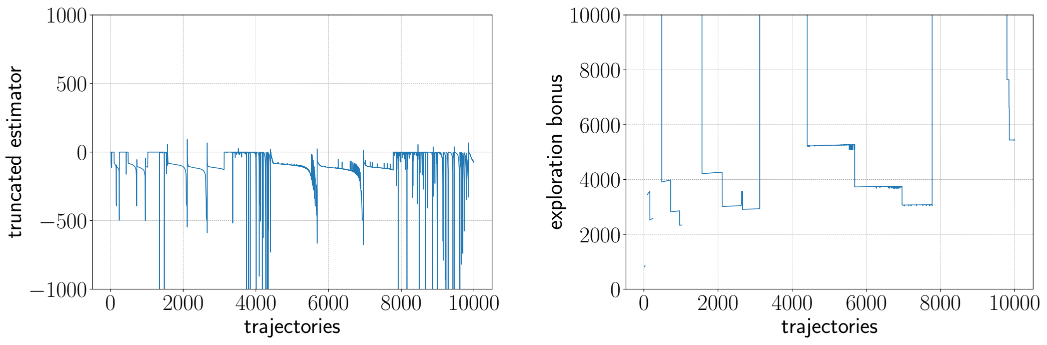

Figure 5.10: Truncated MIS estimator (left) and exploration bonus (right) of the arms selected by OPTIMIST at each iteration of the Inverted Pendulum experiment.

the transition model $\mathcal{P}$, which is unknown in a model free setting. For the sake of clarity, we report here the definition of the Rényi divergence between the distributions $p_{\boldsymbol{\theta}'}(\tau)$ and $p_{\boldsymbol{\theta}}(\tau)$, induced over trajectories $\tau \in \mathcal{T}$ by the target $\pi_{\boldsymbol{\theta}'}$ and behavioural $\pi_{\boldsymbol{\theta}}$:

$$d_\alpha(p_{\boldsymbol{\theta}'}||p_{\boldsymbol{\theta}}) = \left(\int_{\mathcal{T}} p_{\boldsymbol{\theta}}(\tau) \left(\frac{p_{\boldsymbol{\theta}'}(\tau)}{p_{\boldsymbol{\theta}}(\tau)} \right)^\alpha \mathrm{d}\tau \right)^{\frac{1}{\alpha-1}}. \tag{5.12}$$

Simple bounds to this quantity, like $d_\alpha(p_{\boldsymbol{\theta}'}||p_{\boldsymbol{\theta}}) \leqslant \sup_{s\in\mathcal{S}} d_\alpha(\pi_{\boldsymbol{\theta}'}(\cdot|s)||\pi_{\boldsymbol{\theta}}(\cdot|s))^H$, besides being hard to compute due to the presence of the supremum, are extremely conservative since the Rényi divergence is raised to the horizon H. In our experiments, we tested action-based OPTIMIST over the LQG problem by estimating the exponentiated 2-Rényi divergence between a proposal Q and a target P distribution as follows:

$$\hat{d}_2(P|Q) = \frac{t}{\widehat{ESS}}, \tag{5.13}$$

where t is the number of samples available, which in our case corresponds to the iteration step number, $w_{P/Q}$ is the importance weight, and $\widehat{ESS}$ [56] is a well known estimator of the *effective sample size* (ESS) [49], given by:

$$\widehat{ESS} = \frac{\left\|w_{P/Q}\right\|_1^2}{\left\|w_{P/Q}\right\|_2^2}. \tag{5.14}$$

Indeed, the effective sample size is defined as [49]:

$$ESS = \frac{t}{\mathbb{V}\mathrm{ar}_{x\sim Q}[w_{P/Q}(x)] + 1} = \frac{t}{d_2(P|Q)}. \tag{5.15}$$

However, its estimator $\widehat{ESS}$ is very rough due to its high (possibly infinite) variance, and turned out to be useless in our case. The main problem of $\hat{d}_2(P|Q)$ (5.13) is that it is consistent with the true Rényi divergence only for close enough target and behavioural policies (P and Q). This means that the estimator does not allow to share the information obtained by a behavioural Q with distant target distributions P. Moreover, by estimating the Rényi from the samples we would lose OPTIMIST theoretical guarantees on the regret. Therefore, this estimator proved to be impractical for the implementation of action-based OPTIMIST and we decided to focus on the parameter-based setting. For future works, a study of a good estimator for the Rényi would be another interesting research direction.

Chapter 6

Conclusions

In this chapter, we take the time to revisit our original contributions to the reinforcement learning literature. Then, we recapitulate the limitations of our work and suggest some possible axis for future research that stem from these limitations.

6.1 Recapitulation

In this thesis we studied the exploration-exploitation problem in PS by leveraging MAB techniques. After a thorough literature review of both MAB and RL, we developed a threefold contribution to the existing work.
The first contribution is of pure theoretical nature: we provided an ad-hoc formalization of the online, episodic policy optimization problem. This formulation does not fall within the traditional MAB framework, but builds upon it. At every decision epoch, the agent selects a parametrization of its policy (or hyperpolicy), as it would pull a bandit arm, draws a trajectory with it, and observes its payoff, i.e., the cumulative return of the trajectory. The goal is to maximize the expected total payoff. The peculiarity of this framework w.r.t. the classic MAB one, is the special structure existing over the arms, induced by the common sample space of the stochastic arms (the agent's policy parametrizations).
The second contribution is both algorithmic and theoretical: we exploited the correlation between arms to guarantee efficient exploration by leveraging the OFU principle and multiple importance sampling. In particular, we devised OPTIMIST, a suitable algorithm capable for learning within a discrete set of arms. An intuitive extension of it, OPTIMIST2 allows learning within a compact

set of arms, by means of iterative discretization of the continuous space. Both algorithms have been backed by theoretical guarantees on the convergence of their cumulative regret, i.e., sublinear regret, under assumptions that are easy to met in practice. Also, they have no equivalents in the PS literature, which strongly lacks studies on both exploration and convergence properties of the algorithms.
The third contribution is mostly experimental: we carried out several numerical simulations to test the algorithms proposed on multiple tasks, and to benchmark them with other classical MAB and RL algorithms. As we hoped, OPTIMIST (and its extension) proved to be very effective in exploring the (hyper)parameter space by leveraging its structure. In particular, its exploitative behaviour revealed to be more efficient than similar MAB algorithms such as UCB1, and more effective of policy search algorithms such as PGPE and PBPOIS. Nonetheless, numerical simulations shed light on several limitations of the algorithms proposed.

6.2 Limitations and future works

The most evident limitation of OPTIMIST, revealed by numerical simulations, consists in the inefficient way it optimizes its upper bound index (4.9). Taking the *argmax* over a discrete set is very expensive and becomes unfeasible for large, multidimensional state spaces. This problem showed up clearly in the Mountain Car and Inverted Pendulum experiments. Given our limited computing capacity and time, we could neither experiment with finer discretization schedules (smaller k), nor choose to optimize the standard deviation too, which would have meant to double the number of arms. Another limitation has been already discussed in Section 5.5. Computing the exploration bonus is unfeasible because we lack proper estimators for the Rényi divergence between a target and a mixture policy. To the best of our knowledge, the estimators available in the literature are insufficient for modelling the distance between probability distributions as the Rényi does. Particularly, when the behavioural is a mixture distribution and it is far away from the target. Finally, comparative experiments on LQG and Continuous Mountain Car showed that OPTIMIST struggles to catch up with the performances of more exploitative algorithms whenever an extensive exploration is not required. In particular, after having performed many epochs and extensively explored the parameter space, it would be desirable to start exploiting more heavily. However (we should say, as expected), OPTIMIST consistently sticks to the OFU principle, which inherently brings to continuous exploration even in tasks in which little exploration is enough.

The limitations discussed above constitute a natural starting point for future developments. Future works should focus on finding more efficient ways to perform optimization in the infinite-arm setting. The authors of GPUCB, which faces a similar optimization problem, suggest the adoption of global search heuristics. Instead, the RL community generally favours gradient descent. However, this latter is known for performing badly in optimizing multimodal functions, like our bound. Nonetheless, there exist many different variations of this technique (natural gradient, stein variational gradient, gradient with momentum etc.), and some of them could reveal effective. On an other track, parallel to the first one, future work could investigate appropriate estimators for the exploration bonus in the action-based setting. Also, they could study how to mitigate OPTIMIST strong exploration drive whenever it proves to be excessive. Finally, an interesting line of research would be to study suitable integrations of our problem formalization and bounds with posterior sampling, as described in 3.1.5. Indeed, exploiting the sample efficiency that characterizes TS seems promising, and already proved to be a more powerful solution than optimism in some RL scenarios, as discussed in 3.2.3.

EXPLAIN CLEVER CASHING? Unfortunately, the time required for optimization is exponential in arm space dimensionality=====spendere qualche parola per dire come mai perchè la complessità è quella presentata? bigger labels 5.3

Appendix A

Proofs

Lemma 2.5.1. *Let P and $\{Q_k\}_{k=1}^K$ be probability measures on the measurable space $(\mathcal{Z}, \mathcal{F})$ such that $P \ll Q_k$ and $d_2(P\|Q_k) < \infty$ for $k = 1, \dots, K$. Let $f : \mathcal{Z} \to \mathbb{R}$ be a bounded function, i.e., $\|f\|_\infty < \infty$. Let $\widehat{\mu}_{BH}$ be the balance heuristic estimator of f, as defined in (2.40), using N_k i.i.d. samples from each Q_k. Then, the variance of $\widehat{\mu}_{BH}$ can be upper bounded as:*

$$\underset{z_{ik} \overset{iid}{\sim} Q_k}{\mathbb{V}\mathrm{ar}} [\widehat{\mu}_{BH}] \leqslant \|f\|_\infty^2 \frac{d_2(P\|\Phi)}{N}, \tag{2.44}$$

where $N = \sum_{k=1}^K N_k$ is the total number of samples and $\Phi = \sum_{k=1}^K \frac{N_k}{N} Q_k$ is a finite mixture.

Proof. The proof is similar to Lemma 4.1 of [57]:

$$\begin{aligned}
\underset{z_{ik} \overset{\text{iid}}{\sim} Q_k}{\mathbb{V}\mathrm{ar}} [\widehat{\mu}_{\text{BH}}] &= \underset{z_{ik} \overset{\text{iid}}{\sim} Q_k}{\mathbb{V}\mathrm{ar}} \left[\frac{1}{N} \sum_{k=1}^K \sum_{i=1}^{N_k} f(z_{ki}) \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})} \right] \\
&= \frac{1}{N^2} \sum_{k=1}^K \sum_{i=1}^{N_k} \underset{z_{ik} \sim Q_k}{\mathbb{V}\mathrm{ar}} \left[f(z_{ki}) \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})} \right] \qquad (A.1) \\
&\leqslant \frac{1}{N^2} \sum_{k=1}^K \sum_{i=1}^{N_k} \underset{z_{ik} \sim Q_k}{\mathbb{E}} \left[\left(f(z_{ki}) \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})} \right)^2 \right] \\
&\leqslant \|f\|_\infty \frac{1}{N^2} \sum_{k=1}^K \sum_{i=1}^{N_k} \underset{z_{ik} \sim Q_k}{\mathbb{E}} \left[\left(\frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})} \right)^2 \right] \\
&= \|f\|_\infty^2 \frac{1}{N} \underset{z \sim \Phi}{\mathbb{E}} \left[\left(\frac{p(z)}{\sum_{j=1}^n \frac{N_j}{N} q_j(z)} \right)^2 \right] \qquad (A.2)
\end{aligned}$$

$$= \|f\|_\infty^2 \frac{d_2(P\|\Phi)}{N},$$

where (A.1) follows from the independence of the z_{ik} and (A.2) is obtained by the definition of Φ and observing that for an arbitrary function g:

$$\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathop{\mathbb{E}}_{z_{ik}\sim Q_k}[g(z_{ik})] = \sum_{k=1}^{K} \frac{N_k}{N} \mathop{\mathbb{E}}_{z_{1k}\sim Q_k}[g(z_{1k})] = \mathop{\mathbb{E}}_{z\sim\Phi}[g(z)]. \tag{A.3}$$

□

Lemma 4.1.1. *Let P and $\{Q_k\}_{k=1}^N$ be probability measures on the measurable space $(\mathcal{Z},\mathcal{F})$ such that $P \ll Q_k$ and there exists $\epsilon \in (0,1]$ s.t. $d_{1+\epsilon}(P\|Q_k) < \infty$ for $k = 1,\dots,K$. Let $f : \mathcal{Z} \to \mathbb{R}_+$ be a bounded non-negative function, i.e., $\|f\|_\infty < \infty$. Let $\check{\mu}_{BH}$ be the truncated balance heuristic estimator of f, as defined in (4.2), using N_k i.i.d. samples from each Q_k. Then, the bias of $\check{\mu}_{BH}$ can be bounded as:*

$$0 \leqslant \mu - \mathop{\mathbb{E}}_{z_{ik}\overset{iid}{\sim}Q_k}[\check{\mu}_{BH}] \leqslant \|f\|_\infty M^{-\epsilon} d_{1+\epsilon}(P\|\Phi)^\epsilon, \tag{4.3}$$

and the variance of $\check{\mu}_{BH}$ can be bounded as:

$$\mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{iid}{\sim}Q_k}[\check{\mu}_{BH}] \leqslant \|f\|_\infty^2 M^{1-\epsilon} \frac{d_{1+\epsilon}(P\|\Phi)^\epsilon}{N}, \tag{4.4}$$

where $N = \sum_{k=1}^K N_k$ is the total number of samples and $\Phi = \sum_{k=1}^K \frac{N_k}{N} Q_k$ is a finite mixture.

Proof. Let us start with the bias term. The first inequality $0 \leqslant \mu - \mathbb{E}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}[\check{\mu}_{\text{BH}}]$ derives from the fact that $\widehat{\mu}_{\text{BH}} \geqslant \check{\mu}_{\text{BH}}$, being $f(z) \geqslant 0$ for all z and observing that $\widehat{\mu}$ is unbiased, i.e., $\mathbb{E}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}[\widehat{\mu}_{\text{BH}}] = \mu$. For the second inequality, let us consider the following derivation:

$$\mu - \mathop{\mathbb{E}}_{x_i\sim q_i}[\check{\mu}] = \mathop{\mathbb{E}}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}[\widehat{\mu}_{\text{BH}}] - \mathop{\mathbb{E}}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}[\check{\mu}_{\text{BH}}]$$

$$= \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathop{\mathbb{E}}_{z_{ik}\sim Q_k}\left[f(z_{ik})\left(\frac{p(z_{ik})}{\sum_{j=1}^K \frac{N_j}{N} q_j(z_{ik})} - \min\left\{M, \frac{p(z_{ik})}{\sum_{j=1}^K \frac{N_j}{N} q_j(z_{ik})}\right\}\right)\right] \tag{A.4}$$

$$= \mathop{\mathbb{E}}_{z\sim\Phi}\left[f(z)\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)} - M\right)\mathbb{1}_{\left\{\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)} \geqslant M\right\}}\right] \tag{A.5}$$

$$\leqslant \mathop{\mathbb{E}}_{z\sim\Phi}\left[f(z)\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\right)\mathbb{1}_{\left\{\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)} \geqslant M\right\}}\right] \tag{A.6}$$

$$
\begin{aligned}
&\leqslant \|f\|_\infty \mathop{\mathbb{E}}_{z\sim\Phi}\left[\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\right)\mathbb{1}_{\left\{\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\geqslant M\right\}}\right] \\
&\leqslant \|f\|_\infty \mathop{\mathbb{E}}_{z\sim\Phi}\left[\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\right)^{1+\epsilon}\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\right)^{-\epsilon}\mathbb{1}_{\left\{\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\geqslant M\right\}}\right] \\
&\leqslant \|f\|_\infty \mathop{\mathbb{E}}_{z\sim\Phi}\left[\left(\frac{p(z)}{\sum_{j=1}^K \frac{N_j}{N} q_j(z)}\right)^{1+\epsilon}\right] M^{-\epsilon} \qquad (A.7) \\
&= \|f\|_\infty d_{1+\epsilon}(P\|\Phi)^{\epsilon} M^{-\epsilon},
\end{aligned}
$$

where (A.5) is an application of equation (A.3), (A.6) derives from recalling that $M \geqslant 0$ and (A.7) is obtained by observing that $x^{-\epsilon}\mathbb{1}_{\{x\geqslant M\}}$ is either 0 and thus the bound holds or at most $M^{-\epsilon}$. For the variance the argument is similar:

$$
\begin{aligned}
\mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}\left[\check{\mu}_{\mathrm{BH}}\right] &= \mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{\text{iid}}{\sim}Q_k}\left[\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} f(z_{ki})\min\left\{M, \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})}\right\}\right] \\
&= \frac{1}{N^2}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\sim Q_k}\left[f(z_{ki})\min\left\{M, \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})}\right\}\right] \\
&\leqslant \frac{1}{N^2}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathop{\mathbb{E}}_{z_{ik}\sim Q_k}\left[\left(f(z_{ki})\min\left\{M, \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})}\right\}\right)^2\right] \\
&\leqslant \|f\|_\infty \frac{1}{N^2}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathop{\mathbb{E}}_{z_{ik}\sim Q_k}\left[\left(\min\left\{M, \frac{p(z_{ki})}{\sum_{j=1}^n \frac{N_j}{N} q_j(z_{ki})}\right\}\right)^2\right] \\
&= \|f\|_\infty^2 \frac{1}{N}\mathop{\mathbb{E}}_{z\sim\Phi}\left[\min\left\{M, \frac{p(z)}{\sum_{j=1}^n \frac{N_j}{N} q_j(z)}\right\}^2\right] \qquad (A.8) \\
&= \|f\|_\infty^2 \frac{1}{N}\mathop{\mathbb{E}}_{z\sim\Phi}\left[\min\left\{M, \frac{p(z)}{\sum_{j=1}^n \frac{N_j}{N} q_j(z)}\right\}^{1+\epsilon}\min\left\{M, \frac{p(z)}{\sum_{j=1}^n \frac{N_j}{N} q_j(z)}\right\}^{1-\epsilon}\right] \\
&\leqslant \|f\|_\infty^2 \frac{1}{N}\mathop{\mathbb{E}}_{z\sim\Phi}\left[\left(\frac{p(z)}{\sum_{j=1}^n \frac{N_j}{N} q_j(z)}\right)^{1+\epsilon}\right] M^{1-\epsilon} \qquad (A.9) \\
&= \|f\|_\infty^2 M^{1-\epsilon}\frac{d_{1+\epsilon}(P\|\Phi)^{\epsilon}}{N},
\end{aligned}
$$

$$(A.10)$$

where (A.8) is again an application of equation (A.3) and A.9 derives from observing that $\min\{x, y\} \leqslant x$ and also $\min\{x, y\} \leqslant y$. □

Theorem 4.1.1. *Let P and $\{Q_k\}_{k=1}^N$ be probability measures on the measurable space $(\mathcal{Z}, \mathcal{F})$ such that $P \ll Q_k$ and there exists $\epsilon \in (0,1]$ s.t. $d_{1+\epsilon}(P\|Q_k) < \infty$ for $k = 1, \dots, K$. Let $f : \mathcal{Z} \to \mathbb{R}_+$ be a bounded non-negative function, i.e., $\|f\|_\infty < \infty$. Let $\check{\mu}_{BH}$ be the truncated balance heuristic estimator of f, as defined in (4.2), using N_k i.i.d. samples from each Q_k. Let $M_N = \left(\frac{N d_{1+\epsilon}(P\|\Phi)^\epsilon}{\log \frac{1}{\delta}}\right)^{\frac{1}{1+\epsilon}}$, then with probability at least $1-\delta$:*

$$\check{\mu}_{BH} \leqslant \mu + \|f\|_\infty \left(\sqrt{2} + \frac{1}{3}\right)\left(\frac{d_{1+\epsilon}(P\|\Phi)\log\frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}, \tag{4.5}$$

and also, with probability at least $1-\delta$:

$$\check{\mu}_{BH} \geqslant \mu - \|f\|_\infty \left(\sqrt{2} + \frac{4}{3}\right)\left(\frac{d_{1+\epsilon}(P\|\Phi)\log\frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}. \tag{4.6}$$

Proof. Let us start with the first inequality. Observing that all samples z_{ik} are independent and that $\check{\mu}_{\mathrm{BH}} \leqslant M\|f\|_\infty$, we can state using Bernstein inequality [14] that with probability at least $1-\delta$ we have:

$$\begin{aligned}
\check{\mu}_{\mathrm{BH}} &\leqslant \mathop{\mathbb{E}}_{z_{ik}\sim Q_k}[\check{\mu}_{\mathrm{BH}}] + \sqrt{2 \mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{\mathrm{iid}}{\sim} Q_k}[\check{\mu}_{\mathrm{BH}}]\log\frac{1}{\delta}} + \|f\|_\infty \frac{M\log\frac{1}{\delta}}{3N} \\
&\leqslant \mu + \|f\|_\infty \sqrt{\frac{2M^{1-\epsilon}d_{1+\epsilon}(P\|\Phi)^\epsilon \log\frac{1}{\delta}}{N}} + \|f\|_\infty \frac{M\log\frac{1}{\delta}}{3N} && \text{(A.11)}\\
&= \mu + \|f\|_\infty \left(\sqrt{2} + \frac{1}{3}\right)\left(\frac{d_{1+\epsilon}(P\|\Phi)\log\frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}, && \text{(A.12)}
\end{aligned}$$

where (A.11) is obtained by substituting the variance with its bound (4.4) and (A.12) is from the choice of M. For the second inequality we just need to consider additionally the bias.

$$\begin{aligned}
\check{\mu}_{\mathrm{BH}} &\geqslant \mathop{\mathbb{E}}_{z_{ik}\sim Q_k}[\check{\mu}_{\mathrm{BH}}] - \sqrt{2 \mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{\mathrm{iid}}{\sim} Q_k}[\check{\mu}_{\mathrm{BH}}]\log\frac{1}{\delta}} - \|f\|_\infty \frac{M\log\frac{1}{\delta}}{3N} \\
&= \mu - \left(\mu - \mathop{\mathbb{E}}_{z_{ik}\sim Q_k}[\check{\mu}_{\mathrm{BH}}]\right) - \sqrt{2 \mathop{\mathbb{V}\mathrm{ar}}_{z_{ik}\overset{\mathrm{iid}}{\sim} Q_k}[\check{\mu}_{\mathrm{BH}}]\log\frac{1}{\delta}} - \|f\|_\infty \frac{M\log\frac{1}{\delta}}{3N} \\
&\geqslant \mu - \|f\|_\infty M^{-\epsilon} d_{1+\epsilon}(P\|\Phi)^\epsilon - \|f\|_\infty\left(\sqrt{2} + \frac{1}{3}\right)\left(\frac{d_{1+\epsilon}(P\|\Phi)\log\frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.13)}\\
&= \mu - \|f\|_\infty \left(\sqrt{2} + \frac{4}{3}\right)\left(\frac{d_{1+\epsilon}(P\|\Phi)\log\frac{1}{\delta}}{N}\right)^{\frac{\epsilon}{1+\epsilon}}, && \text{(A.14)}
\end{aligned}$$

where (A.13) comes from substituting the bias with its bound (4.3). □

Theorem 4.4.1. *Let $\mathcal{X}$ be a discrete arm set with $|\mathcal{X}| = K \in \mathbb{N}_+$. Under Assumption (4.4.1), Algorithm (4.1) with confidence schedule $\delta_t = \frac{3\delta}{t^2\pi^2 K}$ guarantees, with probability at least $1-\delta$:*

$$\textit{Regret}(T) \leqslant \Delta_0 + CT^{\frac{1}{1+\epsilon}}\left[v_\epsilon\left(2\log T + \log\frac{\pi^2 K}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}},$$

where $C = (1+\epsilon)\left(2\sqrt{2}+\frac{5}{3}\right)\|f\|_\infty$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

Proof. Fix an $\epsilon > 0$. To ease the notation, let $c^- := \|f\|_\infty\left(\sqrt{2}+\frac{1}{3}\right)$, $c^+ := \|f\|_\infty\left(\sqrt{2}+\frac{4}{3}\right)$, and $\beta_t(\boldsymbol{x}) := \left(\frac{d_{1+\epsilon}(p_{\boldsymbol{x}}\|\Phi_t)\log\frac{1}{\delta_t}}{t}\right)^{\frac{\epsilon}{1+\epsilon}}$. We start by showing that, with probability at least $1-\delta$:

$$-c^+\beta_t(\boldsymbol{x}) \leqslant \check{\mu}_t(\boldsymbol{x}) - \mu(\boldsymbol{x}) \leqslant c^-\beta_t(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x}\in\mathcal{X} \text{ and } t = 1,\dots,T. \tag{A.15}$$

Indeed:

$$\begin{aligned}
\mathbb{P}\left(\bigcap_{k=1}^{K}\bigcap_{t=1}^{T}\left[\check{\mu}_t(\boldsymbol{x}_k)-\mu(\boldsymbol{x}_k)\leqslant c^-\beta_t(\boldsymbol{x}_k)\right]\right) &= 1-\mathbb{P}\left(\bigcup_{k=1}^{K}\bigcup_{t=1}^{T}\left[\check{\mu}_t(\boldsymbol{x}_k)-\mu(\boldsymbol{x}_k)> c^-\beta_t(\boldsymbol{x}_k)\right]\right)\\
&\geqslant 1-K\sum_{t=1}^{T}\mathbb{P}\left(\check{\mu}_t(\boldsymbol{x}_1)-\mu(\boldsymbol{x}_1)> c^-\beta_t(\boldsymbol{x}_1)\right) \qquad (A.16)\\
&\geqslant 1-K\sum_{t=1}^{T}\delta_t \qquad (A.17)\\
&\geqslant 1-\frac{\delta}{2}, \qquad (A.18)
\end{aligned}$$

where (A.16) is from a double union bound (over time and over the finite elements of $\mathcal{X}$), (A.17) is from Theorem 4.1.1, and (A.18) is by hypothesis on δ_t and $\sum_{t=1}^{T}\frac{1}{t^2}\leqslant\sum_{t=1}^{\infty}\frac{1}{t^2}=\frac{\pi^2}{6}$. Similarly:

$$\begin{aligned}
\mathbb{P}\left(\bigcap_{k=1}^{K}\bigcap_{t=1}^{T}\left[\check{\mu}_t(\boldsymbol{x}_k)-\mu(\boldsymbol{x}_k)\geqslant -c^+\beta_t(\boldsymbol{x}_k)\right]\right) &= 1-\mathbb{P}\left(\bigcup_{k=1}^{K}\bigcup_{t=1}^{T}\left[\check{\mu}_t(\boldsymbol{x}_k)-\mu(\boldsymbol{x}_k)< -c^+\beta_t(\boldsymbol{x}_k)\right]\right)\\
&\geqslant 1-K\sum_{t=1}^{T}\mathbb{P}\left(\check{\mu}_t(\boldsymbol{x}_1)-\mu(\boldsymbol{x}_1)< -c^+\beta_t(\boldsymbol{x}_1)\right)\\
&\geqslant 1-K\sum_{t=1}^{T}\delta_t\\
&\geqslant 1-\frac{\delta}{2}.
\end{aligned}$$

Hence, by union bound over the two inequalities, (A.15) holds with probability at least $1-\delta$. This allows to lower bound the instantaneous regret with the same probability:

$$\Delta_t = \mu(\boldsymbol{x}^*) - \mu(\boldsymbol{x}) \leqslant \breve{\mu}_t(\boldsymbol{x}^*) + c^+\beta_t(\boldsymbol{x}^*) - \mu(\boldsymbol{x}_t) \tag{A.19}$$
$$\leqslant \breve{\mu}_t(\boldsymbol{x}_t) + c^+\beta_t(\boldsymbol{x}_t) - \mu(\boldsymbol{x}_t) \tag{A.20}$$
$$\leqslant (c^- + c^+)\beta(\boldsymbol{x}_t) \qquad \text{for all } t = 1, \dots, T, \tag{A.21}$$

where (A.19) and (A.21) are from (A.15), while (A.20) is by hypothesis, as $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathcal{X}}(\breve{\mu}_t(\boldsymbol{x}) + c^+\beta_t(\boldsymbol{x}))$. Note that the union bound over the elements of $\mathcal{X}$ in (A.15) was necessary for (A.19) as the optimal arm $\boldsymbol{x}^*$ may not be unique. Finally, with probability at least $1-\delta$:

$$\begin{aligned}
\mathit{Regret}(T) &= \sum_{t=0}^{T} \Delta_t \\
&= \Delta_0 + \sum_{t=1}^{T} \Delta_t \\
&\leqslant \Delta_0 + (c^+ + c^-) \sum_{t=1}^{T} \beta_t(\boldsymbol{x}_t) && \text{(A.22)}\\
&\leqslant \Delta_0 + (c^+ + c^-) v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{\log\frac{1}{\delta_t}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.23)}\\
&= \Delta_0 + (c^+ + c^-) v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{2\log t + \log\frac{\pi^2 K}{3\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.24)}\\
&\leqslant \Delta_0 + (c^+ + c^-) \left[v_\epsilon\left(2\log T + \log\frac{\pi^2 K}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} t^{-\frac{\epsilon}{1+\epsilon}} \\
&\leqslant \Delta_0 + (c^+ + c^-) \left[v_\epsilon\left(2\log T + \log\frac{\pi^2 K}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}} (1+\epsilon)T^{\frac{1}{1+\epsilon}}, && \text{(A.25)}
\end{aligned}$$

where (A.22) is from (A.21) and holds with probability no less than $1-\delta$, (A.23) is from Assumption 4.4.1, (A.24) is by definition of δ_t, and (A.25) is from:

$$\sum_{t=1}^{T} t^{-\alpha} \leqslant \int_1^{T+1} t^{-\alpha}\mathrm{d}t = \frac{1}{1-\alpha}\left((T+1)^{1-\alpha} - 1\right) \leqslant \frac{T^{1-\alpha}}{1-\alpha} \quad \text{for all } 0 < \alpha < 1, \tag{A.26}$$

with $\alpha = \frac{\epsilon}{1+\epsilon}$. The proof is completed by renaming $C \leftarrow (1+\epsilon)(c^+ + c^-) = (1+\epsilon)(2\sqrt{2} + \frac{5}{3})\left\|f\right\|_\infty$. □

Lemma 4.4.1. *In the policy optimization problem, Assumption 4.4.2 can be replaced by:*

$$\sup_{s\in\mathcal{S},\boldsymbol{\theta}\in\Theta} \mathop{\mathbb{E}}_{a\sim\pi_{\boldsymbol{\theta}}} \left[\left|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\right|\right] \leqslant \boldsymbol{u}_1, \tag{4.12}$$

in the action-based paradigm, and by:

$$\sup_{\boldsymbol{\xi}\in\Xi} \mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim\rho_{\boldsymbol{\xi}}} \left[|\nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta})|\right] \leqslant \boldsymbol{u}_2, \tag{4.13}$$

in the parameter-based paradigm, where $\boldsymbol{u}_1$ *and* $\boldsymbol{u}_2$ *are d-dimensional vectors and the inequalities are component-wise.*

Proof. We consider the infinite-horizon case ($H = \infty, \gamma < 1$), as the finite-horizon case is w.l.o.g. under mild assumptions. To show Lipschitz continuity in the action-based paradigm, it is enough to bound $\|\nabla_{\boldsymbol{\theta}} J\|_\infty$ under (4.12). From the Policy Gradient Theorem [87]:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\boldsymbol{\theta}}\\ a\sim\pi_{\boldsymbol{\theta}}}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q_{\boldsymbol{\theta}}(s,a)\right], \tag{A.27}$$

where $\rho_{\boldsymbol{\theta}}$ is the discounted state-occupancy measure under policy $\pi_{\boldsymbol{\theta}}$ and $Q_{\boldsymbol{\theta}}$ is the action-value function [87], modeling the reward that can be obtained starting from state s, taking action a and following $\pi_{\boldsymbol{\theta}}$ thereafter. From (A.27), for every $\boldsymbol{\theta} \in \Theta$:

$$\begin{aligned} |\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})| &\leqslant \frac{R_{\max}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\boldsymbol{\theta}}\\ a\sim\pi_{\boldsymbol{\theta}}}} \left[|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(s,a)|\right] && \text{(A.28)} \\ &\leqslant \frac{R_{\max}}{(1-\gamma)^2} \sup_{s\in\mathcal{S}} \mathop{\mathbb{E}}_{a\sim\pi_{\boldsymbol{\theta}}} \left[|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(s,a)|\right] \\ &= \frac{\boldsymbol{u}_1 R_{\max}}{(1-\gamma)^2}, && \text{(A.29)} \end{aligned}$$

where the inequalities are component-wise, (A.28) is from the trivial fact $\|Q_{\boldsymbol{\theta}}\|_\infty \leqslant \frac{R_{\max}}{(1-\gamma)}$, and (A.29) is from assumption (4.12). It follows that $L = \frac{\|\boldsymbol{u}_1\|_\infty R_{\max}}{(1-\gamma)^2}$ is a valid Lipschitz constant under the l_1 norm. The commonly used Gaussian policy:

$$\pi_{\boldsymbol{\theta}}(a|s) = \mathcal{N}(\boldsymbol{\theta}^T\boldsymbol{\phi}(s), \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{a-\boldsymbol{\theta}^T\boldsymbol{\phi}(s)}{\sigma}\right)^2\right\}, \tag{A.30}$$

where $\phi(s)$ is a vector of component-wise bounded state features, i.e., $\sup_{s\in\mathcal{S}} |\boldsymbol{\phi}(s)| \leqslant \boldsymbol{\phi}_{\max}$, satisfies assumption (4.12):

$$\begin{aligned} \mathop{\mathbb{E}}_{a\sim\pi_{\boldsymbol{\theta}}} \left[|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)|\right] &= \mathop{\mathbb{E}}_{a\sim\pi_{\boldsymbol{\theta}}} \left[\frac{|\boldsymbol{\phi}(s)(a-\boldsymbol{\theta}^T\boldsymbol{\phi}(s))|}{\sigma^2}\right] \\ &\leqslant \frac{|\boldsymbol{\phi}(s)|}{\sigma} \mathop{\mathbb{E}}_{a\sim\pi_{\boldsymbol{\theta}}} \left[\left|\frac{a-\boldsymbol{\theta}^T\boldsymbol{\phi}(s)}{\sigma}\right|\right] \\ &\leqslant \frac{|\boldsymbol{\phi}(s)|}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}} e^{-x^2}|x|\mathrm{d}x && \text{(A.31)} \end{aligned}$$

$$\leqslant \frac{2\boldsymbol{\phi}_{\max}}{\sqrt{2\pi}\sigma} := \boldsymbol{u}_1, \tag{A.32}$$

where inequalities are component-wise and (A.31) is by the substitution $x \leftarrow \frac{a-\boldsymbol{\theta}^T\boldsymbol{\phi}(s)}{\sigma}$. Even when σ must be learned, proper parametrization (e.g., $\sigma \propto \exp\{\boldsymbol{\theta}\}$), together with the compactness of Θ, allows to satisfy assumption (4.12).

To show Lipschitz continuity for the parameter-based paradigm, it is enough to bound $\left\|\nabla_{\boldsymbol{\xi}} \mathbb{E}_{\boldsymbol{\theta}\sim\nu_{\boldsymbol{\xi}}}[J(\boldsymbol{\theta})]\right\|_{\infty}$ under (4.13). For every $\boldsymbol{\xi} \in \Xi$:

$$\begin{aligned}\left|\nabla_{\boldsymbol{\xi}} \underset{\boldsymbol{\theta}\sim\nu_{\boldsymbol{\xi}}}{\mathbb{E}}[J(\boldsymbol{\theta})]\right| &= \left|\underset{\boldsymbol{\theta}\sim\nu_{\boldsymbol{\xi}}}{\mathbb{E}}[\nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta})J(\boldsymbol{\theta})]\right| \\ &\leqslant \frac{R_{\max}}{(1-\gamma)} \underset{\boldsymbol{\theta}\sim\nu_{\boldsymbol{\xi}}}{\mathbb{E}}[|\nabla_{\boldsymbol{\xi}} \log \nu_{\boldsymbol{\xi}}(\boldsymbol{\theta})|] &\text{(A.33)} \\ &\leqslant \frac{\boldsymbol{u}_2 R_{\max}}{(1-\gamma)}, &\text{(A.34)}\end{aligned}$$

where the inequalities are component-wise, (A.33) is from the trivial fact $J(\boldsymbol{\theta}) \leqslant \frac{R_{\max}}{1-\gamma}$, and (A.34) is from assumption (4.13). It follows that $L = \frac{\|\boldsymbol{u}_2 R_{\max}\|_{\infty}}{(1-\gamma)}$ is a valid Lipschitz constant under the l_1 norm. A Gaussian hyperpolicy $\nu_{\boldsymbol{\xi}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\xi}, \operatorname{diag}(\boldsymbol{\sigma}))$ satisfies assumption (4.13) with $\boldsymbol{u}_2 = \frac{2}{\sqrt{2\pi}\boldsymbol{\sigma}}$. The proof of this fact is analogous to that of (A.32). Even when $\boldsymbol{\sigma}$ must be learned, proper parametrization (e.g., $\boldsymbol{\sigma} \propto \exp\{\boldsymbol{\xi}\}$), together with the compactness of Ξ, allows to satisfy assumption (4.13). □

Theorem 4.4.2. *Let $\mathcal{X}$ be a d-dimensional compact arm set with $\mathcal{X} \subseteq [-D, D]^d$. Under Assumptions (4.4.1) and (4.4.2), Algorithm (4.1) with confidence schedule $\delta_t = \frac{6\delta}{\pi^2 t^2(1+d^d t^{2d})}$ guarantees, with probability at least $1-\delta$:*

$$\mathit{Regret}(T) \leqslant \Delta_0 + CT^{\frac{1}{1+\epsilon}}\left[v_{\epsilon}\left(2(d+1)\log T + d\log d + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}} + \frac{\pi^2 LD}{6},$$

where $C = (1+\epsilon)\left(2\sqrt{2}+\frac{5}{3}\right)\|f\|_{\infty}$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

Proof. Fix an $\epsilon > 0$. Let c^-, c^+ and $\beta_t(\boldsymbol{x})$ be defined as in the proof of Theorem 4.4.1. The finite cardinality of $\mathcal{X}$ allowed to perform a union bound over the arms that was crucial for the proof of Theorem 4.4.1. We cannot do the same here as $\mathcal{X}$ has infinite cardinality. To overcome this problem, we follow the line of reasoning proposed by [81]. First, we can say something about the arms that are actually selected by the algorithm, which are finite. From Theorem 4.1.1, by a union bound over $t = 1, \ldots, T$, we have that, with probability at least

$1 - \sum_{t=1}^{T} \delta_t$:

$$\breve{\mu}_t(\boldsymbol{x}_t) - \mu(\boldsymbol{x}_t) \leqslant c^- \beta_t(\boldsymbol{x}_t) \qquad \text{for all } t = 1, \ldots, T. \tag{A.35}$$

We also need a specular inequality for the optimal arm. Unfortunately, we cannot assume there exists a unique optimal arm $\boldsymbol{x}^*$.[1] Even worse, a dense set of optimal arms may exist. To overcome this problem, we introduce, *only for the purposes of the proof*, a discretization of the arm space. Let $\widetilde{\mathcal{X}}_t$ be a d-dimensional regular grid of τ_t^d vertexes, where $(\tau_t \in \mathbb{N}_+)_{t=1}^T$ is a discretization schedule. Let $[\boldsymbol{x}]_t$ be the closest vertex to $\boldsymbol{x}$ in $\widetilde{\mathcal{X}}_t$. From Assumption 4.4.2:

$$|\mu(\boldsymbol{x}) - \mu([\boldsymbol{x}]_t)| \leqslant L \left\| \boldsymbol{x} - [\boldsymbol{x}]_t \right\|_1 \leqslant \frac{LDd}{\tau_t}, \tag{A.36}$$

as each voxel of the grid has side $\frac{2D}{\tau_t}$ and no point can be further from a vertex than d half-sides according to the l_1 norm. Now fix a $t \geqslant 1$ and an optimal arm $\boldsymbol{x}^*$. With probability at least $1 - \delta_t$:

$$\begin{aligned} \mu(\boldsymbol{x}^*) - \breve{\mu}_t([\boldsymbol{x}^*]_t) &= \mu(\boldsymbol{x}^*) - \mu([\boldsymbol{x}^*]_t) + \mu([\boldsymbol{x}^*]_t) - \breve{\mu}_t([\boldsymbol{x}^*]_t) \\ &\leqslant \mu([\boldsymbol{x}^*]_t) - \breve{\mu}_t([\boldsymbol{x}^*]_t) + |\mu(\boldsymbol{x}^*) - \mu([\boldsymbol{x}^*]_t)| \\ &\leqslant c^+ \beta_t([\boldsymbol{x}^*]_t) + \frac{LDd}{\tau_t}, \end{aligned} \tag{A.37}$$

where the inequality (A.37) is from Theorem 4.1.1 and (A.36). Since any voxel may contain an optimal arm, we must perform a union bound over the $\lceil \tau \rceil^d$ vertexes of $\widetilde{\mathcal{X}}_t$, and a subsequent one over $t, \ldots, T$. Hence, with probability at least $1 - \sum_{t=1}^{T} \tau_t^d \delta_t$:

$$\mu(\boldsymbol{x}^*) - \breve{\mu}_t([\boldsymbol{x}^*]_t) \leqslant c^+ \beta_t([\boldsymbol{x}^*]_t) + \frac{LDd}{\tau_t} \quad \text{for } t = 1, \ldots, T \text{ and every } \boldsymbol{x}^* \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mu(\boldsymbol{x}). \tag{A.38}$$

We can now proceed to bound the instantaneous regret. With probability at least $1 - \sum_{t=1}^{T} \delta_t (1 + \tau_t^d)$:

$$\begin{aligned} \Delta_t = \mu(\boldsymbol{x}^*) - \mu(\boldsymbol{x}_t) &\leqslant \breve{\mu}_t([\boldsymbol{x}^*]_t) + c^+ \beta_t([\boldsymbol{x}^*]_t) + \frac{LDd}{\tau_t} - \mu(\boldsymbol{x}_t) && \text{(A.39)} \\ &\leqslant \breve{\mu}_t(\boldsymbol{x}_t) + c^+ \beta_t(\boldsymbol{x}_t) + \frac{LDd}{\tau_t} - \mu(\boldsymbol{x}_t) && \text{(A.40)} \\ &\leqslant (c^+ + c^-) \beta_t(\boldsymbol{x}_t) + \frac{LDd}{\tau_t}, && \text{(A.41)} \end{aligned}$$

[1]Instead, $\mu(\boldsymbol{x}^*)$ is always unique.

where (A.39) is from (A.38) and holds with probability at least $1 - \sum_{t=1}^{T} \tau_t^d \delta_t$, (A.40) is by hypothesis, as $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathcal{X}}(\breve{\mu}_t(\boldsymbol{x}) + c^+\beta_t(\boldsymbol{x}))$, and (A.41) is from (A.36) and holds with probability at least $1 - \sum_{t=1}^{T} \delta_t$. Hence, (A.41) holds with probability no less than $1 - \sum_{t=1}^{T} \tau_t^d \delta_t - \sum_{t=1}^{T} \delta_t = 1 - \sum_{t=1}^{T} \delta_t(1 + \tau_t^d)$. Let us pick as a discretization schedule $\tau_t = dt^2$. This has no impact whatsoever on the algorithm, as the discretization is only hypothetical. With this τ_t and the confidence schedule proposed in the statement of the theorem, it is easy to verify that (A.41) holds with probability at least $1 - \delta$.

Finally, we can bound the regret. With probability at least $1 - \delta$:

$$
\begin{aligned}
\mathit{Regret}(T) &\leqslant \Delta_0 + \sum_{t=1}^{T} \Delta_t \\
&\leqslant \Delta_0 + (c^+ + c^-)\sum_{t=1}^{T} \beta_t(\boldsymbol{x}_t) + LDd\sum_{t=1}^{T} \frac{1}{\tau_t} && \text{(A.42)} \\
&\leqslant (c^+ + c^-)\sum_{t=1}^{T} \beta_t(x_t) + \frac{\pi^2 LD}{6} && \text{(A.43)} \\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{\log\frac{1}{\delta_t}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} + \frac{\pi^2 LD}{6} && \text{(A.44)} \\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{\log(1 + d^d t^{2d}) + 2\log t + \log\frac{\pi^2}{6\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.45)} \\
&\quad + \frac{\pi^2 LD}{6} && \text{(A.46)} \\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{\log(2d^d t^{2d}) + 2\log t + \log\frac{\pi^2}{6\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.47)} \\
&\quad + \frac{\pi^2 LD}{6} && \text{(A.48)} \\
&= \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} \left(\frac{2(d+1)\log t + d\log d + \log\frac{\pi^2}{3\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.49)} \\
&\quad + \frac{\pi^2 LD}{6} \\
&\leqslant \Delta_0 + (c^+ + c^-)\left[v_\epsilon\left(2(d+1)\log T + d\log d + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}} \sum_{t=1}^{T} t^{-\frac{\epsilon}{1+\epsilon}} && \text{(A.50)} \\
&\quad + \frac{\pi^2 LD}{6}
\end{aligned}
$$

$$\leqslant \Delta_0 + (c^+ + c^-)\left[v_\epsilon\left(2(d+1)\log T + d\log d + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}}(1+\epsilon)T^{\frac{1}{1+\epsilon}} \tag{A.51}$$

$$+\frac{\pi^2 LD}{6}, \tag{A.52}$$

where (A.42) is from (A.41) and holds with probability at least $1-\delta$, (A.43) is from the choice of τ_t and $\sum_{t=1}^{T} t^{-2} \leqslant \sum_{t=1}^{\infty} t^{-2} = \frac{\pi^2}{6}$, (A.44) is from Assumption 4.4.1, (A.46) is from the choice of δ_t, (A.48) is from $\log(1+x) \leqslant \log(2x)$, which holds for every $x \geqslant 1$, and (A.52) is from (A.26) with $\alpha = \frac{\epsilon}{1+\epsilon}$. The proof is completed by renaming $C \leftarrow (1+\epsilon)(c^+ + c^-) = (1+\epsilon)(2\sqrt{2}+\frac{5}{3})\,\|f\|_\infty$. $\square$

Theorem 4.4.3. *Let $\mathcal{X}$ be a d-dimensional compact arm set with $\mathcal{X} \subseteq [-D,D]^d$. For any $\kappa \geqslant 2$, under Assumptions (4.4.1) and (4.4.2), Algorithm (4.2) with confidence schedule $\delta_t = \frac{6\delta}{\pi^2 t^2\left(1+\lceil t^{1/\kappa}\rceil^d\right)}$ and discretization schedule $\tau_t = \lceil t^{\frac{1}{\kappa}}\rceil$ guarantees, with probability at least $1-\delta$:*

$$\textit{Regret}(T) \leqslant \Delta_0 + C_1 T^{\left(1-\frac{1}{\kappa}\right)}d + C_2 T^{\frac{1}{1+\epsilon}}\cdot\left[v_\epsilon\left((2+{}^{d}\!/\!_{\kappa})\log T + d\log 2 + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}},$$

where $C_1 = \frac{\kappa}{\kappa-1}LD$, $C_2 = (1+\epsilon)\left(2\sqrt{2}+\frac{5}{3}\right)\|f\|_\infty$, and Δ_0 is the instantaneous regret of the initial arm $\boldsymbol{x}_0$.

Proof. The proof follows the one of Theorem 4.4.2 up to (A.38), except from the fact that the discretization is actually performed by the algorithm. That is, with probability at least $1-\sum_{t=1}^{T}\delta_t(1+\tau_t^d)$:

$$\begin{aligned}&\breve{\mu}_t(\boldsymbol{x}_t) - \mu(\boldsymbol{x}_t) \leqslant c^-\beta_t(\boldsymbol{x}_t) \qquad\qquad \text{and}\\ &\mu(\boldsymbol{x}^*) - \breve{\mu}_t([\boldsymbol{x}^*]_t) \leqslant c^+\beta_t([\boldsymbol{x}^*]_t) + \frac{LDd}{\tau_t} \quad \text{for } t=1,\dots,T \text{ and every } \boldsymbol{x}^* \in \arg\max_{\boldsymbol{x}\in\mathcal{X}}\mu(\boldsymbol{x}).\end{aligned} \tag{A.53}$$

This is enough to bound the instantaneous regret. With probability at least $1-\sum_{t=1}^{T}\delta_t(1+\tau_t^d)$:

$$\Delta_t = \mu(\boldsymbol{x}^*) - \mu(\boldsymbol{x}_t) \leqslant \breve{\mu}_t([\boldsymbol{x}^*]_t) + c^+\beta_t([\boldsymbol{x}^*]_t) + \frac{LDd}{\tau_t} - \mu(\boldsymbol{x}_t) \tag{A.54}$$

$$\leqslant \breve{\mu}_t(\boldsymbol{x}_t) + c^+\beta_t(\boldsymbol{x}_t) + \frac{LDd}{\tau_t} - \mu(\boldsymbol{x}_t) \tag{A.55}$$

$$\leqslant (c^+ + c^-)\beta_t(\boldsymbol{x}_t) + \frac{LDd}{\tau_t}, \tag{A.56}$$

where (A.52) and (A.55) are from (A.53) and hold simultaneously with probability at least $1-\sum_{t=1}^{T}\delta_t(1+\tau_t^d)$, and (A.54) is by hypothesis, as $\boldsymbol{x}_t =$

$\arg\max_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_t}(\breve{\mu}_t(\boldsymbol{x}) + c^+\beta_t(\boldsymbol{x})$. Note that the latter is true only by virtue of the fact that both $[\boldsymbol{x}^*]_t$ and $\boldsymbol{x}_t$ belong to $\widetilde{\mathcal{X}}_t$, as the optimization step of Algorithm 4.2 is restricted to $\widetilde{\mathcal{X}}_t$.

Finally, we can bound the regret. With probability at least $1-\delta$:

$$\begin{aligned}
\mathit{Regret}(T) &= \Delta_0 + \sum_{t=1}^{T}\Delta_t \\
&\leqslant \Delta_0 + (c^+ + c^-)\sum_{t=1}^{T}\beta_t(\boldsymbol{x}_t) + LDd\sum_{t=1}^{T}\frac{1}{\tau_t} && \text{(A.57)}\\
&\leqslant \Delta_0 + (c^+ + c^-)\sum_{t=1}^{T}\beta_t(\boldsymbol{x}_t) + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d && \text{(A.58)}\\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}}\sum_{t=1}^{T}\left(\frac{\log\frac{1}{\delta_t}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d && \text{(A.59)}\\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}}\sum_{t=1}^{T}\left(\frac{2\log t + \log\left(1+\left\lceil t^{\frac{1}{\kappa}}\right\rceil^d\right) + \log\frac{\pi^2}{6\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.60)}\\
&\quad + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d && \text{(A.61)}\\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}}\sum_{t=1}^{T}\left(\frac{2\log t + d\log\left(t^{\frac{1}{\kappa}}+1\right) + \log\frac{\pi^2}{3\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.62)}\\
&\quad + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d && \text{(A.63)}\\
&\leqslant \Delta_0 + (c^+ + c^-)v_\epsilon^{\frac{\epsilon}{1+\epsilon}}\sum_{t=1}^{T}\left(\frac{\left(2+\frac{d}{\kappa}\right)\log t + d\log 2 + \log\frac{\pi^2}{3\delta}}{t}\right)^{\frac{\epsilon}{1+\epsilon}} && \text{(A.64)}\\
&\quad + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d && \text{(A.65)}\\
&\leqslant \Delta_0 + (c^+ + c^-)\left[v_\epsilon\left(\left(2+\frac{d}{\kappa}\right)\log T + d\log 2 + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}}\sum_{t=1}^{T}t^{-\frac{\epsilon}{1+\epsilon}} && \text{(A.66)}\\
&\quad + \frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d,\\
&\leqslant \Delta_0 + (c^+ + c^-)\left[v_\epsilon\left(\left(2+\frac{d}{\kappa}\right)\log T + d\log 2 + \log\frac{\pi^2}{3\delta}\right)\right]^{\frac{\epsilon}{1+\epsilon}}(1+\epsilon)T^{\frac{1}{1+\epsilon}} && \text{(A.67)}
\end{aligned}$$

$$+\frac{\kappa}{\kappa-1}LDT^{\left(1-\frac{1}{\kappa}\right)}d, \tag{A.68}$$

where (A.57) is from (A.56) and holds with probability at least $1-\delta$ with the proposed δ_t and τ_t, (A.58) is from the proposed τ_t and (A.26) with $\alpha = {}^1/_\kappa$, (A.59) is from Assumption 4.4.1, (A.61) is from the proposed δ_t, (A.63) and (A.65) are from the fact $\log(x+1) \leqslant \log(2x)$ for $x \geqslant 1$, and (A.65) is from (A.26) with $\alpha = \frac{\epsilon}{1+\epsilon}$. The proof is completed by renaming $C_1 \leftarrow (1+\epsilon)(c^+ + c^-)\left\|f\right\|_\infty = (1+\epsilon)(2\sqrt{2}+\frac{5}{3})\left\|f\right\|_\infty$ and $C_2 \leftarrow \frac{\kappa}{\kappa-1}LD$. □

Lemma A.0.1. *Let $\Psi = \sum_{l=1}^L \zeta_l P_l$ and $\Phi = \sum_{k=1}^K \beta_k Q_k$, with $\zeta_l \in [0,1]$, $\sum_{l=1}^L \zeta_l = 1$, $\beta_k \in [0,1]$ and $\sum_{k=1}^K \beta_k = 1$, be two finite mixtures of the probability measures $\{P_l\}_{l=1}^L$ and $\{Q_k\}_{k=1}^K$ respectively. Let $\{\psi_{ij}\}_{\substack{i=1,2,\dots,L\\ j=1,2,\dots,K}}$ and $\{\phi_{ij}\}_{\substack{i=1,2,\dots,L\\ j=1,2,\dots,K}}$ be two sets of variational parameters s.t. $\phi_{ij} \geqslant 0$, $\psi_{ij} \geqslant 0$, $\sum_{k=1}^K \phi_{ij} = \zeta_l$ and $\sum_{l=1}^L \psi_{ij} = \beta_k$. Then, for any $\alpha \geqslant 1$, it holds that:*

$$d_\alpha(\Psi\|\Phi)^{\alpha-1} \leqslant \sum_{l=1}^L \sum_{k=1}^K \phi_{lk}^\alpha \psi_{lk}^{1-\alpha} d_\alpha(P_l\|Q_k)^{\alpha-1}.$$

Proof. The proof follows the idea of the variational bound for the KL-divergence proposed in [40]. Using the variational parameters we can express the two mixtures as:

$$\Psi = \sum_{l=1}^L \sum_{k=1}^K \phi_{lk} P_l,$$
$$\Phi = \sum_{l=1}^L \sum_{k=1}^K \psi_{lk} Q_k.$$

We use the convexity of the d_α and we apply Jensen inequality:

$$\begin{aligned}
d_\alpha(\Psi\|\Phi)^{\alpha-1} &= \int \left(\frac{\Psi}{\Phi}\right)^\alpha \mathrm{d}\Phi \\
&= \int \left(\sum_{l=1}^L \sum_{k=1}^K \frac{\phi_{lk}P_l}{\psi_{lk}Q_k}\frac{\psi_{lk}Q_k}{\Phi}\right)^\alpha \mathrm{d}\Phi \\
&\leqslant \int \sum_{l=1}^L \sum_{k=1}^K \frac{\psi_{lk}Q_k}{\Phi}\left(\frac{\phi_{lk}P_l}{\psi_{lk}Q_k}\right)^\alpha \mathrm{d}\Phi \\
&= \sum_{i=1}^n \sum_{j=1}^m \phi_{lk}^\alpha \psi_{lk}^{1-\alpha} \int \left(\frac{P_l}{Q_k}\right)^\alpha \mathrm{d}Q_k \\
&= \sum_{i=1}^n \sum_{j=1}^m \phi_{lk}^\alpha \psi_{lk}^{1-\alpha} d_\alpha(P_l\|Q_k)^{\alpha-1},
\end{aligned} \tag{A.69}$$

where (A.69) is obtained by Jensen inequality observing that $\frac{\psi_{lk}Q_k}{\Phi}$ is a distribution over $\{1, ..., L\} \times \{1, ..., K\}$. $\square$

We now consider the case in which f has just one mixture component, i.e., $n = 1$. In this case, we have that $\sum_{i=1}^{n} \psi_{ij} = \psi_j = b_j$, therefore the result reduces to:

$$d_\alpha(f\|g)^{\alpha-1} \leqslant \sum_{j=1}^{m} \phi_j^\alpha b_j^{1-\alpha} d_\alpha(f\|g_j)^{\alpha-1}. \tag{A.70}$$

We can now minimize the bound over the ϕ_j, subject to $\sum_{j=1}^{m} \phi_j = 1$, we get the following result.

Theorem 5.1.1. *Let P be a probability measure and $\Phi = \sum_{k=1}^{K} \beta_k Q_k$, with $\beta_k \in [0, 1]$ and $\sum_{k=1}^{K} \beta_k = 1$, be a finite mixture of the probability measures $\{Q_k\}_{k=1}^{K}$. Then, for any $\alpha \geqslant 1$, the exponentiated α-Rényi divergence can be bounded as:*

$$d_\alpha(P\|\Phi) \leqslant \frac{1}{\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)}}. \tag{5.3}$$

Proof. We now consider the case in which Ψ has just one mixture component, i.e., $L = 1$ and we abbreviate $\Psi = P$. In this case, we have that $\sum_{l=1}^{L} \psi_{kl} = \psi_k = \beta_k$, therefore the result reduces to:

$$d_\alpha(P\|\Phi)^{\alpha-1} \leqslant \sum_{k=1}^{K} \phi_k^\alpha \beta_k^{1-\alpha} d_\alpha(P\|Q_k)^{\alpha-1}. \tag{A.71}$$

We can now minimize the bound over the ϕ_k, subject to $\sum_{k=1}^{K} \phi_k = 1$. We use the Lagrange multipliers.

$$\mathcal{L}(\{\phi_k\}_{k=1,2,...,K}, \lambda) = \sum_{k=1}^{K} \phi_k^\alpha \beta_k^{1-\alpha} d_\alpha(P\|Q_k)^{\alpha-1} - \lambda \left(\sum_{k=1}^{K} \phi_k - 1 \right)$$

We take the partial derivatives w.r.t. the ϕ_k and the Lagrange multiplier λ.

$$\frac{\partial \mathcal{L}}{\partial \phi_k} = \alpha \phi_k^{\alpha-1} \beta_j^{1-\alpha} d_\alpha(P\|Q_k)^{\alpha-1} - \lambda = 0 \implies \phi_k = \frac{\lambda^{\frac{1}{\alpha-1}} \beta_j}{\alpha^{\frac{1}{\alpha-1}} d_\alpha(P\|Q_k)}.$$

We now replace the expression of ϕ_k into the constraint.

$$\sum_{j=1}^{K} \phi_k = \frac{\lambda^{\frac{1}{\alpha-1}}}{\alpha^{\frac{1}{\alpha-1}}} \sum_{k=1}^{L} \frac{\beta_k}{d_\alpha(P\|Q_k)} = 1 \implies \lambda = \frac{\alpha}{\left(\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)} \right)^{\alpha-1}}.$$

And finally we get the expression for ϕ_k:

$$\phi_k = \frac{\frac{\beta_k}{d_\alpha(P\|Q_k)}}{\sum_{h=1}^{K} \frac{\beta_h}{d_\alpha(P\|Q_h)}}. \tag{A.72}$$

We can now compute the bound value:

$$\begin{aligned}\sum_{k=1}^{K} \phi_k^{\alpha} \beta_k^{1-\alpha} d_\alpha(P\|Q_k)^{\alpha-1} &= \sum_{k=1}^{K} \frac{\frac{\beta_k^\alpha}{d_\alpha(P\|Q_k)^\alpha}}{\left(\sum_{h=1}^{K} \frac{\beta_h}{d_\alpha(P\|Q_h)}\right)^\alpha} \beta_k^{1-\alpha} d_\alpha(P\|Q_k)^{\alpha-1} \\ &= \frac{\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)}}{\left(\sum_{h=1}^{K} \frac{\beta_h}{d_\alpha(P\|Q_h)}\right)^\alpha} \\ &= \frac{1}{\left(\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)}\right)^{\alpha-1}}.\end{aligned}$$

As a consequence the bound becomes:

$$d_\alpha(P\|\Phi)^{\alpha-1} \leqslant \frac{1}{\left(\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)}\right)^{\alpha-1}} \implies d_\alpha(P\|\Phi) \leqslant \frac{1}{\sum_{k=1}^{K} \frac{\beta_k}{d_\alpha(P\|Q_k)}},$$

which is the weighted harmonic mean of the exponentiated divergences. $\square$

Bibliography

[1] Agrawal, R. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization 33*, 6 (1995), 1926–1951.

[2] Agrawal, S., and Goyal, N. Further optimal regret bounds for thompson sampling. In *Artificial intelligence and statistics* (2013), pp. 99–107.

[3] Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation 10*, 2 (1998), 251–276.

[4] Amari, S.-I., and Douglas, S. C. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)* (1998), vol. 2, IEEE, pp. 1213–1216.

[5] Antos, A., Szepesvári, C., and Munos, R. Fitted q-iteration in continuous action-space mdps. In *Advances in neural information processing systems* (2008), pp. 9–16.

[6] Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning 47*, 2-3 (2002), 235–256.

[7] Auer, P., and Ortner, R. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems* (2007), pp. 49–56.

[8] Auer, P., and Ortner, R. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica 61*, 1-2 (2010), 55–65.

[9] Baird III, L. C. Advantage updating. Tech. rep., WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

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

[11] 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), pp. 1471–1479.

[12] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American Statistical Association 112*, 518 (2017), 859–877.

[13] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424* (2015).

[14] Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence.* Oxford university press, 2013.

[15] Brafman, R. I., and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research 3*, Oct (2002), 213–231.

[16] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[17] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[18] Bubeck, S., Cesa-Bianchi, N., et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning 5*, 1 (2012), 1–122.

[19] Bubeck, S., Cesa-Bianchi, N., and Lugosi, G. Bandits with heavy tail. *IEEE Transactions on Information Theory 59*, 11 (2013), 7711–7717.

[20] Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. X-armed bandits. *Journal of Machine Learning Research 12*, May (2011), 1655–1695.

[21] Cesa-Bianchi, N., Gentile, C., Lugosi, G., and Neu, G. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems* (2017), pp. 6284–6293.

[22] Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems* (2005), pp. 1281–1288.

[23] Choshen, L., Fox, L., and Loewenstein, Y. Dora the explorer: Directed outreaching reinforcement action-selection. *arXiv preprint arXiv:1804.04012* (2018).

[24] Cochran, W. G. *Sampling techniques.* John Wiley & Sons, 2007.

[25] Cortes, C., Mansour, Y., and Mohri, M. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 442–450.

[26] Dayan, P., and Hinton, G. E. Using expectation-maximization for reinforcement learning. *Neural Computation 9*, 2 (1997), 271–278.

[27] Degris, T., White, M., and Sutton, R. S. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839* (2012).

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

[29] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. `https://github.com/openai/baselines`, 2017.

[30] Dorato, P., Abdallah, C. T., Cerone, V., and Jacobson, D. H. *Linear-quadratic control: an introduction.* Prentice Hall Englewood Cliffs, NJ, 1995.

[31] 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), pp. 1329–1338.

[32] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (2018), pp. 1406–1415.

[33] Garivier, A., and Cappé, O. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory* (2011), pp. 359–376.

[34] Gil, M., Alajaji, F., and Linder, T. Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences 249* (2013), 124–131.

[35] Glynn, P. W. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM 33*, 10 (1990), 75–84.

[36] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning.* MIT press, 2016.

[37] Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42*, 6 (2012), 1291–1307.

[38] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (2017), pp. 1352–1361.

[39] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (2018), pp. 1856–1865.

[40] Hershey, J. R., and Olsen, P. A. Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* (2007), vol. 4, IEEE, pp. IV–317.

[41] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems* (2016), pp. 1109–1117.

[42] Ionides, E. L. Truncated importance sampling. *Journal of Computational and Graphical Statistics 17*, 2 (2008), 295–311.

[43] Kearns, M., and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine learning 49*, 2-3 (2002), 209–232.

[44] Kimura, H. Efficient non-linear control by combining q-learning with local linear controllers. In *Proceedings of the 16th International Conference on Machine Learning* (1999), pp. 210–219.

[45] Kleinberg, R., Slivkins, A., and Upfal, E. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing* (2008), ACM, pp. 681–690.

[46] Kleinberg, R., Slivkins, A., and Upfal, E. Bandits and experts in metric spaces. *arXiv preprint arXiv:1312.1277* (2013).

[47] Kleinberg, R. D. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems* (2005), pp. 697–704.

[48] Kober, J., and Peters, J. R. Policy search for motor primitives in robotics. In *Advances in neural information processing systems* (2009), pp. 849–856.

[49] Kong, A. A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep 348* (1992).

[50] Kupcsik, A. G., Deisenroth, M. P., Peters, J., and Neumann, G. Data-efficient generalization of robot skills with contextual policy search. In *Twenty-Seventh AAAI Conference on Artificial Intelligence* (2013).

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

[52] Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning.* Springer, 2012, pp. 45–73.

[53] Lattimore, T., and Szepesvári, C. *Bandit Algorithms.* Cambridge University Press (preprint), 2019.

[54] Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-Y. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems* (2012), pp. 206–214.

[55] Magureanu, S., Combes, R., and Proutiere, A. Lipschitz bandits: Regret lower bounds and optimal algorithms. *arXiv preprint arXiv:1405.4758* (2014).

[56] Martino, L., Elvira, V., and Louzada, F. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing 131* (2017), 386–401.

[57] Metelli, A. M., Papini, M., Faccio, F., and Restelli, M. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems* (2018), pp. 5447–5459.

[58] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (2016), pp. 1928–1937.

[59] Moré, J. J., and Thuente, D. J. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS) 20*, 3 (1994), 286–307.

[60] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* (2016).

[61] Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems* (2016), pp. 4026–4034.

[62] Osband, I., Russo, D., and Roy, B. V. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* (2013), pp. 3003–3011.

[63] Osband, I., and Van Roy, B. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300* (2015).

[64] Osband, I., and Van Roy, B. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 2701–2710.

[65] Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 2721–2730.

[66] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation 11*, 2 (2007), 265–286.

[67] Owen, A. B. *Monte Carlo theory, methods and examples.* 2013.

[68] Peters, J., and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks 21*, 4 (2008), 682–697.

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

[70] Rényi, A. On measures of entropy and information. Tech. rep., Hungarian Academy of Sciences Budapest Hungary, 1961.

[71] Rummery, G. A., and Niranjan, M. *On-line Q-learning using connectionist systems*, vol. 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[72] Russo, D., and Van Roy, B. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems* (2013), pp. 2256–2264.

[73] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning 11*, 1 (2018), 1–96.

[74] Sallans, B., and Hinton, G. E. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research 5*, Aug (2004), 1063–1088.

[75] Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats* (1991), pp. 222–227.

[76] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning* (2015), pp. 1889–1897.

[77] Scott, D. W. *Multivariate density estimation: theory, practice, and visualization.* John Wiley & Sons, 2015.

[78] Sehnke, F., Osendorfer, C., Rückstiess, T., Graves, A., Peters, J., and Schmidhuber, J. Policy gradients with parameter-based exploration for control. In *International Conference on Artificial Neural Networks* (2008), Springer, pp. 387–396.

[79] Sehnke, F., Osendorfer, C., Rückstiess, T., Graves, A., Peters, J., and Schmidhuber, J. Parameter-exploring policy gradients. *Neural Networks 23*, 4 (2010), 551–559.

[80] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *ICML* (2014).

[81] Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Haifa, Israel, June 2010), J. Fürnkranz and T. Joachims, Eds., Omnipress, pp. 1015–1022.

[82] Still, S., and Precup, D. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences 131*, 3 (2012), 139–148.

[83] Strens, M. A bayesian framework for reinforcement learning. In *ICML* (2000), vol. 2000, pp. 943–950.

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

[85] Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin 2*, 4 (1991), 160–163.

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

[87] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (2000), pp. 1057–1063.

[88] Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration:

A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems* (2017), pp. 2753–2762.

[89] Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika 25*, 3/4 (1933), 285–294.

[90] Thrun, S. B. Efficient exploration in reinforcement learning.

[91] Tornio, M., and Raiko, T. Variational bayesian approach for nonlinear identification and control. In *Proc. of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, NMPC FS06* (2006), Citeseer, pp. 41–46.

[92] Veach, E., and Guibas, L. J. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95* (1995), ACM Press, pp. 419–428.

[93] Watkins, C. J. C. H. *Learning from delayed rewards.* PhD thesis, King's College, Cambridge, 1989.

[94] Wawrzyński, P. *Intensive reinforcement learning.* PhD thesis, Institute of Control and Computation Engineering, Supervisor: Andrzej Pacut„ 2005.

[95] Whitehead, S. D., and Ballard, D. H. *A study of cooperative mechanisms for faster reinforcement learning.* University of Rochester, Department of Computer Science Rochester, NY, 1991.

[96] Williams, C. K., and Rasmussen, C. E. *Gaussian processes for machine learning*, vol. 2. MIT Press Cambridge, MA, 2006.

[97] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning 8*, 3-4 (1992), 229–256.

[98] Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems* (2011), pp. 262–270.

[99] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Aaai* (2008), vol. 8, Chicago, IL, USA, pp. 1433–1438.