POLITECNICO DI MILANO
School of Industrial and Information Engineering
Department of Electronics, Information, Bioengineering
Master of Science in
Computer Science and Engineering

Learning to Explore Multiple Environments via Reward-Free Reinforcement Learning

Supervisor:
MARCELLO RESTELLI

Assistant Supervisor:
MIRCO MUTTI

Master Graduation Thesis by:
MATTIA MANCASSOLA
Student Id n. 928393

Academic Year 2019-2020

A Giulia.

RINGRAZIAMENTI

Ringrazio il Prof. Marcello Restelli e il Dott. Mirco Mutti per il loro costante supporto durante la stesura di questa tesi. I loro preziosi consigli sono stati un contributo fondamentale, non solo in termini di puro contenuto, ma anche di formazione personale.
Ringrazio inoltre la mia famiglia, senza la quale il mio intero percorso universitario sarebbe stato una mera fantasia, e che da sempre rappresenta un punto saldo nel quale trovare sostegno ed affetto.
Ringrazio infine i miei amici e tutti i colleghi di università, che hanno reso preziosi ed insostituibili questi ultimi anni passati insieme.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

i.i.d. independent identically distributed
cdf cumulative density function
RL Reinforcement Learning
k-NN k-Nearest Neighbors
IW Importance-Weighted
KL Kullback-Leibler
TV Total Variation
MDP Markov Decision Process
CMP Controlled Markov Process
VaR Value at Risk
CVaR Conditional Value at Risk
MSVE Maximum State Visitation Entropy
DQN Deep Q-Networks
TRPO Trust Region Policy Optimization
PPO Proximal Policy Optimization
MAML Model-Agnostic Meta-Learning
DIAYN Diversity Is All You Need
UML Unsupervised Meta-Learning
POIS Policy Optimization via Importance Sampling
MEPOL Maximum Entropy POLicy optimization
MEMENTO Multiple Environments Maximum ENTropy Optimization
MORL Multi-Objective Reinforcement Learning
MOMDP Multi-Objective Markov Decision Process

ABSTRACT

Reinforcement learning is a very active area of research that studies how to learn intelligent agents that can solve sequential decision-making problems via simple interactions within an environment while being guided by a feedback signal, called reward. In recent years, it has shared the growth of interest with the other fields of artificial intelligence thanks to the rise of deep learning techniques. However, the area is far from being considered completely solved, as many technical and theoretical challenges still need to be addressed. Among these, we can find the *ability to learn without external feedback*, meaning that the agent should be able, as humans, to learn by interacting with the surrounding environment without being guided by explicit signals, but only because of intrinsic motivation. Another crucial point of interest is represented by the *ability of the agent to generalize its behavior among different environments*. The purpose of this thesis is to deal with both these aspects, combining them in a single setting. We address the problem of learning to explore a class of multiple reward-free environments with a unique general strategy, which aims to provide a universal initialization to subsequent reinforcement learning problems specified over the same class. Notably, the problem is inherently multi-objective as we can trade off the exploration performance between environments in many ways. In this work, we foster an exploration strategy that is sensitive to the most adverse cases within the class. Hence, we cast the exploration problem as the maximization of the mean of a critical percentile of the state visitation entropy induced by the exploration strategy over the class of environments. Then, we present a policy gradient algorithm, MEMENTO, to optimize the introduced objective through mediated interactions with the class. Finally, we empirically demonstrate the ability of the algorithm in learning to explore challenging classes of continuous environments and we show that reinforcement learning greatly benefits from the pre-trained exploration strategy when compared to learning from scratch.

ESTRATTO IN LINGUA ITALIANA

L'apprendimento per rinforzo [72] è una tecnica di apprendimento automatico, avente come proposito quello di realizzare agenti autonomi, che siano in grado di prendere decisioni a seconda dello stato in cui si trovano. La modalità attraverso cui l'agente apprende tale capacità consiste nell'interazione con l'ambiente in cui è immerso. Solitamente, a seguito di un'interazione, segue un cosiddetto segnale di rinforzo, ovvero una ricompensa o una penalità per l'agente, rispettivamente nel caso in cui esso abbia preso una decisione che sia conforme al comportamento desiderato oppure no. In questo modo, l'agente, dopo un certo numero di interazioni con l'ambiente, sarà in grado di prendere la decisione che, secondo quanto appreso sino a quel momento, porti ad un segnale di rinforzo positivo. Il campo di applicazione dell'approccio appena descritto è molto vasto e spazia dai bot di trading ai rover in missione spaziale. In linea di massima, infatti, ogni entità in grado di elaborare strategie, naturale o artificiale che sia, può essere modellata come un agente immerso in un ambiente, in grado di prendere decisioni e dunque di guadagnare o perdere qualcosa.

MOTIVAZIONE La tecnica di apprendimento tramite segnale di rinforzo, come detto, è stata impiegata con successo in numerosi settori, anche in domini caratterizzati da una complessità non indifferente [69, 7, 2]. Tuttavia, per quanto intuitiva, la metodologia suddetta non è di semplice attuazione. La difficoltà maggiore deriva dal fatto che solitamente la condizione da soddisfare affinché l'agente riceva un segnale positivo o negativo viene introdotta da un progettista umano. Si tratta dunque di una forma di supervisione che comporta una sfida notevole, siccome nella maggior parte dei casi non è semplice progettare una funzione che permetta all'agente di ottenere il comportamento desiderato. È infatti pratica comune, per ovviare a tale complessità, premiare l'agente solo in caso di successo, inserendo un segnale di rinforzo sparso. Potremmo ad esempio premiare l'agente ogni volta che riesce ad afferrare un oggetto e a collocarlo in una determinata posizione. Sulla base di quanto detto finora, è naturale l'estensione ad uno scenario in cui il segnale di rinforzo sia invece completamente assente. In una situazione di questo tipo, l'agente deve essere in grado di esplorare l'ambiente senza la necessità di ricevere qualcosa in cambio, bensì potremmo dire per pura curiosità, così come farebbe un neonato se lasciato in mezzo ad una stanza con dei giocattoli. L'esplorazione che ne deriva consentirà all'agente di essere più malleabile e pronto a situazioni future, in cui dovrà apprendere nuove abilità più complesse e strutturate, anche attraverso l'utilizzo di segnali di rinforzo.

AMBIENTI MULTIPLI Quest'ultimo approccio offre quindi un vantaggio notevole in termini di flessibilità e capacità di generalizzazione. Tuttavia, questa generalizzazione è solitamente limitata ad un singolo ambiente. Un ulteriore tassello di complessità può essere aggiunto considerando un numero di ambienti che sia maggiore di uno. Sino ad ora, infatti, pur implicitamente, abbiamo preso in considerazione un agente immerso in un ambiente, in cui impara sulla base delle interazioni con esso. La limitazione di questo metodo sta nel fatto che l'agente potrebbe acquisire delle abilità che, se applicate in un contesto diverso, non porterebbero allo stesso risultato o addirittura che comporterebbero una drastica riduzione dell'efficacia.

SENSITIVITÀ AL RISCHIO L'aspetto di cui sopra risulta particolarmente cruciale in domini caratterizzati dalla presenza di frangenti relativamente rari e molto sfavorevoli. In tali circostanze, vorremmo che il nostro modello di agente fosse abbastanza robusto da poter gestire sia il caso medio sia il caso peggiore. Numerosi approcci sono stati proposti, tra cui il cosiddetto *Valore a Rischio Condizionato* o *Conditional Value at Risk (CVaR)* [62]. Sostanzialmente, durante l'apprendimento, esso riduce il proprio obiettivo ad una certa percentuale dei possibili casi peggiori, individuati in base alla metrica che stiamo utilizzando per valutare la prestazioni del nostro modello. L'idea di fondo è che un agente che consideri, o addirittura incentri, il suo apprendimento su tale misura di rischio possa comportarsi adeguatamente nel caso in cui in futuro si venga a trovare in una situazione appartenente a quella percentuale di casi più sfavorevoli.

STATO DELL'ARTE Nella letteratura sono stati proposti svariati metodi per costruire agenti in grado di esplorare intrinsecamente, ovvero senza necessità di segnali esterni [71, 10, 56, 13, 25]. Lo stato dell'arte affronta il problema con un'esplorazione *agnostica rispetto al compito*, volta cioè a garantire un'esplorazione il più uniforme possibile, di modo che l'agente generalizzi rispetto ai compiti che gli verranno assegnati in futuro [36, 46, 86, 51, 30, 67]. Questa soluzione, tuttavia, non considera uno scenario con più ambienti. Vi è poi una vasta letteratura volta a costruire agenti in grado di far fronte a situazioni avverse. Vi sono algoritmi che propongono l'ottimizzazione del CVaR per gestire il rischio indotto dalla volatilità dei segnali di rinforzo [74, 17] o da cambiamenti nella dinamica dell'ambiente [61]. Attualmente, tuttavia, nessuno sembra aver impiegato tale misura per prendere in esame la difficoltà dovuta alla presenza di una configurazione particolarmente avversa all'interno di una classe di ambienti. Infine, esiste una branca relativamente recente di apprendimento con rinforzo, chiamata *meta reinforcement learning* [82, 23, 26], nella quale l'obiettivo consiste nel fare in modo che l'agente impari ad imparare. Più precisamente, l'intento è quello di creare agenti che siano in grado di apprendere un nuovo compito facendo affidamento su altri compiti imparati sino a quel momento, facendo leva sullo sviluppo di una capacità di adat-

tamento rapido tra un compito e l'altro. Sebbene questo paradigma sia già stato utilizzato per affrontare il problema dell'esplorazione, gli algoritmi proposti assumono la presenza di un segnale di rinforzo, con l'eccezione di [31]. Ad ogni modo, nessuno prende in considerazione uno scenario caratterizzato da una molteplicità di ambienti.

OBIETTIVO Questa tesi si pone come obiettivo quello di estendere la generalizzazione dell'apprendimento in contesti privi di rinforzo, affrontando il problema dell'esplorazione di più ambienti con un'unica strategia. Oltre a definire formalmente il problema, ne forniamo una caratterizzazione teorica, offrendo una metrica per quantificare la difficoltà di esplorazione in una classe di ambienti. Inoltre, mostriamo come e quando un agente possa beneficiare dall'approccio proposto per risolvere qualunque compito futuro in uno qualsiasi degli ambienti della classe, riducendo l'onere della progettazione di un appropriato segnale di rinforzo e migliorando allo stesso tempo l'efficienza di apprendimento.

CONTRIBUTI In questa tesi forniamo i seguenti contributi:

- Formuliamo il problema dandone una definizione formale in un contesto avverso al rischio;
- Proponiamo un obiettivo per l'esplorazione senza segnali di rinforzo all'interno di una classe di ambienti, per ottenere un agente in grado di risolvere qualunque compito futuro definito trasversalmente all'interno della classe stessa;
- Presentiamo un nuovo algoritmo per creare agenti che agiscano in modo ottimale rispetto all'obiettivo introdotto;
- Mostriamo i meriti della soluzione proposta attraverso un'estesa analisi empirica, comprendente una serie di esperimenti su domini continui a bassa e alta complessità.

STRUTTURA DELLA TESI La tesi è organizzata come segue. Nel Capitolo 2, provvediamo a fornire al lettore tutte le conoscenze necessarie per una piena comprensione dei capitoli successivi. In particolare, introdurremo le principali misure entropiche della teoria dell'informazione, alcune metriche utili a calcolare la distanza tra distribuzioni di probabilità, e altre ancora volte a fornire una misura del rischio. Inoltre, verranno presentati i concetti alla base dell'apprendimento per rinforzo, successivamente estesi mediante una generalizzazione di tale paradigma. Nel Capitolo 3, descriviamo l'attuale stato dell'arte nell'affrontare i problemi di esplorazione con segnale di rinforzo sparso e in assenza di segnale di rinforzo, di apprendimento avverso al rischio, e di generalizzazione tramite adattamento rapido. Successivamente, nel Capitolo 4, presentiamo i nostri contributi, quali una

formulazione del problema in un'ottica avversa al rischio, una caratterizzazione teorica del problema, la formulazione di un obiettivo che si confà all'esigenza di esplorare una classe di ambienti in assenza di segnale di rinforzo, e un algoritmo di apprendimento che lo ottimizzi. Nel Capitolo 5 procediamo fornendo una esaustiva analisi sperimentale della metodologia introdotta, includendo domini continui a bassa e alta complessità. Infine, nel Capitolo 6, concludiamo con un breve riepilogo e alcune riflessioni su possibili sviluppi futuri. Nell'Appendice A, riportiamo le dimostrazioni dei teoremi che sono state omesse dal testo principale per non appesantire l'esposizione. Nell'Appendice B, includiamo alcuni esperimenti aggiuntivi, aventi un'impostazione differente e che, non essendo centrali per questa tesi, non sono stati inseriti nella discussione principale. Concludendo, nell'Appendice C, completiamo le informazioni del Capitolo 5 con ulteriori dettagli implementativi.

1

INTRODUCTION

Reinforcement learning [72] is an autonomous learning technique that is aimed at building intelligent agents that are able to make sequential decisions while interacting with an environment in which they are immersed. Typically, the interaction is guided by a reinforcing signal, which informs the agent whether the decision is compliant with the desired behavior or not. In this way, the agent will be capable of taking the most remunerative decisions according to the previous interactions. The field of application of this approach is wide and it ranges from trading bots to rovers for space missions. In fact, in principle, every natural or artificial strategic entity can be modeled as an agent immersed in an environment, capable of taking decisions and therefore to earn or lose something.

INTRINSIC MOTIVATION Reinforcement learning, as we said, has been proved to be successful in many fields, including domains characterized by a high-level complexity [69, 7, 2]. Nonetheless, albeit intuitive, such methodology is not always easy to implement. A major obstacle is given by the reward signal, which is typically introduced by a human designer. Thus, it is a form of supervision that involves a significant challenge, as in most cases it is not easy to design a reward function inducing the desired behavior to the agent. In fact, it is common to avoid such complications by rewarding the agent only in case of success, hence introducing a sparse reward signal. For instance, we could reward the agent every time it manages to grab an object and place it in a specific position. Moreover, it is natural to imagine a scenario in which the reinforcing signal is instead completely absent. In such context, the agent should be able to explore the environment without the need to receive something in return, but we could say for mere curiosity, as a child would do if he/she was left in the middle of a room with some toys. The deriving explorative behavior will enable the agent to be more malleable and ready for future circumstances, in which it will have to learn new complex and more structured abilities, also by means of reinforcing signals.

MULTIPLE ENVIRONMENTS The latter approach offers a significant advantage in terms of flexibility and generalization. However, it is usually limited to a single environment. Considering more than one environment introduces additional complexity. In fact, even though implicitly, so far we contemplated an agent immersed in a single environment, learning from the interactions with it. The limitation of

this method lies in the fact that the agent could learn to behave in a way that is not helpful for other contexts, or even harmful with respect to learning from scratch.

RISK-SENSITIVITY The issue raised above is crucial in domains that are characterized by the presence of relatively rare and particularly unfavourable scenarios. In such circumstances, we would rather have a model that is both good on average and robust to the worst case. Several approaches have been proposed to measure such risk, e.g., the so-called *Conditional Value at Risk* (CVaR) [62]. Essentially, during the learning phase, it considers only a certain percentage of the worst possible cases, identified by means of the metric that we are using to evaluate our model. The underlying intuition is that an agent that weighs or even centers its learning on such measure of risk could behave adequately in a worst-case situation, avoiding a severe decrease in the performance.

STATE OF THE ART Several methods have been proposed in the literature to build agents able to intrinsically explore the environment, i.e., without the need for external signals [71, 10, 56, 13, 25]. The state of the art addresses the problem with a *task-agnostic* exploration, namely aimed at guaranteeing an exploration that is as uniform as possible over the states of the environment [36, 46, 86, 51, 30, 67]. In this way, the agent can generalize with respect to the tasks that it will have to handle in the future. Nevertheless, this solution does not contemplate a scenario with multiple environments. There is also wide literature aimed at building agents that are able to cope with adverse situations. Some algorithms propose an optimization for the CVaR to manage the risk induced by the volatility of reinforce signals [74, 17] or by changes in the environment dynamics [61]. However, none of them is directly studying the difficulty caused by the presence of a particularly adverse configuration inside a class of environments. Finally, *meta reinforcement learning* [82, 23, 26] aims at producing agents that learn to learn. More precisely, the intent is to create agents that are able to learn a new task by relying on other tasks learned in the past. Thus, the idea is to learn how to quickly adapt from task to task. However, current approaches still assume the presence of a reward signal, with the remarkable exception of [31]. In any case, to the best of our knowledge, none of the existing works combine reward-free meta-training with a multiple-environments setting.

GOAL The goal of this thesis is to extend the generality of reward-free learning by addressing the problem of learning to explore multiple environments with a single exploration strategy. In addition to providing a formal definition of the problem, we also formulate a theoretical characterization, in which we propose a metric to quantify the difficulty of exploration of a class of environments. Moreover, we show

how and when an agent can benefit from the proposed approach to solve any subsequent task specified over any environment in the class, reducing the burden on manual reward design while improving learning efficiency over reinforcement learning from scratch.

CONTRIBUTION In this thesis, we provide the following contributions:

- We provide a formal definition of the problem in a risk-averse fashion;
- We propose a new objective for exploration over a class of reward-free environments, in order to obtain an agent that is able to solve any subsequent task defined over any environment in the class;
- We present a novel algorithm to build optimal behaving agents with respect to this objective;
- We show the merits of the proposed solution by means of an extensive empirical analysis, which includes experiments on both low-dimensional and high-dimensional complexity.

OVERVIEW The thesis is structured as follows. In Chapter 2, we provide the essential background to completely understand the subsequent chapters. In particular, we introduce the main entropic measures of the information theory, some metrics useful to compute the distance between probability distributions, and some others aimed at providing a measure of risk. Besides that, we present the concepts that are at the basis of reinforcement learning, which we then extend by taking into consideration a generalization of such paradigm. In Chapter 3, we describe the current state of the art to address the problems of sparse reward and reward-free exploration, risk-averse learning, and generalization via fast-adaptation. Then, in Chapter 4, we provide a formal definition of the problem in a risk-averse manner, accompanied by a theoretical characterization. We also formulate an objective suitable to the requirement of exploring a class of reward-free environments, and a learning algorithm to optimize it. In Chapter 5, we proceed with an exhaustive experimental evaluation of the introduced methodology, including domains with both low-dimensional and high-dimensional complexity. Finally, in Chapter 6, we conclude with a summary and some thoughts about possible future developments. In Appendix A, we report the proofs of the theorems that we omitted from the main text to avoid weighing the exposition. In Appendix B, we include an evaluation of an orthogonal setting with respect to the one we focus in this thesis. To conclude, in Appendix C, we complete the information of Chapter 5 with some further implementation details.

2

ESSENTIAL BACKGROUND

In this chapter, we provide all the concepts that are crucial to understand the content of the thesis. In Section 2.1, we introduce the information entropy measure in its different formulations and a way to compute its estimate. Furthermore, we present two of the most common measures of risk and two metrics to compute the distance between probability distributions, that we will use to theoretically characterize the problem in Section 4.3. Then, in Section 2.2, we gradually present the reinforcement learning framework. Finally, in Section 2.3, we discuss a generalization of standard reinforcement learning.

2.1 PROBABILITY DISTRIBUTIONS AND INFORMATION ENTROPY

In this section, we introduce the information entropy measure, developed by Claude Shannon in [68]. We start from the discrete case and then proceed with the continuous one. Afterwards, we discuss some entropy estimators that we will take into consideration in the following sections. Finally, we describe two common measures of risk, which allow to quantify the performance that can be obtained in the tail of a distribution.

2.1.1 *Discrete Entropy*

Let X be a discrete random variable with possible outcomes $\{x_1, \dots, x_N\}$, whose probability is denoted by $P(x_i)$. The entropy of X is:

$$H(X) = -\sum_{i=1}^{N} P(x_i) \log_b P(x_i), \tag{2.1}$$

where b is the logarithm base. From now on, we consider $b = e$, the Euler's number, hence the natural logarithm. A unit of entropy is called "nat" in this case.

The entropy measure is at the base of information theory and of ergodic theory. In general, it quantifies the expected information we need to determine the value of a random variable. If this quantity is high, the entropy is high. That is why it can be also interpreted as a measure of the disorder of a random variable. We can read the concept of disorder as unpredictability of its outcome. Intuitively, the surprise deriving from the outcome of a random variable X is higher if we

have no reasons to suspect that one outcome is more probable than the others. In fact, the entropy is maximal when the $P(x_i)$ follows a uniform distribution, meaning that all outcomes are equally possible. It is instead minimal when $P(x_i)$ follows a Dirac distribution, meaning that it is equal to zero everywhere except in one value.

2.1.2 *Differential Entropy*

The differential entropy of a continuous random variable X with probability density function $p(x)$ is defined as:

$$h(X) = -\int_{\mathcal{X}} p(x) \log p(x) dx, \tag{2.2}$$

where $\mathcal{X}$ is the support region of the random variable. Notably, differential entropy is not a direct extension of discrete entropy. Going from the discrete case to the continuous one requires to introduce a reference measure. The consequence is that we lose many nice properties. For example, differential entropy is neither (necessarily) non-negative nor (necessarily) invariant to coordinate transformations.

The Gaussian probability distribution plays a major role in information theory. Interestingly, of all the probability distributions with variance σ^2, it can be shown [9] that it has the largest differential entropy. The differential entropy of a multivariate Gaussian is:

$$H(X) = \frac{p}{2} + \frac{p}{2} \log(2\pi) + \frac{1}{2} \log(|\Sigma|),$$

where p is the length of the vectors whose components are independent identically distributed (i.i.d.) Gaussian random variables, and $|\Sigma|$ is the determinant of the covariance matrix. Note that the entropy increases as the variance increases. This is not a surprise, since the variance is indeed a measure of the uncertainty.

2.1.3 *Relative Entropy*

Some of the difficulties associated with Shannon entropy in the continuous case can be mitigated by relative entropy, which is also called Kullback-Leibler (KL) divergence [43], and it is defined as:

$$D_{KL}(P||Q) = \int_{\mathbb{R}^k} p(x) \log \frac{p(x)}{q(x)} dx, \tag{2.3}$$

where $p(x)$ and $q(x)$ are the probability densities of the distributions P and Q respectively. It is always non-negative, and equality holds only when the two distributions are equivalent. However, note that it is not a true distance between distributions, as it is not symmetric and it does not satisfy the triangle inequality. It is rather a measure of entropy increase due to the use of an approximation to the true

distribution instead of the true distribution itself. We will indeed use this measure to deal with a scenario in which P is a target distribution and Q is the distribution with which we approximate P.

Note that if we compute the relative entropy of a given distribution with given mean and variance with respect to a Gaussian distribution with the same mean and the same variance, we obtain a measure that indicates how much our distribution is "non-normal", which can be written as:

$$D_{KL}(P_X \,||\, \mathcal{N}(\mu_X, \sigma_X^2)) = \log 2\pi e\sigma^2 - h(X),$$

where $h(X)$ is the differential entropy. This shows again that the Gaussian distribution is the one that maximizes the differential entropy.

2.1.4 *Non-parametric Differential Entropy Estimation*

If we do not know the density function f of a distribution, we can still obtain an estimate of the differential entropy given a realization of $X = \{x_t\}_{t=0}^{T-1}$ [4]. Especially, we can turn to non-parametric, k-Nearest Neighbors (k-NN) entropy estimators of the form [70]:

$$\widehat{H}_k(f) = -\frac{1}{T}\sum_{t=0}^{T-1} \ln \frac{k \cdot \Gamma(\frac{p}{2}+1)}{N \cdot |x_t - x_t^{k-NN}|^p \cdot \pi^{\frac{p}{2}}} + \ln k - \Psi(k), \tag{2.4}$$

where $\log k - \Psi(k)$ is a bias correction term in which Ψ is the Digamma function, and p is the dimension of the random vector. The estimator is known to be asymptotically unbiased and consistent [70]. As mentioned in Section 2.1.3, in some cases we might have samples from a sampling distribution f that is different from the target distribution f'. We can provide an estimate of $H(f')$ by means of an Importance-Weighted (IW) k-NN estimator of the form [1]:

$$\widehat{H}_k(f'|f)^{IW} = -\sum_{t=0}^{T-1} \frac{\sum_{j\in\mathcal{N}_t^k} w_j}{k} \ln \frac{\Gamma(\frac{p}{2}+1)\sum_{j\in\mathcal{N}_t^k} w_j}{|x_t - x_t^{k-NN}|^p \cdot \pi^{\frac{p}{2}}} + \ln k - \Psi(k), \tag{2.5}$$

where $\mathcal{N}_t^k$ is the set of indices of the k-NN of x_t, and w_j are the normalized importance weights of samples x_j, which are defined as:

$$w_j = \frac{f'(x_j)/f(x_j)}{\sum_{t=0}^{T-1} f'(x_t)/f(x_t)}.$$

The estimator in (2.5) is known to be asymptotically unbiased for any choice of k [51, 1]. We can also estimate the Kullback-Leibler (KL) divergence, described in Section 2.1.3, as:

$$\widehat{D}_{KL}(f||f') = \frac{1}{T}\sum_{t=0}^{T-1} \ln \frac{k/T}{\sum_{j\in\mathcal{N}_t^k} w_j}. \tag{2.6}$$

Note that, when $f' = f$, $\widehat{D}_{KL}(f||f') = 0$ and $\widehat{H}_k(f'|f)^{IW} = \widehat{H}_k(f)$.

2.1.5 *Wasserstein and Total Variation Metrics*

We now present two common metrics for measuring closeness between two probability distributions. Recall that, for the reasons pointed out in Section 2.1.3, the KL divergence is not a true distance.

The first metric is the *Total Variation* (TV). It is a widely used probability metric, mainly because it admits useful bounding techniques and also very natural interpretations (it ranges in $[0, 1]$). It is defined as:

$$d_{TV}(p, q) = \frac{1}{2} \int_X |\, \mathrm{d}(p - q)|. \tag{2.7}$$

Before we continue, it is worth introducing an additional concept, which will be resumed later on. Let X, Y be two metric sets with metric functions d_X, d_Y. We say a function $f : X \to Y$ is L_f-*Lipschitz continuous* if it holds for some constant L_f

$$d_Y(f(x'), f(x)) \leq L_f d_X(x', x), \quad \forall (x', x) \in X^2, \tag{2.8}$$

where the smallest L_f is the Lipschitz constant and the Lipschitz semi-norm is $\|f\|_L = \sup_{x', x \in X} \left\{ \frac{d_Y(f(x'), f(x))}{d_X(x', x)} : x' \neq x \right\}$.

We can now detail the second metric, which is the *Wasserstein* [41, 81]. Let p, q be two probability measures on a measurable space $(X, \mathcal{F})$, it can be defined as:

$$d_{W_1}(p, q) = \sup_f \left\{ \left| \int_X f \, \mathrm{d}(p - q) \right| : \|f\|_L \leq 1 \right\}. \tag{2.9}$$

2.1.6 *Value at Risk and Conditional Value at Risk*

In the literature, many risk measures have been introduced to quantify the inherent risk in a certain setting. One of the most successful one is the *Conditional Value at Risk* (CVaR) [62]. The CVaR is widely used in the world of finance, but in the last years it has been adopted in many other fields and it will play a central role in this thesis.

Before talking about the CVaR, however, it is worth introducing another measure of risk first, which is called *Value at Risk* (VaR). Formally, we can define the VaR as:

$$VaR_\alpha(X) = inf \, \{x \mid F_X(x) \geq \alpha\}, \tag{2.10}$$

where X is a random variable distributed according to a cumulative density function (cdf) $F_X(x) = Pr(X \leq x)$, and $\alpha \in (0, 1)$ is the confidence level. Essentially, given a continuous distribution, the VaR measures the risk as the maximum value with respect to a given

confidence level α. Typically, with "value" one means a cost or a loss, but it can be whatever other metric, e.g., entropy.

Now, we can go back to the CVaR, that can be defined as:

$$CVaR_\alpha(X) = \mathbb{E}[X \mid X \leq VaR_\alpha(X)]. \tag{2.11}$$

Interestingly, the two terms are clearly related. In fact, even though VaR has been also widely used as a risk measure, CVaR has taken over in the last years thanks to its computational advantages and to superior theoretical properties. For instance, CVaR is a coherent risk measure, satisfying the properties of monotonicity, sub-additivity, homogeneity, and translational invariance [58]. On the contrary, VaR is coherent only when it is based on the standard deviation of a normal distribution. In Figure 2.1 we provide an illustration of the two risk measures.

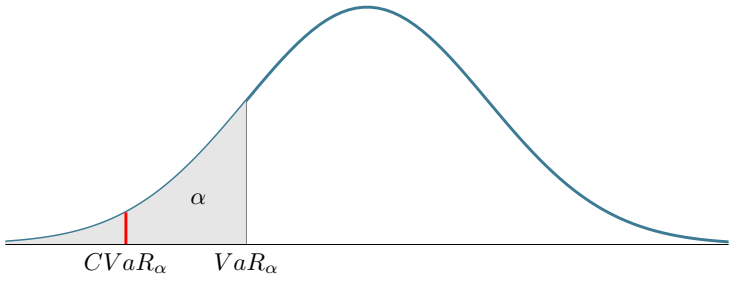


Figure 2.1: Graphic illustration of the VaR_α and $CVaR_\alpha$ terms.

If we specify X to be the random variable for some metric, which is typically but not necessarily a loss, the CVaR is nothing but the expected value $\mathbb{E}[X]$ that we would obtain in the bottom α-percentile of the possible outcomes, i.e., what we call the worst case. Intuitively, by optimizing for it, we are shifting our focus on the left tail-end of the distribution, with the aim of moving to the right the red line.

2.2 REINFORCEMENT LEARNING AND MARKOV DECISION PROCESSES

In this section, we introduce the reader to the Reinforcement Learning (RL) field. First, we provide a brief characterization of the area. Then, we present the core elements of RL, namely those components and abstractions that define its functioning. Finally, we explain how a RL problem can be addressed, by supplying an overview of the main algorithms and paradigms.

2.2.1 *Reinforcement Learning*

Reinforcement Learning is concurrently a problem, a set of methods to solve the problem, and the field that studies the problem and

the methods. The problem can be formalized as a *sequential decision-making* problem, i.e., a setting in which a decision-maker has to make a sequence of decisions in order to achieve a goal. The result of each decision can be deterministic or non-deterministic and in some cases delayed in time. Many real-world situations can be represented as a sequential decision-making problem, ranging from automatic control to robotics or operational research. The Markov Decision Process (MDP) [72] is a mathematical framework used to model sequential decision-making problems. The intent of an MDP is to capture the most important aspects coming from the interaction between the decision-maker and the environment in which it is immersed, which are observations, actions and reward signals. Reinforcement learning is then considering these aspects and learning what to do, namely how to learn a behavior that associates an observation to an action in order to maximize the reward from sampled interaction, i.e., without the knowledge of the MDP.

2.2.2 *Markov Decision Processes*

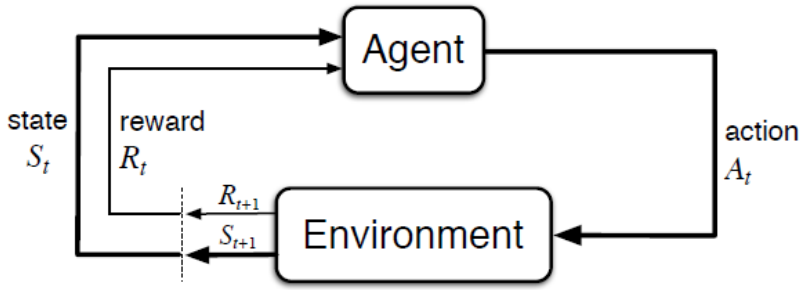


Figure 2.2: The agent-environment interaction in a Markov decision process, from [72].

In an MDP, an agent, i.e., the decision-maker, repeatedly interacts with an environment. At each step t, the agent makes an action A_t according to the current state S_t. At that point, the environment answers by returning a new state S_{t+1} and a reward signal R_{t+1}. The first becomes the new state of the agent, while the second provides an evaluation of the taken action A_t. Then the loop starts again. An illustration of this process is shown in Figure 2.2.

More formally, we can define an MDP as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0 \rangle$, where:

- $\mathcal{S} \subseteq \mathcal{R}^{n_s}$ is a n_s-dimensional continuous space, named *state space*;
- $\mathcal{A} \subseteq \mathcal{R}^{n_a}$ is a n_a-dimensional continuous space, named *action space*;

- $\mathcal{P}$ is a function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ [1], named *transition model*, where $P(s'|s, a)$ is the probability of reaching the next state s' given the values of the preceding state s and action a;
- $\mathcal{R}$ is a function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{R})$, named *reward model*, where $R(s, a)$ is the expected scalar reward obtained after executing the action a in the state s;
- $\gamma \in [0, 1]$ is the *discount factor* for future rewards;
- $d_0 \in \Delta(\mathcal{S})$ is the *initial state distribution*.

For completeness of notation, we define the *state-action space* $\mathcal{SA}$ as the Cartesian product between the state space and the action space. We can now define a *trajectory* as a sequence $\langle s_t, a_t, r_{t+1}\rangle_{i=0,\dots,T}$, where s_t, a_t, r_{t+1} are respectively the current state, the taken action and the resulting reward. The length of a trajectory is called *time horizon* T and it can be either finite or infinite. The notation above considers a state and action space that are both continuous. However, it is worth noting that they can be formalized as discrete sets. In this case, the MDP is said to be *finite*, since the sets $\mathcal{S}, \mathcal{A}, \mathcal{R}$ all have a finite number of elements.

The function $\mathcal{P}$, i.e., the transition model, defines the *dynamics* of the MDP. Generally, there are two main characteristics which are proper to the dynamics of an environment. The first one is called *Markov property*. It is satisfied when the probability of the next state depends only on the immediately preceding state and action and not from the earlier ones. Essentially, it means that the state s_t comprises all the information about the trajectory $s_0, \dots, s_{t-1}$ which are useful to make future decisions:

$$Pr(S_{t+1} = s_{t+1}|S_t = s_t, \dots, S_0 = s_0) = Pr(S_{t+1} = s_{t+1}|S_t = s_t). \quad (2.12)$$

The second characteristic is called *stationarity*. An MDP is stationary if its dynamics does not change over time. From now on, we will assume stationarity to be satisfied, but it is worth to mention that in the literature also non-stationary transition model have been considered [12].

The function $\mathcal{R}$, i.e., the reward model, is responsible for the output of the reward signal $R(s, a)$, thus it plays a central role in learning a behavior that satisfies our expectations. The reward signal is indeed the main factor capable of changing the way in which the agent behaves: if an action is followed by a low reward, when the agent will find itself in the same situation in the future, it will probably opt for another action. Note that the agent's goal consists of maximizing the total amount of rewards it receives. This means that this process is evaluated in the long term, hence maximizing a cumulative reward. Formally, we can define the latter by introducing the concept of *return* G_t:

$$G_t = R_{t+1} + R_{t+2} + \dots + R_T. \quad (2.13)$$

1 $\Delta(X)$ denotes the batch of distributions over the set of X.

The above is called *finite-horizon undiscounted return*, and it is just the sum of rewards obtained in a fixed window of steps. However, as we mentioned before, the time horizon T is not necessarily finite. There are cases in which the agent-environment interaction does not split in what are called *episodes*, i.e., sub-sequences of states, actions and rewards with a terminal state, but instead goes on indefinitely. In those cases, an *infinite-horizon discounted return* is used, defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.14}$$

where γ is the discount factor. The latter has two main advantages. First, it is an intuitive way to account for the uncertainty about the future: the further we look in the future, the more uncertain we are. This is why we discount future rewards multiplying them by γ^k (recall that $\gamma \in [0, 1]$). Second, it is mathematically convenient, because it makes the infinite-sum in (2.14) to converge under reasonable conditions.

It is worth to say that there exist other versions of the MDP framework, according to the setting we are using. For instance, if we do not have access to a reward function $\mathcal{R}$, we resort to a so-called *Controlled Markov Process* (CMP), which therefore assumes the form of a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, d_0 \rangle$.

To conclude, by recalling the definition of Lipschitz continuity in (2.8), we can provide the definition of Lipschitz MDP. A Lipschitz MDP is a standard MDP augmented by a Lipschitz continuous transition model and reward model, i.e.:

$$\begin{aligned} &\forall (s, s', a, a') \in \mathcal{S}^2 \times \mathcal{A}^2, \\ &\qquad d_{W_1}(\mathcal{P}(\cdot|s, a), \mathcal{P}(\cdot|s', a')) \leq L_{\mathcal{P}}\, d_{\mathcal{SA}}((s, a), (s', a')), \\ &\forall (s, s', a, a') \in \mathcal{S}^2 \times \mathcal{A}^2, \\ &\qquad |\mathcal{R}(s, a) - \mathcal{R}(s', a')| \leq L_{\mathcal{R}}\, d_{\mathcal{SA}}((s, a), (s', a')). \end{aligned} \tag{2.15}$$

2.2.3 *Policies*

So far, we have not given a name to the behavior learned by the agent during its interaction with the environment. This decision strategy is called a *policy*. Formally, a policy is a Markovian stationary function of the form

$$\pi : \mathcal{S} \to \Delta(\mathcal{A}). \tag{2.16}$$

It can be defined also as a distribution over actions given states. If the agent is following the policy π, it means that the probability that A_t given that $S_t = s$ is defined as $\pi(a|s) = P[A_t = a|S_t = s]$. A policy can be *deterministic* or *stochastic*. In the former case, for each state $s \in \mathcal{S}$ there exists some action $a \in \mathcal{A}$ such that $\pi(a|s) = 1$. Instead, the latter allows to define also randomized decision strategies.

An agent that follows a policy π on an MDP induces a probability distribution over the state space $\mathcal{S}$, which determines the probability of being in a state s at a given time-step t. Recalling that d_0 is the initial state distribution of an MDP, we can define the induced state distribution as:

$$d_t^\pi(s) = Pr(s_t = s|\pi) = \int_{\mathcal{S}} d_{t-1}^\pi(s') \int_{\mathcal{A}} \pi(a|s') Pr(s|s', a) da ds', \quad (2.17)$$

or, indicating with τ a generic trajectory:

$$d_t^\pi(s) = Pr(s_t = s|\pi) = \int_{\mathcal{T}} Pr(\tau, s_t = s|\pi) d\tau, \quad (2.18)$$

where

$$Pr(\tau, s_t = s|\pi) = d_0(s_0) \prod_{i=0}^{t} \pi(a_i|s_i) Pr(s_{i+1}|s_i, a_i). \quad (2.19)$$

If the MDP satisfies certain conditions (i.e., it is aperiodic and recurrent [60]), it admits a unique steady-state distribution called *steady-state distribution* d^π. The steady-state distribution is the fixed point of Equation (2.17), which means that, for a finite time-step t, the state distribution remains the same from t thereafter:

$$d^\pi(s) = \lim_{t \to \infty} d_t^\pi = d^\pi(s). \quad (2.20)$$

Many real-world problems that we would like to solve with RL have a huge or even infinite number of states and actions. In such cases, it becomes impossible to define the policy for every state-action. Thus, we resort to *function approximation*, meaning that we sample from a desired function to build an approximation of it. In other words, we want to obtain a more compact representation that generalizes across states and actions. The function to be approximated can be a policy or a *value function* (or both). The concept of value function will be introduced in Section 2.2.4. We propose two of the most common representations of stochastic policies, which are *categorical* policies and *diagonal Gaussian* policies. As the name can suggest, categorical policies are used when the action space is discrete, while diagonal Gaussian policies are used with continuous action spaces.

2.2.3.1 *Categorical Policy*

A categorical random variable is a discrete random variable with more than two possible outcomes. It thus follows that a categorical distribution is essentially an extension of the Bernoulli distribution. With that said, we can easily define a categorical policy as a classifier over discrete actions. Especially, a *softmax function* is used to transform the logits flowing out from our model $f(\boldsymbol{\theta}, s)$ (e.g., a neural network) into a vector of action probabilities:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{exp(f(\boldsymbol{\theta}, s))}{\sum_{i=0}^{|\mathcal{A}|-1} exp(f_i(\boldsymbol{\theta}, s))}. \quad (2.21)$$

2.2.3.2 *Diagonal Gaussian Policy*

A multivariate Gaussian distribution is characterized by a mean vector $\mu_{\boldsymbol{\theta}}$, and a covariance matrix Σ. A diagonal Gaussian policy can be defined through a multivariate Gaussian distribution as:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi^n|\Sigma|}} exp(-\frac{1}{2}(a-\mu_{\boldsymbol{\theta}})^T\Sigma^{-1}(a-\mu_{\boldsymbol{\theta}})), \tag{2.22}$$

where $|\Sigma|$ is the determinant of the covariance matrix and $n = n_a$ is the dimension of the continuous action vector. The Gaussian policy might be supported by a neural network that maps the observations to the mean actions $\mu_{\boldsymbol{\theta}}(s)$. The covariance matrix can be represented in different ways, but in all cases the output is the logarithm of the standard deviations. The reason is that $\log\sigma$ can take values in $(-\infty, \infty)$, hence it makes the process of training easier without losing any information.

2.2.4 *Value Functions*

At this point, we can introduce a concept that is widely used in RL, namely the *value function*. If the reward tells the agent the short-term utility of taking an action in a given state, a value function evaluates a state according to the long-term chain of rewards that can be accumulated in the future starting from that state. More precisely, the *state value function* of a state s under a policy π is the expected return when starting in s and then following π:

$$V^\pi(s) = \mathop{\mathbb{E}}_{\pi}[G_t|S_t = s], \tag{2.23}$$

which we can define by means of the Bellman expectation equation [6]:

$$V^\pi(s) = \int_{\mathcal{A}} \pi(a|s)(R(s,a) + \gamma \int_{\mathcal{S}} Pr(s'|s,a)V^\pi(s')ds'. \tag{2.24}$$

Similarly, we can define the *state-action value function*, which encapsulates the same meaning, but in this case we evaluate the long-term utility of taking an action a in a state s under a policy π:

$$\begin{aligned} Q^\pi(s) &= \mathop{\mathbb{E}}_{\pi}[G_t|S_t = s, A_t = a] \\ &= R(s,a) + \gamma \int_{\mathcal{S}} Pr(s'|s,a) \int_{\mathcal{A}} \pi(a'|s')Q^\pi(s',a')ds'da'. \end{aligned} \tag{2.25}$$

Interestingly, these measures can be adopted to provide a partial ordering over policies. In fact, a policy π is better than another policy π' if its expected return is greater or equal to that of π' for each state s, namely:

$$\pi \geq \pi' \iff V^\pi(s) \geq V^{\pi'}(s) \quad \forall s \in \mathcal{S}.$$

Crucially, for any MDP there always exists a policy which is better or equal to all the other policies. This policy is called the *optimal policy* and we will indicate it as π^*. Note that there can be more than one optimal policy, but they all share the same *optimal state value function*:

$$\begin{aligned} V^*(s) &= max_\pi V^\pi(s) \quad \forall s \in \mathcal{S} \\ &= max_a\{R(s,a) + \gamma \int_{\mathcal{S}} Pr(s'|s,a)V^*(s')ds'\}, \end{aligned} \tag{2.26}$$

and the same *optimal state-action value function*:

$$\begin{aligned} Q^*(s,a) &= max_\pi Q^\pi(s,a) \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \\ &= max_a\{R(s,a) + \gamma \int_{\mathcal{S}} Pr(s'|s,a)max_{a'}Q^*(s',a')ds'da'\}. \end{aligned} \tag{2.27}$$

Indeed, once we know $Q^*(s,a)$, we can easily define the optimal policy as the one that acts greedily with respect to it:

$$\pi^*(a|s) = \arg\max_a Q^*(s,a), \quad \forall s \in \mathcal{S}. \tag{2.28}$$

An optimal value function thus specifies the best possible performance in an MDP. In fact, an MDP is said to be "solved" when we know the optimal value functions.

2.2.5 *The RL Problem*

At this point, we can easily define the main objective when we try to solve a RL problem. Independently from the way in which we measure the return and from the kind of policy we are using, the goal in RL is to find a policy that maximizes the expected return. We can thus measure the performance of an agent as:

$$\mathcal{J}^\pi = \mathop{\mathbb{E}}_{\tau\sim\pi}[G(\tau)] = \int_{\mathcal{T}} Pr(\tau|\pi)G(\tau)d\tau. \tag{2.29}$$

The optimal policy is then expressed as:

$$\pi^* = \arg\max_\pi \mathcal{J}^\pi. \tag{2.30}$$

Note that this optimality criterion is weaker than the one in (2.28), because in this case the policy does not need to be optimal in all the states, but only in the ones that are visited with a non-zero probability. Moreover, it is worth to know that there is always a deterministic optimal policy for any MDP [60].

2.2.6 *Methods to solve MDPs*

We said that solving an MDP means to find the optimal state-action value function, or the optimal policy alike. There are plenty of ways to address this problem, thus it is worth to provide a taxonomy that takes into consideration all the different settings we can encounter. The first distinction is the following:

- *Model-based* learning: the agent tries to understand the environment in which it is immersed to create a model of it. In other words, it tries to build an approximation of the transition model $\mathcal{P}$ and of the reward function $\mathcal{R}$. The main advantage of having a model consists in the possibility to perform planning, looking in advance at what can happen and acting accordingly. This results also in a higher sample efficiency. However, if this is relatively easy to do when the ground-truth model is given, leading to remarkable results such as with AlphaZero [69], in the remaining cases, which are the majority, the agent has to build an approximation of it from pure experience, which sets several challenges. Dynamic programming can be used to obtain an optimal policy assuming to have a perfect model of the environment. Nonetheless, it is rarely used in practice, due to its assumption and to its poor computational efficiency;

- *Model-free* learning: the agent learns without any knowledge about the model.

A second distinction can be done by considering the following two paradigms:

- *Value-based* learning: the focus is on the value functions, which once estimated are used to perform the actions. One of the most successful method is *Q-learning* [83], which, as the name suggests, learns an approximation of the optimal state-action value function in (2.25), and then selects the action that is optimal according to the approximation. DQN [50] is a practical implementation of Q-learning, where value functions are represented by neural networks;

- *Policy-based* learning: the focus is on the policy itself. We consider a parameterized policy $\pi_{\boldsymbol{\theta}}$ and we directly update its parameters. Note that in this case we are directly optimizing the thing we want to obtain, as opposed to a value-based method. Nevertheless, even if a value function is in this case not required to select an action, it can be used to learn the policy parameters, in which case we talk of an *actor-critic* method. Policy-based methods have several advantages over the value-based ones, but one of the most straightforward is that a policy may be a simpler function to approximate than an action-value. Another significant benefit of using policy parameterization is the possibility to inject a prior knowledge about the form of the policy that we would like to obtain. Moreover, policy-based methods are more suitable for scaling RL in high-dimensional continuous action spaces, as well as for episodic tasks, i.e., for tasks that can be broken down into a sequence of separate episodes. Finally, they allow to work with stochastic policies in a more natural way. One example of successful application of this approach is PPO [66].

To conclude, another distinction is between:

- *On-policy* learning: the so-called *behavioral policy*, i.e., the one used by the agent to interact with the environment, is also the one that we aim to optimize. Policy-based methods typically follow this approach, updating the policy with the data collected through the latest version of the same policy;

- *Off-policy* learning: the agent acts according to a behavioral policy, but the policy that is actually optimized is different and it is usually called *target policy*. The Q-learning algorithm follows this approach.

2.2.7 *Policy Gradient Methods*

Policy gradient methods, introduced by [84], obviously belongs to the policy-based class, according to one of the distinctions made in the previous section. They indeed operate directly in the parameter space of parameterized policies. In particular, the name comes from the fact that they learn the parameters of the policy by means of *gradient ascent*, i.e., by updating the parameters towards the maximum growth direction of the performance measure (2.29). The search of the optimal policy is performed into the so-called *policy space*, which in this case is restricted to all the stochastic, differentiable and parameterized policies and can be formally defined as:

$$\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^{n_{\boldsymbol{\theta}}}\}, \tag{2.31}$$

where $n_{\boldsymbol{\theta}}$ is the number of parameters. Hence, given the parameters of the current policy $\boldsymbol{\theta}_t$ (e.g., the weights of a neural network) and indicating as $\mathcal{J}^{\pi_{\boldsymbol{\theta}}}$ the performance measure with respect to the policy parameters, and with $\nabla_{\boldsymbol{\theta}}\mathcal{J}^{\pi_{\boldsymbol{\theta}}}$ its respective gradient, we can write the updating procedure as:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha\nabla_{\boldsymbol{\theta}}\mathcal{J}^{\pi_{\boldsymbol{\theta}}}, \tag{2.32}$$

where α value is the learning rate.

2.2.8 *Derivation of the Policy Gradient*

In order to actually use the update in (2.32), we need an expression of the policy gradient that we can numerically compute. Starting from the performance measure definition in (2.29), we can derive:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{J}^{\pi_{\boldsymbol{\theta}}} &= \nabla_{\boldsymbol{\theta}} \int_{\mathcal{T}} Pr(\tau|\pi_{\boldsymbol{\theta}})G(\tau)d\tau \\
&= \int_{\mathcal{T}} \nabla_{\boldsymbol{\theta}} Pr(\tau|\pi_{\boldsymbol{\theta}})G(\tau)d\tau \\
&= \int_{\mathcal{T}} Pr(\tau|\pi_{\boldsymbol{\theta}})\nabla_{\boldsymbol{\theta}} \log Pr(\tau|\pi_{\boldsymbol{\theta}})G(\tau)d\tau \\
&= \mathop{\mathbb{E}}_{\tau\sim Pr(\cdot|\pi_{\boldsymbol{\theta}})}\Big[G(\tau)\nabla_{\boldsymbol{\theta}} \log Pr(\tau|\pi_{\boldsymbol{\theta}})\Big],
\end{aligned} \tag{2.33}$$

where in the third step we used the log-derivative trick $\nabla_{\boldsymbol{\theta}} p(x;\boldsymbol{\theta}) = p(x;\boldsymbol{\theta}) \log p(x;\boldsymbol{\theta})$. Once we have this formulation, we can see that we reduced the gradient to an expectation, which we can estimate with a sample average. Let us recall that the gradient of the log-probability of a trajectory τ is:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} Pr(\tau|\pi_{\boldsymbol{\theta}}) &= \cancel{\nabla_{\boldsymbol{\theta}} \log d_0(s_0)} + \sum_{t=0}^{T-1} \cancel{(\nabla_{\boldsymbol{\theta}} \log P(s_{t+1}|s_t, a_t)} \\ &\quad + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)) \\ &= \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t), \end{aligned} \tag{2.34}$$

and that $G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$. Significantly, Equation (2.34) states that $\nabla_{\boldsymbol{\theta}} Pr(\tau|\pi_{\boldsymbol{\theta}})$ does not depend on the transition model. Finally, we can write the resulting expression:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}^{\pi_{\boldsymbol{\theta}}} = \mathop{\mathbb{E}}_{\tau \sim P(\tau|\pi_{\boldsymbol{\theta}})} \Big[\sum_{t=0}^{T-1} \gamma^t r_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \Big]. \tag{2.35}$$

Note that in (2.34) we remove the terms that do not depend on the policy parameters $\boldsymbol{\theta}$. The resulting equation highlights a crucial result for policy gradients. This result holds for stochastic policies only and that is why they are so important in policy gradients methods.

To conclude, the estimator in (2.35) is only one among the possible policy gradient estimators [57]. Some other estimators apply some tweaks to the $\gamma^t r_t$ term, in order to deal with the inherently present noise in Monte-Carlo estimates. For instance, the REINFORCE algorithm [84] uses exactly what we delineated in (2.35), but with an additional baseline b, which we can subtract from the $\gamma^t r_t$ term to reduce the variance. Below we report the general procedure of a policy gradient algorithm.

Algorithm 1: Policy Gradient Algorithm

Input: initial policy $\pi_{\boldsymbol{\theta}_0}$, learning rate α
for $h = 0, 1, \ldots,$ until convergence **do**
  Explore: collect sample trajectories using $\pi_{\boldsymbol{\theta}_h}$
  Evaluate: compute (estimate) the policy the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{J}^{\pi_{\boldsymbol{\theta}_h}}$
  Update: update the policy parameters $\boldsymbol{\theta}_{h+1} \leftarrow \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}^{\pi_{\boldsymbol{\theta}}}$
end for
Output: optimal policy $\pi_{\boldsymbol{\theta}_h}$

2.3 THE PARETO FRONTIER: A MULTI-OBJECTIVE PERSPECTIVE

Many real-world problems are characterized by the presence of multiple objectives, sometimes conflicting with each other. In such cases, the standard RL framework [72] is not suitable, due to its single-objective focus. That is why throughout the years a new branch has

began to develop, named Multi-Objective Reinforcement Learning (MORL). Essentially, MORL considers Multi-Objective MDPs (MOMDPs) to solve sequential decision-making problems with multiple, possibly conflicting, objectives. Formally, a MOMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \boldsymbol{\mathcal{R}}, \boldsymbol{\gamma}, d_0 \rangle$, where $\boldsymbol{\mathcal{R}} = [\mathcal{R}_1, \dots, \mathcal{R}_q]^T$ and $\boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_q]^T$ are q-dimensional column vectors of reward functions and discount factors. Accordingly, a policy π is associated to q expected returns $\boldsymbol{\mathcal{J}}^\pi = [\mathcal{J}_1^\pi, \dots, \mathcal{J}_q^\pi]$.

In Section 2.2.4, we gave the definition of optimal policy and we said it is not necessarily unique. If there are N optimal policies, they all share the same optimal state value function. In a MOMDP this does not hold anymore, as a single policy cannot maximize simultaneously all the objectives, due to their conflicting nature. Here comes to play the concept of *Pareto frontier*. In order to understand it better, let's first introduce the notion of *Pareto dominance*. Formally, a policy π *strongly dominates* a policy π' if it is superior on all objectives:

$$\pi \succ \pi' \iff \forall i \in \{1, \dots, q\}, \mathcal{J}_i^\pi > \mathcal{J}_i^{\pi'}. \tag{2.36}$$

Instead, a policy π *weakly dominates* a policy π' if it is not worse on all the objectives:

$$\pi \succeq \pi' \iff \forall i \in \{1, \dots, q\}, \mathcal{J}_i^\pi \geq \mathcal{J}_i^{\pi'} \wedge \exists i \in \{1, \dots, q\}, \mathcal{J}_i^\pi = \mathcal{J}_i^{\pi'}. \tag{2.37}$$

Finally, we define a *Pareto-optimal* policy as:

$$\pi^* = \pi \quad s.t. \quad \nexists \pi', \pi' \succeq \pi. \tag{2.38}$$

At this point we can introduce the concept of Pareto frontier. Since in general there exist multiple *(locally) Pareto-optimal* policies, where with "locally" we mean that the dominance is restricted to a neighborhood of π, the objective we would like to achieve consists in finding the set of all these Pareto-optimal policies. The Pareto frontier is nothing but the set obtained by mapping the Pareto-optimal policies π^* with their corresponding $\mathcal{J}^{\pi^*}$. In [54], Parisi et al. propose a policy-based approach to learn a continuous approximation of the Pareto frontier in MOMDPs. They consider a function ϕ_ρ that defines a manifold in the policy parameter space, from which another mapping is done by means of the expected return $\mathcal{J}(\boldsymbol{\theta})$, obtaining the corresponding image in the objective space. Given this setting, they try to find the parameterization ρ that minimizes the distance from the real Pareto frontier using a gradient-ascent procedure.

3

STATE OF THE ART

In this chapter, we provide an overview of the latest methods and approaches in the literature that are relevant to the problems we will encounter in the next sections. Since this thesis is at the intersection of three different areas, the overview will briefly cover all of them. In particular, in Section 3.1, we discuss the recent developments of the exploration problem in reinforcement learning, with a focus on intrinsic motivation and task-agnostic exploration. Then, in Section 3.2, we introduce those methods whose aim is to build robust policies, where the meaning of robust can be generally understood as the capability to deal with situations more complex than the average. Finally, in Section 3.3, we introduce the paradigm of meta-learning (or learning to learn) and the methods that use it to learn policies that are able to quickly learn new tasks or to adapt to new environments.

3.1 EXPLORATION IN REINFORCEMENT LEARNING: FROM SPARSE-REWARD TO REWARD-FREE

In the previous sections, we just assumed the presence of some kind of reward signal, and given this assumption we formulated several expressions. This is indeed how RL was originally thought and many astonishing results have been obtained in the recent years [50, 69]. However, a reward function is not always granted but somehow a luxury. In fact, even if the intuition behind its functioning is unequivocal, its design is not as straightforward [32]. Typically, the way in which we avoid this obstacle is to adopt a very simple formulation, such as giving a positive reward (e.g., +1) each time the agent solves a task. The drawback of this solution is that the reward signal becomes sparse, meaning that the agent has to perform a certain number of actions without receiving any feedback about its behavior. This approach is feasible if the state space is not too large, otherwise the agent will struggle to reach the goal even once. To further complicate the matter, in some cases the reward might not be present at all while training [40]. In both the sparse-reward or the reward-free scenarios, exploration becomes the main challenge for the agent.

To address this exploration problem, several works have been proposed. We provide a brief but exhaustive overview of the main methods used to tackle the exploration problem in sparse-reward and reward-free RL.

3.1.1 *Prediction-Error Methods*

Prediction-error methods use the error coming from the prediction of the next state as an intrinsic bonus, following the intuition that the agent will prefer to go in states that it does not know well, hence where the error is higher. Generally, the error might be defined as:

$$R_{intrinsic}(s_t, s_{t+1}) = ||f(s_{t+1}) - M(f(s_t), a_t)||_2,$$

where $f(\cdot)$ is the feature space and M our model of the environment, typically a neural network. The main challenge in this setting consists of learning an appropriate function f. In [56], the idea is to learn it end-to-end by means of an inverse dynamics model. Essentially, the predictions are not made in the raw observation space (e.g., pixels), but instead only the information that are directly related to the agent's actions are represented in the feature space. This solves the so-called *white-noise* problem [64], which occurs due to local stochasticity and that pushes the agent to stall its exploration.

3.1.2 *Count-Based Methods*

Count-based methods add an intrinsic bonus when the agent encounters a new or less visited state. The simplest way in which such idea can be put in practice is by counting the number of times a state s has been visited. At that point, the intrinsic bonus can be computed as:

$$R_{intrinsic}(s_t) = \frac{1}{\sqrt{N(s_t)}},$$

where $N(s_t)$ is the number of times the state s_t has been visited. The obvious limitation of this solution is that it works well only for discrete state spaces with small dimensions. Several approaches have been proposed in the literature to deal with this shortcoming, by hashing the state space [75] or by means of density models [5, 53]. One drawback of density models is their computational complexity. Burda et al. [13] managed to reduce this complexity and to obtain better results overall. They use a fixed random neural network to produce random continuous features, and a second network to predict the output of the random one. The intrinsic bonus is given by the prediction error. In this sense, it can be considered a prediction-error method as well. The count-based aspect is due to the fact that the higher the error, the lower the number of visits of the second network to the considered state.

3.1.3 *Information Gain Methods*

Information gain methods add an intrinsic bonus that is proportional to the amount of uncertainty about the environment that has been

reduced in the last step. The uncertainty is typically measured by means of the *mutual information*, which can be computed as:

$$I(X;Y) = H(X) - H(X|Y),$$

where X and Y are two random variables and H is the entropy. Note that we can interpret the mutual information between X and Y as the entropy reduction on X given Y. In [29], they use the conditional mutual information $I(A;G|S)$ to quantify the dependence of the policy on the goal G and they minimize it in order to obtain a policy that is able to exploit a multi-goals structure to allow for an efficient exploration of new environments.

Another relevant work is the one in [25]. They propose an algorithm called Diversity Is All You Need (DIAYN), aimed at learning new skills without a reward function. Especially, as its name suggests, it seeks for a set of skills which is as diverse as possible, thus maximizing the exploration of possible behaviors. The way in which they achieve such a diversity is by optimizing a mutual information objective of the form:

$$\begin{aligned} \textit{maximize } \mathcal{F}(\boldsymbol{\theta}) &= I(S;Z) + H(A|S) - I(A;Z|S) \\ &= H(Z) - H(Z|S) + H(A|S,Z), \end{aligned}$$

where S and A represent states and actions respectively, $Z \sim p(z)$ is a latent variable representing the skill (where they fix $p(z)$ to be uniform), $I(\cdot)$ is the mutual information, and $H(\cdot)$ is the entropy. Intuitively, the above objective pushes towards distinguishable skills, i.e., which visit different states, by maximizing the mutual information between S and Z. Concurrently, the mutual information between skills and actions given the state is minimized to guarantee that skills are distinguishable with respect to states and not actions. Finally, the entropy of the actions conditioned to the states is maximized to obtain skills that are different from one another.

Actually, as they cannot integrate over all states and skills, they approximate $p(z|s)$ with a learned discriminator $q_\phi(z|s)$. Then, they exploit the Jensen's inequality [3] to write:

$$\begin{aligned} \mathcal{F}(\boldsymbol{\theta}) &= H(A|S,Z) - H(Z|S) + H(Z) \\ &= H(A|Z,S) + \mathop{\mathbb{E}}_{\substack{z\sim p(z) \\ s\sim\pi(z)}} \big[\log p(z|s)\big] - \mathop{\mathbb{E}}_{z\sim p(z)} \big[\log p(z)\big] \\ &\geq H(A|Z,S) + \mathop{\mathbb{E}}_{\substack{z\sim p(z) \\ s\sim\pi(z)}} \big[\log q_\phi(z|s) - \log p(z)\big] \end{aligned} \tag{3.1}$$

3.1.4 *A Limitation Of Intrinsic Bonuses*

As outlined in [24], both count-based methods and prediction-error methods suffer the problem of *detachment*, possibly making them

unable to methodically explore the environment. In fact, the inherently consumable nature of intrinsic bonuses might cause the agent to become detached from the frontiers with high rewards, meaning that the agent will hardly rediscover a frontier after having exhausted the intrinsic bonuses. This is particularly limiting when the reward is not only sparse, but even not present at all. Different methodologies have to be developed for this case, which we outline in next section.

3.1.5 *Task-Agnostic Exploration Methods*

Task-agnostic exploration methods address the problem of exploration by seeking for an intrinsic objective that, when optimized, can lead to a policy which allows the agent to solve many subsequent tasks. The typical objective is strive for a policy that induces a state distribution which is as uniform as possible. Usually, this objective is measured in terms of the entropy function.

The first work to propose such an approach is [36]. Their objective is given by:

$$\pi^* \in \arg\max_{\pi \in \Pi} H(d_\pi), \tag{3.2}$$

where Π is the policy space and d_π is the state distribution induced by π. As one might notice, optimizing the objective in (3.2) is not straightforward. Ideally, we could use (2.17) and then compute the entropy directly, but in practice it is not feasible except for simple discrete domains, since we would need to estimate the transition model. In [36], Hazan et al. resort to compute an estimate of the state distribution with a density model. Especially, they provide results for both a count-based density model and a kernel-based density model [55]. Their algorithm learns an optimal mixture of policies by means of the conditional gradient method [27].

Several works have then followed this line of research. For instance, in the discrete setting we can find [76], which introduce the active exploration problem in MDPs, providing an algorithm that seeks for a policy maximizing the accuracy of the mean predictions given an unknown environment, and whose states are characterized by random variables that need to be estimated. Another relevant work that operates in the discrete case is the one of [52]. They adopt the objective in (3.2), but instead of using the conditional gradient method, they maximize a lower bound to the policy entropy, and also propose a constraint formulation to lower the time with which the policy reaches its full exploration efficiency.

As regards the continuous case, in [46] the objective is reformulated as the problem of matching a given target distribution $p^*(s)$, which encodes our belief about the future tasks. If the target distribution is the uniform one, the objective is equivalent to the entropy maximization.

The disparity between the state marginal distribution $\rho(s)$ (defined as the probability that the policy visits the state s) and the target distribution is measured with the KL divergence. Thus, the objective becomes:

$$\begin{aligned} &\min_{\pi \in \Pi} D_{KL}(\rho_\pi(s) \,||\, p^*(s)) \\ &= \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{\rho_\pi(s)} \log p^*(s) + H_\pi(s). \end{aligned} \tag{3.3}$$

Their approach also relies on a density model to estimate the entropy and the output is still a mixture of policies (at least for complex target distributions). Other more recent works are [86], whose algorithm outputs a single exploratory policy by maximizing the *Rényi entropy* over the state-action space, and [30], where Guo et al. introduce a version of the Shannon entropy, named *geometry-aware* Shannon entropy, that relies on a symmetric similarity function $k : \mathcal{S} \times \mathcal{S} \to [0, 1]$ to capture the underlying geometry of the state space $\mathcal{S}$.

Finally, [51] propose an algorithm, called Maximum Entropy POLicy optimization (MEPOL), to learn a single exploration policy rather than a mixture of policies. Especially, it gets rid of any explicit model to estimate the entropy, a procedure which is known to have some flaws [4], which make the solutions that use it hardly scalable to complex domains. Their objective is defined as:

$$max_{\pi \in \Pi} H(\bar{d}_T), \qquad \bar{d}_T = \frac{1}{T} \sum_{t=0}^{T-1} d_t^\pi, \tag{3.4}$$

where $\bar{d}_T = \frac{1}{T} \sum_{t=1}^{T} d_t^\pi$ is the finite-horizon marginal distribution across all steps. As said for (3.2), optimizing (3.4) is not trivial, and if done directly it would preclude high-dimensional domains. That is why in [51], Mutti et al. opt for a policy-search model-free algorithm that uses a non-parametric differential entropy estimator of the form seen in (2.5) and that is here reported for convenience, after making some changes to the notation:

$$\widehat{H}^{\text{IW}}_{\tau_i} = -\sum_{t=0}^{T-1} \frac{\sum_{j \in \mathcal{N}_t^k} w_j}{k} \ln \frac{\sum_{j \in \mathcal{N}_t^k} w_j}{V_t^k} + \ln k - \Psi(k), \tag{3.5}$$

where

$$V_t^k = \frac{\left\| s_{t,\tau_i} - s_{t,\tau_i}^{k\text{-NN}} \right\|^p \cdot \pi^{\frac{p}{2}}}{\Gamma(\frac{p}{2} + 1)} \tag{3.6}$$

is the volume of the hyper-sphere of radius $R_t = \left\| s_{t,\tau_i} - s_{t,\tau_i}^{k\text{-NN}} \right\|$, which is the Euclidean distance between s_{t,τ_i} and its k-nearest neighbor $s_{t,\tau_i}^{k\text{-NN}}$. In fact, they show how a Euclidean metric suffices to get reliable entropy estimates in continuous control domains. MEPOL relies on the above estimator to assess the quality of a policy given a batch of interactions, and tries to maximize it within a parametric policy space. The intuition behind the use of a k-NN estimator to provide an estimate

of the entropy is that the entropy of a continuous distribution can be somehow inferred by looking at how random samples drawn from it end up lying over the support surface [4]. Maximizing the entropy via a k-NN estimator thus means to seek for a uniform coverage of the support. The way in which MEPOL does this consists in combining the idea of two successful policy-search methods: POIS [49], to perform the optimization offline via importance sampling, allowing for an efficient exploitation of the samples collected with previous policies, and TRPO [65], as the IW estimator is optimized via gradient ascent by performing subsequent optimizations withing a trust-region around the current policy. The trust-region constraint is obtained by imposing:

$$\widehat{D}_{KL}(\pi_{\boldsymbol{\theta}'}||\pi_{\boldsymbol{\theta}}) = \frac{1}{T}\sum_{t=0}^{T-1} \ln \frac{k/T}{\sum_{j\in\mathcal{N}_t^k} w_j} \leq \delta, \tag{3.7}$$

where $\widehat{D}_{KL}(\pi_{\boldsymbol{\theta}'}||\pi_{\boldsymbol{\theta}})$ is a non-parametric IW k-NN estimate of the KL divergence [1].

Interestingly, [34] adopt the same entropy estimator, but the k-NN computation is performed on the latent representation space obtained by means of contrastive learning [33] (and more recently [15]). This allows them to test their algorithm in a visual-based setting. A similar approach is followed in [67], where the k-NN computation is performed in the representation space of a randomly initialized encoder. More precisely, they define the intrinsic reward as:

$$r^i(s_i) = \log(\left\|y_i - y_i^{k-NN}\right\|_2 + 1), \tag{3.8}$$

where $y_i = f_{\boldsymbol{\theta}}(s_i)$ is a fixed representation from a random encoder, and y_i^{k-NN} its k-nearest neighbor within a set of N representations. Interestingly, the parameters $\boldsymbol{\theta}$ of the encoder are randomly initialized and fixed during the training. As regards the choice of a random initialization, they refer to several works in the literature that have shown its effectiveness in different areas, RL included [28, 45]. They opt for a fixed representation with the aim of having a more stable intrinsic reward, as the distance between two states remains unchanged during the training. The choice of an encoder architecture must be sought in the intent of extracting features that are relevant to discriminate between states. The intuition comes from the strong inductive bias of deep convolutional networks [80, 14].

The works of [51] and [67] will be resumed in Chapter 4, since their contribution is significant in this thesis. For the sake of completeness, we provide in Algorithm 2 the pseudo-code of MEPOL.

3.2 RISK-SENSITIVE ALGORITHMS IN REINFORCEMENT LEARNING

The idea of risk-sensitive RL, also sometimes called *Risk-Averse* or *Risk-Safe* RL, is a natural development of the concepts that we introduced in

Algorithm 2: MEPOL

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

IS-Optimizer

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

the previous sections. In Section 2.2.5, we said that, generally speaking, the goal in RL is to find a policy that maximizes the *expected* return. The term "expected" is highlighted because in many real-world problems it represents the weakness of what we introduced as the standard RL approach. There are several applications in which the environments can contain many uncertainties, due to their stochastic nature, and if the optimization is based on the *average* case, they can go unnoticed. In other words, particularly unlucky situations may occur and if our agent is not robust enough to deal with those situations, bad things can happen. One straightforward example can be found by considering the world of finance: when we decide to make an investment, we would like to maximize the expected return, but also to avoid to lose everything due to an unexpected event. Hence, the trading-bot should be robust enough to perform reasonably well even in the so-called *worst-case* scenario.

Risk-sensitive RL tries to address these limitations. In order to do that, it makes use of some mathematical tools, which provide a measure of the risk. The basic insight is to shift the focus of the optimization process to the worst-case, or at least to take it into consideration, e.g., by introducing one or more constraints. Two risk measures which have been increasingly adopted in the last years are the VaR and CVaR. Due to the superior properties of CVaR over VaR, which we highlighted in Section 2.1.6, CVaR has increasingly taken the place of VaR in risk-averse RL. The first work proposing such change of direction is [62], which shows the effectiveness of a CVaR minimization in a portfolio optimization setting. It addresses the problem by means of linear

programming. Other works have then built on this idea [38, 39], improving the computational efficiency limits imposed by solving a linear programming problem. However, they still make use of a stochastic program formulation. Hence, in their respective field of application, the parameters do not affect the distribution of the outcomes.

The first work to propose a policy-gradient algorithm for CVaR optimization is [74]. They derive a novel formula for the CVaR gradient, along with a Monte-Carlo based estimator and a stochastic gradient descent procedure for the optimization, which is guaranteed to converge to a local optimum. Then, they empirically test their solution in a RL domain, learning a risk-sensitive policy for Tetris. We report below the general formulation of the CVaR gradient:

$$\nabla_{\boldsymbol{\theta}} CVaR_\alpha(X) = \mathop{\mathbb{E}}_{\boldsymbol{\theta}}\Big[\nabla_{\boldsymbol{\theta}} f_X(X;\boldsymbol{\theta})(X - VaR_\alpha(X)\Big| X \leq VaR_\alpha(X)\Big], \quad (3.9)$$

where X is a random variable that would typically assume the form of some kind of performance measure, $f_X(X;\boldsymbol{\theta})$ its probability density function, and b the baseline. The Monte-Carlo estimate of the gradient is given by:

$$\widehat{\nabla}_{\boldsymbol{\theta}} CVaR_\alpha(X) = \frac{1}{\alpha N}\sum_{i=1}^{N} f_{\tau_j}\big(X_i - \widehat{VaR_\alpha}\big)\mathbb{1}\big(X_i \leq \widehat{VaR_\alpha}\big), \quad (3.10)$$

where $f_{\tau_j} = \sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau_i}, s_{t,\tau_i})$. Remarkably, the estimator in (3.10) is asymptotically unbiased and consistent [74], but it is hampered by the estimation error of the VaR term to be subtracted to each X_i in finite sample regimes [42]. In Algorithm 3 we detail the full procedure.

Algorithm 3: CVaRSGD

Input: initial parameters $\boldsymbol{\theta}^0 \in \mathbb{R}^k$, step size ϵ, percentile α, reward function $r(\boldsymbol{x}, y) : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}$, projection operator $\Gamma : \mathbb{R}^k \to \mathbb{R}^k$
for $i = 0, 1, \ldots,$ until convergence **do**
  sample n_i i.i.d. realizations $x_{n_i}, y_{n_i} \sim f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{\theta}^i)$
  compute $r_1, \ldots, r_N = Sort(r(\boldsymbol{x_1}, y_1), \ldots, r(\boldsymbol{x_N}, y_N))$
  estimate $VaR_\alpha \sim r_{\lceil \alpha N \rceil}$
  for $j = 0, 1, \ldots, k$ **do**
    compute a Monte-Carlo estimate of the CVaR gradient $\Delta_{j;n_i}$
    update the parameters $\boldsymbol{\theta}_j^{i+1} \leftarrow \Gamma(\boldsymbol{\theta}_j^i + \epsilon_i \Delta_{j;n_i})$
  end for
end for
Output: risk-sensitive policy $\pi_{\boldsymbol{\theta}}$

Concurrently with [74], a similar solution has been endorsed in [17]. They propose both a policy-gradient algorithm and an actor-critic algorithm having as objective the mean-CVaR optimization, in which the CVaR becomes a constraint rather than the objective function:

$$\underset{\pi_{\boldsymbol{\theta}} \in \Pi_\Theta}{\text{maximize}} \;\; \mathcal{F}_{\text{mean-cvar}}(\pi_{\boldsymbol{\theta}}) = \Big\{ \mathbb{E}[X] \quad \text{s.t. } CVaR_\alpha[X] \geq \xi \Big\}, \quad (3.11)$$

where ξ is the parameter to control risk aversion. The nature of this parameter changes according to the context in which it is used. For instance, in the financial framework, it typically represents the *loss tolerance*.

Knowing from [62] that CVaR can be equivalently written as:

$$CVaR_\alpha(X) = \max_{\nu \in \mathbb{R}} \left\{ \nu + \frac{1}{\epsilon} \mathop{\mathbb{E}}_{x \sim f_X} \left[(x - \nu)^- \right] \right\}, \tag{3.12}$$

where $(z)^- = min(0, z)$, and exploiting the Lagrangian relaxation method [8] to get an unconstrained form, they write the optimization problem as:

$$\min_{\lambda \geq 0} \max_{\boldsymbol{\theta} \in \Theta, \nu \in \mathbb{R}} \mathop{\mathbb{E}}_{x \sim f_{X;\boldsymbol{\theta}}} \left[x \right] + \lambda \left(\nu + \frac{1}{\epsilon} \mathop{\mathbb{E}}_{x \sim f_{X;\boldsymbol{\theta}}} \left[\left(x - \nu \right)^- \right] - \xi \right). \tag{3.13}$$

As opposed to [74], they thus adopt a fully principled, i.e., unbiased and consistent, policy update formulation. The optimization is performed via a three-timescale gradient ascent procedure for the policy gradient algorithm so that the VaR parameter ν update is on the fastest time-scale, the policy parameter $\boldsymbol{\theta}$ update on the intermediate time-scale, and the Lagrange multiplier λ update on the slowest time-scale. Instead, as regards the actor-critic algorithm, the hierarchy remains the same, but the fastest time-scale is taken by the critic update.

The same constrained formulation is presented in [37] to learn robust options, which are temporally extended sequence of actions [73]. Also [18], building on [20], propose a safe and active model-based learner, where the safe component is obtained by incorporating a CVaR constraint, which is again relaxed through the Lagrangian method.

Finally, it is worth to mention the work of [61], as it shares some of the intent that are central to this thesis (we will provide further explanations in Chapter 4). They take into consideration an inherent issue of model-based methods, i.e., the possibility to obtain control strategies that are not robust to model risks, which are caused by a discrepancy between the real-world target domain and the simulated source domain. To address this problem, they opt for an adversarial training on an ensemble of models. Formally, given the source domains (MDPs) and a distribution $p_{\mathcal{M}}$ over them, they wish to learn a policy $\pi^*_{\boldsymbol{\theta}}$ that performs well for all $\mathcal{M} \sim p_{\mathcal{M}}$. To accomplish this result, they iteratively sample a model from the source domain, they collect trajectories from it, and they add them to a dataset $\mathcal{D}$. Then, they perform the gradient update on a subset of $\mathcal{D}$, that specifically contains only the trajectories whose return is lower or equal to the α-percentile value of returns. In fact, as in [74], they optimize for the CVaR, maximizing the expected return for the worst α-percentile of $\boldsymbol{\mathcal{M}}$.

3.3 META REINFORCEMENT LEARNING

Meta reinforcement learning [82, 23] is a recent field of RL, directly aimed at making a further step towards general algorithms. The goal of traditional *meta-learning* is to learn a model that is able to quickly learn new skills or adapt to new environments. The idea comes from psychology, where this process is called *learning to learn* [35]. In their work, [11] highlight how one shortcoming of standard deep RL is the use of weak inductive biases, namely the set of assumptions used by the agent when it encounters an unseen situation to make a decision. Logically, the stronger the inductive bias the faster the learning. However, the catch is that this is true only if the the set of assumptions the agent use contains the correct assumption. The intuition of meta RL is to induce a stronger inductive bias by taking advantage of the past experience. This is something that naturally happens in our life. For instance, if we have to learn how to use a new object, we would probably succeed faster if we have some previous experience with similar objects or other related ones. We are thus guided by our bias, that allows to reduce the number of trials (variance), making the process of learning relatively faster.

In [82, 23], they use a model-based approach, in which they adopt a recurrent architecture, which is trained on a set of tasks sampled from a common distribution. Another common approach is the optimization-based one. The work of [26] belongs to this class of solutions. They propose an algorithm called Model-Agnostic Meta-Learning (MAML), where with model-agnostic they mean that it is compatible with any gradient-descent trainable model, and also transversal with respect to the area of application. In fact, they empirically show its efficacy in the classification, regression, and reinforcement learning settings. Formally, they consider a set of tasks $\mathcal{T}$, where each task $\mathcal{T}_i \in \mathcal{T}$ is an MDP with horizon T, drawn from a given distribution $p(\mathcal{T})$ and whose loss is defined as:

$$\mathcal{L}_{\mathcal{T}_i}(\pi_{\boldsymbol{\theta}}) = -\mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\theta}}}\Big[\sum_{t=0}^{T-1} R_i(\tau_t)\Big].$$

For each task $\mathcal{T}_i$, K trajectories are sampled with $\pi_{\boldsymbol{\theta}}$, the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(\pi_{\boldsymbol{\theta}})$ is evaluated and the adapted parameters are updated as:

$$\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(\pi_{\boldsymbol{\theta}}).$$

Then, new trajectories are sampled using $\pi_{\boldsymbol{\theta}_i'}$. Once the previous operations have been completed for all the tasks, the parameters $\boldsymbol{\theta}$ are finally updated using the new trajectories:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta\nabla_{\boldsymbol{\theta}}\sum_{\mathcal{T}_i\sim p(\mathcal{T})}\mathcal{L}_{\mathcal{T}_i}(\pi_{\boldsymbol{\theta}_i'}).$$

The rationale is to perform the meta-optimization over the model parameters $\boldsymbol{\theta}$, while the objective depends on the updated model

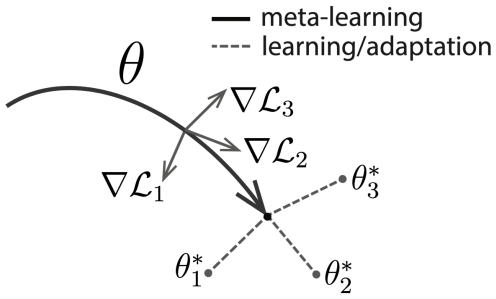


Figure 3.1: An intuitive illustration of the MAML optimization, from [26].

parameters θ'. This procedure results in a model whose parameters can be adapted to a new task in a small number of gradient steps.

One typical assumption made by meta RL algorithms is the possibility to sample from a pre-specified task distribution. Just like we said about the difficulty of designing a suitable reward function, we can claim the same about the design of a task distribution. To address this limitation, [31] propose an unsupervised meta-learning (UML) method to automatize the task design process. Formally, they consider the task distribution as a mapping from a latent variable $z \sim p(z)$, which represents a single task, to a corresponding reward function $r_z(s, a)$. Especially, they rely on DIAYN [25] (see Section 3.1.3) to train a discriminator $D(z|s)$ that predicts which z was used to generate a given trajectory τ. Once trained, they iteratively sample a task $z \sim p(z)$, extract the reward function $r_z(s|a) = \log(D(z|s))$ and they provide a task to a standard meta RL algorithm, MAML.

4

MULTI-ENVIRONMENT EXPLORATION WITHOUT REWARDS

In this chapter, we present a novel model-free policy-gradient algorithm to learn a single exploration strategy over a class of reward-free environments. In Section 4.1, we formally introduce the problem and we frame it into the current literature. In Section 4.2, we define the actual objective our algorithm seeks to optimize. Then, in Section 4.3, we theoretically characterize the problem, proposing a way to quantify the difficulty of exploration given a class of environments. Finally, in Section 4.4, we exhaustively describe how our algorithm works to optimize the previously introduced objective.

4.1 A MULTI-OBJECTIVE PROBLEM

In this section, we frame the problem we want to address. As we said in Section 3.1, the typical RL [72] setting involves a learning agent interacting with an environment in order to maximize a reward signal. In principle, the reward signal is a given and perfectly encodes the task. In practice, the reward is usually hand-crafted, and designing it to make the agent learn a desirable behavior is often a huge challenge. This poses a serious roadblock on the way of autonomous learning, as any task requires a costly and specific formulation, while the synergy between solving one RL problem and another is very limited. To address this crucial limitation, Jin at al. [40] have formulated the *reward-free* RL problem. In this setting, the agent is tasked with mastering an environment without rewards, so that the knowledge it acquires can be eventually deployed to solve any RL problem one could specify in this same environment. Mastering the environment has been formulated in (i) learning to model its transition dynamics [40, 77, 78, 87], or (ii) learning to explore it with a general, task-agnostic, strategy [36, 51]. Although they overcome the reliance on a reward function, previous solutions to reward-free RL are still severely environment-specific.

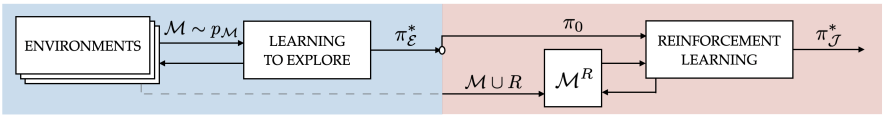


Figure 4.1: Illustration of the two-phase learning problem.

In this work, we aim to push the generality of reward-free learning even further by addressing the problem of *learning to explore multiple environments* with a single exploration strategy. In other words, we

consider a class of reward-free environments that belong to the same domain but differ in their transition dynamics; given that, the ultimate goal of the agent is to learn an exploration strategy that helps to solve any subsequent RL task specified over any environment in the class, reducing the burden on manual reward design while improving learning efficiency over RL from scratch. In Figure 4.1, we depict the learning problem, pointing out the two phases that constitute the overall process. More precisely, on the left, we highlight the process of learning to explore multiple environments. The obtained exploration policy $\pi^*_{\mathcal{E}}$ conveys a pre-trained initialization to the subsequent RL process (right), which outputs a reward maximizing policy $\pi^*_{\mathcal{J}}$ for an MDP $\mathcal{M}^R$ built upon any environment of the class.

In order to accomplish this goal, we extend a reward-free objective meant for environment-specific exploration, which is the *Maximum State Visitation Entropy* (MSVE) [36]. The underlying intuition is that a desirable property for a general exploration strategy is to visit with high probability any state where the agent might be rewarded in a subsequent RL task. When dealing with multiple environments, this becomes a *multi-objective* problem, as one could establish any combination of preferences over the environments in the class.

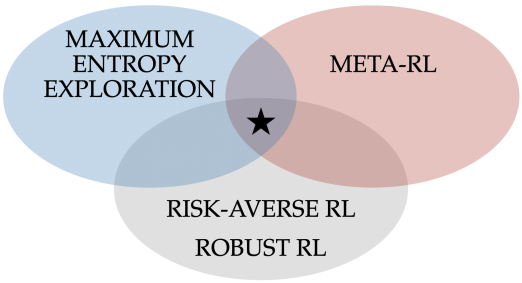


Figure 4.2: Where our work (star) stands in the literature.

We can depict the point in the literature in which our work lies as in Figure 4.2, i.e., coarsely at the intersection of reward-free exploration (see Section 3.1), robust and risk-averse RL (see Section 3.2), and meta RL (see Section 3.3).

4.2 LEARNING TO EXPLORE MULTIPLE ENVIRONMENTS

In this section, we present the objective for learning to explore multiple environments with a unique general strategy.

Let $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, \dots, \mathcal{M}_I\}$ be a class of unknown CMPs, in which every element $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, d_0)$ has a specific transition model $\mathcal{P}_i$, while $\mathcal{S}, \mathcal{A}, d_0$ are common across the class. More precisely, $\mathcal{S}$ and $\mathcal{A}$ are potentially infinite and homogeneous (hence with the same features) inside the class. At each turn, the agent is able to interact with a single environment $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. The selection of the environment to interact with is mediated by a distribution $p_{\boldsymbol{\mathcal{M}}}$ over $\boldsymbol{\mathcal{M}}$, outside the

control of the agent. The aim of the agent is to learn an exploration strategy that is general across all MDPs $\mathcal{M}^{\mathcal{R}}$ one can build upon $\boldsymbol{\mathcal{M}}$. In a single-environment setting, this problem has been assimilated to learning a policy that maximizes the entropy of the induced state visitation frequencies [36, 46, 51].

Recalling what we said in Section 2.3, we can contemplate a dual multi-objective scenario if we consider $\boldsymbol{\mathcal{M}}$ with the MORL paradigm. The duality is given by the multiplicity of the environments and the multiple dimensions that we would consider for each environment. We could try to learn an approximation of the Pareto frontier with the aim of obtaining a set of policies that allows for a comprehensive exploration of the environment. In fact, one considerable advantage of multiple-policy methods over single-policy methods is the ability to represent the Pareto-optimal manifold, allowing for a posteriori selection of the best policy given the task. Essentially, instead of considering all the features in a uniform way, we can optimize for a trade-off between them, as if we were assigning them some weights. Formally, we could write the objective as:

$$\mathcal{E}_{i,j}(\pi) = \mathop{\mathbb{E}}_{\tau \sim p_\pi, \mathcal{M}_i} [H_\tau(j)], \tag{4.1}$$

where i and j denote the environment and the feature respectively. In this thesis, we will narrow down the problem by exploring uniformly on j and adopting a risk-sensitive approach on i. However, a preliminary study on the multi-objective features exploration is provided in Appendix B.

We can straightforwardly extend the MSVE objective to multiple environments by considering the expectation over the class of CMPs:

$$\mathcal{E}_{\boldsymbol{\mathcal{M}}}(\pi) = \mathop{\mathbb{E}}_{\substack{\mathcal{M} \sim p_{\boldsymbol{\mathcal{M}}} \\ \tau \sim p_{\pi, \mathcal{M}}}} \big[H_\tau\big], \tag{4.2}$$

where the usual entropy objective over the single environment $\mathcal{M}_i$ can be easily recovered by setting $p_{\mathcal{M}_i} = 1$. However, the objective function (4.2) does not account for the tail behavior of H_τ, i.e., for the exploration performance in environments of $\boldsymbol{\mathcal{M}}$ that are rare or particularly unfavorable. This is decidedly undesirable as the agent may be tasked with an MDP built upon one of these adverse environments in the subsequent RL phase, where even an optimal strategy w.r.t. (4.2) may fail to provide sufficient exploration. To overcome this limitation, we look for a more nuanced exploration objective that balances the expected performance with the sensitivity to the tail behavior. By taking inspiration from the risk-averse optimization literature [62], we consider the CVaR of the state visitation entropy induced by π over $\boldsymbol{\mathcal{M}}$:

$$\begin{aligned} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi) &= CVaR_\alpha(H_\tau) \\ &= \mathop{\mathbb{E}}_{\substack{\mathcal{M} \sim p_{\boldsymbol{\mathcal{M}}} \\ \tau \sim p_{\pi, \mathcal{M}}}} \big[H_\tau \mid H_\tau \leq VaR_\alpha(H_\tau)\big], \end{aligned} \tag{4.3}$$

where α is the confidence level and $\mathcal{E}^1_{\mathcal{M}}(\pi) := \mathcal{E}_{\mathcal{M}}(\pi)$. The lower we set the value of α, the more we hedge against the possibility of a bad exploration outcome in some $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can thus define our objective as learning a policy

$$\pi^*_{\mathcal{E}} \in \arg\max \mathcal{E}^{\alpha}_{\mathcal{M}}(\pi) \tag{4.4}$$

through mere interactions with $\boldsymbol{\mathcal{M}}$.

4.3 THEORETICAL ANALYSIS OF THE PROBLEM

Before presenting a method to optimize the objective in (4.3), it is worth discussing the problem and, especially, what makes a class of CMPs hard to explore with a unique strategy. Intuitively, learning to explore a class $\boldsymbol{\mathcal{M}}$ with a policy π is challenging when the state distributions induced by π in different $\mathcal{M} \in \boldsymbol{\mathcal{M}}$ are diverse. The more diverse they are, the more their entropy can vary, and the harder is to get a π with a large entropy across the class. To measure this diversity, we are interested in the supremum over the distances between the state distributions $(d_\pi^{\mathcal{M}_1}, \dots, d_\pi^{\mathcal{M}_I})$ that a single policy $\pi \in \Pi$ realizes over the class $\boldsymbol{\mathcal{M}}$. We call this measure the *diameter* $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ of the class $\boldsymbol{\mathcal{M}}$. Since we have infinitely many policies in Π, computing $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ is particularly arduous. However, we are able to provide an upper bound to $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ defined through a Wasserstein metric (see Section 2.1.5), under common regularities assumptions for the class $\boldsymbol{\mathcal{M}}$.

Assumption 1. *The class $\boldsymbol{\mathcal{M}}$ is L_{P^π}-Lipschitz continuous,*

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

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

Theorem 4.3.1. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs satisfying Assumption 1. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by the policy π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ of the class as*

$$\begin{aligned}\mathcal{D}_{\boldsymbol{\mathcal{M}}} &:= \sup_{\pi \in \Pi,\, \mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}} d_{W_1}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) \\ &\leq \sup_{P', P \in \boldsymbol{\mathcal{M}}} \frac{1 - L^T_{P^\pi}}{1 - L_{P^\pi}} \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)).\end{aligned}$$

Theorem 4.3.1 provides a measure to quantify the hardness of the exploration problem in a specific class of CMPs, and to possibly compare one class with another. However, the value of $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ might result, due to the supremum over Π, from a policy that is far away from the policies we actually deploy while learning, say $(\pi_0, \dots, \pi^*_{\mathcal{E}})$. To get a finer assessment of the hardness of $\boldsymbol{\mathcal{M}}$ we face in practice, it is

worth considering a policy-specific measure to track during the optimization. We call this measure the *π-diameter* $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ of the class $\boldsymbol{\mathcal{M}}$. Theorem 4.3.2 provides an upper bound to $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ defined through a convenient TV metric.

Theorem 4.3.2. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy, and let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ of the class as*

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

We can build on Theorem 4.3.2 to relate the result to the performance measure of MEMENTO, i.e., the entropy, as follows:

Theorem 4.3.3. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy and $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ the corresponding π-diameter of $\boldsymbol{\mathcal{M}}$. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$, and let $\sigma_{\boldsymbol{\mathcal{M}}} \leq \sigma_{\mathcal{M}} := \inf_{s \in \mathcal{S}} d_\pi^{\mathcal{M}}(s), \forall \mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the entropy gap of the policy π within the model class $\boldsymbol{\mathcal{M}}$ as*

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

4.4 A POLICY GRADIENT APPROACH

In this section, we present an algorithm, called *Multiple Environments Maximum ENTropy Optimization* (MEMENTO), to optimize the exploration objective in (4.3) through mediated interactions with a class of continuous environments.

MEMENTO operates as a typical policy gradient approach [21]. It directly searches for an optimal policy by navigating a set of parametric differentiable policies $\Pi_\Theta := \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^n\}$. Given a probability distribution $p_{\boldsymbol{\mathcal{M}}}$, the algorithm operates by iteratively sampling an environment $\mathcal{M}_i \in \boldsymbol{\mathcal{M}}$ drawn according to $p_{\boldsymbol{\mathcal{M}}}$ and then sampling B trajectories of length T from it using $\pi_{\boldsymbol{\theta}}$, where B is the dimension of each mini-batch. The reason why we introduce an additional parameter B, instead of considering one trajectory at a time, is due to the fact that a significant amount of samples is needed to obtain a reliable estimate of the entropy, noting that the entropy estimator is only asymptotically unbiased. The estimate of the entropy of each mini-batch $\widehat{H}_{\tau_j}$ is then computed by means of the estimator in (4.7) and appended to the dataset $\mathcal{D}$. Once we obtain the final dataset $\mathcal{D}$, we can straightforwardly derive a risk-sensitive policy update by just

subsampling from it, so that to keep only the realizations below the α-percentile. This can be easily done by sorting $\mathcal{D}$ in ascending order and considering only the first αN mini-batches. Then, we can compute the gradient as follows:

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) = \frac{1}{\alpha N} \sum_{i=1}^{N} f_{\tau_i} \, \widehat{H}_{\tau_i} \, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{VaR}_{\alpha}(H_{\tau})). \tag{4.5}$$

As in [51], the operations carried out once all the trajectories have been sampled are executed in a fully off-policy manner, in which we repeat the same steps until either a trust-region boundary of the form

$$\widehat{D}_{KL}(\pi_{\boldsymbol{\theta}'} || \pi_{\boldsymbol{\theta}}) = \frac{1}{T} \sum_{t=0}^{T-1} \ln \frac{k/T}{\sum_{j \in \mathcal{N}_t^k} w_j} \leq \delta \tag{4.6}$$

is reached, or the off-policy iterations exceeds a specified limit. The value of (4.6) is computed as in [51], by considering the entire batch of trajectories collected to execute the off-policy optimization steps as a single trajectory.

The policy parameters $\boldsymbol{\theta}$ are iteratively updated in the gradient direction, until a stationary point is reached. This update has the form

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \beta \widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}),$$

where β is a learning rate, and $\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ is the gradient of (4.3) w.r.t. $\boldsymbol{\theta}$. The following proposition provides the formula of $\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$. The derivation follows closely the one in [74, Proposition 1], which we have adapted to our objective function of interest (4.3).

Proposition 4.4.1. *Let $\pi_{\boldsymbol{\theta}} \in \Pi_{\Theta}$. The policy gradient of the exploration objective $\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ w.r.t. $\boldsymbol{\theta}$ is given by*

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

However, in this work we do not assume full knowledge of the class of CMPs $\boldsymbol{\mathcal{M}}$, and the expected value in Proposition 4.4.1 cannot be computed without having access to $p_{\boldsymbol{\mathcal{M}}}$ and $p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}}$. Instead, MEMENTO computes the policy update via a Monte Carlo estimation of $\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ from the sampled interactions $\{(\mathcal{M}_i, \tau_i)\}_{i=1}^{N}$ with the class of environments $\boldsymbol{\mathcal{M}}$. The policy gradient estimate itself relies on a Monte Carlo estimate of each entropy value H_{τ_i} from τ_i, and a Monte Carlo estimate of $VaR_{\alpha}(H_{\tau})$ given the estimated $\{H_{\tau_i}\}_{i=1}^{N}$. The following paragraphs describe how these estimates are carried out, while Algorithm 4 provides the pseudo-code of MEMENTO.

Algorithm 4: MEMENTO

Input: initial policy $\pi_{\boldsymbol{\theta}_0}$, exploration horizon T, number of trajectories N, batch-size B, percentile α, learning rate β, trust-region threshold δ, sampling distribution $p_{\boldsymbol{\mathcal{M}}}$
for epoch $= 0, 1, \ldots,$ until convergence **do**
  for $i = 1, 2, \ldots, N$ **do**
    sample an environment $\mathcal{M}_i \sim p_{\boldsymbol{\mathcal{M}}}$
    for $j = 1, 2, \ldots, B$ **do**
      sample a trajectory $\tau_j \sim p_{\pi_{\boldsymbol{\theta}}, \mathcal{M}_i}$ of length T
    end for
  end for
  initialize dataset $\mathcal{D} = \emptyset$, off-policy step $h = 0$ and $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
  while $\widehat{D}_{KL}(\pi_{\boldsymbol{\theta}_0} || \pi_{\boldsymbol{\theta}_h}) \leq \delta$ **do**
    for $j = 1, 2, \ldots, B$ **do**
      estimate H_{τ_j} with (4.7)
      append $\widehat{H}_{\tau_j}$ to $\mathcal{D}$
    end for
    sort $\mathcal{D}$ and split it in $\mathcal{D}_\alpha$ and $\mathcal{D}_{1-\alpha}$
    compute a gradient step $\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \beta \widehat{\nabla}_{\boldsymbol{\theta}_h} \mathcal{E}^{\alpha}_{\mathcal{M}}(\pi_{\boldsymbol{\theta}_h})$
    $h \leftarrow h + 1$
  end while
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_h$
end for
Output: exploration policy $\pi_{\boldsymbol{\theta}_h}$

4.4.1 *Entropy Estimation*

We would like to compute the entropy H_{τ_i} of the state visitation frequencies d_{τ_i} from a single realization $\{s_{t,\tau_i}\}_{t=0}^{T-1} \subset \tau_i$. This estimation is notoriously challenging when the state space is continuous and high-dimensional $\mathcal{S} \subseteq \mathbb{R}^p$. Taking inspiration from recent works pursuing the MSVE objective [51, 34, 67], we employ a principled k-NN entropy estimator [70] of the form

$$\widehat{H}^{\text{IW}}_{\tau_i} = -\sum_{t=0}^{T-1} \frac{\sum_{j \in \mathcal{N}_t^k} w_j}{k} \ln \frac{\Gamma(\frac{p}{2}+1) \sum_{j \in \mathcal{N}_t^k} w_j}{\left\| s_{t,\tau_i} - s_{t,\tau_i}^{k\text{-NN}} \right\|^p \pi^{\frac{p}{2}}} + \ln k - \Psi(k) \quad (4.7)$$

where Γ is the Gamma function, $\| \cdot \|$ is the Euclidean distance, $s_{t,\tau_i}^{k\text{-NN}} \in \tau_i$ is the k-nearest neighbor of s_{t,τ_i}, $\ln k - \Psi(k)$ is a bias correction term in which Ψ is the Digamma function, $\mathcal{N}_i^k$ is the set of indices of the k-NN of s_{t,τ_i}, and w_j are the normalized importance weights of samples s_{j,τ_i}. To compute these importance weights, we consider a dataset $\mathcal{D} = \{s_{t,\tau_i}\}_{t=0}^{T-1}$ by looking each state encountered in a trajectory as an unweighted particle. Then, we expand it as $\mathcal{D}_{\tau_i} = \{(\tau_{i,t}, s_t)\}_{t=0}^{T-1}$, where $\tau_{i,t} = (s_{0,\tau_i}, \ldots, s_{t,\tau_i})$ is the portion of the trajectory that leads to state s_{t,τ_i}. This allows to associate each particle s_{t,τ_i} to its importance

weight $\widehat{w}_t$ and normalized importance weight w_t for any pair of target ($\pi_{\boldsymbol{\theta}'}$) and sampling ($\pi_{\boldsymbol{\theta}}$) policies:

$$\widehat{w}_t = \frac{p(\tau_{i,t}|\pi_{\boldsymbol{\theta}'})}{p(\tau_{i,t}|\pi_{\boldsymbol{\theta}})} = \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_{z,\tau_i}|s_{z,\tau_i})}{\pi_{\boldsymbol{\theta}}(a_{z,\tau_i}|s_{z,\tau_i})}, \qquad w_t = \frac{\widehat{w}_t}{\sum_{n=0}^{T-1} \widehat{w}_n}.$$

As we already discussed in Section 3.1.5, the intuition behind the estimator in (4.7) is straightforward: we can suppose the state visitation frequencies d_{τ_i} to have a high entropy as long as the average distance between any encountered state and its k-NN is large. Despite its simplicity, a Euclidean metric suffices to get reliable entropy estimates in continuous control domains [51]. More sophisticated metrics can be used to deal with rich-observations, e.g., in visual-based domains [34, 67].

4.4.2 *VaR Estimation and Baseline*

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

$$\widehat{VaR}_{\alpha}(H_\tau) = H_{[\lceil \alpha N \rceil]}. \tag{4.8}$$

Albeit asymptotically unbiased, the VaR estimator in (4.8) is known to suffer from a large variance in finite sample regimes [42], which is aggravated by the error in the upstream entropy estimates, which provide the order statistics. This variance is mostly harmless when we use the estimate to filter out entropy values beyond the α-percentile, i.e., the condition $H_\tau \leq VaR_{\alpha}(H_\tau)$ in Proposition 4.4.1. Instead, its impact is significant when we subtract it from the values within the α-percentile, i.e., the term $H_\tau - VaR_{\alpha}(H_\tau)$ in Proposition 4.4.1. To mitigate this issue, we consider a convenient baseline $b = -VaR_{\alpha}(H_\tau)$ to be subtracted from the latter, which gives the Monte Carlo policy gradient estimator

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\mathcal{M}}(\pi_{\boldsymbol{\theta}}) = \sum_{i=1}^{N} f_{\tau_i} \, \widehat{H}_{\tau_i} \, \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{VaR}_{\alpha}(H_\tau)), \tag{4.9}$$

where $f_{\tau_i} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau_i}|s_{t,\tau_i})$. Notably, the baseline b trades off a lower estimation error for a slight additional bias in the estimation (4.9). We found that this baseline leads to empirically good results and we can corroborate its use with some theoretical motivations. First, let us report for convenience the two alternatives policy gradient estimator, i.e, with and without baseline:

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

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

where $f_{\tau_i} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau_i}|s_{t,\tau_i})$. The former ($\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$) is known to be asymptotically unbiased [74], but it is hampered by the estimation error of the VaR term to be subtracted to each $\widehat{H}_{\tau_i}$ in finite sample regimes [42]. The latter ($\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$) introduces some bias in the estimate, but it crucially avoids the estimation error of the VaR term to be subtracted, as it cancels out with the baseline b. The following proposition, along with related lemmas, assesses the critical number of samples (n^*) for which an upper bound to the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ is lower to the estimation error of $\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$.

Lemma 4.4.2. *The expected bias of the policy gradient estimate $\widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ can be upper bounded as*

$$\mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[\text{bias}\right] = \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \left[\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})\right] \leq \mathcal{U}\alpha b,$$

where $\mathcal{U}$ is a constant such that $f_{\tau_i} \leq \mathcal{U}$ for all τ_i.

Proof. This Lemma can be easily derived by means of

$$\begin{aligned}
\mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \left[\text{bias}\right] &= \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \left[\nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \widehat{\nabla}^{b}_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})\right] \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \mathop{\mathbb{E}}_{\substack{\mathcal{M}_i\sim\boldsymbol{\mathcal{M}} \\ \tau_i\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}_i}}} \left[\frac{1}{\alpha N} \sum_{i=1}^{N} f_{\tau_i} \left(\widehat{H}_{\tau_i} - VaR_{\alpha}(H_{\tau_i}) - b\right) \mathbb{1}(\widehat{H}_{\tau_i} \leq \widehat{VaR}_{\alpha}(H_{\tau_i}))\right] \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau} \left(\widehat{H}_{\tau} - VaR_{\alpha}(H_{\tau}) - b\right) \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{VaR}_{\alpha}(H_{\tau}))\right] \qquad (4.10) \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) - \nabla_{\boldsymbol{\theta}} \mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}}) + \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau}\, b\, \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{VaR}_{\alpha}(H_{\tau}))\right] \qquad (4.11) \\
&= \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[f_{\tau}\, b\, \mathbb{1}(\widehat{H}_{\tau} \leq \widehat{VaR}_{\alpha}(H_{\tau}))\right] \leq \mathcal{U}\alpha b, \qquad (4.12)
\end{aligned}$$

where (4.11) follows from (4.10) by noting that the estimator without the baseline term is unbiased [74], and (4.12) is obtained by upper bounding f_{τ} with $\mathcal{U}$ and noting that $\mathbb{E}_{\substack{\mathcal{M}\sim\boldsymbol{\mathcal{M}} \\ \tau\sim p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}}} \left[\mathbb{1}(\widehat{H}_{\tau} \leq \widehat{VaR}_{\alpha}(H_{\tau}))\right] = \alpha$. ∎

Lemma 4.4.3 (VaR concentration bound from [44])**.** *Let X be a continuous random variable with a pdf f_X for which there exist $\eta, \Delta > 0$ such that $f_X(x) > \eta$ for all $x \in \left[VaR_{\alpha}(X) - \frac{\Delta}{2}, VaR_{\alpha}(X) + \frac{\Delta}{2}\right]$. Then, for any $\epsilon > 0$ we have*

$$Pr\left[|\widehat{VaR}_{\alpha}(X)_{\alpha} - VaR_{\alpha}(X)| \geq \epsilon\right] \leq 2\exp\left(-2n\eta^2 \min(\epsilon^2, \Delta^2)\right),$$

where $n \in \mathbb{N}$ is the number of samples employed to estimate $\widehat{VaR}_{\alpha}(X)$.

Proposition 4.4.4. *Let $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ and $\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ be policy gradient estimates with and without a baseline. Let f_H be the pdf of H_τ, for which there exist $\eta, \Delta > 0$ such that $f_H(H_\tau) > \eta$ for all $H_\tau \in \left[VaR_\alpha(H_\tau) - \frac{\Delta}{2}, VaR_\alpha(H_\tau) + \frac{\Delta}{2}\right]$. The number of samples n^* for which the estimation error ϵ of $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ is lower than the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ with at least probability $\delta \in (0,1)$ is given by*

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

Proof. The proof is straightforward by considering the estimation error ϵ of $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ equal to the upper bound of the bias of $\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ from Lemma 4.4.2, i.e., $\epsilon = \mathcal{U}\alpha b$. Then, we set

$$\delta = 2\exp\left(-2n^*\eta^2 \min(\mathcal{U}^2\alpha^2 b^2, \Delta^2)\right)$$

from Lemma 4.4.3, which gives the result through simple calculations. ■

The Proposition 4.4.4 proves that there is little incentive to choose the policy gradient estimator $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$ when the number of trajectories is lower than n^*, as its estimation error would exceed the bias introduced by the alternative estimator $\widehat{\nabla}^{b}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$. Unfortunately, it is not easy to compute n^* in our setting, as we do not assume to know the distribution of H_τ, but the requirement is arguably seldom matched in practice.

In Chapter 5 we will empirically show that the baseline $b = -VaR_\alpha(H_\tau)$ might benefit the variance of the policy gradient estimation, at the expense of the additional bias, which is anyway lower than the estimation error of $\widehat{\nabla}_{\boldsymbol{\theta}}\mathcal{E}^{\alpha}_{\boldsymbol{\mathcal{M}}}$.

5

EXPERIMENTAL ANALYSIS

In this chapter, we provide an extensive empirical evaluation of the proposed methodology over the two-phase learning process described in Figure 4.1. Especially, we investigate the ability to learn a general exploration strategy over a class of continuous environments (learning to explore), and how this pre-trained strategy is beneficial to RL. In Section 5.1, we first provide a brief overview of how the empirical evaluation is organized, followed by an exhaustive analysis of an illustrative domain in Section 5.2. Then, in Section 5.3, we assess the ability of MEMENTO to deal with larger classes of environments. Afterwards, in Section 5.4, we further extend our study by evaluating MEMENTO on a class of high-dimensional domains, including a visual-based one. Finally, in Section 5.5, we show how MEMENTO outperforms two algorithms that belong to the meta RL paradigm.

5.1 OVERVIEW

In the next sections, we will assess the ability of MEMENTO to learn to explore a class of multiple reward-free environments based on the exploration objective in (4.3). In other words, we seek to empirically show the claim presented in Chapter 4, namely the fact that by optimizing for the CVaR, we can obtain a more robust exploratory policy. This policy is general across different environments, and it does not suffer a drop in performance due to some adversarial configurations.

To this end, we first consider a class $\mathcal{M}$ composed of two different configurations of a continuous 2D gridworld domain, which we call the *GridWorld with Slope*. We use this simple setting as representative for the entire batch of experiments, discussing how the choice of the percentile of interest affects the exploration strategy. Then, we verify the ability of our method to scale with the size of the class of environments, by considering a class of 10 continuous gridworlds. We call this domain *MultiGrid*. Afterwards, we verify the ability of our method to scale with the dimensionality of the environments in the class, by considering a class of 29D Ant domains, and by subsequently doubling down on the dimensionality of the observations with a class of MiniGrid vision-based domains. Finally, we provide a comparison with two meta RL algorithms, which are MAML [26] and UML [31], of which we gave an outline in Section 3.3.

For all the experiments but one, we use a Gaussian distribution with diagonal covariance matrix to represent the parametric policy $\pi_{\boldsymbol{\theta}}$. It takes as input the environment state features and outputs an action vector $a \sim \mathcal{N}(\mu, \sigma^2)$. The mean μ is state-dependent and is the downstream output of a densely connected neural network. The standard deviation is state-independent and it is represented by a separated trainable vector. The dimension of μ, σ, and a vectors is equal to the action-space dimension of the environment. The only exception is given by the MiniGrid domain, because it is visual and discrete. In that case, we use a convolutional architecture, followed by a standard feed-forward neural network with output given by a categorical distribution. More details are provided in the corresponding Section 5.4.2.

In any considered domain, we highlight the significant benefit that the exploration strategy provides to RL problems specified in the same class of environments. Moreover, for both *GridWorld with Slope* and *MultiGrid*, we show that the exploration strategy learned with our approach is superior for RL w.r.t. a policy meta-trained with MAML or UML on the same class.

We report the parameters used in the experiments in Appendix C.

5.2 AN ILLUSTRATIVE DOMAIN: GRIDWORLD WITH SLOPE

In *GridWorld with Slope* (2D states, 2D actions), which we coded from scratch, the agent can move inside a map composed of four rooms connected by four narrow hallways, by choosing at each step how much to move on the x and y axes. One side of the environment measures 2 units and the agent can cover at most 0.2 units in a single step. Thus, the agent needs around 10 steps to go from one side to the other on a straight line. When the agent collides with the external borders or with the internal walls, it is re-positioned according to a custom function. This is done not only to make the interaction more realistic, but also to limit the possibility to have a negative infinite entropy resulting from the k-NN computation, which can occur whenever more than k samples lie on the same position. This precaution is particularly useful in our scenario, due to the way in which we create the two configurations GWS and GWN. Indeed, GWS is characterized by a south-facing slope, while GWN is characterized by a north-facing slope, both of which are present only in the upper half of the environment. Since the initial position of the agent is sampled in a small square in the top-right corner, it is easy to see that in the first epochs in the GWN environment, the agent would repeatedly collide with the top-border, leading in general to a much more lower entropy w.r.t. to GWS.

We only apply the slope to the upper half of the environment to obtain a convenient trade-off between the overall exploration complexity

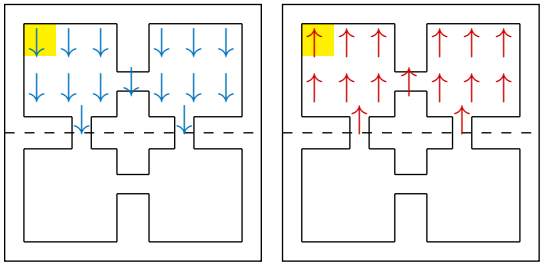


Figure 5.1: An informal illustration of the *GridWorld with Slope* domain.

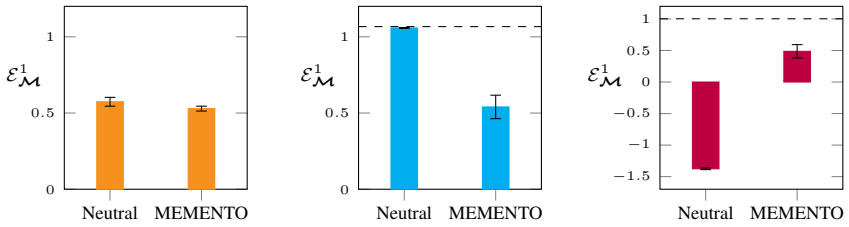
and the *risk* caused by the adverse configuration. Indeed, we noted that by applying the slope to the whole GridWorld, the advantage in terms of exploration entailed by the risk-averse approach is even higher, but it struggles to explore the bottom states of the environment within a reasonable number of samples. The slope is computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{2}, \frac{\Delta_{max}}{20})$, where $\Delta_{max} = 0.2$ is the maximum step that the agent can perform. A visual representation of the setting can be found in Figure 5.1, where the shaded areas denote the initial state distribution and the arrows render a slope that favors or contrasts the agent's movement.

This class of environments is unbalanced (and thus interesting to our purpose) for two reasons: first, the GWN configuration is more challenging from a pure exploration standpoint, since the slope prevents the agent from easily reaching the two bottom rooms; secondly, the distribution over the class is also unbalanced, as it is $p_{\mathcal{M}} = [Pr(\text{GWS}), Pr(\text{GWN})] = [0.8, 0.2]$, meaning that the adversarial configuration is sampled only with a probability of 0.2, as a representative of the worst-case scenario.

We execute the algorithm over 200 trajectories with exploration horizon $T = 400$. Being the batch-size $B = 5$, we have $N = 40$ mini-batches. Due to the sampling distribution $p_{\mathcal{M}}$, in each epoch we have on average 32 mini-batches containing trajectories deriving from the interaction of the agent with GWS and 8 mini-batches containing trajectories deriving from the interaction of the agent with GWN. We set the percentile $\alpha = 0.35$, hence the policy update is done on a dataset $\mathcal{D}_\alpha$ containing 14 mini-batches. The α-percentile results from an empirical analysis that will be discussed in Section 5.2.1. Essentially, we choose α to ensure that $\mathcal{D}_\alpha$ will likely contain trajectories from both the configurations.

We compare MEMENTO against *Neutral*, which is a simplified version of MEMENTO with $\alpha = 1$,[1] to highlight the importance of percentile sensitivity w.r.t. a naïve approach to the multiple environments scenario. The methods are evaluated in terms of the state visitation entropy $\mathcal{E}^1_{\mathcal{M}}$ induced by the exploration strategies they learn. In Figure 5.2, we compare the performance of the optimal exploration strategy obtained

1 The pseudocode is identical to Algorithm 4 except that all trajectories affect the gradient estimate in (4.9).



(a) $p_{\mathcal{M}} = [0.8, 0.2]$ **(b)** $p_{\mathcal{M}} = [1, 0]$ **(c)** $p_{\mathcal{M}} = [0, 1]$

Figure 5.2: Comparison of the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ obtained by MEMENTO ($\alpha = 0.35$) and Neutral ($\alpha = 1$) in the *GridWorld with Slope* domain. The polices are trained (150 epochs, 8×10^4 samples per epoch) on the configuration **(a)** and tested on **(a, b, c)**. The dashed lines in **(b, c)** represent the optimal performance in that specific configuration. We provide 95% c.i. over 4 runs.

by running MEMENTO ($\alpha = 0.35$) and Neutral ($\alpha = 1$) for 150 epochs on the GridWorld with Slope class ($p_{\mathcal{M}} = [0.8, 0.2]$). We show that the two methods achieve a very similar expected performance over the class (Figure 5.2a). However, this expected performance is the result of a (weighted) average of very different contributions. As anticipated, Neutral has a strong performance in GWS ($p_{\mathcal{M}} = [1, 0]$, Figure 5.2b), which is close to the configuration-specific optimum (dashed line), but it displays a bad showing in the adverse GWN ($p_{\mathcal{M}} = [0, 1]$, Figure 5.2c). Conversely, MEMENTO learns a strategy that is much more robust to the configuration, showing a similar performance in GWS and GWN, as the percentile sensitivity prioritizes the worst case during training.

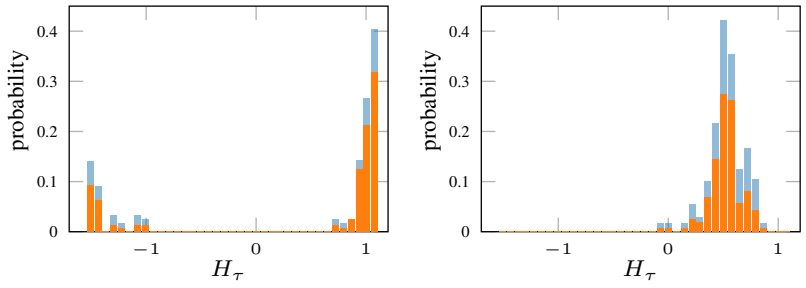
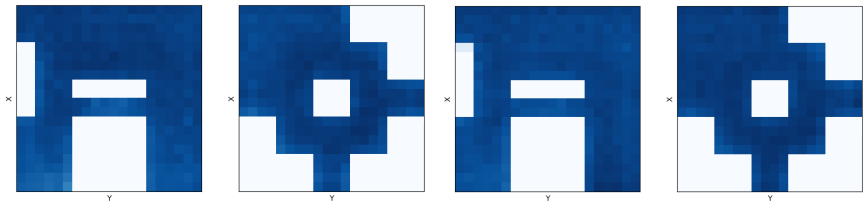


Figure 5.3: Empirical distributions of Neutral (left) and MEMENTO (right)

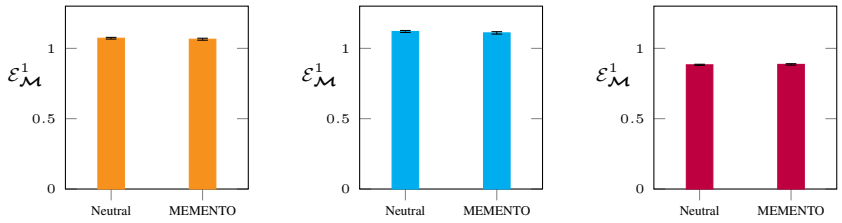
To confirm this conclusion, it is worth looking at the actual distribution that is generating the expected performance in Figure 5.2a. To this end, in Figure 5.3, we provide the empirical distribution of the trajectory-wise performance (H_τ), considering a batch of 200 trajectories with $p_{\mathcal{M}} = [0.8, 0.2]$. It clearly shows that Neutral is heavy-tailed towards lower outcomes, whereas MEMENTO concentrates around the mean. This suggests that with a smart choice of α we can induce a good exploration outcome for every trajectory (and any configuration), while without percentile sensitivity we cannot hedge against the risk of particularly bad outcomes.

However, let us point out that not all classes of environments would expose such an issue for a naïve, risk-neutral approach. To provide



(a) MEMENTO **(b)** Neutral

Figure 5.4: Heatmaps of the state visitations (200 trajectories, $T = 400$) induced by the exploration policies trained with MEMENTO ($\alpha = 0.35$) **(a)** and Neutral ($\alpha = 1$) **(b)** in the *GridWorld Counterexample* domain.



(a) $p_{\mathcal{M}} = [0.8, 0.2]$ **(b)** $p_{\mathcal{M}} = [1, 0]$ **(c)** $p_{\mathcal{M}} = [0, 1]$

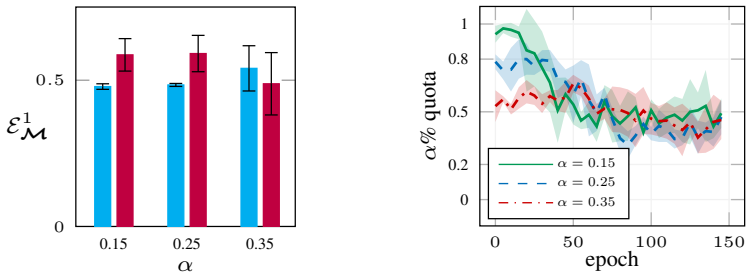
Figure 5.5: Comparison of the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ obtained by MEMENTO ($\alpha = 0.35$) and Neutral ($\alpha = 1$) in the *GridWorld Counterexample* domain. The polices are trained (50 epochs, 8×10^4 samples per epoch) on the configuration **(a)** and tested on **(a, b, c)**. We provide 95% c.i. over 4 runs.

some evidence of this fact, we provide a convenient example. We consider two GridWorld environments that differ for the shape of the traversable area (see Figure 5.4), sampled according to $p_{\mathcal{M}} = [0.8, 0.2]$, and we run MEMENTO with $\alpha = 0.35$ and Neutral ($\alpha = 1$), obtaining the two corresponding exploration policies. In Figure 5.5 we show the performance (measured by $\mathcal{E}^1_{\mathcal{M}}$) obtained by executing those policies on each setting. Clearly, regardless of what configuration we consider, there is no advantage deriving from the use of a risk-averse approach as MEMENTO, meaning that the class of environments $\mathcal{M}$ is balanced in terms of hardness of exploration.

Nonetheless, it is fair to assume that, in terms of risk, real classes of environments will be more similar to the *GridWorld with Slope* domain than to the one of the counterexample. Those are the settings we care about, as they require nuanced solutions (e.g., MEMENTO) for scenarios with multiple environments.

5.2.1 *On the Value of the Percentile*

In this section, we discuss how the choice of the percentile (i.e., parameter α) impacts the results reported in Figure 5.2.



(a) α sensitivity **(b)** α% composition

Figure 5.6: The behaviour of MEMENTO with different values of α. We provide 95% c.i. over 4 runs.

We repeatedly train MEMENTO (for 150 epochs) in the *GridWorld with Slope* domain considering different values for α, and we compare the resulting exploration performance $\mathcal{E}^1_{\mathcal{M}}$ as in Figure 5.2. In Figure 5.6a, we can see that the lower α we choose, the more we prioritize GWN (right bar for every α) at the expense of GWS (left bar). Note that this trend carries on with increasing α, ending in the values of Figures 5.2b, 5.2c. The reason for this behavior is quite straightforward, and we empirically support it with the results in Figure 5.6b: the smaller is α, the larger is the share of trajectories from the adverse configuration (GWN) ending up in the percentile at first, and thus the more GWN affects the policy update (see the gradient in (4.9)).

5.2.2 *The Baseline*

In Section 4.4.2 we provided some theoretical corroborations to support the claim that the bias introduced by our policy gradient estimator is lower than the one introduced by the need of estimating the VaR (see Proposition 4.4.4). In this section, we empirically show that the baseline $b = -VaR_\alpha(H_\tau)$ represents a valid or even better alternative despite its simplicity.

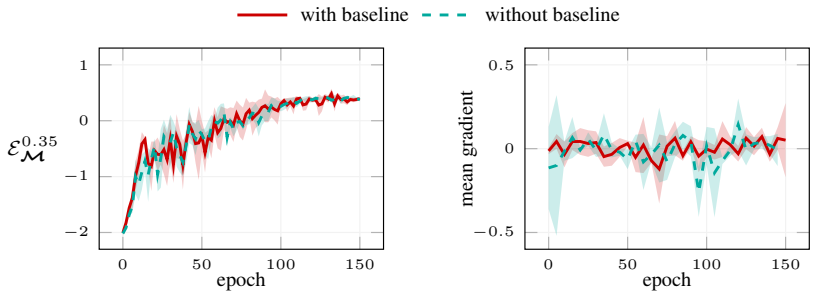
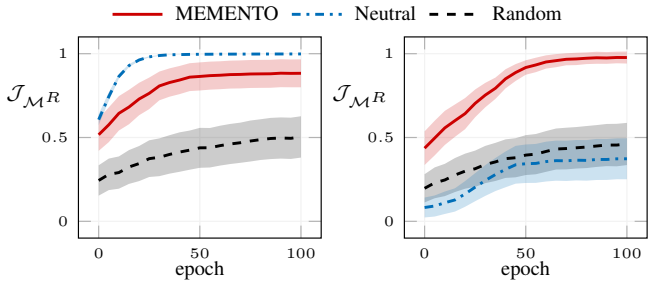
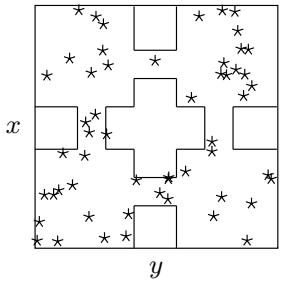


Figure 5.7: Comparison of the exploration performance $\mathcal{E}^{0.35}_{\mathcal{M}}$ (left) and sampled gradients of the policy mean (right) achieved by MEMENTO ($\alpha = 0.35$) with and without the baseline $b = -VaR_\alpha(H_\tau)$ in the policy gradient estimation (4.9). We provide 95% c.i. over 4 runs.

In Figure 5.7 (left), we can see that the exploration performance $\mathcal{E}^{\alpha}_{\mathcal{M}}$ obtained by MEMENTO with and without the baseline is essentially



(a) RL on GWS (left) and GWN (right)



(b) Sampled goals

Figure 5.8: Comparison of the average return $\mathcal{J}_{\mathcal{M}^R}$ as a function of learning epochs (1.2×10^4 samples per epoch) achieved by TRPO initialized with MEMENTO ($\alpha = 0.35$), Neutral ($\alpha = 1$), and random exploration strategies, when dealing with a set of RL tasks specified on the *GridWorld with Slope* domain **(a)**. We provide 95% c.i. over 50 randomly sampled goal locations **(b)**.

the same in the illustrative *GridWorld with Slope* domain. Whereas Figure 5.7 (right) suggests a slightly inferior variance for the policy gradient estimate employed by MEMENTO with the baseline.

5.2.3 *RL with a General Exploration Strategy*

In this section, we illustrate how beneficial is the pre-trained exploration strategy to deal with any RL problem one could specify in the GridWorld with Slope class.

To this end, we design a family of MDPs $\mathcal{M}^R$, where $\mathcal{M} \in \{\text{GWS}, \text{GWN}\}$, and R is any sparse reward function that gives 1 when the agent reaches the area nearby a random goal location ($d \leq 0.1$) and 0 otherwise. On this family, we compare the performance achieved by TRPO [65] with different initializations: the exploration strategies learned by MEMENTO ($\alpha = 0.35$) and Neutral ($\alpha = 1$), or a random policy (Random). These three variations are evaluated in terms of their average return $\mathcal{J}_{\mathcal{M}^R}$, which is defined as $\mathcal{J}^{\pi}_{\mathcal{M}^R} = \mathbb{E}_{\tau \sim p_{\pi,\mathcal{M}}}\left[R(\tau)\right]$, over 50 randomly generated goal locations, of which we provide an illustration in Figure 5.8b.

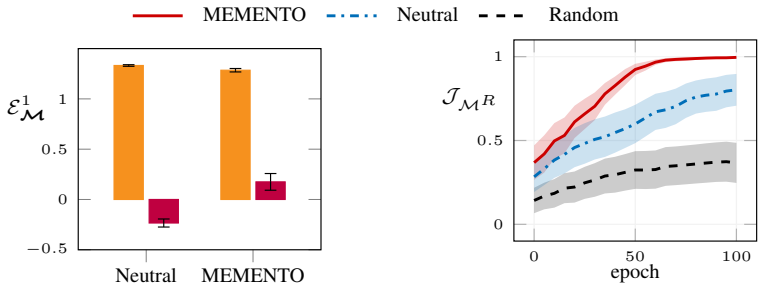
In Figure 5.8a, we show the results of the comparison. As expected, the performance of TRPO with Neutral is competitive in the GWS

configuration, but it falls sharply in the GWN configuration, where it is not significantly better than TRPO with Random. Instead, the performance of TRPO with MEMENTO is strong on both GWS and GWN. Despite the simplicity of the domain, solving an RL problem in GWN with an adverse goal location is far-fetched for both a random initialization and a naïve solution to the reward-free exploration over multiple environments.

5.3 SCALING TO LARGER CLASSES OF ENVIRONMENTS

In this section, we show that the previous analyses extend straightforwardly to larger classes of environments.

To this end, we consider a class $\mathcal{M}$ composed of ten different configurations of the continuous gridworlds presented in Section 5.2, which we call the *MultiGrid* domain. These environments differ for both the shape and the type of slope to which they are subject to. The adverse configuration is still GWN, but the slope is computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{2.6}, \frac{\Delta_{max}}{20})$, where $\Delta_{max} = 0.2$. The other 9 gridworlds have instead a different arrangement of the walls (see the heatmaps in Figure 5.10) and the slope, computed as $s \sim \mathcal{N}(\frac{\Delta_{max}}{3.2}, \frac{\Delta_{max}}{20})$ with $\Delta_{max} = 0.2$, is applied over the entire environment. Two configurations are subject to south-facing slope, three to east-facing slope, one to south-east-facing slope and three to no slope at all.

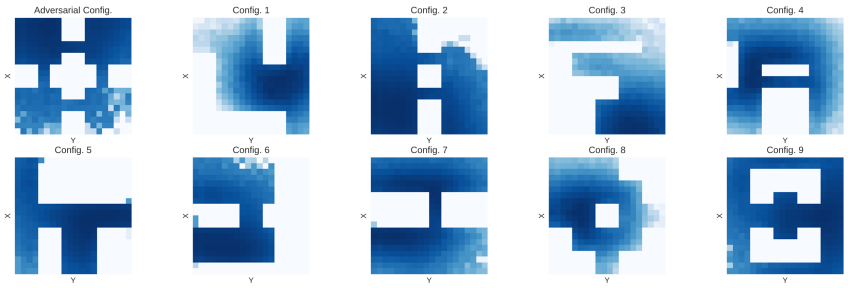


(a) Learning to explore **(b)** RL

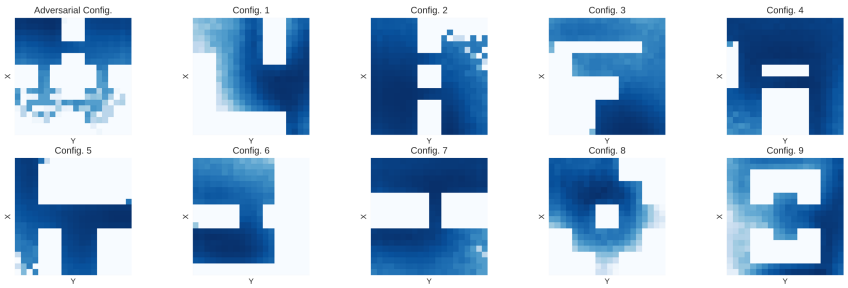
Figure 5.9: Comparison of the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ (95% c.i. over 4 runs), and the average return $\mathcal{J}_{\mathcal{M}^R}$ (95% c.i. over 50 tasks) obtained by TRPO with corresponding initialization, achieved by MEMENTO ($\alpha = 0.1$) and Neutral ($\alpha = 1$) in the *MultiGrid* domain (50 epochs per 2×10^5 samples).

As before, we compare MEMENTO ($\alpha = 0.1$) and Neutral ($\alpha = 1$) on the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ achieved by the optimal strategy, in this case considering a uniformly distributed $p_{\mathcal{M}}$. While the average performance of Neutral is slightly higher across the class (Figure 5.9a, left bar), MEMENTO still has a decisive advantage in the worst-case configuration (Figure 5.9a, right bar). Likewise, just as in Section 5.2.3, this advantage transfer to the RL phase, where we compare the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO with MEMENTO, Neutral, and Random initializations over 50 random goal locations in the GWN configuration,

which are the same as in the *GridWorld with Slope* experiment, i.e, the ones in Figure 5.8b. The results of the RL comparison are reported in Figure 5.9b.



(a) MEMENTO



(b) Neutral

Figure 5.10: Heatmaps of the state visitations (200 trajectories, $T = 400$) induced by the exploration policies trained with MEMENTO ($\alpha = 0.1$) **(a)** and Neutral ($\alpha = 1$) **(b)** in the *MultiGrid* domain. We provide the average over 4 seeds.

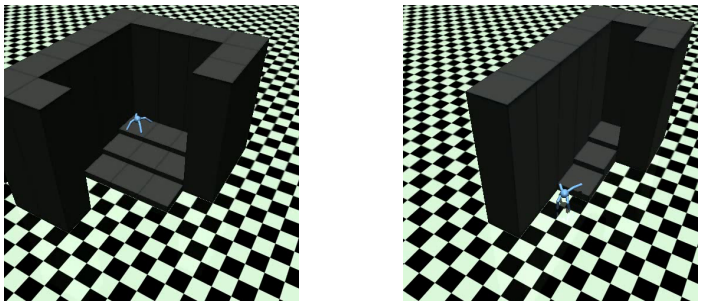
In Figure 5.10, we report the state-visitation frequencies achieved by MEMENTO (Figure 5.10a) and Neutral (Figure 5.10b) in each configuration of the *MultiGrid* domain. Clearly, MEMENTO manages to obtain a better exploration in the adversarial configuration w.r.t. Neutral, especially in the bottom part of the environment, which is indeed the most difficult part to visit. On the other environments, the performance is overall comparable.

5.4 SCALING TO INCREASING DIMENSIONS

In this section, we show that the previous analyses nicely scales to classes of (challenging) high-dimensional domains, thanks to the flexibility of the entropy estimator (4.7).

5.4.1 *Ant*

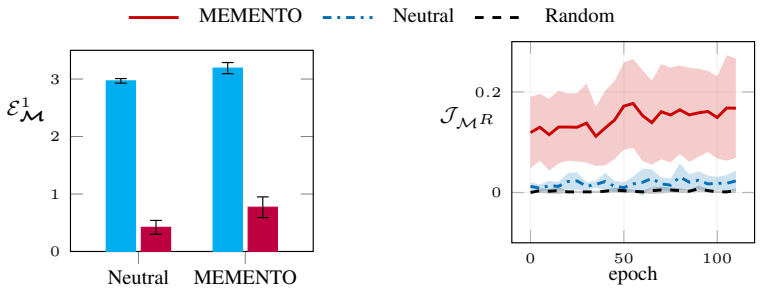
To this end, we first consider a class $\mathcal{M}$ consisting of two Ant environments, with 29D states and 8D actions. A visualization of both the environments is reported in Figure 5.11. In order to build them, we adapt the Ant-Maze environment of *rllab* [22], exploiting its malleability to



(a) Ant Stairs Down **(b)** Ant Stairs Up

Figure 5.11: Illustration of the *Ant Stairs* domain. We show a render of the *Ant Stairs Down* environment **(a)** and of the adverse *Ant Stairs Up* environment **(b)**.

build two configurations that fit our purposes. In the first, sampled with probability $p_{\mathcal{M}_1} = 0.8$, the Ant faces a wide descending staircase (*Ant Stairs Down*), made up of 3×3 blocks of decreasing height and a final 1×3 flat area. In the second, the Ant faces a narrow ascending staircase (*Ant Stairs Up*, sampled with probability $p_{\mathcal{M}_2} = 0.2$), which is significantly harder to explore than the former. It is made up of an initial square (the initial position of the Ant), followed by three blocks of increasing height. Note that each block has a side length slightly greater than the Ant size. In the mold of the gridworlds in Section 5.2, these two configurations are specifically designed to create an imbalance in the class.



(a) Learning to explore **(b)** RL

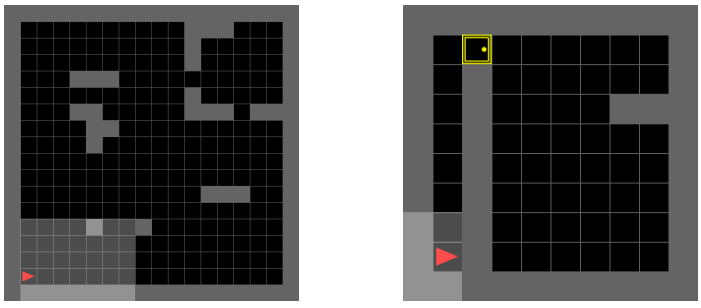
Figure 5.12: Comparison of the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ (95% c.i. over 4 runs), and the average return $\mathcal{J}_{\mathcal{M}^R}$ (95% c.i. over 8 tasks) obtained by TRPO with corresponding initialization, achieved by MEMENTO ($\alpha = 0.2$) and Neutral ($\alpha = 1$) in the *Ant* domain (400 epochs per 6×10^4 samples).

As in Section 5.2, we compare MEMENTO ($\alpha = 0.2$) against Neutral ($\alpha = 1$) on the exploration performance $\mathcal{E}^1_{\mathcal{M}}$ achieved after 400 epochs. Note that $\mathcal{E}^{\alpha}_{\mathcal{M}}$ is maximized over the x, y spatial coordinates of the ant's torso during the *learning to explore* phase. MEMENTO fares slightly better than Neutral both in the worst-case configuration (Figure 5.12a, right bar) and, surprisingly, in the easier one (Figure 5.12a, left bar). We do not expect the latter fact to happen in general: arguably, MEMENTO will always perform better (or on par) in the worst case, but on average it will be worse. Then, we design a set of incrementally challenging

sparse-rewards RL tasks in the *Ant Stairs Up*, which give reward 1 upon reaching a certain step of the staircase. Also in this setting, TRPO with MEMENTO initialization outperforms TRPO with Neutral and Random in terms of the average return $\mathcal{J}_{\mathcal{M}^R}$ (Figure 5.12b). Note that these sparse-reward continuous control tasks are particularly arduous: TRPO with Neutral and Random barely learns anything, while even TRPO with MEMENTO does not handily reach the optimal average return (1) within 100 epochs.

5.4.2 *MiniGrid - A Visual Domain*

In this section, we double down on the dimensionality of the observations, pursuing reward-free exploration of a class of multiple vision-based domains. We use the MiniGrid suite [16], which consists of a set of fast and light-weighted gridworld environments. The environments are partially observable, with the dimension of the agent's field of view having size $7 \times 7 \times 3$. Both the observation space $\mathcal{S}$ and the action space $\mathcal{A}$ are discrete, and in each tile of the environment there can be only one object at the same time. The set of objects is $O = \{wall, floor, lava, door, key, ball, box, goal\}$. The agent can move inside the grid and interact with these objects according to their properties. In particular, the actions comprise turning left, turning right, moving forward, picking up an object, dropping an object and toggling, i.e., interacting with the objects (e.g., to open a door). The strength of MiniGrid is its malleability: besides the off-the-shelf environments, it allows to easily develop custom implementations. We exploit this property to build two custom environments, of which we provide an illustration in Figure 5.13. The first one has a size of 18×18, and it simply contains some sparse walls. The second one is smaller, 10×10, and is characterized by the presence of a door at the top of a narrow hallway. The door is closed but not locked, meaning that the agent can open it without using a key. In both the environments, the agent is the red triangle starting in the bottom-left corner.



(a) Easy configuration **(b)** Adverse configuration

Figure 5.13: Illustration of the MiniGrid domain.

Before introducing the experimental setting, it is worth making a brief description of the architecture that we will use to perform this analysis. Indeed, we have to handle $7 \times 7 \times 3$ RGB images as

inputs, and the set of all the possible inputs is discrete and finite. Hence, we need to change our policy, which is a Gaussian policy, to one that is able to process images and output discrete actions. We adopt the architecture recently proposed by [67] and that we shortly exposed in Section 3.1.5. Thus, we use a random encoder made up of 3 convolutional layers with kernel 2, stride 1, and padding 0, each activated by a ReLU function, and with 16, 32 and 64 filters respectively. The first ReLU is followed by a 2D max pooling layer with kernel 2. The output of the encoder is a 64 dimensional tensor, which is then fed to a feed-forward neural network with two fully-connected layers with hidden dimension 64 and a Tanh activation function.

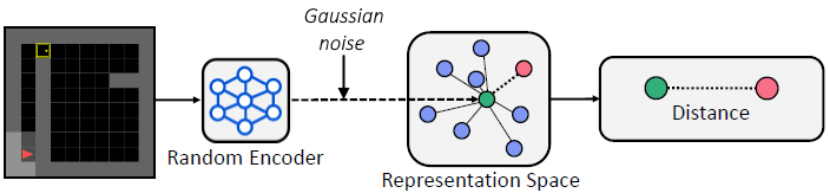
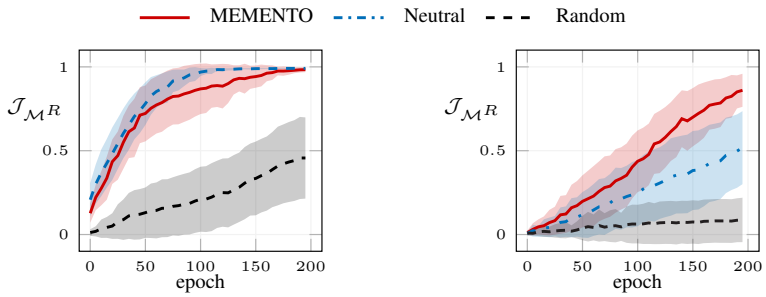


Figure 5.14: Illustration of the procedure to perform the k-NN computation in the representation space generated by a fixed random encoder, adapted from [67].

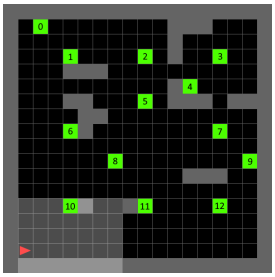
As regards the training procedure, everything remains as we described it in Section 4.4, except for two differences. The first difference is that the k-NN computation is performed on the representation space generated by a fixed random encoder. Note that this random encoder is not part of the policy. It is randomly initialized and not updated during the training in order to produce a more stable entropy estimate. In addition, before computing the distances, we apply to its output a random Gaussian noise $\epsilon \sim \mathcal{N}(0.001, 0.001)$ truncated in $[0, 0.001]$. We do this to avoid the aliasing problem, which occurs when we have many samples (more than k) in the same position, thus having zero distance and producing a negative infinite entropy estimate. The homogeneity of the MiniGrid environments in terms of features make this problem more frequent. In Figure 5.14, we report a visualization of the procedure. The second difference is the addition of a bootstrapping procedure for the easy configuration, meaning that we use only a subset of the mini-batches of the easy configuration to update the policy. Especially, we randomly sample a number of mini-batches that is equal to the dimension of the $\mathcal{D}_\alpha$ dataset so that Neutral uses the same number of samples of MEMENTO. The reason why we avail this method is to avoid a clear advantage for Neutral in learning effective representations, since it usually access more samples than MEMENTO. Note that it is not a stretch, since we are essentially balancing the information available to the two algorithms.

Following the line of the experiments in the *GridWorld with Slope* domain, we thus create an unbalanced couple of environments. The distribution over the class is unbalanced, as it is $p_{\mathcal{M}} = [0.8, 0.2]$, where 0.2 refers to the adverse configuration of Figure 5.13b. Moreover, the adverse configuration owes its difficulty to two factors: first, the

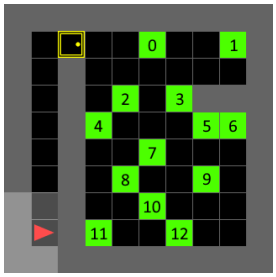


(a) RL (easy configuration)

(b) RL (adverse configuration)



(c) Goals (easy configuration)



(d) Goals (adverse configuration)

Figure 5.15: Comparison of the average return $\mathcal{J}_{\mathcal{M}^R}$ as a function of learning epochs (7.5×10^3 samples per epoch) achieved by TRPO initialized with MEMENTO ($\alpha = 0.3$), Neutral ($\alpha = 1$), and random exploration strategies (300 epochs per 1.5×10^4 samples), when dealing with a set of RL tasks specified on both the configurations of the *MiniGrid* domain **(a)**, **(b)**. We provide 95% c.i. over 13 goal locations **(c)**, **(d).**

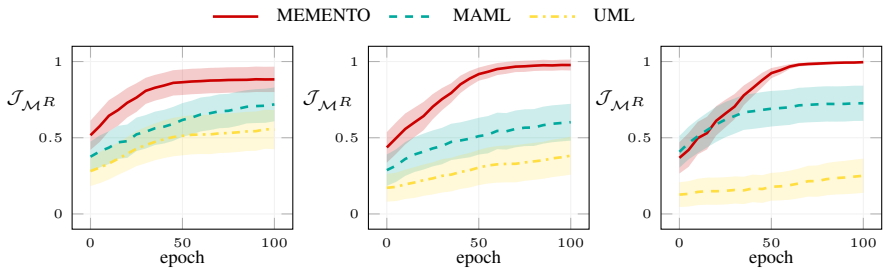
closed door is by itself a challenging obstacle; second, we modify the movement of the agent so that the direction is given by the bottom of the triangle instead of the top. The intuition is that by doing this we are essentially changing the shape of the agent, hence causing the policy to struggle in the exploration. We train the policy for 300 epochs, using 100 trajectories with horizon $T = 150$. For MEMENTO, we set $\alpha = 0.3$. As usual, we then compare the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO with the MEMENTO, Neutral and Random initialization. We manually place 13 goals in the environment to obtain a uniform coverage. In Figures 5.15c, 5.15d we show their locations. The agent receives a reward of $+1$ if its position matches the one of the goal, otherwise 0. In Figures 5.15a, 5.15b we report the RL results on both the easy and the adverse configurations.

Also in this case, TRPO with MEMENTO initialization outperforms Neutral and Random in terms of the average return $\mathcal{J}_{\mathcal{M}^R}$ in the adverse configuration, while having a comparable performance to Neutral in the easy one. Interestingly, in the adverse environment, MEMENTO has an initial performance very similar to the others, but it then allows to quickly learn. Note also that the tasks are not straightforward, as TRPO with random initialization barely manages to have an average return $\mathcal{J}_{\mathcal{M}^R}$ that is greater than 0.

5.5 COMPARISON WITH META-RL

In this section, we compare the exploration strategy learned with MEMENTO w.r.t. a policy trained with two meta RL algorithms, which are MAML [26] and UML [31]. The motivation behind the use of UML has to be sought in the aim of finding a more sensible comparison. In fact, in contrast to MAML, it does not require manual task design, since the reward function is automatically extracted from an unsupervised skills discovery method (i.e., DIAYN). Hence, not assuming the reward during the meta-training phase, UML is surely a more sound comparison. Nonetheless, it is worth to underline the fact that the meta RL paradigm as a whole is not the most significant comparison, since with MEMENTO we are not focusing on adaptation, but on finding a policy that is naturally capable of exploring a class of unknown CMPs.

We meta-train a policy with MAML on the same *GridWorld with Slope* ($p_{\mathcal{M}} = [0.8, 0.2]$) and *MultiGrid* (uniformly distributed $p_{\mathcal{M}}$) domains that we have previously presented. During meta-training, we provide MAML with full access to the tasks (i.e., reward functions) that we will consider in the meta-testing phase. Note that this gives MAML a great edge over MEMENTO and UML, which operate reward-free training. As in [26], we compute the gradient updates with vanilla policy gradient and TRPO as meta-optimizer. Since our setting resembles what [26] call *2D navigation*, we use the same meta-batch size of 20, i.e., considering 20 tasks for each batch of tasks, for a total of 200 number of batches. The policy is trained to maximize the performance after 1 policy gradient update using 30 trajectories. As regards UML, we use DIAYN [25] to acquire the task proposals and then we feed them into MAML with the setting we just described. We use the default hyperparameters of DIAYN and a number of skills equal to the one used in MAML, i.e., 20.



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

Figure 5.16: Comparison of the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO (1.2×10^4 samples per epoch) initialized with a MEMENTO exploration strategy ($\alpha = 0.35$ **(a)**, $\alpha = 0.1$ **(b)**) and a MAML and UML meta-policy, when dealing with a set of RL tasks in the *GridWorld with Slope* **(a)** and the *MultiGrid* **(b)**. We provide 95% c.i. over 50 tasks.

As in previous sections, we consider the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO initialized with the exploration strategy learned by MEMENTO or the meta-policy learned by MAML or UML. TRPO with MEMENTO

fares clearly better than TRPO with MAML in all the configurations (Figures 5.16a, 5.16b). Note that as opposed to [26], where the reward function is dense, we keep the one used up to now, i.e., assigning $+1$ only when the agent is in the surroundings of the goal and 0 otherwise. MEMENTO is also remarkably superior to UML, which, interestingly, performs worse than MAML in all the configurations, with a substantial drop in the *MultiGrid* domain. In Figure 5.17 we report the discriminability term $\log q_\phi(z|s)$ during the training of DIAYN for both *GridWorld with Slope* and *MultiGrid*: while DIAYN somewhat manage to increase the trajectories discriminability in the former, it is not able to tackle the additional complexity provided by *MultiGrid*.

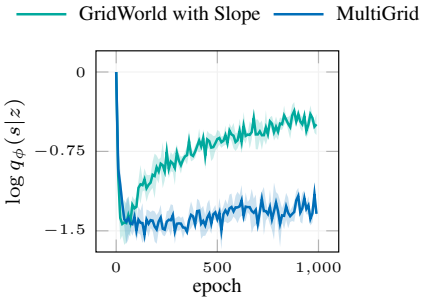
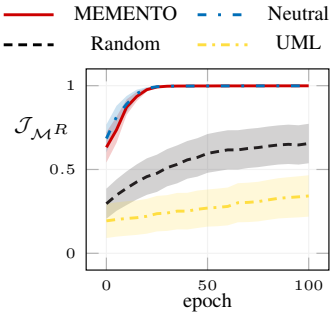
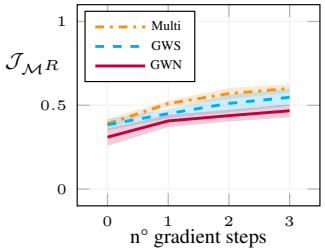


Figure 5.17: Comparison of the discriminability term $\log q_\phi(z|s)$ achieved by DIAYN in the *GridWorld with Slope* and the *MultiGrid*. We provide 95% c.i. over 4 runs.



(a) MultiGrid - An easy configuration



(b) MAML: fast adaptation

Figure 5.18: Comparison of the average return $\mathcal{J}_{\mathcal{M}^R}$ achieved by TRPO (1.2×10^4 samples per epoch) initialized with a MEMENTO exploration strategy ($\alpha = 0.1$), Neutral ($\alpha = 1$), and a UML meta-policy, when dealing with a set of RL tasks in one of the non-adverse configurations of the *MultiGrid* domain. We provide 95% c.i. over 50 tasks **(a)**. We illustrate the fast-adapting behaviour of the MAML policy in **(b)**.

In Figure 5.18, we provide two additional plots. The one on the left is another comparison on the *MultiGrid* domain, but in this case we execute TRPO initialized with MEMENTO, Neutral, Random and UML on one of the other nine environments, i.e., one that is not adverse. Reasonably (given the results in 5.3), MEMENTO and Neutral have an excellent and similar performance. Instead, despite a small improvement w.r.t. 5.16b, UML still struggles. Actually, it performs worse than Random, meaning that the learned reward functions might induce

a negative bias in this case. In Figure 5.18b, we instead provide a visualization of the fast-adapting behavior of the MAML policy, which is indeed the objective for which it is thought.

6

CONCLUSIONS

In this thesis, we addressed the problem of learning to explore a class of multiple reward-free environments with a unique general strategy. First, we formulated the problem within a tractable MSVE objective with percentile sensitivity, so that to obtain a unique exploration strategy that is transverse w.r.t. a class of reward-free environments, i.e, that does not suffer the presence of a particularly adverse configuration. Then, we presented a policy gradient algorithm, MEMENTO, to optimize this objective, exploiting a non-parametric entropy estimator. Finally, we provided an extensive experimental analysis to show its ability in learning to explore and the benefits it brings to subsequent RL problems, where the rewards are sparsely defined. During this analysis, we assess the ability of MEMENTO to both scale to larger classes of environments, testing it on 10 different configurations, and to high-dimensional domains, testing it on a Ant environment with 29D states and 8D actions. Moreover, we also prove its efficacy in a visual-based domain, i.e., MiniGrid. We also compare MEMENTO with two meta RL algorithms, which do not succeed in outperforming it, despite one of them has even access to the reward functions.

As a final note, it is worth mentioning some alternative settings in which MEMENTO can be employed with benefit. In this work, we focused on a specific solution for an essentially multi-objective problem, by establishing a preference over the environments through the CVaR objective. Instead, as already discussed in Section 4.2, a future direction could pursue learning a direct approximation of the Pareto frontier [54] of the exploration strategies over multiple environments. Alternatively, we could replace the class of environments with a single CMP specified under uncertainty [63], and deal with the *robust reward-free exploration* problem with little or no modifications to MEMENTO. Another promising direction is to assume some control over the class distribution during the learning to explore process, either by an external supervisor or by the agent itself [48]. Lastly, future work may establish regret guarantees for the reward-free exploration problem over multiple environments, in a similar flavor to the reward-free RL problem in a single environment [40].

We believe that this work motivates the importance of designing specific solutions to the relevant reward-free exploration problem over multiple environments, and that it represents a further step towards the achievement of artificial general intelligence.

BIBLIOGRAPHY

[1] Jiří Ajgl and Miroslav Šimandl. "Differential entropy estimation by particles." In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 11991–11996 (cit. on pp. 7, 26).

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. "Learning dexterous in-hand manipulation." In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20 (cit. on pp. XVI, 1).

[3] David Barber and Felix Agakov. "The IM algorithm: a variational approach to Information Maximization." In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. 2003, pp. 201–208 (cit. on p. 23).

[4] Jan Beirlant, Edward J Dudewicz, László Györfi, and Edward C Van der Meulen. "Nonparametric entropy estimation: An overview." In: *International Journal of Mathematical and Statistical Sciences* 6.1 (1997), pp. 17–39 (cit. on pp. 7, 25, 26).

[5] Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. "Unifying count-based exploration and intrinsic motivation." In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 1479–1487 (cit. on p. 22).

[6] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. ISBN: 9780691079516 (cit. on p. 14).

[7] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. "Dota 2 with large scale deep reinforcement learning." In: *arXiv preprint arXiv:1912.06680* (2019) (cit. on pp. XVI, 1).

[8] Dimitri P Bertsekas. "Nonlinear programming." In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334 (cit. on p. 29).

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738 (cit. on p. 6).

[10] Andrea Bonarini, Alessandro Lazaric, and Marcello Restelli. "Incremental skill acquisition for self-motivated learning animats." In: *International Conference on Simulation of Adaptive Behavior*. Springer. 2006, pp. 357–368 (cit. on pp. XVII, 2).

[11] Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. "Reinforcement learning, fast and slow." In: *Trends in cognitive sciences* 23.5 (2019), pp. 408–422 (cit. on p. 30).

[12] Bruce Lee Bowerman. "Nonstationary Markov decision processes and related topics in nonstationary Markov chains." In: *PhD thesis, Iowa State University* (1974) (cit. on p. 11).

[13] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. "Exploration by random network distillation." In: *International Conference on Learning Representations*. 2018 (cit. on pp. XVII, 2, 22).

[14] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. "Deep clustering for unsupervised learning of visual features." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149 (cit. on p. 26).

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on p. 26).

[16] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for OpenAI Gym*. https://github.com/maximecb/gym-minigrid. 2018 (cit. on p. 54).

[17] Yinlam Chow and Mohammad Ghavamzadeh. "Algorithms for CVaR optimization in MDPs." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*. 2014, pp. 3509–3517 (cit. on pp. XVII, 2, 28).

[18] Alexander I Cowen-Rivers, Daniel Palenicek, Vincent Moens, Mohammed Abdullah, Aivar Sootla, Jun Wang, and Haitham Ammar. "Samba: Safe model-based & active reinforcement learning." In: *arXiv preprint arXiv:2006.09436* (2020) (cit. on p. 29).

[19] Imre Csiszár and Zsolt Talata. "Context tree estimation for not necessarily finite memory processes, via BIC and MDL." In: *IEEE Transactions on Information theory* (2006) (cit. on p. 75).

[20] Marc Deisenroth and Carl E Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer. 2011, pp. 465–472 (cit. on p. 29).

[21] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. "A survey on policy search for robotics." In: *Foundations and trends in Robotics* 2.1-2 (2013), pp. 388–403 (cit. on p. 37).

[22] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. "Benchmarking deep reinforcement learning for continuous control." In: *Proceedings of the International Conference on Machine Learning*. 2016 (cit. on p. 52).

[23] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. "RL^2: Fast reinforcement learning via slow reinforcement learning." In: *arXiv preprint arXiv:1611.02779* (2016) (cit. on pp. XVII, 2, 30).

[24] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. "First return, then explore." In: *Nature* 590.7847 (2021), pp. 580–586 (cit. on p. 23).

[25] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. "Diversity is All You Need: Learning Skills without a Reward Function." In: *International Conference on Learning Representations*. 2018 (cit. on pp. XVII, 2, 23, 31, 57).

[26] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135 (cit. on pp. XVII, 2, 30, 31, 44, 57, 58).

[27] Marguerite Frank, Philip Wolfe, et al. "An algorithm for quadratic programming." In: *Naval research logistics quarterly* 3.1-2 (1956), pp. 95–110 (cit. on p. 24).

[28] Adam Gaier and David Ha. "Weight Agnostic Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019 (cit. on p. 26).

[29] Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Yoshua Bengio, and Sergey Levine. "InfoBot: Transfer and Exploration via the Information Bottleneck." In: *International Conference on Learning Representations*. 2018 (cit. on p. 23).

[30] Zhaohan Daniel Guo, Mohammad Gheshlagi Azar, Alaa Saade, Shantanu Thakoor, Bilal Piot, Bernardo Avila Pires, Michal Valko, Thomas Mesnard, Tor Lattimore, and Rémi Munos. "Geometric Entropic Exploration." In: *arXiv preprint arXiv:2101.02055* (2021) (cit. on pp. XVII, 2, 25).

[31] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. "Unsupervised meta-learning for reinforcement learning." In: *arXiv preprint arXiv:1806.04640* (2018) (cit. on pp. XVIII, 2, 31, 44, 57).

[32] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca D Dragan. "Inverse reward design." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6768–6777 (cit. on p. 21).

[33] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742 (cit. on p. 26).

[34] Liu Hao and Abbeel Pieter. "Behavior From the Void: Unsupervised Active Pre-Training." In: *arXiv preprint arXiv:2103.04551* (2021) (cit. on pp. 26, 39, 40).

[35] Harry F Harlow. "The formation of learning sets." In: *Psychological review* 56.1 (1949), p. 51 (cit. on p. 30).

[36] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. "Provably efficient maximum entropy exploration." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2681–2691 (cit. on pp. XVII, 2, 24, 33–35).

[37] Takuya Hiraoka, Takahisa Imagawa, Tatsuya Mori, Takashi Onishi, and Yoshimasa Tsuruoka. "Learning robust options by conditional value at risk optimization." In: *arXiv preprint arXiv:1905.09191* (2019) (cit. on p. 29).

[38] L Jeff Hong and Guangwu Liu. "Simulating sensitivities of conditional value at risk." In: *Management Science* 55.2 (2009), pp. 281–293 (cit. on p. 28).

[39] Garud Iyengar and Alfred Ka Chun Ma. "Fast gradient descent method for mean-CVaR optimization." In: *Annals of Operations Research* 205.1 (2013), pp. 203–212 (cit. on p. 28).

[40] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. "Reward-free exploration for reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4870–4879 (cit. on pp. 21, 33, 61).

[41] Leonid Vasilevich Kantorovich and SG Rubinshtein. "On a space of totally additive functions." In: *Vestnik of the St. Petersburg University: Mathematics* 13.7 (1958), pp. 52–59 (cit. on p. 8).

[42] Ravi Kumar Kolla, LA Prashanth, Sanjay P Bhat, and Krishna Jagannathan. "Concentration bounds for empirical conditional value-at-risk: The unbounded case." In: *Operations Research Letters* 47.1 (2019), pp. 16–20 (cit. on pp. 28, 40, 41).

[43] S. Kullback and R. A. Leibler. "On Information and Sufficiency." In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79 –86 (cit. on p. 6).

[44] Prashanth L.A., Krishna Jagannathan, and Ravi Kolla. "Concentration bounds for CVaR estimation: The cases of light-tailed and heavy-tailed distributions." In: *Proceedings of the International Conference on Machine Learning*. 2020 (cit. on p. 41).

[45] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. "Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning." In: *International Conference on Learning Representations*. 2019 (cit. on p. 26).

[46] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. "Efficient exploration via state marginal matching." In: *arXiv preprint arXiv:1906.05274* (2019) (cit. on pp. XVII, 2, 24, 35).

[47] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents." In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 523–562 (cit. on p. 78).

[48] Alberto Maria Metelli, Mirco Mutti, and Marcello Restelli. "Configurable Markov decision processes." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3491–3500 (cit. on pp. 61, 73).

[49] Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. "Policy Optimization via Importance Sampling." In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 5442–5454 (cit. on p. 26).

[50] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." In: *nature* 518.7540 (2015), pp. 529–533 (cit. on pp. 16, 21).

[51] Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. "A Policy Gradient Method for Task-Agnostic Exploration." In: *arXiv preprint arXiv:2007.04640* (2020) (cit. on pp. XVII, 2, 7, 25, 26, 33, 35, 38–40).

[52] Mirco Mutti and Marcello Restelli. "An Intrinsically-Motivated Approach for Learning Highly Exploring and Fast Mixing Policies." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5232–5239 (cit. on p. 24).

[53] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. "Count-based exploration with neural density models." In: *International conference on machine learning*. PMLR. 2017, pp. 2721–2730 (cit. on p. 22).

[54] Simone Parisi, Matteo Pirotta, and Marcello Restelli. "Multi-objective reinforcement learning through continuous pareto manifold approximation." In: *Journal of Artificial Intelligence Research* 57 (2016), pp. 187–227 (cit. on pp. 19, 61).

[55] Emanuel Parzen. "On estimation of a probability density function and mode." In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076 (cit. on p. 24).

[56] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. "Curiosity-driven exploration by self-supervised prediction." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2778–2787 (cit. on pp. XVII, 2, 22).

[57] Jan Peters and Stefan Schaal. "Policy gradient methods for robotics." In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 2219–2225 (cit. on p. 18).

[58] Georg Ch Pflug. "Some remarks on the value-at-risk and the conditional value-at-risk." In: *Probabilistic constrained optimization*. Springer, 2000, pp. 272–281 (cit. on p. 9).

[59] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. "Policy gradient in lipschitz markov decision processes." In: *Machine Learning* 100.2 (2015), pp. 255–283 (cit. on pp. 71, 73).

[60] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014 (cit. on pp. 13, 15).

[61] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles." In: *Proceedings of the International Conference on Learning Representations*. 2016 (cit. on pp. XVII, 2, 29).

[62] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk." In: *Journal of risk* 2 (2000), pp. 21–42 (cit. on pp. XVII, 2, 8, 27, 29, 35, 76).

[63] Jay K Satia and Roy E Lave Jr. "Markovian decision processes with uncertain transition probabilities." In: *Operations Research* 21.3 (1973), pp. 728–740 (cit. on p. 61).

[64] Jürgen Schmidhuber. "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)." In: *IEEE Transactions on Autonomous Mental Development* 2.3 (2010), pp. 230–247 (cit. on p. 22).

[65] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization." In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897 (cit. on pp. 26, 50).

[66] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347* (2017) (cit. on p. 16).

[67] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. "State Entropy Maximization with Random Encoders for Efficient Exploration." In: *arXiv preprint arXiv:2102.09430* (2021) (cit. on pp. XVII, 2, 26, 39, 40, 55).

[68] Claude E Shannon. "A mathematical theory of communication." In: *The Bell system technical journal* 27.3 (1948), pp. 379–423 (cit. on p. 5).

[69] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." In: *Science* 362.6419 (2018), pp. 1140–1144 (cit. on pp. XVI, 1, 16, 21).

[70] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. "Nearest neighbor estimates of entropy." In: *American journal of mathematical and management sciences* 23.3-4 (2003), pp. 301–321 (cit. on pp. 7, 39).

[71] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. "Intrinsically motivated reinforcement learning: An evolutionary perspective." In: *IEEE Transactions on Autonomous Mental Development* 2.2 (2010), pp. 70–82 (cit. on pp. XVII, 2).

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on pp. XVI, 1, 10, 18, 33).

[73] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211 (cit. on p. 29).

[74] Aviv Tamar, Yonatan Glassner, and Shie Mannor. "Optimizing the CVaR via sampling." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015 (cit. on pp. XVII, 2, 28, 29, 38, 41, 76).

[75] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. "# exploration: A study of count-based exploration for deep reinforcement learning." In: *31st Conference on Neural Information Processing Systems (NIPS)*. Vol. 30. 2017, pp. 1–18 (cit. on p. 22).

[76] Jean Tarbouriech and Alessandro Lazaric. "Active exploration in markov decision processes." In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 974–982 (cit. on p. 24).

[77] Jean Tarbouriech, Matteo Pirotta, Michal Valko, DeepMind Paris, and Alessandro Lazaric. "Improved Sample Complexity for Incremental Autonomous Exploration in MDPs." In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 33).

[78] Jean Tarbouriech, Shubhanshu Shekhar, Matteo Pirotta, Mohammad Ghavamzadeh, and Alessandro Lazaric. "Active model estimation in markov decision processes." In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 1019–1028 (cit. on p. 33).

[79] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033 (cit. on p. 78).

[80] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454 (cit. on p. 26).

[81] Cédric Villani. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media, 2008 (cit. on p. 8).

[82] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. "Learning to reinforcement learn." In: *arXiv preprint arXiv:1611.05763* (2016) (cit. on pp. XVII, 2, 30).

[83] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8.3-4 (1992), pp. 279–292 (cit. on p. 16).

[84] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256 (cit. on pp. 17, 18).

[85] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning." In: *Conference on Robot Learning*. PMLR. 2020, pp. 1094–1100 (cit. on p. 78).

[86] Chuheng Zhang, Yuanying Cai, Longbo Huang, and Jian Li. "Exploration by Maximizing R\'enyi Entropy for Zero-Shot Meta RL." In: *arXiv preprint arXiv:2006.06193* (2020) (cit. on pp. XVII, 2, 25).

[87] Xuezhou Zhang, Yuzhe Ma, and Adish Singla. "Task-agnostic Exploration in Reinforcement Learning." In: *Advances in Neural Information Processing Systems* 33 (2020) (cit. on p. 33).

A

THEOREM PROOFS

Theorem 4.3.1. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs satisfying Assumption 1. Let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by the policy π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}$ of the class as*

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

Proof. The proof follows techniques from [59]. Let us report a preliminary result which states that the function $h_f(\bar{s}) = \int_{\mathcal{A}} \pi(\bar{a}|\bar{s}) \int_{\mathcal{S}} P(s|\bar{s}, \bar{a})\, \mathrm{d}s\, \mathrm{d}\bar{a}$ has a Lipschitz constant equal to L_{P^π} [59, Lemma 3]:

$$\begin{aligned}\left|h_f(\bar{s}') - h_f(\bar{s})\right| &= \left| \int_{\mathcal{S}} f(s) \int_{\mathcal{A}} \pi(a|\bar{s}') P(s|\bar{s}', a)\, \mathrm{d}a\, \mathrm{d}s \right. \\ &\qquad \left. - \int_{\mathcal{S}} f(s) \int_{\mathcal{A}} \pi(a|\bar{s}) P(s|\bar{s}, a)\, \mathrm{d}a\, \mathrm{d}s \right| \\ &= \left| \int_{\mathcal{S}} f(s) \left(P^\pi(s|\bar{s}') - P^\pi(s|\bar{s}) \right) \mathrm{d}s \right| \leq L_{P^\pi} d_{\mathcal{S}}(\bar{s}', \bar{s}),\end{aligned} \tag{A.1}$$

where $d_{\mathcal{S}}$ is a metric over $\mathcal{S}$ and $P^\pi(s|\bar{s}) = \int_{\mathcal{A}} \pi(\bar{a}|\bar{s}) P(s|\bar{s}, \bar{a})\, \mathrm{d}\bar{a}$. Then, we note that the marginal state distribution over T steps $d_\pi^{\mathcal{M}}$ can be written as a sum of the contributions $d_{\pi,t}^{\mathcal{M}}$ related to any time step $t \in [T]$, which is

$$d_\pi^{\mathcal{M}}(s) = \frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}}(s). \tag{A.2}$$

Hence, we can look at the Wasserstein distance of the state distributions for some $t \in [T]$ and $\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}$. We obtain

$$d_{W_1}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) = \sup_f \left\{ \left| \int_{\mathcal{S}} \left(d^{\mathcal{M}'}_{\pi,t}(s) - d^{\mathcal{M}}_{\pi,t}(s) \right) f(s)\, \mathrm{d}s \right| : \|f\|_L \leq 1 \right\} \tag{A.3}$$

$$\begin{aligned} &= \sup_f \left\{ \left| \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \Big(d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) \pi(\bar{a}|\bar{s}) P'(s|\bar{s},\bar{a}) \right. \right. \\ &\quad \left. \left. - d^{\mathcal{M}}_{\pi,t-1}(\bar{s}) \pi(\bar{a}|\bar{s}) P(s|\bar{s},\bar{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right| : \|f\|_L \leq 1 \right\} \\ &= \sup_f \left\{ \left| \int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\bar{a}|\bar{s}) \Big(P'(s|\bar{s},\bar{a}) - P(s|\bar{s},\bar{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right. \right. \end{aligned} \tag{A.4}$$

$$\left. \left. + \int_{\mathcal{S}} \left(d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) - d^{\mathcal{M}}_{\pi,t-1}(\bar{s}) \right) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\bar{a}|\bar{s}) P(s|\bar{s},\bar{a}) f(s)\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right| : \|f\|_L \leq 1 \right\} \tag{A.5}$$

$$\begin{aligned} &\leq \sup_f \left\{ \left| \int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\bar{a}|\bar{s}) \Big(P'(s|\bar{s},\bar{a}) - P(s|\bar{s},\bar{a}) \Big) f(s)\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right| : \|f\|_L \leq 1 \right\} \\ &\quad + \sup_f \left\{ \left| \int_{\mathcal{S}} \left(d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) - d^{\mathcal{M}}_{\pi,t-1}(\bar{s}) \right) \int_{\mathcal{A}} \int_{\mathcal{S}} \pi(\bar{a}|\bar{s}) P(s|\bar{s},\bar{a}) f(s)\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right| : \|f\|_L \leq 1 \right\} \\ &\leq \sup_f \left\{ \int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) \int_{\mathcal{A}} \pi(\bar{a}|\bar{s})\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s} \right. \\ &\quad \sup_{\bar{s}\in\mathcal{S},\bar{a}\in\mathcal{A}} \left\{ \left| \int_{\mathcal{S}} \Big(P'(s|\bar{s},\bar{a}) - P(s|\bar{s},\bar{a}) \Big) f(s)\, \mathrm{d}s \right| \right\} : \|f\|_L \leq 1 \Big\} \\ &\quad + L_{P^\pi} \sup_f \left\{ \left| \int_{\mathcal{S}} \left(d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) - d^{\mathcal{M}}_{\pi,t-1}(\bar{s}) \right) \frac{h_f(\bar{s})}{L_{P^\pi}}\, \mathrm{d}\bar{s} \right| : \|f\|_L \leq 1 \right\} \end{aligned} \tag{A.6}$$

$$= \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L_{P^\pi} d_{W_1}(d^{\mathcal{M}'}_{\pi,t-1}, d^{\mathcal{M}}_{\pi,t-1}), \tag{A.7}$$

where we have plugged the common temporal relation $d^{\mathcal{M}}_{\pi,t}(s') = \int_{\mathcal{S}} \int_{\mathcal{A}} d^{\mathcal{M}}_{\pi,t-1}(s)\pi(a|s)P(s'|s,a)\, \mathrm{d}s\, \mathrm{d}a$ into (A.3), we sum and subtract $\int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\bar{s})\pi(\bar{a}|\bar{s})P(s|\bar{s},\bar{a})\, \mathrm{d}s\, \mathrm{d}\bar{a}\, \mathrm{d}\bar{s}$ to get (A.4), (A.5), and we apply the inequality in (A.1) to obtain (A.6) and then (A.7). To get rid of the dependence to the state distributions $d^{\mathcal{M}'}_{\pi,t-1}$ and $d^{\mathcal{M}}_{\pi,t-1}$, we repeatedly unroll (A.7) to get

$$d_{W_1}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) \leq \left(\sum_{j=0}^{t} L^j_{P^\pi} \right) \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^t_{P^\pi} d_{W_1}(D', D) \tag{A.8}$$

$$= \left(\frac{1 - L^t_{P^\pi}}{1 - L_{P^\pi}} \right) \sup_{s\in\mathcal{S},a\in\mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L^t_{P^\pi} d_{W_1}(D', D), \tag{A.9}$$

where we note that $d_{W_1}(d^{\mathcal{M}'}_{\pi,0}, d^{\mathcal{M}}_{\pi,0}) = d_{W_1}(D', D)$ to derive (A.8), and we assume $L_{P^\pi} < 1$ (Assumption 1) to get (A.9) from (A.8). As a side

note, when the state and action spaces are discrete, a natural choice of a metric is $d_{\mathcal{S}}(s', s) = \mathbb{1}(s' \neq s)$ and $d_{\mathcal{A}} = \mathbb{1}(a' \neq a)$, which results in the Wasserstein distance being equivalent to the total variation, the constant $L_{P^\pi} = 1$, and $\sum_{j=0}^{t} L_{P^\pi}^j = t$. More details over the Lipschitz constant L_{P^π} can be found in [59]. Finally, we can exploit the result in (A.9) to write

$$d_{W_1}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) = \sup_f \left\{ \left| \int_{\mathcal{S}} \left(\frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}'}(s) - \frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}}(s) \right) f(s) \, \mathrm{d}s \right| : \|f\|_L \leq 1 \right\} \tag{A.10}$$

$$\begin{aligned} &\leq \frac{1}{T} \sum_{t=0}^{T-1} \sup_f \left\{ \left| \int_{\mathcal{S}} \left(d_{\pi,t}^{\mathcal{M}'}(s) - d_{\pi,t}^{\mathcal{M}}(s) \right) f(s) \, \mathrm{d}s \right| : \|f\|_L \leq 1 \right\} \\ &\leq \frac{1}{T} \sum_{t=0}^{T-1} \frac{1 - L_{P^\pi}^t}{1 - L_{P^\pi}} \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L_{P^\pi}^t d_{W_1}(D', D) \\ &\leq \frac{1 - L_{P^\pi}^T}{1 - L_{P^\pi}} \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_{W_1}(P'(\cdot|s,a), P(\cdot|s,a)) + L_{P^\pi}^T d_{W_1}(D', D), \end{aligned} \tag{A.11}$$

in which we use (A.2) to get (A.10). The final result follows from (A.11) by assuming the initial state distribution D to be shared across all the CMPs in $\boldsymbol{\mathcal{M}}$, and taking the supremum over $P', P \in \boldsymbol{\mathcal{M}}$. ■

Theorem 4.3.2. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy, and let $d_\pi^{\mathcal{M}}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. We can upper bound the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ of the class as*

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

Proof. The proof follows techniques from [48], especially Proposition 3.1. Without loss of generality, we take $\mathcal{M}', \mathcal{M} \in \boldsymbol{\mathcal{M}}$. With some overloading of notation, we will alternatively identify a CMP with the tuple $\mathcal{M}$ or its transition model P. Let us start considering the TV between the marginal state distributions induced by π over $\mathcal{M}', \mathcal{M}$, we can write

$$\begin{aligned} d_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) &= \frac{1}{2} \int_{\mathcal{S}} \left| d_\pi^{\mathcal{M}'}(s) - d_\pi^{\mathcal{M}}(s) \right| \mathrm{d}s \\ &= \frac{1}{2} \int_{\mathcal{S}} \left| \frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}'}(s) - \frac{1}{T} \sum_{t=0}^{T-1} d_{\pi,t}^{\mathcal{M}}(s) \right| \mathrm{d}s \qquad \text{(A.12)} \\ &\leq \frac{1}{2T} \sum_{t=0}^{T-1} \int_{\mathcal{S}} \left| d_{\pi,t}^{\mathcal{M}'}(s) - d_{\pi,t}^{\mathcal{M}}(s) \right| \mathrm{d}s \\ &= \frac{1}{T} \sum_{t=0}^{T-1} d_{TV}(d_{\pi,t}^{\mathcal{M}'}, d_{\pi,t}^{\mathcal{M}}), \qquad \text{(A.13)} \end{aligned}$$

where we use (A.2) to get (A.12). Then, we provide an upper bound to each term of the final sum in (A.13), i.e.,

$$d_{TV}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t}) = \frac{1}{2}\int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi,t}(s) - d^{\mathcal{M}}_{\pi,t}(s) \right| \mathrm{d}s$$

$$= \frac{1}{2}\int_{\mathcal{S}} \left| \int_{\mathcal{A}}\int_{\mathcal{S}} d^{\mathcal{M}'}_{\pi,t-1}(\bar{s})\pi(\bar{a}|\bar{s})P'(s|\bar{s},\bar{a}) - d^{\mathcal{M}}_{\pi,t-1}(\bar{s})\pi(\bar{a}|\bar{s})P(s|\bar{s},\bar{a}) \right| \mathrm{d}\bar{s}\,\mathrm{d}\bar{a}\,\mathrm{d}s \tag{A.14}$$

$$\leq \frac{1}{2}\int_{\mathcal{S}} \left| d^{\mathcal{M}'}_{\pi,t-1}(\bar{s}) - d^{\mathcal{M}}_{\pi,t-1}(\bar{s}) \right| \int_{\mathcal{A}}\int_{\mathcal{S}} \pi(\bar{a}|\bar{s})P'(s|\bar{s},\bar{a})\,\mathrm{d}\bar{s}\,\mathrm{d}\bar{a}\,\mathrm{d}s \tag{A.15}$$

$$+ \frac{1}{2}\int_{\mathcal{S}}\int_{\mathcal{A}} d^{\mathcal{M}}_{\pi,t-1}(\bar{s})\pi(\bar{a}|\bar{s}) \int_{\mathcal{S}} \left| P'(s|\bar{s},\bar{a}) - P(s|\bar{s},\bar{a}) \right| \mathrm{d}\bar{s}\,\mathrm{d}\bar{a}\,\mathrm{d}s \tag{A.16}$$

$$= d_{TV}(d^{\mathcal{M}'}_{\pi,t-1}, d^{\mathcal{M}}_{\pi,t-1}) + \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,t-1} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] \tag{A.17}$$

$$= \sum_{j=1}^{t-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,j} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D), \tag{A.18}$$

where we use the temporal relation $d^{\mathcal{M}}_{\pi,t}(s') = \int_{\mathcal{S}}\int_{\mathcal{A}} d^{\mathcal{M}}_{\pi,t-1}(s)\pi(a|s)P(s'|s,a)\,\mathrm{d}s\,\mathrm{d}a$ to get (A.14), in which we sum and subtract $\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d^{\mathcal{M}}_{\pi,t-1}(\bar{s})\pi(\bar{a}|\bar{s})P(s|\bar{s},\bar{a})\,\mathrm{d}s\,\mathrm{d}\bar{a}\,\mathrm{d}\bar{s}$ to obtain (A.15) and (A.16), and we repeatedly unroll (A.17) to write (A.18), noting that $d_{TV}(d^{\mathcal{M}'}_{\pi,0}, d^{\mathcal{M}}_{\pi,0}) = d_{TV}(D', D)$. Finally, we can plug (A.18) in (A.13) to get

$$d_{TV}(d^{\mathcal{M}'}_{\pi}, d^{\mathcal{M}}_{\pi}) \leq \frac{1}{T}\sum_{t=0}^{T-1} d_{TV}(d^{\mathcal{M}'}_{\pi,t}, d^{\mathcal{M}}_{\pi,t})$$

$$\leq \frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{t-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi,j} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D)$$

$$\leq \sum_{t=0}^{T-1}\int_{\mathcal{S}} \frac{1}{T}\sum_{j=0}^{T-1} d^{\mathcal{M}}_{\pi,j}(s) \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] \mathrm{d}s + d_{TV}(D', D) \tag{A.19}$$

$$= \sum_{t=0}^{T-1} \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D) \tag{A.20}$$

$$= T \mathop{\mathbb{E}}_{\substack{s\sim d^{\mathcal{M}}_{\pi} \\ a\sim\pi(\cdot|s)}} \Big[d_{TV}(P'(\cdot|s,a), P(\cdot|s,a)) \Big] + d_{TV}(D', D), \tag{A.21}$$

in which we have used (A.2) to obtain (A.20) from (A.19). The final result is straightforward from (A.20) by assuming the initial state distribution D to be shared across all the CMPs in $\boldsymbol{\mathcal{M}}$, and taking the supremum over $P', P \in \boldsymbol{\mathcal{M}}$. ■

Theorem 4.3.3. *Let $\boldsymbol{\mathcal{M}}$ be a class of CMPs, let $\pi \in \Pi$ be a policy and $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ the corresponding π-diameter of $\boldsymbol{\mathcal{M}}$. Let $d^{\mathcal{M}}_{\pi}$ be the marginal state distribution over T steps induced by π in $\mathcal{M} \in \boldsymbol{\mathcal{M}}$, and let $\sigma_{\boldsymbol{\mathcal{M}}} \leq \sigma_{\mathcal{M}} :=$*

$\inf_{s\in\mathcal{S}} d_\pi^{\mathcal{M}}(s), \forall \mathcal{M} \in \boldsymbol{\mathcal{M}}$. *We can upper bound the entropy gap of the policy* π *within the model class* $\boldsymbol{\mathcal{M}}$ *as*

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

Proof. Let us expand the entropy gap of the policy π as

$$\begin{aligned}
&\sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left|H(d_\pi^{\mathcal{M}'}) - H(d_\pi^{\mathcal{M}})\right| \\
&= \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{\left| -\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\log d_\pi^{\mathcal{M}'}(s)\,\mathrm{d}s + \int_{\mathcal{S}} d_\pi^{\mathcal{M}}(s)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s\right|\right\} && \text{(A.22)}\\
&\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{\left| \int_{\mathcal{S}} \left(d_\pi^{\mathcal{M}}(s) - d_\pi^{\mathcal{M}'}(s)\right)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s\right|\right. \\
&\quad \left. + \left|\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\left(\log d_\pi^{\mathcal{M}'}(s) - \log d_\pi^{\mathcal{M}}(s)\right)\mathrm{d}s\right|\right\} && \text{(A.23)}\\
&\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{ -\log\sigma_{\mathcal{M}} \int_{\mathcal{S}} \left|d_\pi^{\mathcal{M}'}(s) - d_\pi^{\mathcal{M}'}(s)\right|\mathrm{d}s + D_{KL}(d_\pi^{\mathcal{M}'}||d_\pi^{\mathcal{M}})\right\} && \text{(A.24)}\\
&\leq \sup_{\mathcal{M}',\mathcal{M}\in\boldsymbol{\mathcal{M}}} \left\{ -\log\sigma_{\mathcal{M}} D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}}) + \left(D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}})\right)^2/\sigma_{\mathcal{M}}\right\} && \text{(A.25)}\\
&\leq \left(\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\right)^2/\sigma_{\boldsymbol{\mathcal{M}}} - \mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)\log\sigma_{\boldsymbol{\mathcal{M}}} && \text{(A.26)}
\end{aligned}$$

in which we sum and subtract $\int_{\mathcal{S}} d_\pi^{\mathcal{M}'}(s)\log d_\pi^{\mathcal{M}}(s)\,\mathrm{d}s$ to obtain (A.23) from (A.22), $\log d_\pi^{\mathcal{M}}(s)$ is upper bounded with $\log\sigma_{\mathcal{M}}$ to get (A.24), and we use the reverse Pinsker's inequality $D_{KL}(p||q) \leq (D_{TV}(p,q))^2/\inf_{x\in\mathcal{X}} q(x)$ [19, p. 1012 and Lemma 6.3] to obtain (A.31). Finally, we get the result by upper bounding $D_{TV}(d_\pi^{\mathcal{M}'}, d_\pi^{\mathcal{M}})$ with the π-diameter $\mathcal{D}_{\boldsymbol{\mathcal{M}}}(\pi)$ and $\sigma_{\mathcal{M}}$ with $\sigma_{\boldsymbol{\mathcal{M}}}$ in (A.25). ■

Proposition 4.4.1. *Let* $\pi_{\boldsymbol{\theta}} \in \Pi_\Theta$. *The policy gradient of the exploration objective* $\mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi_{\boldsymbol{\theta}})$ *w.r.t.* $\boldsymbol{\theta}$ *is given by*

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

Proof. Let us start from expanding the exploration objective (4.3) to write

$$\begin{aligned}
\mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi) &= CVaR_\alpha(H_\tau) \\
&= \mathop{\mathbb{E}}_{\substack{\mathcal{M}\sim p_{\boldsymbol{\mathcal{M}}}\\ \tau\sim p_{\pi,\mathcal{M}}}} \left[H_\tau \mid H_\tau \leq VaR_\alpha(H_\tau)\right] \\
&= \frac{1}{\alpha}\int_{-\infty}^{VaR_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)h\,\mathrm{d}h, && \text{(A.27)}
\end{aligned}$$

where $p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}$ is the probability density function (pdf) of the random variable H_τ when the policy $\pi_{\boldsymbol{\theta}}$ is deployed on the class of environments $\boldsymbol{\mathcal{M}}$, and the last equality comes from the definition of CVaR [62]. Before computing the gradient of (A.27), we derive a preliminary result for later use, i.e.,

$$\nabla_{\boldsymbol{\theta}} \int_{-\infty}^{VaR_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)\,\mathrm{d}h = \int_{-\infty}^{VaR_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)\,\mathrm{d}h + \nabla_{\boldsymbol{\theta}} VaR_\alpha(H_\tau) p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(VaR_\alpha(H_\tau)) = 0, \tag{A.28}$$

which follows directly from the Leibniz integral rule, noting that $VaR_\alpha(H_\tau)$ depends on $\boldsymbol{\theta}$ through the pdf of H_τ. We now take the gradient of (A.27) to get

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi) &= \nabla_{\boldsymbol{\theta}} \frac{1}{\alpha} \int_{-\infty}^{VaR_\alpha(H_\tau)} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) h\,\mathrm{d}h \\ &= \frac{1}{\alpha} \int_{-\infty}^{VaR_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) h\,\mathrm{d}h \\ &\quad + \frac{1}{\alpha} \nabla_{\boldsymbol{\theta}} VaR_\alpha(H_\tau) VaR_\alpha(H_\tau) p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(VaR_\alpha(H_\tau)) \qquad \text{(A.29)} \\ &= \frac{1}{\alpha} \int_{-\infty}^{VaR_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) \Big(h - VaR_\alpha(H_\tau) \Big)\,\mathrm{d}h, \qquad \text{(A.30)}\end{aligned}$$

where (A.29) follows from the Leibniz integral rule, and (A.30) is obtained from (A.29) through (A.28), which we can rearrange to write $p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(VaR_\alpha(H_\tau)) = \frac{1}{\nabla_{\boldsymbol{\theta}} VaR_\alpha(H_\tau)} \int_{-\infty}^{VaR_\alpha(H_\tau)} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)\,\mathrm{d}h$. All of the steps above are straightforward replications of the derivations by Tamar, Glassner, and Mannor [74], Proposition 1. To conclude the proof we just have to compute the term $\nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h)$, which is specific to our setting. Especially, we note that

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\boldsymbol{\mathcal{M}}}(h) &= \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \delta(h - H_\tau)\,\mathrm{d}\tau\,\mathrm{d}\mathcal{M} \qquad \text{(A.31)} \\ &= \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \nabla_{\boldsymbol{\theta}} \log p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \delta(h - H_\tau)\,\mathrm{d}\tau\,\mathrm{d}\mathcal{M} \\ &= \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau}|s_{t,\tau}) \Big) \delta(h - H_\tau)\,\mathrm{d}\tau\,\mathrm{d}\mathcal{M}, \qquad \text{(A.32)}\end{aligned}$$

where (A.31) and (A.32) are straightforward from the definitions in Section 2.2, and $\mathcal{T}$ is the set of feasible trajectories of length T. Finally, the result follows by plugging (A.32) into (A.30), which gives

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{E}^\alpha_{\boldsymbol{\mathcal{M}}}(\pi) = \frac{1}{\alpha} \int_{\boldsymbol{\mathcal{M}}} p_{\boldsymbol{\mathcal{M}}}(\mathcal{M}) \int_{\mathcal{T}} p_{\pi_{\boldsymbol{\theta}},\mathcal{M}}(\tau) \int_{-\infty}^{VaR_\alpha(H_\tau)} \delta(h - H_\tau) \\ \Big(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{t,\tau}|s_{t,\tau}) \Big) \Big(h - VaR_\alpha(H_\tau) \Big)\,\mathrm{d}h\,\mathrm{d}\tau\,\mathrm{d}\mathcal{M}.\end{aligned}$$

■

B

ADDITIONAL EXPERIMENTS

In this Appendix, we discuss some additional experiments, which we decided to not include in Chapter 5 because they have a different setting, which is not central in this thesis. However, as stated in Section 4.2 and in the conclusions of Chapter 6, they can represent a good starting point for a future work.

We carried out these experiments using the Meta-World benchmark [85]. It consists of 50 robotic manipulation tasks implemented in the MuJoCo physics engine [79], and it is thought to evaluate state of the art multi-task and meta-learning algorithms. Multi-task learning and meta-learning have essentially the same goal: expand the set of the learned skills. However, multi-task RL tries to do that by learning a fixed set of skills with minimal data, while meta RL (see Section 3.3) relies on past experience to quickly adapt to new tasks. Yu et al. [85] introduce this benchmark to overcome some limitations of previous suites, such as the Arcade Learning Environment [47], which is too heterogeneous in terms of visual appearance, controls and objectives to allow for a substantial efficiency gain thanks to positive transfer between the different tasks. Instead, Meta-World proposes tasks that share the same environment and control structure, but that are at the same time different from one another. In particular, all the tasks comprise a simulated Sawyer robotic arm, whose action space $\mathcal{A}$ includes the 3D coordinates of the end-effector position and the possibility to grab an object by opening and closing it. The observation space $\mathcal{S}$ includes the 3D Cartesian positions of the end-effector and of one or two objects, while the 3D position of the goal is hidden and used to compute a dense reward function (which we do not use). In Figure B.1, we report an illustration of one of the 50 tasks.

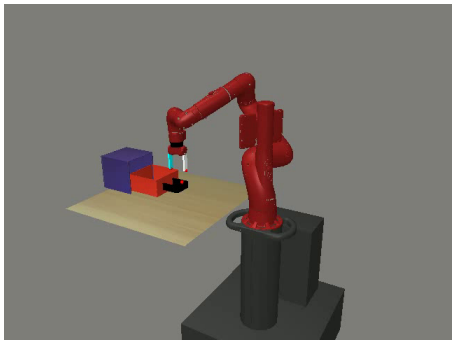


Figure B.1: An illustration of one of the 50 tasks contained in the Meta-World benchmark [85].

B.1 DRAWER-CLOSE ENVIRONMENT

The first experiment that we want to describe is exactly the one described in Figure B.1. It is called *drawer-close-v1* and it consists of the robotic arm and a drawer, which is initially open. The goal, as planned by the Meta-World suite, is to close the drawer, going from a point (a) to a point (b). Our setting is as usual a reward-free domain, in which we want to explore as evenly as possible. We proceed by maximizing the entropy by means of the estimator in (4.7). As in the Ant experiment (see Section 5.4), we perform the maximization only on a subset of the features. Interestingly, the difference here is that we are dealing with multiple entities, i.e., we have features of both the robotic arm and the drawer. We initially maximized the entropy over the 3D position of the end-effector, with the intent of exploring the whole environment and eventually interacting with the drawer. However, exploring a three-dimensional space as large as the one here considered requires an extremely high number of samples. Note indeed that the actions are scaled by a factor of $\frac{1}{100}$, making the exploration very slow. This is an obvious design choice, since the movement of the robotic arm must be smooth. Even if we assume to be able to fully explore the environment, the interaction with the object can be very poor or even absent, since the arm has no incentive to do it. We actually noticed that also maximizing the entropy over both the arm's and the object's features is not enough to guarantee an exhaustive interaction. That is why we ended up with this solution: we narrowed down the portion of navigable environment and we maximized the entropy over the 3D position of the end-effector. Once obtained a policy capable of exploring the narrowed region in a uniform way, we used it as initialization to launch another experiment, this time maximizing the entropy only over the y coordinate of the drawer. Clearly, these choices inject some domain knowledge in the solution, which is something that is not always possible. This recall the point made in Section 4.2, namely the viable option to learn an approximation of the Pareto frontier instead of manually choose the features of interest.

For the first exploration phase, we trained the algorithm for 40 epochs, sampling 10 trajectories with a time horizon $T = 10000$, a mini-batch of size $B = 2$ and $k = 50$. In Figure B.2, we report the heatmap of the state visitation induced by the obtained policy, where we can distinguish the parallelepiped that represents the narrowed region. Then, as we said, we launched again the algorithm, by using the previously obtained policy as initialization and maximizing the entropy over the y coordinate of the drawer. Note that in this case we are maximizing the entropy over the other two dimensions as well (x, z), but they would not impact the process, because the drawer has no lateral or vertical movements. We set the parameters to $N_{traj} = 50$, $T = 1000$, $B = 5$, and $k = 1000$. The reason behind such an high value of k is that by limiting the entropy estimation to only the feature y, which covers a very small range, we incur in the *aliasing* problem.

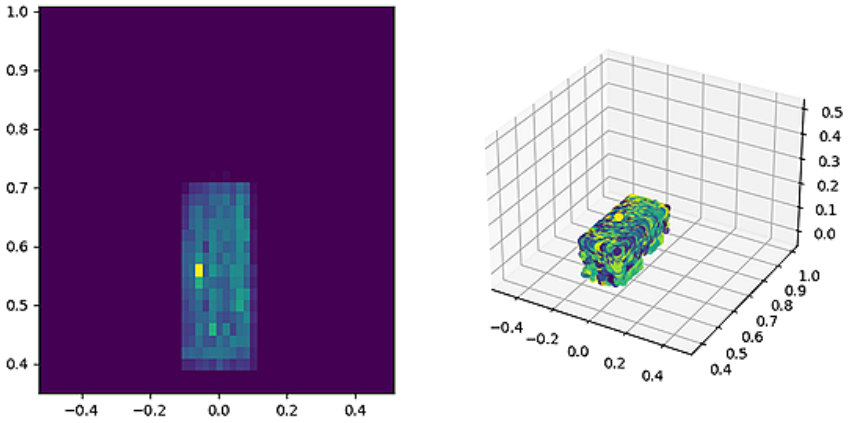


Figure B.2: Heatmap of the state visitation induced by the exploration policy in the *drawer-close-v1* domain when optimizing on the 3D position of the robotic arm.

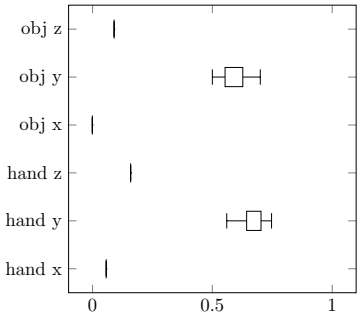


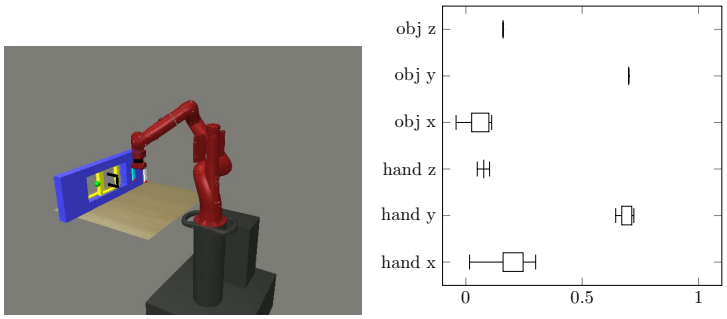
Figure B.3: Box plot showing the range of the features in the *drawer-close-v1* domain when optimizing on the y coordinate of the drawer.

The aliasing problem occurs when too many samples are in the same position, leading the k-NN computation to output an infinite value. At this point, the resulting behavior was quite satisfying. However, the Sawyer arm learned to move the drawer in a way that was not natural, e.g., by applying some pressure on the border and then moving back and forth. The issue was mainly due to the simulated physics of the environment. We thus increased the mass of the drawer and its friction with the table, obtaining a perfect behavior of the agent: it was able to grab the drawer's handle and go back and forth, as one can guess from the box plot in Figure B.3. We can clearly notice that the only two features with a significant range of motion is the y coordinate of both the end-effector (*hand* in the plot) and the drawer (*obj* in the plot).

B.2 WINDOW-OPEN ENVIRONMENT

The second experiment uses the environment called *window-open-v1*. As the name suggests, in this case we have a window that is initially closed (see Figure B.4a). Following the same intuitions as in the drawer-close domain, we reduced the navigable space and we initially trained a policy to explore it. Then, we used the policy as initialization to

launch another experiment, where we maximized the entropy over the x coordinate of the window. Also in this case, we had to make some tweaks in order to obtain a natural interaction. We increased the size of the handle, otherwise too small to be grabbed (note that the Meta-World tasks are thought for a dense reward setting, where the reward is computed according to the distance between the end-effector and the position of the goal). Furthermore, we changed the initial position of the window, setting it at the center of its range. In this way the agent can learn to move it in both directions on the x axis, without stalling in one of the two corners. In Figure B.4b, we provide a box plot, which shows how the positions of the robotic arm and the window change during one epoch. Noticeably, the x coordinate of the end-effector and the x coordinate of the window are the ones with the largest range.



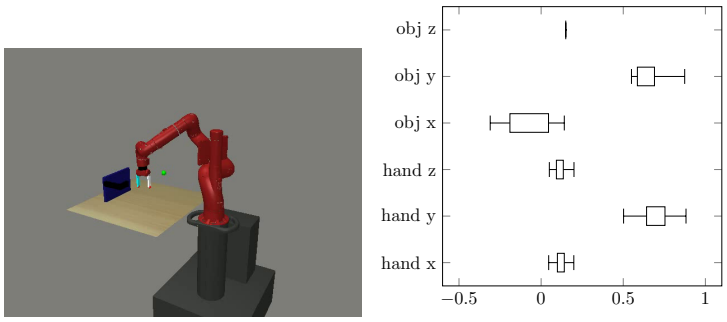
(a) The environment **(b)** Features' range

Figure B.4: Illustration of the *window-open-v1* environment **(a)**, and a box plot showing the range of the features when optimizing on the x coordinate of the window **(b)**.

B.3 DOOR-OPEN ENVIRONMENT

The last experiment uses the environment called *door-open-v1*. It consists of the usual robotic arm and a door, which is initially closed (see Figure B.5a). We followed the exact same procedure of training, maximizing (in the second phase) on the x and y coordinates of the door. We made two main changes to the default setting. First, we increased the friction with the floor (i.e., the table) as with the *drawer-close-v1* domain. Second, in the preliminary exploration, we had to adapt the narrowed region so that the agent could not get on the right of the door to avoid some brutal movements, which are probably due to the physics of the pin. Again, in Figure B.5b, we show the range of the different features. Understandably, all the features but *obj z* are subject to a movement, with the x position of the door being the most explored. Note that in this case we are maximizing over two features, which are subject to change in different moments, as the door first moves on the x axis and then on the y axis. This seems to make the interaction quite difficult. In fact, as opposed to the previous two domains, the movement induced by the final policy is not really smooth. An easy

future development might consider a combination of the two features, e.g., via polar coordinates.



(a) The environment **(b)** Features' range

Figure B.5: Illustration of the *door-open-v1* environment **(a)**, and a box plot showing the range of the features when optimizing on the x, y coordinates of the door **(b)**.

C

IMPLEMENTATION DETAILS

C.1 HYPERPARAMETER VALUES

C.1.1 *Learning to Explore*

In Table C.1, we report the parameters of MEMENTO and Neutral that are used in the *learning to explore* phase of the experiments with low-dimensional complexity, i.e., *GridWorld with Slope* and *MultiGrid*. In Table C.2, we report the parameters of MEMENTO and Neutral that are used in the *learning to explore* phase of the experiments with high-dimensional complexity, i.e., *Ant* and *MiniGrid*.

Table C.1: MEMENTO and Neutral Parameters for Low-Dimensional Domains

	GridWorld with Slope	MultiGrid
Number of epochs	150	50
Horizon (T)	400	400
Number of traj. (N)	200	500
Mini-batch dimension (B)	5	5
α-percentile	0.35	0.1
Sampling dist. ($p_{\mathcal{M}}$)	[0.8,0.2]	[0.1,0.1,...,0.1]
KL threshold (δ)	15	15
Learning rate (β)	10^{-5}	10^{-5}
Number of neighbors (k)	30	30
Policy hidden layer sizes	(300,300)	(300,300)
Policy hidden layer act. function	ReLU	ReLU
Number of seeds	4	4

C.1.2 *Reinforcement Learning*

In Table C.3, we report the TRPO parameters that are used in the RL phase of the experiments.

Table C.2: MEMENTO and Neutral Parameters for High-Dimensional Domains

	Ant	MiniGrid
Number of epochs	400	300
Horizon (T)	400	150
Number of traj. (N)	150	100
Mini-batch dimension (B)	5	5
α-percentile	0.2	0.3
Sampling dist. ($p_{\mathcal{M}}$)	[0.8,0.2]	[0.8,0.2]
KL threshold (δ)	15	15
Learning rate (β)	10^{-5}	10^{-5}
Number of neighbors (k)	500	50
Policy hidden layer sizes	(400,300)	*
Policy hidden layer act. function	ReLU	*
Number of seeds	4	4

* See Section 5.4.2 for full details on the architecture.

Table C.3: TRPO Parameters for Goal-Based RL

	GridWorld with Slope	MultiGrid	Ant	MiniGrid
Number of Iter.	100	100	100	200
Horizon	400	400	400	150
Sim. steps per Iter.	1.2×10^4	1.2×10^4	4×10^5	7.5×10^3
δ_{KL}	10^{-4}	10^{-4}	10^{-2}	10^{-4}
Discount (γ)	0.99	0.99	0.99	0.99
Number of seeds	50	50	8	13
Number of goals	50	50	8	13

C.1.3 *Meta-RL*

In Table C.4 and Table C.5, we report the MAML and UML parameters that are used in the experiments described in Section 5.5, in order to meta-train a policy on the *GridWorld with Slope* and *MultiGrid* domains. For MAML experiments, we adopted the codebase at https://github.com/tristandeleu/pytorch-maml-rl, while for DIAYN we used the original implementation.

Table C.4: MAML Parameters

	GridWorld with Slope	MultiGrid
Number of batches	200	200
Meta batch size	20	20
Fast batch size	30	30
Num. of Grad. Step	1	1
Horizon	400	400
Fast learning rate	0.1	0.1
Policy hidden layer sizes	(64,64)	(64,64)
Policy hidden layer act. function	Tanh	Tanh
Number of seeds	4	4

Table C.5: DIAYN Parameters

	GridWorld with Slope	MultiGrid
Number of epochs	1000	1000
Horizon	400	400
Number of skills	20	20
Learning rate	3×10^{-4}	3×10^{-4}
Discount (γ)	0.99	0.99
Policy hidden layer sizes	(300,300)	(300,300)
Policy hidden layer act. function	ReLU	ReLU
Number of seeds	4	4