POLITECNICO DI MILANO
SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science in Computer Science and Engineering
Department of Electronics, Information and Bioengineering

Reinforcement Learning in Configurable Environments: an Information Theoretic approach

AI & R Lab
The Artificial Intelligence and Robotics Lab
Politecnico di Milano

Supervisor: Prof. Marcello Restelli
Co-supervisor: Dott. Alberto Maria Metelli

Author:
Emanuele Ghelfi, 875550

Academic Year 2017-2018

A mio padre...

Abstract

The general goal of Reinforcement Learning (RL) is to design agents able to learn a behaviour from interactions with an environment. Most of the problems tackled by Reinforcement Learning are typically modeled as Markov Decision Processes in which the environment is considered a fixed entity and cannot be controlled. Nevertheless, there exist several real-world examples in which a partial control on the environment can be exercised by the agent itself or by an external supervisor. For instance, in a car race the driver can set up his/her vehicle to better suit his/her needs. With the phrase environment configuration we refer to the activity of altering some environmental parameters to improve the performance of the agent's policy. This scenario has been recently formalized as a Configurable Markov Decision Process (CMDP).
The aim of this thesis is to further investigate the framework of Configurable Markov Decision Processes. We propose a new information theoretic algorithm, namely Relative Entropy Model Policy Search (REMPS), able to manage CMDPs with continuous action and state spaces.
We propose a theoretical analysis of REMPS deriving the performance gap between the ideal case of the algorithm and the approximated case. Moreover, we empirically evaluate the performance of our approach in three scenarios, showing that it outperforms a naïve gradient method in several situations.

Estratto in Lingua Italiana

L'*Apprendimento per Rinforzo* (Sutton and Barto, 1998) è un campo dell'*Intelligenza Artificiale* che tratta il problema dell'apprendimento tramite interazione con l'ambiente. Questa disciplina considera *problemi di decisioni sequenziali*, modellizzati come *Processi Decisionali di Markov* (Puterman, 1994). Il decisore, indicato come *agente*, deve stabilire quale azione effettuare considerando l'incertezza dell'ambiente e la sua esperienza. Definiamo, in questo contesto, l'apprendimento come processo di adattamento dinamico del comportamento di un agente allo scopo di raggiungere un obiettivo. Il comportamento dell'agente, che descrive quali azioni prendere in certe situazioni (stati), è chiamato *politica*. Lo scopo degli algoritmi di Apprendimento per Rinforzo è quello di sviluppare agenti in grado di apprendere tramite interazione con l'ambiente, focalizzandosi sull'apprendimento di una politica che massimizzi una metrica di prestazione.

Nella maggior parte dei problemi affrontati in letteratura, l'ambiente è considerato un'entità fissa, che non può essere controllata. Nonostante questo, esistono diversi esempi reali nei quali può essere esercitato un controllo parziale dell'ambiante dall'agente stesso o da un supervisore esterno. L'ambiente può essere quindi *configurato* per massimizzare la velocità di apprendimento o la prestazione finale dell'agente. Questo scenario è stato recentemente formalizzato come *Processo Decisionale di Markov Configurabile* (Metelli et al., 2018a). Risolvere un Processo Decisionale di Markov Configurabile significa trovare la politica dell'agente e la configurazione dell'ambiente che, congiuntamente, massimizzino le prestazioni. In (Metelli et al., 2018a) gli autori presentano un algoritmo di apprendimento sicuro, *Safe Policy Model Iteration* (SPMI), per risolvere questo tipo di problema. Questo approccio è riuscito a mostrare i vantaggi della configurazione in esempi illustrativi, anche se è ben lungi dall'essere applicabile in scenari reali. Anzitutto, SPMI è applicabile solo a problemi con spazi di stati e azioni finiti, mentre molti esempi interessanti di Processi Decisionali di Markov Configurabili hanno almeno uno spazio di stati continuo. Secondariamente, questo algoritmo richiede una conoscenza esatta della dinamica dell'ambiente. Questa limitazione è rilevante, in quanto in realtà non conosciamo quasi mai la dinamica reale dell'ambiente e, anche se un modello può essere disponibile, questo può essere troppo approssimato o complesso per essere utilizzato (ad esempio il modello fluido-dinamico di una macchina).

In questa tesi vogliamo investigare più approfonditamente la relazione tra la politica e la configurazione dell'ambiente e trovare tecniche per ottimizzare questi componenti in modo congiunto. Presentiamo un algoritmo, derivato da *Relative Entropy Policy Search* (Peters et al., 2010) (REPS), chiamato *Relative Entropy Model Policy Search* (REMPS) che utilizza una formulazione primale-duale per aggiornare i parametri di politica e modello verso un ottimo locale. In aggiunta presentiamo una strategia di proiezione che tiene in considerazione l'effetto congiunto di politica e funzione di transizione. Studiamo le proprietà teoriche del nostro approccio analizzando l'errore dovuto alla differenza di prestazioni dal caso approssimato rispetto al caso ideale. La valutazione sperimentale di REMPS è effettuata su tre domini. Il primo dominio, chiamato catena, è semplice e viene utilizzato per visualizzare il comportamento del nostro algoritmo. Il secondo dominio, Cart-Pole, è un banco di prova classico per gli algoritmi di RL. Il terzo dominio, TORCS, è più complesso e mostra le prestazioni del nostro approccio in un problema di guida e configurazione automatica. Nei risultati sperimentali mostriamo che il nostro approccio è migliore rispetto a metodi a gradiente in alcune situazioni.

La tesi è strutturata come segue. Il Capitolo 1 presenta il contesto, le motivazioni e lo scopo della tesi. Il Capitolo 2 è un'introduzione all'apprendimento per rinforzo e ai Processi Decisionali di Markov, insieme ad alcune importanti estensioni. Nel Capitolo 3 presentiamo lo Stato dell'Arte dei Processi Decisionali di Markov Configurabili e Relative Entropy Policy Search, siccome il nostro approccio si fonda su queste basi. Nel Capitolo 4 sviluppiamo i contributi principale di questa tesi: l'algoritmo Relative Entropy Model Policy Search e la sua analisi teorica. Nel Capitolo 5 mostriamo la valutazione sperimentale dell'algoritmo su tre esempi illustrativi. Il Capitolo 6 contiene la conclusione, nella quale descriviamo brevemente il lavoro svolto e proponiamo possibili estensioni e direzioni di ricerca.

Ringraziamenti

Ringrazio in primo luogo la mia famiglia che mi ha sempre sostenuto sia moralmente che economicamente durante i miei studi.
In secondo luogo desidero ringraziare il prof. Marcello Restelli e il Dott. Alberto Maria Metelli che in questi mesi mi hanno dato tutto il supporto necessario per riuscire a concludere la stesura di questa tesi. A loro sono veramente grato per avermi introdotto al mondo della ricerca e in particolare al mondo del Reinforcement Learning.
Vorrei ringraziare tutti i miei compagni di università, tra questi in particolare Leonardo, Emiliano, Edoardo, Samuele, Giulio. E' anche grazie allo stimolo continuo, alla collaborazione e al confronto che si impara.
Un ringraziamento speciale va alla "Nonna" Luciana per aver allietato le mie giornate di studio con il suo buonissimo ragù.
Un ultimo ringraziamento, ma non di certo per importanza, va alla mia ragazza Giulia per avermi sopportato in questi mesi e per essermi sempre stata vicino.

Emanuele
Milano, 20 Dicembre 2018

Notation

Column vectors are denoted by bold, lowercase letter (e.g. $\boldsymbol{x}$). The notation $\boldsymbol{x}^T$ is for row vectors. Matrices are denoted by bold, uppercase letters (e.g. $\boldsymbol{M}$).

$\boldsymbol{\omega}$	model parameter vector $\boldsymbol{\omega} \in \mathbb{R}^{d'}$
$\boldsymbol{\theta}$	policy parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{d}$
$\Delta(\mathcal{X})$	set of probability distributions over space $\mathcal{X}$
$\mathbb{P}(x)$	Probability of event x
$\mathcal{A}$	action space
$\mathcal{P}$	configuration space, also referred to as model space
$\mathcal{S}$	state space
Ω	family of model parameters $\Omega \subseteq \mathbb{R}^{d'}$
Π	policy space
π	policy
$\pi_{\boldsymbol{\theta}}$	policy parametrized by $\boldsymbol{\theta}$
Θ	family of policy parameters $\Theta \subseteq \mathbb{R}^{d}$
$\underset{x \sim \mathcal{D}}{\mathbb{E}}[f(x)]$	Expected value of the function f given that samples x are distributed according to $\mathcal{D}$
a	action
a_t	action at time t
G_t	Return from time step t
P	model configuration, environment dynamics
$P_{\boldsymbol{\omega}}$	model parametrized by $\boldsymbol{\omega}$
$r(s, a)$	reward received after performing action a in state s

$R(s, a, s')$	reward received after performing action a in state s and landing in s'
s	state
s_t	state at time t
t	discrete time step t
$x \sim \mathcal{D}(\cdot)$	sample x comes from the distribution $\mathcal{D}(\cdot)$

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do.

Steve Jobs

Reinforcement Learning (Sutton and Barto, 1998) (RL) is a field of artificial intelligence (AI) and machine learning (ML) that deals with the problem of learning from interactions. In this context we define learning as the dynamic adaptation process of a behaviour in order to achieve some objective. We, as human beings continually face this problem. For example, as infants we were unable to walk. Nonetheless, we gradually understand the effect of our action through trials and errors, even without a supervisor teaching us how to do it. We start from no knowledge about how our actions influence the world, but in some months we learn how to reach our goal which, in this case, is walking.

Reinforcement Learning lies between the areas of neuroscience, artificial intelligence, optimal control, psychology, operation research and statistics. RL considers *Sequential Decision Making problems*, modeled as *Markov Decision Processes* (Puterman, 1994) (MDPs), which are problems arising in many real-world scenarios. In this settings the decision maker, referred to as the *agent*, has to take decisions accounting for the environment uncertainty and its experience. Agents are *goal-directed*, they need only a notion of goal, a numerical signal to be maximized. Unlike *supervised learning*, in RL there is no need to provide good examples, it is the agent which learns how to map situations to actions. The mapping from situation (states) to actions is called *policy* in literature and it represents the agent's behaviour. Solving an MDP means finding the agent's policy by maximizing the total reward.

The RL approach has proven to be successful in several domains, such as robotics (Kober et al., 2013), finance (Moody et al., 1998), videogames (Atari, Dota) (Mnih et al., 2013, 2015; Lillicrap et al., 2015a), board games (Alpha Go) (Silver et al., 2016, 2017).

Motivations

In most of the problems tackled by RL the environment is considered a *fixed* entity that cannot be controlled. Nevertheless, there exist several real-world examples in which a partial control on the environment can be exercised by the agent itself or by an external *supervisor*. With the phrase *environment configuration*, we refer to the activity of altering some environmental parameters to improve the performance of the agent. Consider, as before, a child learning to walk. His/her parents (supervisors) can dynamically configure the environment, trying to help their child. They can help their child during his/her first steps, they can reduce the pain while falling by using a mat, or they can help their child by setting up goals of increasing difficulty (e.g., walking further).
We can easily notice that most of the existing environments are consist of a *fixed* part and a *configurable* part. For example, in a car race the *fixed* parts are the physics laws, while the *configurable* parts are the wing angle, the type of tyres, the brake settings. This process of environment configuration is common in many other scenarios (e.g., robotics, e-learning). In these examples there is an entity entitled to configuration activity that configures some environment features (transition function, reward, task difficulty) in order to improve the learning speed or the final performance of the agent.

State of the Art

The scenarios presented above have been recently formalized as *Configurable Markov Decision Processes* (Metelli et al., 2018a) (CMDP). Solving a CMDP means to find the agent's policy π together with the environment configuration P which, jointly, maximize the total reward. In (Metelli et al., 2018a), a safe-learning algorithm, *Safe Policy Model Iteration* (SPMI), is presented to solve the learning problem in the CMDP framework. The basic idea is to optimize a lower bound on the performance improvement so that a monotonic performance improvement is guaranteed. Although this approach succeeded in showing the advantages of configuring the environment in some illustrative examples, it is quite far from being applicable to real-world scenarios. We believe that the most significant limitations of SPMI are two. First, it is only applicable to problems with a finite state-actions space, while the most interesting examples of CMDPs have, at least, continuous state space (e.g., the car configuration problem). Second, it requires full knowledge of the environment dynamics. This latter limitation is the most relevant as, in reality we almost never know the true environment dynamics, and even if a model is available it might be too approximate and hardy usable, being very complex and computationally expansive (e.g., the fluido-dynamical model of a car).

Goal

The aim of this thesis is to further investigate the close relationship between policy and environment and to find ways to optimize them jointly in order to achieve high performance. Our approach is intended to manage CMDPs with continuous action and state spaces and it does not require full knowledge of the model, an approximated (learned) model can be exploited. These features permits the application of our algorithm on real-world cases.

Contribution

The contribution of this thesis are algorithmic, theoretical and experimental. We present an algorithm derived from *Relative Entropy Policy Search* (REPS) (Peters et al., 2010), namely *Relative Entropy Model Policy Search* (REMPS), which exploits a primal-dual formulation to update the model and policy parameters toward a local optimum. Moreover, we present a projection strategy suitable for CMDP that takes into account the joint effect of the policy and the transition function.
We derive some theoretical guarantees for the single step of REMPS, obtaining a bound on the difference of performance between the exact case and the approximated case.
We show the experimental results of our algorithm on some standard RL benchmarks to highlight the importance of configuring the environment.

Thesis Outline

The structure of this thesis is organized as follows. In Chapter 2 we present the Reinforcement Learning and Markov Decision Processes frameworks, along with some notable extensions. We start from the Markov Decision Process formalization, we introduce the main components such as the transition function, the reward and the policy. Finally we describe the main approaches for solving MDPs, namely Linear Programming, Dynamic Programming and Policy Search.
In Chapter 3 we depict the State of the Art of Configurable Markov Decision Processes since our algorithm builds upon them. We focus on the motivations underlying the framework proposal and we outline some limitations of the state of the art method for solving CMDPs. The second part of the chapter is devoted to REPS and its extensions.
In Chapter 4 we present the main contributions of this thesis: the REMPS algorithm and its theoretical analysis. In the first part of the chapter we formalize the CMDP learning problem in an information theoretic fashion. Motivated by two theorems we consider three projection strategies suited for CMDPs. The main goal of the theoretical analysis is to provide a finite-sample analysis of the single step of REMPS.
In Chapter 5 we show the experimental evaluations of our algorithm on three illustrative examples.
Chapter 6 contains the conclusion, in which we briefly describe our work together with some possible extensions and research directions.
Appendix A reports the proof of the Linear Programming solution presented in Section 2.2.
Appendix B contains the extension of two main estimators for the policy gradient to the case of the model gradient. In Appendix C we show the derivation of the REMPS solution in closed form.

Chapter 2

Reinforcement Learning

All models are wrong, but some are useful.

George E. P. Box

In this chapter, we present the basics of the Reinforcement Learning framework needed for contextualize the work of the following chapters. We will first introduce the Markov Decision Process model, Section 2.1, then we present the main approaches for the *exact* solution of Markov Decision Processes, namely *Linear Programming*, Section 2.2, and *Dynamic Programming*, Section 2.3. In Section 2.4 we present *Policy Search* algorithms, that represents an approximate solution to the Markov Decision Processes problem. At the end of this chapter we will consider some notable extensions to the Markov Decision Process framework, Section 2.5.

2.1 Markov Decision Processes

A Markov Decision Process (Puterman, 2014) (MDP) is a formal framework for modelling sequential decision making problems. The decision maker is usually called *agent*. In MDPs an agent interacts with an *environment* through *actions* and receives a *reward* based on the action and on the current state of the environment. The goal of the agent is to maximize the cumulative sum, possibly discounted, of rewards in a given *horizon* (possibly infinite). The task the agent has to learn is defined through the rewards it receives.

2.1.1 Formal Model

In this work we consider finite-time MDPs in which time is divided in discrete steps. At each time step $t = 0, 1, ..., H$ the agent receives a representation s_t of the state of the

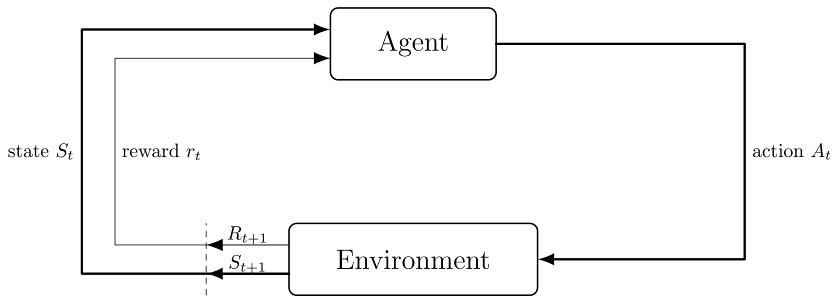


Figure 2.1: Agent Environment interface, from (Sutton and Barto, 1998).

environment, $s_t \in \mathcal{S}$, takes an action, $a_t \in \mathcal{A}$, receives a reward, $r_t \in \mathbb{R}$, and lands in the next state s_{t+1} according to the environment dynamics $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$. Here H denotes the horizon length, $H \in \mathbb{R}^+ \cup \{+\infty\}$.
Formally an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, R, \gamma, P, \mu \rangle$, where:

- $\mathcal{S}$ is the *state space* (discrete or continuous);
- $\mathcal{A}$ is the *action space* (discrete or continuous);
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *reward function*;
- $\gamma \in [0, 1]$ is the *discount factor* for future rewards;
- $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the *transition function*, where $P(\cdot|s, a)$ is the probability distribution over the next states given that the agent executes action a in state s;
- $\mu \in \Delta(\mathcal{S})$ is the initial state distribution.

Formally the reward function is defined over $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, so a reward realization is $R(s_t, a_t, s_{t+1})$. Usually we forget the dependence on the state s_{t+1} by taking the expectation over the next states according to the environment dynamics:

$$r(s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1} \sim P(\cdot|s_t, a_t)} \left[R(s_t, a_t, s_{t+1}) \right]. \tag{2.1}$$

A trajectory is sequence $\tau = \langle s_t, a_t, R_t \rangle_{t=0,\dots,H}$ in which s_t represent the current state, a_t the action taken, R_t the reward received at time step t.

2.1.2 Transition Model

The transition function must satisfy two properties in order for the problem to be an MDP:

Markovian: the current state and action are a sufficient statistic to define the probability over the next states;

Stationarity: the environment dynamics does not change over time.

These usually are assumptions made in order to simplify the problem. The Markovian property can be ensured by adding enough information to the state. The Stationarity property can be ensured in a similar way, if a task is non stationary it can be translated in a stationary task by adding the time component to the state description.

2.1.3 Policy

The agent interacts with the environment by means of a *policy* that defines its *behaviour*. At each decision epoch a decision maker takes action according to its behaviour b. In the most general case the action a_t depends on the whole history from time 0 to time t. We denote the set of histories with $\mathbb{H}_t$. A *stochastic* behaviour b_t can be defined as:

$$b_t : \mathbb{H}_t \to \Delta(\mathcal{A})\,. \tag{2.2}$$

A behaviour is said to be *Markovian* if the distribution over actions depends only on the current state:

$$b_t : \mathcal{S} \to \Delta(\mathcal{A})\,. \tag{2.3}$$

A policy is *stationary* if the distribution over actions does not depends on the time. In this work we will denote a *stationary Markovian* policy with π:

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})\,. \tag{2.4}$$

It is useful to consider *parameterized* policies, where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of policy parameters. To denote a parameterized policy we use the notation $\pi_{\boldsymbol{\theta}}$ and we omit the subscript when it is clear from the context that the considered policy is parameterized. The usage of a parameterization implicitly defines a set of policy in which we are interested, we denote the policy family with Π and we have $\pi \in \Pi$. Common policy parameterizations are:

Linear (deterministic), the resulting action is a linear combination of the state features $\phi(s)$:

$$\pi(s) = \boldsymbol{\theta}^T \boldsymbol{\phi}(s). \tag{2.5}$$

Gaussian, the resulting action has a gaussian distribution, in which the mean and the variance depend on state features:

$$\pi(\cdot|s) \sim \mathcal{N}(m(\boldsymbol{\phi}(s)), v(\boldsymbol{\phi}(s))). \tag{2.6}$$

The Gaussian parameterization is useful for continuous action spaces.

Boltzmann, used for discrete action spaces, the resulting action is a soft-max acting on the weighted state features:

$$\pi(a_i|s) = \frac{e^{\boldsymbol{\theta}_i^T \boldsymbol{\phi}(s)}}{\sum_j e^{\boldsymbol{\theta}_j^T \boldsymbol{\phi}(s)}}\,, \tag{2.7}$$

where $\boldsymbol{\theta}_i$ is the set of parameters associated to action a_i.

In all of these parametrizations the state features might be non-linear features depending on some parameters, e.g. coming from a neural network; radial basis function (RBF) features, tile coding features.

2.1.4 State distribution

Behaving accordingly to a policy in an MDP induces a distribution over the state space. Here we recall the formulation of the γ-discounted future state occupancy (Sutton et al., 1999):

$$\hat{d}^{\pi}_{\mu,\gamma}(s) = \sum_{t=0}^{+\infty} \gamma^t \mathbb{P}(s_t = s|\mu, P, \pi)\,. \tag{2.8}$$

The previous formula defines $\hat{d}^{\pi}_{\mu}$ of a state s as the sum of the discounted probability of being in s in a time step t given the policy, the initial state distribution and the transition model. Formally $\hat{d}^{\pi}_{\mu}$ is not a distribution as it does not sum up to 1. We normalize it obtaining the γ discounted state distribution:

$$d^{\pi}_{\mu,\gamma}(s) = (1-\gamma)\hat{d}^{\pi}_{\mu}(s)\,. \tag{2.9}$$

2.1.5 State Kernel

The transition model and the policy naturally define the *state kernel*, that has a primary role in our work. Formally the state kernel is a function $P^{\pi} : \mathcal{S} \rightarrow \Delta(\mathcal{S})$. It defines, for each state, a probability measures over the set of states. It consider jointly the effect of the transition model and the policy. The state kernel is obtained by marginalizing the transition model over the actions:

$$P^{\pi}(s'|s) = \int_{\mathcal{A}} \pi(a|s) P(s'|s,a) \mathrm{d}a\,. \tag{2.10}$$

2.1.6 Goal and Rewards

In Reinforcement Learning the agent's goal is to maximize the total amount of rewards it receives. This is based on the reward hypothesis (Sutton and Barto, 1998):

> *That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).*

Even if a scalar reward signal might seem limiting, this concept has proven to be very flexible and applicable to a wide range of tasks. For example, in locomotion tasks the reward can be defined to be proportional to the amount of distance traveled from the previous step to the current step. In making a robot reaching a goal the reward can be defined as -1 for every step until the agent finds the goal, encouraging the agent to reach the goal as fast as possible. It is very important in RL that the reward is given in such a way that it describes what is the goal of the task, it should not describe how to do something. However, in some cases the reward definition might be not trivial. Think for example to a reward for a driving

task. In the case of driving it is not trivial to define what does it mean to drive in a good way.
RL tasks can be divided in *episodic* and *continuing*. In episodic tasks we have that H, the horizon length, is finite, while in continuing tasks $H = +\infty$. The *return*, G_t quantifies the agent performance. For *episodic* tasks the return from time step t can be defined as the sum of rewards received until the end of the episode:

$$G_t = \sum_{k=t}^{H} r_k \,. \tag{2.11}$$

It is easy to see that for *continuing tasks* this formulation of return diverges since we have an infinite sum of rewards. To deal with *continuing tasks* we need to introduce the notion of *discounting*. The *discount factor* $\gamma \in [0,1]$ quantifies the present values of future payoffs. We introduce the *discounted return* as the cumulative, discounted, sum of rewards until the end of the episode:

$$G_t = \sum_{k=0}^{H} \gamma^k r_{t+k+1} \,. \tag{2.12}$$

The discount factor, besides having the interpretation mentioned before, can be seen as the probability that the process continues for another step. From the agent point of view, if the discount factor is near to zero greedy actions are profitable since future rewards do not have high values. If the discount factor is near to 1 the agent is *far-sighted*, it is possible for him to sacrifice an action related to a good immediate reward now for a higher reward in the future steps. We indicate with $G(\tau)$ the return associated with the trajectory τ.

2.1.7 Policy and Value Functions

Policy evaluation is the process of quantifying how good a policy is, it is a key step in almost all RL algorithms. The performance of a policy is defined as the expected value of the return under state and action distribution:

$$J^{\pi} = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu \\ a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim P(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \right] . \tag{2.13}$$

The simple idea formalized above is that policy π_1 it is better then policy π_2 if, on average, it collects more rewards. Solving an MDP means to find the policy π^* such that:

$$\pi^* \in \arg\max_{\pi} J^{\pi} \,. \tag{2.14}$$

When we have a parametrized policy we can cast the problem to the point of view of policy parameters. We have that J depends on the policy parameters:

$$J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu \\ a_t \sim \pi_{\theta}(\cdot|s_t) \\ s_{t+1} \sim P(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \right] . \tag{2.15}$$

$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \,. \tag{2.16}$$

The *Value function* (or state-value function) provides a utility measure to a state following a policy. The value of a state s under policy π, denoted as $v^\pi(s)$, is the expected discounted return starting from s following the behaviour prescribed by π:

$$v^\pi(s) = \mathop{\mathbb{E}}_{\pi}\left[G_t | s_t = s\right] \,. \tag{2.17}$$

In this way the value function embeds long-term information. Similarly, we define the value of a state-action pair (s,a) under policy π as the expected discounted return starting from s, executing action a and following the behaviour π. This utility measure is denoted as *Q-function* or *action-value function*:

$$q^\pi(s,a) = \mathop{\mathbb{E}}_{\pi}\left[G_t | s_t = s, a_t = a\right] \,. \tag{2.18}$$

The *value function* is not suitable for control as it does not provide information on which action to take. The *Q-function* provides a utility to all actions in a particular state. Value function and Q-function are clearly strictly related, the former is obtained by averaging the latter over the action distribution defined by the policy:

$$v^\pi(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[q^\pi(s,a)\right] \,. \tag{2.19}$$

Using the state value function it is possible to re-write the performance of a policy:

$$J^\pi = \mathop{\mathbb{E}}_{s\sim\mu}\left[v^\pi(s)\right] \,. \tag{2.20}$$

The value function has also a recursive formulation, denoted as *Bellman expectation operator for v^π* (Bellman, 1957):

$$v^\pi(s) = \mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim P(\cdot|s,a)}}\left[R(s,a,s') + \gamma v^\pi(s')\right] \,. \tag{2.21}$$

We can express in a recursive manner also the Q-function, the associated operator is denoted as *Bellman expectation operator for q^π*:

$$q^\pi(s,a) = \mathop{\mathbb{E}}_{\substack{a'\sim\pi(\cdot|s')\\ s'\sim P(\cdot|s,a)}}\left[R(s,a,s') + \gamma q^\pi(s',a')\right] \,. \tag{2.22}$$

The Bellman expectation operators have several properties (see 2.1.8). The operators are mainly used in iterative policy evaluation (see 2.3.1). We give the formal definition of the Bellman Expectation operators.

Definition 2.1.1 (Bellman Expectation Operator for v^π)**.** *The Bellman expectation operator for v^π is defined as $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$, maps state-value functions to state-value functions:*

$$(T^\pi v)(s) = \mathop{\mathbb{E}}_{\substack{a\sim\pi(s)\\ s'\sim P(\cdot|s,a)}}\left[R(s,a,s') + \gamma v(s')\right] \,. \tag{2.23}$$

Definition 2.1.2 (Bellman Expectation Operator for q^π)**.** *The Bellman expectation operator for q^π is defined as $T^\pi : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, maps Q-functions to Q-functions:*

$$(T^\pi q)(s,a) = \mathop{\mathbb{E}}_{\substack{a'\sim\pi(s') \\ s'\sim P(\cdot|s,a)}} \left[R(s,a,s') + \gamma q(s',a')\right] . \tag{2.24}$$

2.1.8 Optimality Conditions

Besides Eq. (2.14) we can define another partial ordering over policies, the ordering induced by *value functions*.

Definition 2.1.3. *Policy π is better or equal ($\succeq$) policy π' if its expected return is greater or equal to that of π' for all states:*

$$\pi \succeq \pi' \iff v^\pi(s) \geq v^{\pi'}(s),\ \forall s \in \mathcal{S} . \tag{2.25}$$

Following the previous definition an *optimal policy* is a policy that is better or equal to all other policies in all states. This optimality condition is stricter than the one presented in (2.14). From the definition of optimal policy the *optimal state-value function* and the *optimal Q-function* follows:

$$v^*(s) = \max_\pi v^\pi(s) \tag{2.26}$$
$$q^*(s,a) = \max_\pi q^\pi(s,a) . \tag{2.27}$$

There is always at least an optimal (deterministic) policy maximizing the state-value function in every state (Puterman, 1994) and all optimal policies share the same *optimal state value-function*. The knowledge of the *optimal* Q-function makes it possible to find the *optimal* deterministic policy selecting in every state the action yielding the highest q-value:

$$\pi^*(s) \in \arg\max_a q^*(s,a) . \tag{2.28}$$

This holds if we have complete freedom in the selection of the policy (i.e. when Π is the set of all Markovian policies). However for practical algorithms we need to restrict our policy space (e.g. parametric policy space), in these cases it is convenient to use the definition of optimal policy in (2.14).
The optimal state-value function and optimal Q-function satisfy the *Bellman optimality equations*:

$$v^*(s) = \max_a q^*(s,a) \tag{2.29}$$
$$= \max_a \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \left[R(s,a,s') + \gamma v^*(s')\right] . \tag{2.30}$$

$$q^*(s,a) = \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \left[R(s,a,s') + \gamma \max_{a'} q^*(s',a')\right] . \tag{2.31}$$

At an intuitive level the *Bellman optimality equation* for v^* expresses the fact that the optimal state-value function must equal the expected return for the best action in that state. The *Bellman optimality equation* for q^*, similarly, expresses the fact that the optimal Q-function must equal the immediate reward plus the discounted return of the best action in the next state according to the environment dynamic.
The right handside of Eq. (2.29) it's defined as *Bellman Optimality Operator*.

Definition 2.1.4 (Bellman Optimality Operator for v^*)**.** *The Bellman optimality operator for v^* is defined as $T^* : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$, maps state-value functions to state-value functions:*

$$(T^*v)(s) = \max_a \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[R(s,a,s') + \gamma v(s')\right] . \tag{2.32}$$

Definition 2.1.5 (Bellman Optimality Operator for q^*)**.** *The Bellman optimality operator for q^* is defined as $T^* : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, maps Q-functions to Q-functions:*

$$(T^*q)(s,a) = \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[R(s,a,s') + \gamma \max_{a'} q(s',a')\right] . \tag{2.33}$$

The Bellman operators (2.1.8, 2.1.7, 2.1.8, 2.1.7) are characterized by the following properties, where we will use T to denote both the expectation operator and the optimality operator:

– Monotonicity:

$$f_1 \leq f_2 \implies Tf_1 \leq Tf_2\,; \tag{2.34}$$

– Max-Norm contraction:

$$\left\|Tf_1 - Tf_1\right\|_\infty \leq \gamma \left\|f_1 - f_2\right\|_\infty ,\forall f_1, f_2\,; \tag{2.35}$$

– v^* is the *unique fixed point* of T^*. v^π is the *unique fixed point* of T^π.

– Convergence:

$$\lim_{k\to\infty} (T^*)^k f = v^* ,\forall f \in \mathbb{R}^{|\mathcal{S}|} \tag{2.36}$$

$$\lim_{k\to\infty} (T^\pi)^k f = v^\pi ,\forall f \in \mathbb{R}^{|\mathcal{S}|} . \tag{2.37}$$

Solving Eq. (2.29) and Eq. (2.31) leads us directly to the MDP solution. Unfortunately these equations are non-linear because of the presence of the $\max$ operator and there is no closed form solution for the general case. There exists many iterative solution methods trying to solve an approximation of the Bellman optimality equations that are discussed in the following sections.

2.2 Linear Programming

The problem of computing an optimal policy for an infinite horizon finite-states MDP can be formulated as linear program (LP) (d'Epenoux, 1963). In this section we will go further in the LP formulation since our algorithm extends on this. The basic idea follows from Definition 2.1.3: we want to find the value function maximizing the value of each state weighted by the initial state distribution subject to a feasibility constraint:

$$\begin{aligned}
&\underset{v}{\text{minimize}} \sum_{s\in\mathcal{S}} \mu(s)v(s) \\
&\text{subject to } v(s) \geq r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)v(s')\,, \forall s \in \mathcal{S},\, \forall a \in \mathcal{A}\,.
\end{aligned}$$

Where we have $|\mathcal{S}|$ variables and $|\mathcal{S}||\mathcal{A}|$ constraints. Notice that the maximization role is taken by the constraints, while we need to minimize since otherwise an optimal solution would have infinite values for all variables v.

Theorem 2.2.1 (Linear Programming Solution) v^* *is the solution of the above linear program.*

It is possible to prove the above theorem using the properties of the Bellman optimality operators, the interested reader can refer to Appendix A. It is also interesting to analyze the Dual Linear Program:

$$\begin{aligned}
&\underset{p}{\text{maximize}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d(s,a)r(s,a) \\
&\text{subject to } \sum_{a\in\mathcal{A}} d(s,a) = \mu(s) + \gamma \sum_{s'\in\mathcal{S}} \sum_{a'\in\mathcal{A}} d(s',a')P(s|s',a'),\ \forall s \in \mathcal{S} \\
&\qquad d(s,a) \geq 0,\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{S}\,.
\end{aligned}$$

In the dual program $d(s,a)$ is the discounted state action distribution (Sutton et al., 1999):

$$d(s,a) = \sum_{t}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a)\,. \tag{2.38}$$

The objective of the dual is the usual objective: maximize the expected discounted sum of rewards. The constraints are needed for ensuring that p is a valid distribution. From the dual program it is possible to extract the optimal policy:

$$\pi^*(s) \in \arg\max_a d(s,a)\,. \tag{2.39}$$

For each state s we have that $d(s,a^*) = 1$ if action a^* is optimal in state s. When there are multiple optimal actions we have a split of the probability between these actions. The LP formulation is an exact way of solving the MDPs, unfortunately this formulation becomes impractical when the states space grows too much.

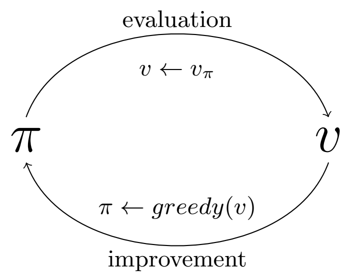


Figure 2.2: Policy Iteration, from (Sutton and Barto, 1998).

2.3 Dynamic Programming

Dynamic programming (Bellman, 1957) (DP) is by far the most common way for solving MDPs when the dynamics and the reward function are known. DP takes into account the sequential or temporal structure of the problem. It is a method for solving complex problems by breaking them into subproblems. A problem, in order to be solved by DP techniques, must have two properties: 1. Optimal Substructure, 2. Overlapping Subproblems. The optimal substructure property means that the principle of optimality applies and the optimal solution can be decomposed into subproblems. The overlapping subproblems property means that subproblems recur many times and the solutions can be stored and reused. In MDPs both properties are satisfied. The **Bellman equation** gives the recursive decomposition while the **state-value function** caches and reuses previous solutions.

2.3.1 Policy Iteration

Policy Iteration (Howard, 1960) is a two stage algorithm: Policy Evaluation (PE) and Policy Improvement (PI). The first stage (Policy Evaluation) aims at evaluating the current policy through the application of the Bellman Expectation Operator. The PE phase is needed since we want to perform an update of the current policy. The update direction can be computed using the action-value function in the following way. Suppose we know $v^\pi(s')$ for a particular s'. To improve the current policy it is enough to compute $q^\pi(s', a)$ for each $a \in \mathcal{A}$ and then to compare it with $v^\pi(s')$. If exists an action a' such that $q(s', a') > v^\pi(s')$, then we can *improve* the current policy using the following update rule:

$$\pi'(s) = \begin{cases} \pi(s) & \text{if } s \neq s' \\ \arg\max_a q^\pi(s, a) & \text{if } s = s' \end{cases} \tag{2.40}$$

Basically this update rule states that the new policy is the same as the old policy except for states in which there exists an action a such that the action-value function q^π is greater than the value function v^π. The justification for the previous statement comes from the *Policy Improvement Theorem.*

Theorem 2.3.1 (Policy Improvement Theorem) *Let π and π' be a pair of policies such that:*

$$q^\pi(s, \pi'(s)) \geq v^\pi(s)\,, \forall s \in \mathcal{S}\,, \tag{2.41}$$

Algorithm 1 Policy Iteration

```
Input: $\mathcal{S}, \mathcal{A}, P, R$
Output: $\pi \approx \pi^*$
1: Initialize $v^\pi$ and $\pi$ randomly
2: repeat
3:     $v^\pi \leftarrow$ Evaluate policy $\pi$ by applying $T^\pi$
4:     $\pi \leftarrow$ Improve policy $\pi$
5: until no improvement found
6: return Policy $\pi$
```

Then the policy π' must be as good or better than π.

Given the previous theorem we can expand the update rule (2.40) by computing the *greedy policy* in all states:

$$\pi'(s) \in \arg\max_a q^\pi(s, a)\,. \tag{2.42}$$

This update rule meets the condition of Theorem 2.3.1, so the updated policy is as good as, or better than the old one. When the new policy is as good as the old, $v^{\pi'} = v^\pi$, we know, thanks to the optimality operator property, that $v^\pi = v^*$, we have obtained the optimal policy. Now we can formalize the Policy Iteration Algorithm, see Algorithm 1. All the previous ideas can be easily extended to the case of stochastic policies.

2.3.2 Value Iteration

The Value Iteration (VI) algorithm performs evaluation and improvement in the same step. While Policy Iteration performs a search in the space of policies, Value Iteration searches in the space of value functions, calculating the policy only in the last step. Value Iteration is obtained simply by turning the Bellman Optimality Operator into an update rule:

$$v_{k+1}(s) = \max_a \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[R(s, a, s') + \gamma v_k(s') \right] \,, \forall s \in \mathcal{S}\,. \tag{2.43}$$

Value Iteration algorithm is reported in Algorithm 2.
Both PI and VI converge to an optimal policy for discounted finite MDPs.

2.4 Policy Search

Policy search (PS) methods explore directly the policy space. In the PS framework the RL problem is formalized as:

$$\pi^* \in \arg\max_{\pi \in \Pi} J^\pi\,. \tag{2.45}$$

Among PS methods it is worth mentioning policy gradient methods (Sutton et al., 1999; Peters and Schaal, 2008b). Policy gradient methods consider a set of parameterized policies

Algorithm 2 Value Iteration

```
Input: $\mathcal{S}, \mathcal{A}, P, R$
Output: $\pi \approx \pi^*$
1: Initialize $v$ randomly
2: repeat
3:     $v \leftarrow$ Apply Bellman Optimality operator $T^* v$
4: until no changes in the value function
5: return Policy $\pi$:
```

$$\pi(s) = \arg\max_a \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[R(s,a,s') + \gamma v(s') \right] , \forall s \in \mathcal{S} \, . \tag{2.44}$$

(see Section 2.1.3) $\pi_{\boldsymbol{\theta}}(a|s)$ where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of policy parameters. In this cases the space of policies is represented by $\Pi = \left\{ \pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d \right\}$. A standard requirement is that the policy must be differentiable in its parameters. A common technique to maximize the performance of a policy is to perform *gradient ascent* over the policy parameters:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^k) \, , \tag{2.46}$$

where $\alpha \geq 0$ is the *learning rate*, also called *step size*. The quantity $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^k)$ is the gradient of the performance of the policy and it can be computed through the policy gradient theorem (Sutton et al., 1999).

Theorem 2.4.1 (Policy Gradient Theorem) *For any MDP:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathcal{S}} d_{\mu,\pi_{\boldsymbol{\theta}}}(s) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) q^{\pi_{\boldsymbol{\theta}}}(s,a) da ds \, . \tag{2.47}$$

In real cases since the state distribution is not known it is not possible to compute analytically the gradient of the performance measure. We need to resort to a sample approximation of it. Gradient ascent method guarantees convergence at least to a local optimum even if in practice local optima can be avoided using a stochastic approximation to the gradient or policies belonging to high dimensional spaces.

Gradient Estimation

REINFORCE method (Williams, 1992) builds on the fact that the outer integral in Eq. (2.47) is the analytical expression of the expected value under the state distribution induced by the policy π. We can rewrite the gradient in the following way (where we use π for $\pi_{\boldsymbol{\theta}}$):

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \mathop{\mathbb{E}}_{s \sim d_{\mu,\pi}(\cdot)} \left[\int_{\mathcal{A}} \nabla \pi(a|s) q^{\pi}(s,a) ds \right] && (2.48) \\ &= \mathop{\mathbb{E}}_{\substack{s \sim d_{\mu,\pi}(\cdot) \\ a \sim \pi(\cdot|s)}} \left[\nabla \log \pi(a|s) q^{\pi}(s,a) \right] , && \text{log trick: } \nabla f = f \nabla \log f \end{aligned}$$

The estimation of the action value function can be performed in a straightforward way from the sum of discounted rewards: $\widehat{q}^{\pi} \approx \sum_{k=0}^{H} \gamma^k r(s_k, a_k)$ Summarizing, we can obtain an approximation of the gradient through the following estimator:

$$\widehat{\nabla_\theta J(\boldsymbol{\theta})}_{RF} = \langle \left(\sum_{k=0}^{H} \nabla_\theta \log \pi(a_k, s_k) \right) \left(\sum_{k=0}^{H} \gamma^k r(s_k, a_k) \right) \rangle_N \,, \tag{2.49}$$

where N is the batch size, the number of collected trajectories, $\langle \cdot \rangle_N$ denotes the sample mean over N trajectories and H is the horizon length.
REINFORCE gradient estimation suffers of high variance, that grows at least cubically with the length of the horizon and quadratically with the magnitude of the reward.
G(PO)MDP (Baxter and Bartlett, 2001) employs a better approximation with lower variance exploiting the observation that future actions do not depend on past rewards (if the policy does not change within an episode) leading to the following estimation:

$$\widehat{\nabla_\theta J(\boldsymbol{\theta})}_{G(PO)MDP} = \langle \sum_{l=0}^{H} \left(\sum_{k=l}^{H} \nabla_\theta \log \pi(a_k | s_k) \right) \left(\gamma^l r(s_l, a_l) \right) \rangle_N \,. \tag{2.50}$$

Natural Policy Gradient

As presented in (Furmston and Barber, 2012), the general form of Policy Gradient updates is:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \alpha \mathbf{G}(\boldsymbol{\theta}^k)^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^k) \,, \tag{2.51}$$

where $\mathbf{G}(\boldsymbol{\theta})$ is a positive definite matrix. The matrix $\mathbf{G}(\boldsymbol{\theta})$ defines the metric used to measure vectors in the parameter space. The parameter update in Equation (2.46) can be derived from this formulation by using $\mathbf{G}(\boldsymbol{\theta}^k) = \boldsymbol{I}$, that is the Euclidean norm. This assumption does not take into account that the space parameterized is actually a Riemannian manifold. As (Amari, 1998) suggested it is better to define a metric based not on the choice of coordinates but rather on the manifold (i.e. the surface) that these coordinates parameterize. Natural Gradient (Kakade, 2002) exploits this structure and uses as metric the Fisher Information Matrix of the trajectory distribution and due to the Markovian structure of the dynamics it is given by:

$$\mathbf{G}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi}_{\mu,\gamma}(\cdot) \\ a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)^T \right] \,. \tag{2.52}$$

Convergence to a local maximum is guaranteed; by choosing a more direct path to the optimal solution in parameter space, the natural gradient has faster convergence and avoids premature convergence of steepest ascent gradient; the natural policy gradient can be shown to be covariant, i.e., independent of the parameterization of the policy; it requires fewer data points for a good gradient estimate.

2.5 MDP extensions

The standard MDP framework has been extended in the literature in order to cover many real cases scenarios. Here we report some notable extensions that can be related and compared to the framework considered in this work.

2.5.1 MDP with imprecise probability

MDP with imprecise probability (MDPIP) (Harmanec, 2002) covers the case in which it is not easy (or even possible) to define a precise probability measure for a given transition $P(\cdot|s,a)$. In this cases the probability parameters are imprecise and therefore the transition model cannot be defined by means of a probability distribution but it must be defined through a set of probability distribution. These sets of distribution are known as transition credal sets $\mathcal{K}(\cdot|s,a)$.
In this framework it is common to use game theoretic approaches maximizing the lowest expected reward with respect to the probability parameters, known as Γ-maximin criterion. In this cases the optimal value function is:

$$v^*(s) = \max_{a \in \mathcal{A}} \left\{ \min_{P(\cdot|s,a) \in \mathcal{K}(\cdot|s,a)} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v^*(s') \right\} . \tag{2.53}$$

2.5.2 Bounded-parameter Markov decision processes

Bounded-parameter MDPs (Givan et al., 2000) can be used to represent variation or uncertainty concerning the parameters of sequential decision problems in cases where no prior probabilities on the parameter values are available. BMDPs form an efficiently solvable special case of the class of MDPIPs. In this context interval value functions are introduced as a natural extension of traditional value functions. An interval value function assigns a closed real interval to each state, representing the fact that the value of that state falls within that interval. An interval value function can be used to bound the performance of a policy over the set of exact MDPs associated with a given bounded-parameter MDP.

2.5.3 Non stationary MDPs

Non stationary MDPs (Bowerman, 1974) are used to model scenarios in which the transition dynamic and the reward function are non stationary, i.e. they change over time. We can model this scenario with a different transition and reward functions for each timestep:

$$P(s'|s,a,t) = P^t(s'|s,a) \tag{2.54}$$
$$R(s,a,s',t) = R^t(s,a,s') . \tag{2.55}$$

A usual assumption made in this family of MDPs is that there is a correlation between contniguous time frames. In this way the agent can use recent experience and forget past experience. In (Hopp et al., 1987) an optimality condition for non stationary MDPs has been defined. Moreover, (Garcia and Smith, 2000), (Cheevaprawatdomrong et al., 2007) and (Ghate and L. Smith, 2013) focused on how to find an optimal policy for non stationary MDPs in an effective way.

Chapter 3

State of the Art

The measure of greatness in a scientific idea is the extent to which it stimulates thought and opens up new lines of research.

Paul A.M. Dirac

In this chapter, we present the framework of Configurable Markov Decision Processes (Metelli et al., 2018a) Section 3.1 since it is the MDP extension that formalizes our learning problem. We recall that our main goal is to find an algorithm able to jointly optimize policy and model parameters by keeping into account their relationship. In Section 3.2, we present the Relative Entropy Policy Search (REPS) algorithm as our algorithm takes inspiration from it.

3.1 Configurable Markov Decision Processes

In most of the problems tackled by RL the environment is considered to be a *fixed* entity that cannot be controlled. Nevertheless, there exist several real-world motivational examples in which a partial control on the environment can be exercised by the agent itself or by an external *supervisor*. With the phrase *environment configuration* we refer to the activity of altering some environmental parameters to improve the performance of the agent.
Configurable Markov Decision Processes (CMDPs) are an extension to the Markov Decision Processes framework that considers the environment optimization. A CMDP is an MDP in which it is possible to *configure* some environmental parameters in order to improve the agent performance or the learning speed. CMDPs arise naturally in many real world cases. Formula One engineers have to configure their cars (e.g. wings, tyres, engine, brakes) to minimize lap time. In industry, machines have to be configured to maximize the production rate. These are cases in which the goal of the configuration is to improve performance, however in the case of car race it is possible for the supervisor to improve the learning speed of the pilot by presenting tracks of different difficulty. This example resembles the child example in Chapter 1, in which parents present to their son goals of

different difficulty, in order to teach him to walk.
In all the above examples the *agent* or a *supervisor* configures some environmental parameters to achieve good performance.
In this work we consider *configurable transition functions* and we focus on how to improve performances by jointly optimizing the environment and the policy.
We remark that there is a profound difference between this scenario and the other MDP extensions presented in Section 2.5. In other MDP extensions the environment is assumed not fixed or not fully known in the case of MDPIP, that is the transition function is not stationary, but there is no possibility to configure it. In the context of CMDPs the environment is not fixed since we want to exploit the configurability structure to obtain a performance gain.

3.1.1 Formal Model

The following definition characterizes the *Configurable Markov Decision Processes*.

Definition 3.1.1 (Configurable Markov Decision Process). *A Configurable Markov Decision Process is a tuple $\mathcal{CM} = \langle \mathcal{S}, \mathcal{A}, R, \mu, \gamma, \mathcal{P}, \Pi \rangle$, where $\langle \mathcal{S}, \mathcal{A}, R, \mu, \gamma \rangle$ is an MDP without the transition model, $\mathcal{P}$ is the model space and Π is the policy space.*

The model space $\mathcal{P}$ will be sometimes referred to as configuration space in the rest of the document. In this scenario the learning subjects are the policy and the model spaces. The performance of a model-policy pair is denoted by $J_\mu^{P,\pi}$ and defined as:

$$J_\mu^{P,\pi} = \frac{1}{1-\gamma} \int_{\mathcal{S}} d_\mu^{P,\pi}(s) \int_{\mathcal{A}} \pi(a|s) \int_{\mathcal{S}} P(s'|s,a) R(s,a,s') \mathrm{d}s \mathrm{d}a \mathrm{d}s \,, \tag{3.1}$$

where $d_\mu^{P,\pi}$ is the discounted state distribution induced by the model P and the policy π (see Section 2.1.4). The goal of a learning process in a CMDP is to find a couple model-policy pair (P^*, π^*) such that:

$$P^*, \pi^* \in \underset{P \in \mathcal{P}, \pi \in \Pi}{\arg\max} J_\mu^{P,\pi} \,. \tag{3.2}$$

Recall that the standard MDP solution is only a half of the CMDP solution, as the MDP solution is defined as:

$$\pi^* \in \underset{\pi \in \Pi}{\arg\max} J_\mu^{P,\pi} \,, \tag{3.3}$$

under a fixed model P. In other words, it requires to find the optimal policy in a fixed environment P.
It is useful now to redefine some important quantities in the MDP theory by considering the contribution of the model:

$$v^{P,\pi}(s) = \underset{\substack{a \sim \pi(\cdot|s) \\ s' \sim P(\cdot|s,a)}}{\mathbb{E}} \left[R(s,a,s') + \gamma v^{P,\pi}(s') \right] , \tag{3.4}$$

$$q^{P,\pi}(s,a) = \underset{s' \sim P(\cdot|s,a)}{\mathbb{E}} \left[R(s,a,s') + \gamma v^{P,\pi}(s') \right] \,. \tag{3.5}$$

These quantities are the same as the ones defined in Section 2.1 but here we emphasize the model contribution. Notice that in the case of fixed model (standard MDP framework) we can forget this dependency but in our context it is better to keep it explicit.
To measure the utility of a transition we introduce the state-action-next-state value function or U-function:

$$u^{P,\pi}(s,a,s') = R(s,a,s') + \gamma v^{P,\pi}(s') \,. \tag{3.6}$$

The U-function is the expected return starting from state s, taking action a, landing in state s' and then following the trajectory induced by the policy π and the model P. It can be considered as an extension of the action-value function that considers the contribution of the model.

3.1.2 Model and Policy spaces

In Section 2.1.3 we discussed about possible policy parameterizations. Working with a bounded (e.g. parameterized) policy space can be beneficial since we have a finite number of parameters to tune. Besides reducing the search space, a bounded policy space can also represent real world limitations on the selection and implementation of the policy. Similarly, when configuring the transition model we can have different scenarios. In the most common case we are limited in the model selection, we can configure some MDP parameters (e.g. brake pressure, tyres type, wing angle) but we cannot configure the main dynamic laws (e.g. physics of the environment). In (Metelli et al., 2018a) three types of scenarios are defined:

Unconstrained: no limitation on the model and policy spaces. This is not the most common scenario, thus it is possible to have no constraints on the model (policy) selection.

Constrained: in this case we have an initial model (policy) P_0 (π_0) and we are not allowed to move too far. The notion of distance in this case can be expressed with the usual divergences between probability measures (e.g. Kullback-Leibler, Total Variation, Wasserstein). In this cases we formalize the model and policy spaces in the following way:

$$\mathcal{P} = \{P : d(P, P_0) < \epsilon_p\} \tag{3.7}$$

$$\Pi = \{\pi : d(\pi, \pi_0) < \epsilon_\pi\} \,. \tag{3.8}$$

Parametric: in the most common case only a limited part of the model can be configured. We can represent this scenario by means of a parametric model space in which the alteration are limited to the choice of the model parameters. We express this type of model and policy spaces by:

$$\mathcal{P}_\Omega = \left\{P_{\boldsymbol{\omega}} : \boldsymbol{\omega} \in \Omega \subseteq \mathbb{R}^k\right\} \tag{3.9}$$

$$\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p\} \,. \tag{3.10}$$

From now on, we use $\boldsymbol{\theta}$ for identifying the policy parameters and $\boldsymbol{\omega}$ for identifying the model parameters.

The parametric case is the scenario mainly considered in this work.

3.1.3 Theoretical Foundations

Safe Policy Model Iteration (SPMI), presented in (Metelli et al., 2018a), is a safe approach for solving the CMDP learning problem. A possible approach to Safe Reinforcement Learning is to employ a lower bound on the performance improvement obtained by moving from a model-policy pair (P, π) to another pair (P', π'). Once obtained a lower bound on the performance improvement it is possible to maximize this lower bound finding the model-policy pair yielding the best possible improvement. It is useful to introduce some important quantities (Kakade and Langford, 2002; Pirotta et al., 2013). We define the policy advantage function, that quantifies how much an action is better than the others in a state:

$$A^{P,\pi}(s,a) = q^{P,\pi}(s,a) - v^{P,\pi}(s)\,. \tag{3.11}$$

We introduce the model advantage function, that quantifies how much landing in a state s' after having performed action a starting in state s is better than the other states:

$$\begin{aligned} A^{P,\pi}(s,a,s') &= R(s,a,s') + \gamma v^{P,\pi}(s') - q^{P,\pi}(s,a) = & (3.12)\\ &= u^{P,\pi}(s,a,s') - q^{P,\pi}(s,a)\,. & (3.13)\end{aligned}$$

We define the *relative advantage* functions for quantifying the one-step improvement attained by a new policy π' or a new model P' when starting from the model P and the policy π:

$$A_{P,\pi}^{P,\pi'}(s) = \mathop{\mathbb{E}}_{a\sim\pi'(\cdot|s)}\left[A^{P,\pi}(s,a)\right], \tag{3.14}$$

$$A_{P,\pi}^{P',\pi}(s,a) = \mathop{\mathbb{E}}_{s'\sim P'(\cdot|s,a)}\left[A^{P,\pi}(s,a,s').\right] \tag{3.15}$$

We define also their expected value under the γ-discounted state distribution:

$$\mathbb{A}_{P,\pi,\mu}^{P,\pi'} = \mathop{\mathbb{E}}_{\substack{a\sim\pi'(\cdot|s)\\ s\sim d_\mu^{P,\pi}}}\left[A^{P,\pi}(s,a)\right], \tag{3.16}$$

$$\mathbb{A}_{P,\pi,\mu}^{P',\pi} = \mathop{\mathbb{E}}_{\substack{(s,a)\sim\delta_\mu^{P,\pi}\\ s'\sim P'(\cdot|s,a)}}\left[A^{P,\pi}(s,a,s')\right], \tag{3.17}$$

Where $\delta_\mu^{P,\pi}$ is the stationary state–action distribution under the model P and the policy π. From the previous theory we can obtain a lower bound of the performance improvement using only information available from the current model-policy.

Theorem 3.1.1 (Coupled Bound) *The performance improvement of model-policy pair* (P', π') *over* (P, π) *can be lower bounded as:*

$$\underbrace{J_\mu^{P',\pi'} - J_\mu^{P,\pi}}_{\textit{Performance improvement}} \geq \underbrace{\frac{\mathbb{A}_{P,\pi,\mu}^{P',\pi'}}{1-\gamma}}_{\textit{Advantage term}} - \underbrace{\frac{\gamma \Delta A_{P,\pi}^{P',\pi'} D_{\mathbb{E}}^{P'^{\pi'},P^{\pi}}}{2(1-\gamma)^2}}_{\textit{Dissimilarity Penalization}}, \tag{3.18}$$

where $\Delta A_{P,\pi}^{P',\pi'} = \sup_{s,s' \in \mathcal{S}} \left| A_{P,\pi}^{P',\pi'}(s') - A_{P,\pi}^{P',\pi'}(s) \right|$, *and*
$D_{\mathbb{E}}^{P'^{\pi'},P^{\pi}} = \mathbb{E}_{s \sim d_{\mu}^{P,\pi}} ||P'^{\pi'}(\cdot|s) - P^{\pi}(\cdot|s)||$.

The bound is composed by two terms. The advantage term quantifies how much the model-policy pair (P', π') is better with respect to the current one. The dissimilarity term is a penalization that prevents the algorithm from moving too far away from the current model-policy. In practice this bound it is difficult to use in an algorithm since it does not separate the dependence of the two main components of a CMDP. The lower bound on the performance improvement used in SPMI is the following.

Theorem 3.1.2 (Decoupled Bound) *The performance improvement of model-policy pair (P', π') over (P, π) can be lower bounded as:*

$$\underbrace{J_{\mu}^{P',\pi'} - J_{\mu}^{P,\pi}}_{\textit{Performance improvement}} \geq B(P', \pi') = \underbrace{\frac{\mathbb{A}_{P,\pi,\mu}^{P',\pi} + \mathbb{A}_{P,\pi,\mu}^{P,\pi'}}{1-\gamma}}_{\textit{Advantage term}} - \underbrace{\frac{\gamma \Delta q^{P,\pi} D}{2(1-\gamma)^2}}_{\textit{Dissimilarity Penalization}}, \tag{3.19}$$

where D is a dissimilarity term keeping into account the dissimilarity of the model and the policy and $\Delta q^{P,\pi} = \sup_{s,s' \in \mathcal{S}, a,a' \in \mathcal{A}} \left| q^{P,\pi}(s', a') - q^{P,\pi}(s, a) \right|$.

This bound effectively decouples the effect of the model and the policy and it is used in SPMI.

3.1.4 Safe Policy Iteration and Safe Model Iteration

In (Metelli et al., 2018a) the lower bound in Theorem 2.3.1 is used in different ways. When limited to policy learning, the resulting algorithm resembles Safe Policy Iteration (SPI) (Pirotta et al., 2013) but uses a new lower bound on performance improvement. The idea is to select a target policy $\bar{\pi}$, the greedy policy with respect to the q-function, and then find the value α^* such that the policy obtained as convex combination between the current policy and the target one maximizes the value of the lower bound. SPI is reported in Algorithm 3. Safe Model Iteration (SMI) is the symmetrical version with respect to SPI, for the model learning problem, that is learning the best model for the current policy. As before, the idea is to select a target model $\bar{P}$, the greedy model with respect to the u-function, and then find the value β^* such that the model obtained as convex combination between the current model and the target one maximizes the value of the lower bound. SMI is reported in Algorithm 4. SMI-SPI is a sequential approach derived from the previous approaches. It does a full model learning with a fixed policy and then a full policy learning with fixed model.
SPI-SMI is the symmetrical version. It is worth noting that these two algorithms do not solve the original CMDP problem, but an approximated version.

Algorithm 3 Safe Policy Iteration

```
Output: Optimal Policy
1: Initialize $\pi^{(0)}$ randomly
2: for t = 0,1,... until convergence do
3:     Evaluate current policy $\pi^{(t)}$
4:     Compute target $\bar{\pi}(s) \in \arg\max_{a \in \mathcal{A}} q^{P,\pi}(s,a)$
5:     Compute terms $\mathbb{A}_{P,\pi,\mu}^{P,\bar{\pi}}, D_{\infty}^{\bar{\pi},\pi}, D_{\mathbb{E}}^{\bar{\pi},\pi}, \Delta q^{P,\pi}$
6:     Compute $\alpha^* = \min\left(\frac{(1-\gamma)}{\gamma \Delta q^{P,\pi}} \frac{\mathbb{A}_{P,\pi,\mu}^{P,\bar{\pi}}}{D_{\infty}^{\bar{\pi},\pi}, D_{\mathbb{E}}^{\bar{\pi},\pi}}, 1\right)$
7:     Update the policy as: $\pi^{(t+1)} = \alpha^* \bar{\pi} + (1-\alpha^*)\, \pi^{(t)}$
8: end for
9: return Optimal Policy $\pi^{(t)}$
```

Algorithm 4 Safe Model Iteration

```
Output: Optimal Model
1: Initialize $P^{(0)}$ randomly
2: for t = 0,1,... until convergence do
3:     Evaluate current model $P^{(t)}$
4:     Compute target $\bar{P}(s,a) \in \arg\max_{s' \in \mathcal{S}} u^{P,\pi}(s,a,s')$
5:     Compute terms $\mathbb{A}_{P,\pi,\mu}^{\bar{P},\pi}, D_{\infty}^{\bar{P},P}, D_{\mathbb{E}}^{\bar{P},P}, \Delta q^{P,\pi}$
6:     Compute $\beta^* = \min\left(\frac{1-\gamma}{\gamma^2 \Delta q^{P,\pi}} \frac{\mathbb{A}_{P,\pi,\mu}^{\bar{P},\pi}}{D_{\infty}^{\bar{P},P}, D_{\mathbb{E}}^{\bar{P},P}}, 1\right)$
7:     Update the model as: $P^{(t+1)} = \beta^* \bar{P} + (1-\beta^*)\, P^{(t)}$
8: end for
9: return Optimal model $P^{(t)}$
```

3.1.5 Safe Policy Model Iteration

Safe Policy Model Iteration (SPMI) requires to compute the target policy $\bar{\pi}$ and model $\bar{P}$, then it computes the next policy and model as linear combination of the current pair and the target pair:

$$\pi' = \alpha\bar{\pi} + (1-\alpha)\pi \tag{3.20}$$

$$P' = \beta\bar{P} + (1-\beta)P\,, \tag{3.21}$$

where α and β are the parameters maximizing the lower bound in 3.19. SPMI is reported in Algorithm 5.

In the SPMI algorithm, it is the value of the lower bound that selects which update has to be performed at each step, such as a policy update, a model update or a policy-model (joint) update. Instead, SPMI alternated (SPMI-alt) forces policy and model updates to be alternated, independently to the bound value. SPMI-sup uses the same update strategy of SPMI, but it optimizes a slightly looser lower bound on performance improvement. This bound is, basically, the straightforward restatement of the one presented in (Pirotta et al.,

Algorithm 5 Safe Policy Model Iteration

```
Output: Optimal Model Policy Pair
1: Initialize $\pi^{(0)}, P^{(0)}$ randomly
2: for t = 0,1,... until convergence do
3:     $\bar{\pi}$ = PolicyChooser($\pi^{(t)}$)
4:     $\bar{P}$ = ModelChooser($P^{(t)}$)
5:     $\alpha^*, \beta^* = \arg\max_{\alpha,\beta} B(\alpha, \beta)$
6:     $\pi^{(t+1)} = \alpha^* \bar{\pi} + (1-\alpha)\pi^{(t)}$
7:     $P^{(t+1)} = \beta^* \bar{P} + (1-\beta)P^{(t)}$
8: end for
9: return Optimal Policy Model Pair $(P^{(t)}, \pi^{(t)})$
```

2013), adapted to deal with a learning problem in the CMDP framework:

$$J_{\mu}^{P',\pi'} - J_{\mu}^{P,\pi} \geq \frac{\mathbb{A}_{P,\pi,\mu}^{P',\pi} + \mathbb{A}_{P,\pi,\mu}^{P,\pi'}}{1-\gamma} - \frac{\gamma \Delta q^{P,\pi} D_{\text{sup}}}{2(1-\gamma)^2}, \tag{3.22}$$

where $D_{\text{sup}} = D_{\infty}^{\pi',\pi^2} + 2D_{\infty}^{\pi',\pi} D_{\infty}^{P',P} + \gamma D_{\infty}^{P',P^2}$ (see (Metelli et al., 2018a) for the definition of these distances), which is obtained by the dissimilarity term of the decoupled bound (Eq. (3.19)), upper bounding $D_{\mathbb{E}}^{\pi',\pi} \leq D_{\infty}^{\pi',\pi}$ and $D_{\mathbb{E}}^{P',P} \leq D_{\infty}^{P',P}$.

3.1.6 Limitations

Although this approach succeeded in showing the advantages of configuring the environment in some illustrative examples, it is quite far from being applicable to real-world scenarios. We believe that the most significant limitations of SPMI are three. First, it is applicable only to problems with a finite state-actions space, while the most interesting examples of CMDPs have, at least, a continuous state space (e.g., the car configuration problem). Second, it requires a full knowledge of the environment dynamics. This latter limitation is the most relevant as, in reality we almost never know the true environment dynamics, and even if a model is available it might be too approximate or hardy usable being very complex and computationally expansive (e.g., the fluido-dynamical model of a car). A third problem could be the high sample complexity of safe methods. Being conservative, safe methods allow only small updates. Therefore the required number of iterations might be too high for online applications.

3.2 Relative Entropy Policy Search

Relative Entropy Policy Search (Peters et al., 2010) (REPS) is an information theoretic approach to the problem of *policy search*. The main idea behind information theoretic approaches is to stay close to the observed data (Deisenroth, 2011). This idea is translated in the constraint that the stationary distribution after the policy update should not jump away from the stationary distribution before the update. The updated policy should stay close to the old one to avoid premature convergence to highly suboptimal policies and to limit the information loss.
Policy Gradient methods (Section 2.4) tackle this problem by allowing only small incremental policy updates. However, they are in some sense myopic, looking only at local information. REPS is a more powerful method based on a *constrained maximization* problem with a closed form solution. By looking at a neighborhood of the current policy it is able to overcome local minima as we will see later in this dissertation.
Natural Policy gradient algorithms (Kakade, 2002) were the first to implement the information theoretic approach but they still require a user-specified learning rate. REPS uses the same information theoretic constraint as Natural Actor Critic (Peters and Schaal, 2008a) (NAC), but updates the policy using a weighted maximum likelihood estimates, thus not requiring a learning rate. The distance index used in the constraint $D_{KL}(d\|\bar{d})$, where d is the distribution we search for and $\bar{d}$ is the distribution from which samples are generated, forces d to be low where $\bar{d}$ is low, intuitively it prevents d to move too far from the old data.

3.2.1 Problem Formulation

REPS considers infinite horizon problems, thus it maximizes the average reward per time step. We denote with $d^\pi(s)$ the stationary state distribution induced by the policy π, with $d^\pi(s,a) = d^\pi(s)\pi(a|s)$ the distribution over states and actions induced by the new policy, with $\bar{d}(s,a)$ the observed data distribution.

Primal The constrained optimization problem considered is the following:

$$\underset{d}{\text{maximize}} \int_{\mathcal{S}}\int_{\mathcal{A}} d^\pi(s,a)r(s,a)\,\mathrm{d}s\,\mathrm{d}a \tag{3.23}$$

$$\text{subject to } D_{KL}(d^\pi||\bar{d}) = \int_{\mathcal{S}}\int_{\mathcal{A}} d^\pi(s,a)\log\frac{d^\pi(s,a)}{\bar{d}(s,a)}\,\mathrm{d}s\,\mathrm{d}a \le \epsilon \tag{3.24}$$

$$\int_{\mathcal{S}} d^\pi(s')\phi(s')\mathrm{d}s' = \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d^\pi(s,a)P(s'|s,a)\phi(s')\,\mathrm{d}s\,\mathrm{d}a\,\mathrm{d}s' \tag{3.25}$$

$$\int_{\mathcal{S}}\int_{\mathcal{A}} d^\pi(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1\,, \tag{3.26}$$

where $\phi : \mathcal{S} \to \mathbb{R}^k$ maps MDP states to stationary features under policy π. The constraint Eq. (3.25) is a consistency constraint that ensures the compatibility of the state action distribution d^π with the transition model P. The key difference to the Linear Programming

formulation (Section 2.2) lies in the use of Eq. (3.24) that bounds the Kullback-Leibler distance of the new distribution from the old one. The choice of ϵ depends on the problem and on the number of samples available.
REPS can act as a global maximizer if provided with an infinite number of samples and with an extremely large ϵ.

Dual The REMPS problem is solved considering the dual:

$$\min_{\theta,\eta} \eta \log \left(\int_{\mathcal{S}} \int_{\mathcal{A}} \bar{d}(s,a) \exp\left(\epsilon + \frac{1}{\eta} \delta_\theta(s,a) \right) \mathrm{d}s \, \mathrm{d}a \right) \tag{3.27}$$

$$\text{subject to } \eta \geq 0 \,, \tag{3.28}$$

where $\delta_\theta(s,a) = r(s,a) + \int_S P(s'|s,a)\theta^T\phi(s')\,ds' - \theta^T\phi(s)$ is the Bellman error. By a simple re-parametrization of the dual with $\hat{\eta} = \eta^{-1}$ we obtain a convex problem that can be efficiently solved by standard optimizers such as Broyden–Fletcher–Goldfarb–Shannon (BFGS).

Theorem 3.2.1 (REPS Solution) *The optimal policy solution of the REMPS problem defined above is:*

$$\pi(a|s) = \frac{\bar{d}(s,a) \exp(\frac{1}{\eta}\delta_\theta(s,a))}{\int_{\mathcal{A}} \bar{d}(s,a') \exp(\frac{1}{\eta}\delta_\theta(s,a'))\, da'} \,, \tag{3.29}$$

where η and θ are determined by the solution of the dual.

The interested reader is referred to (Peters et al., 2010) for the derivation of the solution. All components of REPS can be expressed by sample averages, thus the application of the algorithm to continuous state and action spaces is straightforward.

3.2.2 Parametric Policy

The solution (3.29) is possible if we have total freedom on the selection of the policy, that is when the policy space Π is unbounded or when the state and action spaces are finite. In order to deal with either continuous state space or continuous action space the distributions $d^\pi(\cdot)$ and $\pi(\cdot|s)$ need to be represented by parametric distributions (Daniel et al., 2012). The parametric distribution is obtained through Moment Projection of the distribution d^π, solution of the REPS problem, that we will denote as d for ease of notation. The Moments Projection is obtained by searching for $\hat{d}, \hat{\pi}$ minimizing the KL-divergence $D_{KL}(d||\hat{d}\hat{\pi})$. The optimization problem is the following:

$$\hat{d}\,, \hat{\pi} = \arg\min_{d',\pi'} \int_{\mathcal{S}} \int_{\mathcal{A}} d(s,a) \log \frac{d(s,a)}{d'(s)\pi'(a|s)} \,\mathrm{d}a\,\mathrm{d}s \tag{3.30}$$

$$= \arg\min_{d',\pi'} - \int_{\mathcal{S}} \int_{\mathcal{A}} d(s,a) \log \left(d'(s)\pi'(a|s) \right) \,\mathrm{d}a\,\mathrm{d}s \tag{3.31}$$

$$= \arg\min_{d',\pi'} - \int_{\mathcal{S}} \int_{\mathcal{A}} \bar{d}(s,a) \frac{d(s,a)}{\bar{d}(s,a)} \log \left(d'(s)\pi'(a|s) \right) \,\mathrm{d}a\,\mathrm{d}s \tag{3.32}$$

$$\approx \arg\min_{d',\pi'} - \sum_{(s,a)\in\mathcal{D}} \exp\left(\frac{1}{\eta}\delta_\theta(s,a)\right) \log\left(d'(s)\pi'(a|s)\right) \ , \tag{3.33}$$

where $\mathcal{D}$ is dataset collected with the distribution $\bar{d}$. The KL-divergence is estimated through importance sampling from samples coming from $\bar{d}$.

3.2.3 REPS extensions

Episode-based REPS

In (Kupcsik et al., 2013) an episode-based version of REPS that is equivalent to stochastic search was presented. This REPS extension employs a meta-policy that defines the distribution from which policy parameters are drawn. The KL–constraint it is then defined at the level of the meta-policy. The Episode-based REPS formulation is:

$$\max_p \sum_{s,\boldsymbol{\psi}} p(s,\boldsymbol{\psi}) G_{s,\boldsymbol{\psi}} \tag{3.34}$$

$$\text{subject to} \sum_{s,\boldsymbol{\psi}} p(s,\boldsymbol{\psi}) \log \frac{p(s,\boldsymbol{\psi})}{q(s,\boldsymbol{\psi})} < \epsilon \tag{3.35}$$

$$\sum_s p(s)\phi(s) = \hat{\phi}, \tag{3.36}$$

where parameters ψ are the parameters of the meta policy, $G_{s,\psi}$ is the full return starting from s with parameters ψ. The distribution p is the distribution over context s and meta policy parameters ψ. The Eq. (3.35) bounds the relative entropy with respect to the observed distribution q. The last constraint is needed for continuous or high dimensional discrete state spaces and matches feature averages instead of single probability values. In Eq. (3.36) $\phi(s)$ is the feature vector describing the context and $\hat{\phi}$ is the feature vector averaged over all observed contexts.

MORE

Model-Based Relative-Entropy Stochastic Search (Abdolmaleki et al., 2015) (MORE) uses the same episode-based formulation of the policy search together with a constraint lower-bounding the entropy of the new policy, enforcing exploration. Moreover MORE learns a quadratic surrogate model of the objective function to compute the solution analytically. MORE formulation is:

$$\max_\pi \int \pi(\boldsymbol{\theta}) G_{\boldsymbol{\theta}} \mathrm{d}\boldsymbol{\theta} \tag{3.37}$$

$$\text{subject to} \int_\Theta \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} < \epsilon \tag{3.38}$$

$$H(\pi) \geq \beta \tag{3.39}$$

$$\int_\Theta \pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} = 1, \tag{3.40}$$

where $G_{\boldsymbol{\theta}}$ denotes the expected return when evaluating parameter vector $\boldsymbol{\theta}$. The term $H(\pi) = -\int \pi(\boldsymbol{\theta}) \log \pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ denotes the entropy of the distribution π and $q(\boldsymbol{\theta})$ is the old distribution.

Trust Region Approaches

REPS can also be related to algorithms using a trust region approach. The trust region is the neighbourhood of the current policy (defined by some distance) such that we can have a reliable estimate of the performance.

TRPO Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) uses this concept together with off policy optimization. This lead to the following formulation:

$$\max_{\boldsymbol{\theta}'} \; L_{\boldsymbol{\theta}'}(\boldsymbol{\theta}) \tag{3.41}$$

$$\text{subject to } \bar{D}^{d_{\boldsymbol{\theta}}}_{KL}(\boldsymbol{\theta}, \boldsymbol{\theta}') \leq \epsilon \,, \tag{3.42}$$

where $\boldsymbol{\theta}$ are the current policy parameter, $\boldsymbol{\theta}'$ are the new policy parameter, $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$ is a surrogate of the performance of the new policy estimated under the old policy and $\bar{D}^{d^{\boldsymbol{\theta}}}_{KL}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ is the expected KL–divergence between the new and the old policy. The theory justifying TRPO actually suggests using a penalty instead of a constraint, i.e., solving the unconstrained optimization problem:

$$\max_{\boldsymbol{\theta}'} \; L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') - \beta D^{\max}_{KL}(\boldsymbol{\theta}, \boldsymbol{\theta}') \,, \tag{3.43}$$

where $D^{\max}_{KL}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \max_s D^{\max}_{KL}(\pi_{\boldsymbol{\theta}}(\cdot|s), \pi_{\boldsymbol{\theta}'}(\cdot|s))$. TRPO uses a hard constraint rather than a penalty because it is hard to choose a single value of β that performs well across different problems or even within a single problem, where the characteristics change over the course of learning.

PPO Proximal Policy Optimization (PPO) (Schulman et al., 2017) implements the concept of trust region by using a clipped objective the prevents excessively large policy updates:

$$L^{clip}_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \mathop{\mathbb{E}}_{\substack{s \sim d^\pi(\cdot) \\ a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)}} \left[\min\left(w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(a|s), \text{clip}\left(w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(a|s), 1-\epsilon, 1+\epsilon \right) \right) A_{\pi_{\boldsymbol{\theta}}}(s,a) \right] \,, \tag{3.44}$$

where $A_\pi(s,a)$ is the advantage function and $w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(a|s) = \frac{\pi_{\boldsymbol{\theta}'(a|s)}}{\pi_{\boldsymbol{\theta}(a|s)}}$ is the importance weight associated with action a in state s.

POIS Policy Optimization via Importance Sampling (POIS) (Metelli et al., 2018b) is a model–free, actor–only, policy optimization algorithm, that mixes online and offline optimization to efficiently exploit the information contained in the collected trajectories. POIS explicitly accounts for the uncertainty introduced by the importance sampling by optimizing a surrogate objective function. The latter captures the trade-off between the estimated performance improvement and the variance injected by the importance sampling. The POIS algorithm uses as penalization the Rényi divergence (Rényi, 1961; Van Erven and Harremos, 2012), an information–theoretic dissimilarity index between probability measures. The surrogate objective function proposed in action-based POIS is:

$$\mathcal{L}_{\boldsymbol{\theta}}^{\text{A-POIS}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p(\cdot|\boldsymbol{\theta})} \left[w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(\tau_i) R(\tau_i) \right] - \lambda \sqrt{\frac{\widehat{d}_2\left(p(\cdot|\boldsymbol{\theta}') \| p(\cdot|\boldsymbol{\theta})\right)}{N}}, \tag{3.45}$$

where $w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(\tau_i)$ is the importance weights associated to the trajectory τ_i, $w_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(\tau_i) = \frac{p(\tau_i|\boldsymbol{\theta}')}{p(\tau_i|\boldsymbol{\theta})} = \prod_{t=0}^{H-1} \frac{\pi_{\boldsymbol{\theta}'}(a_{\tau_i,t}|s_{\tau_i,t})}{\pi_{\boldsymbol{\theta}}(a_{\tau_i,t}|s_{\tau_i,t})}$, and the term $\widehat{d}_2\left(p(\cdot|\boldsymbol{\theta}') \| p(\cdot|\boldsymbol{\theta})\right)$ is the approximated Rényi divergence between the new stationary distribution and the old one:

$$\widehat{d}_2\left(p(\cdot|\boldsymbol{\theta}') \| p(\cdot|\boldsymbol{\theta})\right) = \frac{1}{N} \sum_{i=1}^{N} \prod_{t=0}^{H-1} d_2\left(\pi_{\boldsymbol{\theta}'}(\cdot|s_{\tau_i,t}) \| \pi_{\boldsymbol{\theta}}(\cdot|s_{\tau_i,t})\right).$$

POIS has a strong theoretical grounding, since its surrogate objective function derives from a statistical bound on the estimated performance, that is able to capture the uncertainty induced by importance sampling. The experimental evaluation in (Metelli et al., 2018b) shows that POIS is able to achieve a performance comparable with TRPO and PPO.

Chapter 4

Relative Entropy Model Policy Search

Our beliefs are based on our experience, which gives us a very incomplete picture of the world, and it's easy to jump to false conclusions.

Pedro Domingos, The Master Algorithm

In this chapter, we will present a novel *information theoretic* approach to solve CMDPs, namely *Relative Entropy Model Policy Search* (REMPS). REMPS is an extension of REPS (see Section 3.2) for the case of model-policy learning, it is able to overcome local maxima/minima and provides an exact update step. REMPS formulates the learning problem as a constrained optimization problem for which we find the solution in closed form.
In Section 4.2 we formalize the optimization problem, we derive the solution and we propose several projection strategies able to work with continuous state and action spaces. In Section 4.3 we propose a theoretical study of the REMPS property, providing a bound on the difference of performance between the ideal formulation of REMPS and the samples approximation formulation.

4.1 Motivations

When an agent interacts with the environment there is a tight connection between the policy and the model configuration. In standard RL applications, where the environment is assumed to be fixed during the learning process, it is common to treat model configuration as instance of hyperparameters selection problem. In this case it is possible to use standard techniques like Bayesian Optimization. Using these techniques we are solving a slightly different problem with respect to the CMDP learning problem (Section 3.1) that is defined as:

$$P^*, \pi^* \in \operatorname*{arg\,max}_{P \in \mathcal{P}_{\boldsymbol{\omega}}, \pi \in \Pi_{\boldsymbol{\theta}}} J^{P,\pi}.$$

If the model configuration is selected before the training phase we are actually searching for the best policy given a model. If the model configuration is selected after the training process we are searching for the best model for a given policy. In general a model configuration induces an optimal policy and a policy induces an optimal configuration but the pairs model-policy found in these ways are, in general, different from the pair model-policy yielding the optimal performance. For this reasons treating model parameters as hyperparameters it is not the correct way to proceed. Furthermore the model-policy learning problem is a different problem with respect to the ones usually solved in the literature.
In principle we can act with gradient methods over the policy and the model (see Section 2.4 and Appendix B) but we argue that this is not a smart way to tackle the CMDP probelm. Gradient methods tackle local optima in the return landscape by using policies with a huge number of parameters, multiple restarts and stochastic gradient ascent updates. However, these methods for avoiding local optima are not theoretically justified. Moreover, while a policy can have an arbitrarily large number of parameters, in the context of CMDP the model parameters are fixed given a problem and usually the cardinality of model parameters is much smaller with respect to the cardinality of policy parameters. Moreover multiple restarts using different values for the model parameters might cause dangerous behaviours.
We describe the rationale behind the choice of an information theoretic approach with an example. Suppose the agent found itself in the model $P_{\boldsymbol{\omega}_0}$ with the best policy $\pi_{\boldsymbol{\theta}^*(\boldsymbol{\omega}_0)}$ for that model. A good choice for the update of the model is $P_{\boldsymbol{\omega}_1}$ such that the learning process starting from $\pi_{\boldsymbol{\theta}^*(\boldsymbol{\omega}_0)}$ in $P_{\boldsymbol{\omega}_1}$ yields good results. In other words the update of a component (model) of a CMDP should take into account future updates of the other (policy). By acting with gradient updates we do not obtain this behaviour: the gradient of the performance in $(P_{\boldsymbol{\omega}_0}, \pi_{\boldsymbol{\theta}^*(\boldsymbol{\omega}_0)})$ with respect to the policy parameters yields zero value (since it is a local optima) and the gradient with respect to the model parameters points towards the direction where the performance of the *current* policy are improved. As said before, we do not want to consider the current policy in the model update, we should consider future policies.
The two issues of gradient methods explained above can be solved by information theoretic approaches. Our REMPS formulation, as we will see later, considers jointly the model and the policy yielding an exact model-policy update in the neighbourhood of the current model-policy pair. REMPS is based on two phases: *optimization* and *projection*. In the optimization phase we seek for the best stationary distribution d (in the space of *all* possible distribution) that is not far from the current distribution more than $\epsilon > 0$ in terms of KL-divergence. The distribution d, might fall outside the space of representable stationary distributions given our model and policy spaces. Therefore, like in (Daniel et al., 2012), in the *projection* phase we perform a projection onto the representable space finding the actual distribution $\widehat{d}$.

4.2 Relative Entropy Model Policy Search

In this section we present the optimization phase and the projection phase of REMPS. We consider a CMDP with parametric model and policy spaces. Model parameters are denoted with $\boldsymbol{\omega}$ and policy parameters with $\boldsymbol{\theta}$. The performance of a configuration-policy pair (P, π)

is defined in terms of the long-term expected reward:

$$J^{P,\pi} = \liminf_{H \to +\infty} \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim P(\cdot|s_t,a_t)}} \left[\frac{1}{H} \sum_{t=0}^{H-1} R(s_t, a_t, s_{t+1}) \right].$$

We express the state-action-next-state stationary distribution as:

$$d^{P,\pi}(s,a,s') = d^{P,\pi}(s)\pi(a|s)P(s'|s,a).$$

4.2.1 Optimization

We define the following constrained optimization problem:

$$\max_{d} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s') R(s,a,s') \mathrm{d}s \mathrm{d}a \mathrm{d}s' \tag{4.1}$$

subject to: (4.2)

$$D_{KL}(d || d^{P,\pi}) \leq \epsilon \tag{4.3}$$

$$\int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s') \mathrm{d}s \mathrm{d}a \mathrm{d}s' = 1\,, \tag{4.4}$$

where $d^{P,\pi}(\cdot,\cdot,\cdot)$ is the sampling distribution and ϵ is a parameter controlling how large a model-policy update can be.
This formulation is derived from the REPS formulation but it takes into account also the model parameters by considering the joint distribution obtained by the model and the policy. The objective is the maximization of the average reward. The KL–constraint prevents the optimized joint distribution from moving too far from the sampling distribution. The last constraint is only needed to ensure that the obtained distribution is well formed. It is worth noting that, differently from REPS, we do not impose a constraint on the validity of the stationary distribution with respect to to the transition model, as we have the possibility to change it, configuring the environment.

REMPS Dual The dual problem is given by:

$$\min_{\eta \in [0,+\infty)} g(\eta) = \eta \log \left(\int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d^{P,\pi}(s,a,s') \exp\left(\epsilon + \frac{R(s,a,s')}{\eta} \right) \mathrm{d}s \mathrm{d}a \mathrm{d}s' \right), \tag{4.5}$$

which is convex in η^{-1}.

In real cases we do not have access to the real sampling distribution $d^{P,\pi}$, so we cannot compute the exact solution of the dual problem. Like in REPS, all components of REMPS can be estimated from samples. The dual function can be rewritten as:

$$g(\eta) = \eta \log \mathop{\mathbb{E}}_{(s,a,s') \sim d^{P,\pi}(\cdot,\cdot,\cdot)} \left[\exp\left(\epsilon + \frac{R(s,a,s')}{\eta} \right) \right] \tag{4.6}$$

$$\approx \eta \log \left[\frac{1}{N} \sum_{(s,a,s')\in D} \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right)\right], \tag{4.7}$$

where D is dataset of triples (s, a, s') collected with the distribution $d^{P,\pi}$. From this formulation it is easy to obtain a sample estimation of the dual since it is the empirical mean of the above exponential function with samples coming from the the current model P and the current policy π. We refer to the approximated dual function with $\widetilde{g}$.
The following theorem derives the solution of the constrained optimization problem in terms of the sampling distribution $d^{P,\pi}$, the reward associated to each sample $R(s, a, s')$, the Lagrange multiplier η and the ϵ parameter.

Theorem 4.2.1 (REMPS solution) *The solution of the REMPS problem is:*

$$d(s,a,s') = \frac{d^{P,\pi}(s,a,s') \exp\left(\frac{R(s,a,s')}{\eta}\right)}{\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d^{P,\pi}(s,a,s') \exp\left(\frac{R(s,a,s')}{\eta}\right) \mathrm{d}s\mathrm{d}a\mathrm{d}s'} \tag{4.8}$$

which is induced by the optimal model and policy:

$$\pi'(a|s) = \frac{\pi(a|s) \int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right) \mathrm{d}s'}{\int_{\mathcal{A}} \pi(a|s) \int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right) \mathrm{d}s'\mathrm{d}a} \tag{4.9}$$

$$P'(s'|a,s) = \frac{P(s'|s,a) \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right)}{\int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right) \mathrm{d}s'}. \tag{4.10}$$

where η is the minimizer of the dual problem 4.5.

The proof is given in Appendix C.

4.2.2 Projection

The REMPS problem is solved in closed form but the solution might lie outside the space of the feasible models and policies. Therefore there is the need to obtain a solution that can be represented.
We consider the case in which both the model and the policy are parametric: $\mathcal{P}_\Omega = \{P_{\boldsymbol{\omega}} : \boldsymbol{\omega} \in \Omega \subseteq \mathbb{R}^k\}$ and $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p\}$, the space of joint distributions induced by $\mathcal{P}_\Omega$ and Π_Θ is $\mathcal{D}_{\Omega,\Theta} = \{d^{P,\pi} : P \in \mathcal{P}_\Omega, \pi \in \Pi_\Theta\}$. We denote with $d^{\boldsymbol{\omega},\boldsymbol{\theta}}$ a distribution belonging to $\mathcal{D}_{\Omega,\Theta}$ to highlight the dependence on the model and policy parameters.
We decide to perform a *moment projection* searching, inside the space $\mathcal{D}_{\Omega,\Theta}$, for the solution that best represents d, the solution of the optimization problem.
This decision is based on the following bound, relating the performance gap of two distributions to the KL–divergence.

Theorem 4.2.2 (Joint bound) *Let us denote with $d^{P,\pi}$ the stationary distribution induced by the model P and policy π and $d^{P',\pi'}$ the stationary distribution induced by the model P'*

and policy π'. Let us assume that the reward is uniformly bounded, that is for $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ it holds that $|R(s,a,s')| < R_{\max}$. The norm of the difference of performance can be upper bounded as:

$$|J^{P,\pi} - J^{P',\pi'}| \leq R_{\max}\sqrt{2D_{KL}(d^{P,\pi}\|d^{P',\pi'})}. \tag{4.11}$$

Proof. Starting from the definition of the difference of the performance:

$$\begin{aligned}
|J^{P,\pi} - J^{P',\pi'}| &\leq \left|\int_{\mathcal{X}} d^{P,\pi}(x)R(x)\mathrm{d}x - \int_{\mathcal{X}} d^{P',\pi'}(x)R(x)\mathrm{d}x\right| && (4.12)\\
&\leq R_{\max}\left|\int_{\mathcal{X}} d^{P,\pi}(x) - d^{P',\pi'}(x)\mathrm{d}x\right| && (4.13)\\
&\leq R_{\max}\,TV(d^{P,\pi}, d^{P',\pi'}) && (4.14)\\
&\leq R_{\max}\sqrt{2D_{KL}(d^{P,\pi}\|d^{P',\pi'})}, && (4.15)
\end{aligned}$$

where we denote with $\mathcal{X}$ the space $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and with x a tuple (s,a,s') for ease of notation. Eq. (4.14) follows from the definition of Total Variation Distance and Eq. (4.15) follows from the Pinsker inequality. □

The previous bound relates the difference of performance between the stationary distributions to the KL–divergence between the two distributions, considering jointly the effect of the model and the policy. Now we derive a similar bound, that uses corollary 3.1 of (Metelli et al., 2018a) and its extension to the case of undiscounted reward.

Theorem 4.2.3 (Disjoint bound) *Let us denote with $d^{P,\pi}$ the stationary distribution induced by the model P and policy π, $d^{P',\pi'}$ the stationary distribution induced by the model P' and policy π'. Let us assume that the reward is uniformly bounded, that is for $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ it holds that $|R(s,a,s')| < R_{\max}$. If (P',π') admits group invertible state kernel $P'^{\pi'}$ the norm of the difference of performance can be upper bounded as:*

$$|J^{P,\pi} - J^{P',\pi'}| \leq R_{\max}\, c_1 \mathbb{E}_{s,a\sim d^{P,\pi}}\left[\sqrt{2D_{KL}(\pi'\|\pi)} + \sqrt{2D_{KL}(P'\|P)}\right], \tag{4.16}$$

where $c_1 = 1 + ||A'^{\#}||_\infty$ and $A'^{\#}$ is the group inverse of the state kernel $P'^{\pi'}$.

Proof. Starting from the definition of the difference of the performance:

$$\begin{aligned}
|J^{P,\pi} - J^{P',\pi'}| &\leq \left|\int_{\mathcal{X}} d^{P,\pi}(x)R(x)\mathrm{d}x - \int_{\mathcal{X}} d^{P',\pi'}(x)R(x)\mathrm{d}x\right| && (4.17)\\
&\leq R_{\max}\left|\int_{\mathcal{X}} d^{P,\pi}(x) - d^{P',\pi'}(x)\mathrm{d}x\right| && (4.18)\\
&\leq R_{\max}\,TV(d^{P,\pi}, d^{P',\pi'}) && (4.19)\\
&\leq R_{\max}\, c_1 \mathbb{E}_{s,a\sim d^{P,\pi}}\left[\sqrt{2D_{KL}(\pi'\|\pi)} + \sqrt{2D_{KL}(P'\|P)}\right] && (4.20)
\end{aligned}$$

where we denote with $\mathcal{X}$ the space $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and with x a tuple (s,a,s') for ease of notation. Eq. (4.19) follows from the definition of Total Variation Distance and Eq. (4.20) follows from the Pinsker inequality. □

We propose three techniques to project the REMPS solution over the space of the feasible distributions $\mathcal{D}_{\Omega,\Theta}$.

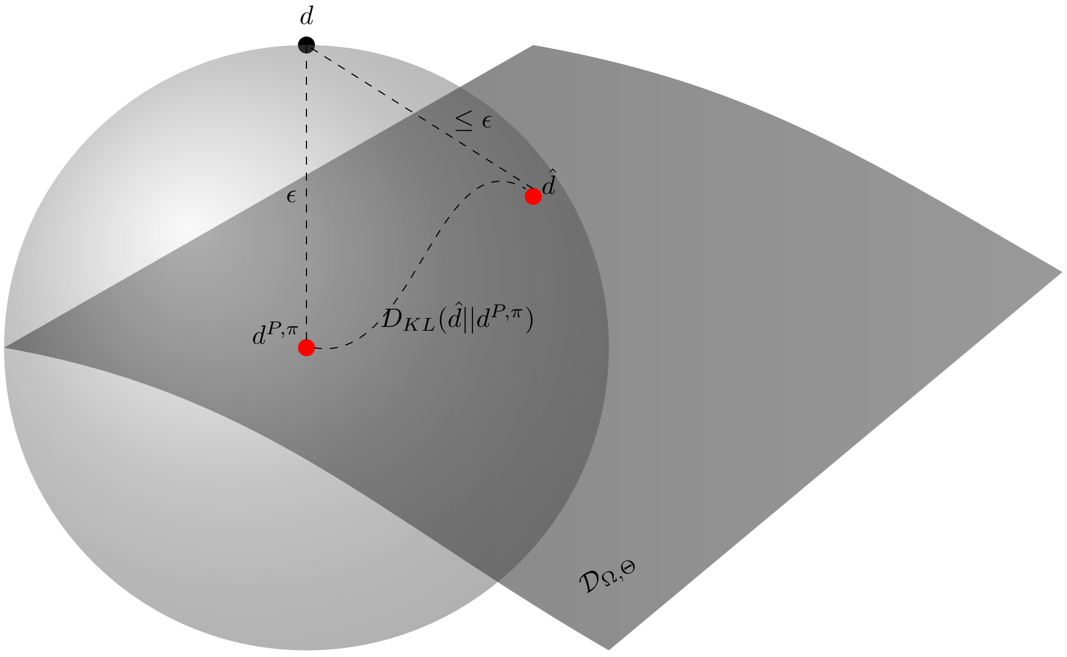


Figure 4.1: Illustration of REMPS and information projection. The ball centred in $d^{P,\pi}$ with radius ϵ represents the KL constraint over the space of distributions. The surface $\mathcal{D}_{\Omega,\Theta}$ represents the space of available distributions. Distances are not euclidean since measured with the KL divergence.

Projection of the stationary distribution

In principle, as REMPS works on the stationary distribution, we should project it directly, i.e., finding the parameters that induce the most similar stationary distribution. The projection problem is:

$$\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}} = \underset{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega}{\arg\min}\, D_{KL}\left(d(s,a,s')\|d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s,a,s')\right)$$

$$s.t.\ d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s) = \int_{\mathcal{S}}\int_{\mathcal{A}} d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s')\pi_{\boldsymbol{\theta}}(a|s')P_{\boldsymbol{\omega}}(s'|s,a)\mathrm{d}a\mathrm{d}s'.$$

However this problem is impractical as the constraint is difficult to enforce in continuous state-spaces even if replaced with the matching of the expectations of some features. Differently, in finite state-action spaces it is possible to enforce its sample based version by introducing a variable for each $d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s)$.

Projection of the state kernel P^π

The projection of the state kernel P^π is a relaxed version with respect to the stationary distribution projection. We minimize the expected KL–divergence between P^π of the distribution d recovered by REMPS and $P^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}$, that is the state kernel distribution defined by our parametric space:

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}} &= \arg\min_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}} d(s) D_{KL}(P^{\pi}(\cdot|s) \| P^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(\cdot|s)) \mathrm{d}s \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}} d(s) \int_{\mathcal{S}} P^{\pi}(s'|s) \log P^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(s'|s) \mathrm{d}s' \mathrm{d}s \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}} d(s) \int_{\mathcal{S}} \int_{\mathcal{A}} \pi'(a|s) P'(s'|s,a) \log P^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(s'|s) \mathrm{d}s' \mathrm{d}s \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s') \log \int_{\mathcal{A}} P_{\boldsymbol{\omega}}(s'|s,a') \pi_{\boldsymbol{\theta}}(a'|s) \mathrm{d}a' \mathrm{d}s \mathrm{d}a \mathrm{d}s'
\end{aligned}$$

The sample-based version requires to compute the state kernel from samples, this can be done for the case of finite actions (and possibly continuous state-space):

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}} &= \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \sum_{(s,a,s')\in\mathcal{D}} \frac{d(s,a,s')}{d^{P,\pi}(s,a,s')} \log \sum_{a'\in\mathcal{A}} P_{\boldsymbol{\omega}}(s'|s,a') \pi_{\boldsymbol{\theta}}(a'|s) \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} \sum_{(s,a,s')\in\mathcal{D}} \exp\left(\frac{R(s,a,s')}{\eta}\right) \log \sum_{a'\in\mathcal{A}} P_{\boldsymbol{\omega}}(s'|s,a') \pi_{\boldsymbol{\theta}}(a'|s),
\end{aligned}$$

where $\mathcal{D}$ is a dataset collected with the distribution $d^{P,\pi}$ and we use importance sampling in order to recover the expected value under the distribution d from samples coming from the sampling distribution $d^{P,\pi}$ and we rewrite the ratio $d/d^{P,\pi}$ as:

$$\frac{d(s,a,s')}{d^{P,\pi}(s,a,s')} \propto \exp\left(\frac{R(s,a,s')}{\eta}\right),$$

neglecting a constant.

Projection of policy and model independently

In this simpler version of the projection problem we minimize the expected KL–divergence between the distributions π' and P' and their corresponding parametric distributions $\pi_{\boldsymbol{\theta}}$ and $P_{\boldsymbol{\omega}}$.

$$\begin{aligned}
\widehat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}\in\Theta} \int_{\mathcal{S}} d(s) D_{KL}(\pi'(\cdot|s) \| \pi_{\boldsymbol{\theta}}(\cdot|s)) \mathrm{d}s = && (4.21) \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \int_{\mathcal{S}} d(s) \int_{\mathcal{A}} \pi'(a|s) \log \frac{\pi(a|s)}{\pi_{\boldsymbol{\theta}}(a|s)} \mathrm{d}a \mathrm{d}s = && (4.22) \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s') \log \frac{\pi'(a|s)}{\pi_{\boldsymbol{\theta}}(a|s)} \mathrm{d}s \mathrm{d}a \mathrm{d}s' = && (4.23) \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s') \log \pi_{\boldsymbol{\theta}}(a|s) \mathrm{d}s \mathrm{d}a \mathrm{d}s', && (4.24)
\end{aligned}$$

that can be estimated from samples:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{(s,a,s')\in\mathcal{D}} \exp\left(\frac{R(s,a,s')}{\eta}\right) \log \pi_{\boldsymbol{\theta}}(a|s), \qquad (4.25)$$

where $\mathcal{D}$ is a dataset collected with $d^{P,\pi}$. Symmetrically, for the transition model:

$$\begin{aligned}\widehat{\boldsymbol{\omega}} &= \arg\min_{\boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}}\int_{\mathcal{A}} d(s,a) D_{KL}(P'(\cdot|s,a), P_{\boldsymbol{\omega}}(\cdot|s,a))\mathrm{d}s\mathrm{d}a = \\ &= \arg\max_{\boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d(s,a,s') \log P_{\boldsymbol{\omega}}(s'|s,a)\mathrm{d}s\mathrm{d}a\mathrm{d}s',\end{aligned}$$

and the sample based optimization works as for the policy. This optimization problem can be easily solved in a sample-based form for both finite and continuous state and action spaces.

4.2.3 Model Approximation

In the previous sections we presented REMPS in the case of known environment. However, the perfect knowledge of the environment is difficult, or even impossible, in practice. Even in cases where an environment model is available it might be too approximate or hardy usable being very complex and computationally expansive.
In REMPS it is possible to use any model approximation method, the only requirement is that it must be possible to learn the mapping $(s, a, \boldsymbol{\omega}) \to s'$, from state, action and environment configuration to a distribution over the next states. Notice that in REMPS the environment model is used only the projection phase, while the optimization phase is model-free.
In order to learn an environment approximation we use a maximum likelihood (ML) approach. We collect a dataset composed by tuples $(s, a, \boldsymbol{\omega}, s')$ and given a parametric model defining a distribution over the state space we find the parameters by maximizing the log-likelihood of the seen transitions. The parametric model can be a Gaussian Process, a Neural Network or some other function approximator model. We denote the model approximation with $\widetilde{P}$.

4.2.4 Discussion

In this section we discuss the main benefits and limitations of REMPS. Being an information theoretic approach, REMPS, has the main goal of maximizing the performance while staying close to the observed data. This translates into the KL constraint between the sampling distribution $d^{P,\pi}$ and the optimized distribution d. REMPS considers jointly the effect of the two CMDP components and finds a distribution, in the space of all possible distributions, maximizing the average reward while satisfying the KL constraint using a primal-dual formulation. However, due to a possible limitation in the representation power of the model and the policy this distribution might be unfeasible. Notice that a lack in representation power of the policy can be easily addressed (e.g. using more parameters), while a limited representation power in the model has to be expected since the number of parameters and their influence is fixed given a task. REMPS solves this problem by performing a KL–projection, that is finding the model-policy pair minimizing the KL distance between their joint distribution $\hat{d}$ and d.

Having access to infinite samples we are sure that d is close the sampling distribution, that is $D_{KL}(d \| d^{P,\pi}) \leq \epsilon$. This constraint might be, in practice, not satisfied due to wrong estimations. We also highlight the fact that performing the Moment Projection we can actually find a suboptimal distribution. By a simple inspection we can easily prove that $D_{KL}(d \| \widehat{d}) \leq \epsilon$. However, being the KL–divergence not symmetric, we have no insight on $D_{KL}(\widehat{d} \| d^{P,\pi})$ that is a more relevant quantity.
In practice we can add a regularization in the projection phase penalizing some type of distance between the new model-policy parameters and the sampling parameters, even if this is not theoretically justified.
The REMPS pseudocode is reported in Algorithm 6.

Algorithm 6 Relative Entropy Model Policy Search

```
Input: $\epsilon$: KL constraint, $P$: model.
1: Initialize $\pi_{\boldsymbol{\theta}_0}, P_{\boldsymbol{\omega}_0}$ randomly
2: for t = 0,1,... until convergence do
3:     Collect samples from $\pi_{\boldsymbol{\theta}_t}, P_{\boldsymbol{\omega}_t}$
4:     Obtain $\eta^*$, the minimizer of the dual problem:
```

$$\eta^* = \min_{\eta\in[0,+\infty)} g(\eta) \tag{4.26}$$

$$= \min_{\eta\in[0,+\infty)} \eta \log\left(\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d^{P,\pi}(s,a,s') \exp\left(\epsilon + \frac{R(s,a,s')}{\eta}\right) \mathrm{d}s\mathrm{d}a\mathrm{d}s'\right). \tag{4.27}$$

```
5:     Project the optimal distribution $d$ onto $\mathcal{D}_{\Omega,\Theta}$ according to the projection strategy.
   a. Projection of the stationary distribution:
```

$$\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}} = \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} D_{KL}(d(s,a,s') \| d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s,a,s'))$$

$$s.t.\ d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s) = \int_{\mathcal{S}}\int_{\mathcal{A}} d^{\boldsymbol{\omega},\boldsymbol{\theta}}(s')\pi_{\boldsymbol{\theta}}(a|s')P_{\boldsymbol{\omega}}(s'|s,a)\mathrm{d}s'\mathrm{d}a.$$

```
   b. Projection of the state kernel:
```

$$\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}} = \operatorname*{arg\,max}_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d(s,a,s') \log \int_{\mathcal{A}} P_{\boldsymbol{\omega}}(s'|s,a')\pi_{\boldsymbol{\theta}}(a'|s)\mathrm{d}a'\mathrm{d}s\mathrm{d}a\mathrm{d}s'.$$

```
   c. Projection of policy and model independently:
```

$$\widehat{\boldsymbol{\theta}} = \operatorname*{arg\,max}_{\boldsymbol{\theta}\in\Theta} \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d(s,a,s') \log \pi_{\boldsymbol{\theta}}(a|s)\mathrm{d}s\mathrm{d}a\mathrm{d}s' \tag{4.28}$$

$$\widehat{\boldsymbol{\omega}} = \operatorname*{arg\,max}_{\boldsymbol{\omega}\in\Omega} \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} d(s,a,s') \log P_{\boldsymbol{\omega}}(s'|s,a)\mathrm{d}s\mathrm{d}a\mathrm{d}s'. \tag{4.29}$$

```
6:     Update policy: $\boldsymbol{\theta}_{t+1} \leftarrow \widehat{\boldsymbol{\theta}}$
7:     Update model: $\boldsymbol{\omega}_{t+1} \leftarrow \widehat{\boldsymbol{\omega}}$
8: end for
9: return Policy-Model Pair $(\pi_{\boldsymbol{\theta}_t}, P_{\boldsymbol{\omega}_t})$
```

4.3 Theoretical Analysis

In this section we derive some theoretical guarantees for the single step of REMPS when it is executed starting from a finite number of samples N. This analysis is based on (Cortes et al., 2010).

Notation Let d be a stationary distribution, we will denote with $\mathcal{D}_d = \left\{ d' = \frac{d \exp\left(\frac{1}{\eta} r\right)}{\int d \exp\left(\frac{1}{\eta} r\right)} : \eta \in [0, +\infty) \right\}$. Given a policy hypothesis space Π and a transition model hypothesis space $\mathcal{P}$, we will denote with $\mathcal{D}_{\mathcal{P},\Pi} = \{d_{P,\pi} : P \in \mathcal{P}, \pi \in \Pi\}$ the set of stationary distributions induced by the policy and model hypothesis spaces. For the sake of brevity we will denote with $\mathcal{X} = \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ the state-action-next state space and with $x = (s, a, s')$ a state-action-next-state triple. Given a set $\mathcal{X}$, we denote with $\Delta(\mathcal{X})$ the set of probability distributions on $\mathcal{X}$.

4.3.1 Problem Formulation

In this section we revisit the REMPS problem formulation in order to understand better its component to perform a theoretical analysis. The REMPS problem takes as input a KL-divergence threshold and provides as output a new stationary distribution on the space $\mathcal{D}_{\mathcal{P},\Pi}$. As we said before, our algorithm is divided in two phases: *optimization* and *projection*.

Optimization The problem we aim to solve in the optimization phase can be stated as follows. Given a KL-divergence threshold $\epsilon > 0$, let $(P, \pi) \in \mathcal{P} \times \Pi$ be the current configuration-policy pair inducing a stationary distribution $d^{P,\pi}$, we seek for a new stationary distribution d that solves the optimization problem $\text{OPT}_{P,\pi}(\epsilon)$:

$$\max_{d \in \Delta\mathcal{X}} J_d = \mathop{\mathbb{E}}_{x \sim d} [R(x)] \tag{4.30}$$

$$\text{s.t.} \;\; D_{KL}(d || d^{P,\pi}) = \mathop{\mathbb{E}}_{x \sim d} \left[\log \frac{d(x)}{d^{P,\pi}(x)} \right] \leq \epsilon. \tag{4.31}$$

The solution of $\text{OPT}_{P,\pi}(\epsilon)$ is:

$$d(x) = \frac{d^{P,\pi}(x) \exp\left(\frac{1}{\eta} R(x)\right)}{\int_{\mathcal{X}} d^{P,\pi}(x) \exp\left(\frac{1}{\eta} R(x)\right) \mathrm{d}x}, x \in \mathcal{X}, \tag{4.32}$$

where η is the unique solution of the dual problem $\text{DUAL}_{P,\pi}(\epsilon)$:

$$\min_{\eta \in [0, +\infty)} \eta \log \mathop{\mathbb{E}}_{x \sim d^{P,\pi}} \exp\left(\frac{1}{\eta} R(x) + \epsilon\right). \tag{4.33}$$

In practice we have access neither to $d^{P,\pi}$ nor to d. Therefore we need to estimate expectations with samples averages from samples collected with the distribution $d^{P,\pi}$. Notice that we have only access to an empirical estimate of $d^{P,\pi}$, which is $\widehat{d}^{P,\pi} = \frac{1}{N}\sum_{i=1}^{N}\delta(x - x_i)$ uniform on the seen xs. For this purpose we perform an importance sampling procedure. We define the weights $w(x_i) = \frac{d(x_i)}{\widehat{d}^{P,\pi}(x_i)} = Nd(x_i)$. Thus, the approximated problem we need to solve is actually $\widetilde{\text{OPT}}_{P,\pi}(\epsilon)$:

$$\max_{d \in \Delta(\{x_i : i \in \{1,2,...,N\}\})} \widetilde{J}_d = \frac{1}{N}\sum_{i=1}^{N} w(x_i)R(x_i) = \sum_{i=1}^{N} d(x_i)R(x_i) \tag{4.34}$$

$$\text{s.t. } \widetilde{D}_{KL}(d\|d^{P,\pi}) = \frac{1}{N}\sum_{i=1}^{N} w(x_i)\log w(x_i) = \tag{4.35}$$

$$= \sum_{i=1}^{N} d(x_i)\log d(x_i) + \log N \le \epsilon. \tag{4.36}$$

This problem yields a solution which is defined only over the seen state-action-next state triples:

$$d(x_i) = \frac{\exp\left(\frac{1}{\widetilde{\eta}}R(x_i)\right)}{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{\widetilde{\eta}}R(x_j)\right)}, \quad i \in \{1, 2, ..., N\}, \tag{4.37}$$

where $\widetilde{\eta}$ is the unique solution of the approximated dual problem $\widetilde{\text{DUAL}}_{P,\pi}(\epsilon)$:

$$\min_{\eta \in [0,+\infty)} \eta \log \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\widetilde{\eta}}R(x_i) + \epsilon\right). \tag{4.38}$$

Once solved this problem, the new distribution over the whole space $\mathcal{X}$ is characterized only by the Lagrange multiplier $\widetilde{\eta}$:

$$\widetilde{d}(x) = \frac{d^{P,\pi}(x)\exp\left(\frac{1}{\widetilde{\eta}}R(x)\right)}{\int_{\mathcal{X}} d^{P,\pi}(x)\exp\left(\frac{1}{\widetilde{\eta}}R(x)\right)}, \; x \in \mathcal{X}. \tag{4.39}$$

We denote the performance of the new stationary distribution $\widetilde{d}$ as with $J_{\widetilde{d}} = \mathbb{E}_{x\sim\widetilde{d}}[R(x)]$. As expected we would like find $\widetilde{d}$ maximizing $J_{\widetilde{d}}$ but we actually use an empirical estimate $\widetilde{J}_{\widetilde{d}}$.

Projection In the *Projection* phase we aim to find the best representation of the stationary distribution we got from the optimization phase given hypothesis space $\mathcal{D}_{\mathcal{P},\Pi}$. Let d be the solution of $\text{OPT}_{P,\pi}(\epsilon)$, the projection problem $\text{PROJ}_{\mathcal{D}_{P,\Pi}}(d)$ can be stated as the moment-projection of d onto $\mathcal{D}_{\mathcal{P},\Pi}$:

$$\max_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} H(d\|d') = \mathop{\mathbb{E}}_{x\sim d}\left[\log d'(x)\right], \tag{4.40}$$

where we considered the cross-entropy $H(d\|d')$ instead of the KL–divergence, since $D_{KL}(d\|d') = H(d\|d') - H(d)$ and the entropy term $H(d)$ is independent on d'. We call d' the solution of this problem, which can be considered the solution of the complete problem REMPS$_{P,\pi}(\epsilon)$ = PROJ$_{\mathcal{D}_{\mathcal{P},\Pi}}(\cdot) \circ$ OPT$_{P,\pi}(\epsilon)$. Clearly, also in the projection phase we need to consider the Monte Carlo estimates again obtained by the very same samples $\{x_i\}_{i=1}^N$ collected with the sampling distribution $d^{P,\pi}$. Let $\widetilde{d}$ be the solution of $\widetilde{\text{OPT}}_{P,\pi}(\epsilon)$, the approximated projection problem $\widetilde{\text{PROJ}}_{\mathcal{D}_{\mathcal{P},\Pi}}(\widetilde{d})$ becomes:

$$\max_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} \widetilde{H}(\widetilde{d} \| d') = \frac{1}{N} \sum_{i=1}^{N} w(x_i) \log d'(x_i). \tag{4.41}$$

We call $\widetilde{d}'$ the solution of this problem, which can be considered the solution of the complete approximated problem $\widetilde{REMPS}_{P,\pi}(\epsilon) = \widetilde{\text{PROJ}}_{\mathcal{D}_{\mathcal{P},\Pi}}(\cdot) \circ \widetilde{\text{OPT}}_{P,\pi}(\epsilon)$.

Off-distribution estimation Given a value of the Lagrange multiplier η inducing the distribution d, let us define the ratio importance weight $\widehat{w}(x)$ and the self-normalized importance weight $\widetilde{w}(x)$ as:

$$\begin{aligned}
\widehat{w}(x) &= \frac{d(x)}{d^{P,\pi}(x)} = \\
&= \frac{\exp\left(\frac{1}{\eta}R(x)\right)}{\int_{\mathcal{X}} d^{P,\pi}(x) \exp\left(\frac{1}{\eta}R(x)\right) \mathrm{d}x}, \\
\widetilde{w}(x) &= \frac{\widehat{w}(x)}{\sum_{i=1}^N \widehat{w}(x_i)} = \\
&= \frac{\exp\left(\frac{1}{\eta}R(x)\right)}{\sum_{i=1}^N \exp\left(\frac{1}{\eta}R(x_i)\right)}.
\end{aligned}$$

Thus, the off-distribution estimator $\widetilde{J}_d$, which is optimized by $\widetilde{\text{OPT}}_{P,\pi}(\epsilon)$ is actually a *self-normalized importance weighting* estimate, opposed to the *ratio importance weighting estimate* $\widehat{J}_d$ which does not appear in the optimization problems, but will be useful in the following:

$$\begin{aligned}
\widehat{J}_d &= \frac{1}{N} \sum_{i=1}^N \widehat{w}(x_i) R(x_i), \\
\widetilde{J}_d &= \sum_{i=1}^N \widetilde{w}(x_i) R(x_i).
\end{aligned}$$

Analogously, we define the KL-divergence estimators:

$$\begin{aligned}
\widehat{D}_{KL}(d \| d^{P,\pi}) &= \frac{1}{N} \sum_{i=1}^N \widehat{w}(x_i) \log \widehat{w}(x_i), \\
\widetilde{D}_{KL}(d \| d^{P,\pi}) &= \sum_{i=1}^N \widetilde{w}(x_i) \log \left(N \widetilde{w}(x_i)\right),
\end{aligned}$$

and, given $d' \in \mathcal{D}_{\mathcal{P},\Pi}$, we define the cross-entropy estimators:

$$\widehat{H}(d\|d') = \frac{1}{N}\sum_{i=1}^{N} \widehat{w}(x_i)\log d'(x_i),$$
$$\widetilde{H}(d\|d') = \sum_{i=1}^{N} \widetilde{w}(x_i)\log d'(x_i).$$

It is well known that the ratio estimation is unbiased while the self-normalized estimator is biased but consistent.

4.3.2 Assumptions

We start providing two assumptions that we will consider in the whole analysis:

Assumption 4.3.1 (Uniformly bounded reward) *For any $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, it holds that: $|R(s,a,s')| \leq r_{\max}$.*

Assumption 4.3.2 (Finite pseudo-dimension) *Given a policy $\pi \in \Pi$ and a transition model $P \in \mathcal{P}$, the pseudo-dimensions of the hypothesis spaces $\{\frac{d}{d^{P,\pi}} : d \in D_{d^{P,\pi}}\}$, $\{\frac{d}{d^{P,\pi}}R : d \in D_{d^{P,\pi}}\}$, $\{\frac{d}{d^{P,\pi}}\log\left(\frac{d}{d^{P,\pi}}\right) : d \in D_{d^{P,\pi}}\}$ and $\{\frac{d}{d^{P,\pi}}\log(d') : d \in D_{d^{P,\pi}}, d' \in \mathcal{D}_{\mathcal{P},\Pi}\}$ are bounded by $v < +\infty$. The first is the space of non-normalized weights. The second is the space average reward estimators, the third is the space of KL–divergence estimators and the last is the space of cross-entropy estimators.*

4.3.3 Sensitivity to the KL constraint

In this section, we analyze how the performance of the solution of the problem $\text{OPT}_{P,\pi}(\epsilon)$ changes when we change the KL-divergence threshold. Suppose that $\epsilon' \leq \epsilon$, the constraint is more restrictive, thus we expect that $J_{d'} \leq J_d$, where d' is the solution of $\text{OPT}_{P,\pi}(\epsilon')$ and d is the solution of $\text{OPT}_{P,\pi}(\epsilon)$, since we are in the ideal case. Let us consider a new class distributions $d_\alpha = \alpha d + (1-\alpha)d_{P,\pi}$, with $\alpha \in [0,1]$. Ideally we could increase α until we satisfy the constraint ϵ' getting the best representation of d fulfilling the constraint (a projection).

Lemma 4.3.1 *Let d and d' be the solution of the problems $OPT_{P,\pi}(\epsilon)$ and $OPT_{P,\pi}(\epsilon')$ with $\epsilon' \leq \epsilon$. Let $d_\alpha = \alpha d + (1-\alpha)d^{P,\pi}$ with $\alpha \in [0,1]$. If $D_{KL}(d_\alpha\|d^{P,\pi}) = \epsilon'$, then $\alpha \geq \frac{\epsilon'}{\epsilon}$.*

Proof. We use the convexity of the KL-divergence: $D_{KL}(\alpha\mu_1 + (1-\alpha)\mu_2\|\alpha\nu_1 + (1-\alpha)\nu_2) \leq \alpha D_{KL}(\mu_1\|\nu_1) + (1-\alpha)D_{KL}(\mu_2\|\nu_2)$ for $\alpha \in [0,1]$. Take $\mu_1 = d$, $\mu_2 = \nu_1 = \nu_2 = d^{P,\pi}$:

$$\epsilon' = D_{KL}(d_\alpha\|q) = D_{KL}(\alpha d + (1-\alpha)d^{P,\pi}\|\alpha d^{P,\pi} + (1-\alpha)d^{P,\pi}) \leq$$
$$\leq \alpha D_{KL}(d\|d^{P,\pi}) + (1-\alpha)D_{KL}(d^{P,\pi}\|d^{P,\pi}) = \alpha D_{KL}(d\|d^{P,\pi}).$$

Therefore, observing that $D_{KL}(d\|d^{P,\pi}) \leq \epsilon$:

$$\alpha \geq \frac{\epsilon'}{D_{KL}(d\|d^{P,\pi})} \geq \frac{\epsilon'}{\epsilon}. \tag{4.42}$$

□

The following results upper bound the reduction of performance.

Proposition 4.3.1 (ϵ sensitivity) *Let d and d' the solutions of OPT$_{P,\pi}(\epsilon)$ and OPT$_{P,\pi}(\epsilon')$ respectively starting from $d^{P,\pi}$ and with $\epsilon' \leq \epsilon$. Then:*

$$J_d - J_{d'} \leq r_{\max} \| d - d^{P,\pi} \|_1 \left(1 - \frac{\epsilon'}{\epsilon} \right). \tag{4.43}$$

Proof. Consider the $\alpha' \in [0,1]$ such that $D_{KL}(d_{\alpha'} \| d^{P,\pi}) = \epsilon'$. We start observing that being d' the optimal solution with constraint ϵ' and since $d_{\alpha'}$ fulfills the constraint, we surely have $J_{d'} \geq J_{d_{\alpha'}}$. Consider the following sequence of inequalities:

$$\begin{aligned} J_d - J_{d'} &\leq J_d - J_{d_{\alpha'}} \\ &\leq r_{\max} \| d - d_{\alpha'} \|_1 \\ &\leq r_{\max} \| (1-\alpha')(d - d^{P,\pi}) \|_1 \\ &= r_{\max} (1-\alpha') \| d - d^{P,\pi} \|_1. \end{aligned}$$

Applying Lemma 4.3.1 we get $1 - \alpha' \leq 1 - \frac{\epsilon'}{\epsilon}$, from which the result follows. □

4.3.4 Finite-Sample Analysis

We have seen in the previous section that we need to solve using samples both phases of the REMPS problem. Starting from $d^{P,\pi}$, OPT$_{P,\pi}(\epsilon)$ yields the solution d whereas $\widetilde{\text{REMPS}}_{P,\pi}(\epsilon)$ provides the solution $\widetilde{d'}$ which is derived from the $\widetilde{\text{OPT}}_{P,\pi}(\epsilon)$ problem yielding $\widetilde{d}$ and the $\widetilde{\text{PROJ}}_{D_{\mathcal{P},\Pi}}(\widetilde{d})$ problem. There are two sources of error in this process. First, $\widetilde{d}$ is obtained from a finite sample and thus it might differ from d (*estimation error*). Second, we limit to a hypothesis space $\mathcal{D}_{\Pi,\mathcal{P}}$ that might not be able to represent $\widetilde{d}$ (*approximation error*). The goal of this analysis is to provide a bound to the quantity $J_d - J_{\widetilde{d'}}$. For this purpose, we consider the following decomposition to isolate the contribution of the OPT$_{P,\pi}(\epsilon)$ and DUAL$_{P,\pi}(\epsilon)$ from the contribution of PROJ$_{D_{\mathcal{P},\Pi}}(\cdot)$:

$$J_d - J_{\widetilde{d'}} = \underbrace{J_d - J_{\widetilde{d}}}_{\text{OPT}} + \underbrace{J_{\widetilde{d}} - J_{\widetilde{d'}}}_{\text{PROJ}}. \tag{4.44}$$

$J_d - J_{\widetilde{d}}$ A typical approach, from Empirical Risk Minimization (ERM), for bounding the estimation error is to sum and subtract the empirical risk of the empirical risk minimizer $\widetilde{J}_{\widetilde{d}}$ and exploit the fact that this quantity is larger (smaller in supervised learning) than the empirical risk of any other hypothesis in the hypothesis space (being ERM), in particular d. However, in our framework the hypothesis space changes since the constraint con the KL–divergence is estimated from samples and, in principle, it can impose more relaxed/tight conditions. For this purpose we introduce a new distribution $\overline{d}$ which is the optimal solution of the OPT$_{P,\pi}(\epsilon)$ problem using the sample constraint. For this reason, $\widetilde{d}$ and $\overline{d}$ are searched in the same hypothesis space and thus we can apply theory from ERM. Clearly, we need to manage the discrepancy between $\overline{d}$ and d; for this, we use the sensitivity analysis we presented before. Let us define the discrepancy in the constraint for a given hypothesis d:

$$\Delta\epsilon(d) = D_{KL}(d\|d^{P,\pi}) - \widetilde{D}_{KL}(d\|d^{P,\pi}). \tag{4.45}$$

As a consequence $\widetilde{D}_{KL}(d\|d^{P,\pi}) \leq \epsilon \iff D_{KL}(d\|d^{P,\pi}) \leq \epsilon + \Delta\epsilon(d)$. Finally, we define $\Delta\epsilon = \sup_{d\in\mathcal{D}_{d^{P,\pi}}} \Delta\epsilon(d)$. We have the usual two cases. i) If $\Delta\epsilon \leq 0$ then the exact constraint is always (i.e., for every hypothesis) tighter and thus $J_{\overline{d}} \geq J_d$. ii) If $\Delta\epsilon > 0$ then there exist at least one hypothesis for which the constraint is looser; thus it might be that $J_{\overline{d}} \leq J_d$. In general, the following result holds.

Lemma 4.3.2 *Let d, $\overline{d}$ as defined before. The following bound holds:*

$$J_d \leq J_{\overline{d}} + 2r_{\max} \max\left\{0, \min\left\{\frac{1}{2}, \frac{\Delta\epsilon}{\epsilon}\right\}\right\}. \tag{4.46}$$

Proof. If $J_d - J_{\overline{d}} \leq 0$ then the theorem holds. Otherwise, it must be that $\Delta\epsilon(d) \geq 0$ (this is because we find $\overline{d}$ as optimal solution having smaller performance with respect to d). We define d_α as in the previous proposition so we get:

$$\begin{aligned}
J_d - J_{\overline{d}} &\leq J_d - J_{d_\alpha} \\
&\leq r_{\max}\left(1 - \frac{\epsilon}{\epsilon + \Delta\epsilon(d)}\right)\|d - d^{P,\pi}\|_1 \\
&\leq r_{\max}\frac{\Delta\epsilon(d)}{\epsilon + \Delta\epsilon(d)}\|d - d^{P,\pi}\|_1 \\
&\leq 2r_{\max}\min\left\{\frac{1}{2}, \frac{\Delta\epsilon(d)}{\epsilon}\right\} \\
&\leq 2r_{\max}\min\left\{\frac{1}{2}, \frac{\Delta\epsilon}{\epsilon}\right\},
\end{aligned}$$

where we exploited the fact that $\|d - d^{P,\pi}\|_1 \leq 2$, $\frac{\Delta\epsilon(d)}{\epsilon+\Delta\epsilon(d)} \leq \frac{\Delta\epsilon(d)}{\epsilon}$, being $\Delta\epsilon(d) \geq 0$, and $\frac{\Delta\epsilon(d)}{\epsilon+\Delta\epsilon(d)} \leq \frac{1}{2}$ being $\Delta\epsilon(d) \leq \epsilon$ and finally $\Delta\epsilon(d) \leq \Delta\epsilon$. Taking the max between the two cases we get the result. □

Notice that $\max\left\{0, \min\left\{\frac{1}{2}, \frac{\Delta\epsilon}{\epsilon}\right\}\right\} \leq \frac{|\Delta\epsilon|}{\epsilon}$ and $\frac{|\Delta\epsilon|}{\epsilon} = \frac{1}{\epsilon}\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi}) - D_{KL}(d\|d^{P,\pi})\right|$, which is convenient for using ERM theory. Now we are ready to bound $J_d - J_{\widetilde{d}}$.

Lemma 4.3.3 *Let d and $\widetilde{d}$ be the solutions of the $OPT_{P,\pi}(\epsilon)$ and $\widetilde{OPT}_{P,\pi}(\epsilon)$ problems, the latter using N i.i.d. samples collected from $d^{P,\pi}$. Let $\epsilon > 0$ be the KL–constraint. Then, it holds that:*

$$J_d - J_{\widetilde{d}} \leq 2\sup_{d\in\mathcal{D}_{d^{P,\pi}}}|J_d - \widetilde{J}_d| + \frac{2r_{\max}}{\epsilon}\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi}) - D_{KL}(d\|d^{P,\pi})\right|. \tag{4.47}$$

Proof. We use a very simple argument of ERM combined with the previous result. Let $\overline{p}$ as defined before, we have:

$$J_d - J_{\widetilde{d}} \leq J_{\overline{d}} - J_{\widetilde{d}} + \frac{2r_{\max}}{\epsilon}\max\left\{0, \min\left\{\frac{1}{2}, \frac{\Delta\epsilon}{\epsilon}\right\}\right\}$$

$$\begin{aligned}
&\leq J_{\widetilde{d}} - J_{\widetilde{d}} + \frac{2r_{\max}}{\epsilon}|\Delta\epsilon| \\
&= J_{\widetilde{d}} - J_{\widetilde{d}} + \frac{2r_{\max}}{\epsilon}|\Delta\epsilon| \pm \widetilde{J}_{\widetilde{d}} \\
&\leq J_{\widetilde{d}} - \widetilde{J}_{\widetilde{d}} + \widetilde{J}_{\widetilde{d}} - J_{\widetilde{d}} + \frac{2r_{\max}}{\epsilon}|\Delta\epsilon| \\
&\leq 2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widetilde{J}_d| + \frac{2r_{\max}}{\epsilon} \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \left| \widetilde{D}_{KL}(d \| d^{P,\pi}) - D_{KL}(d \| d^{P,\pi}) \right|,
\end{aligned}$$

where we exploited the fact that $\widetilde{J}_{\widetilde{d}} \leq \widetilde{J}_{\widetilde{d}}$, being $\widetilde{d}$ the ERM over the same hypothesis space. □

$J_{\widetilde{d}} - J_{\widetilde{d}'}$ For bounding this second term is useful to recall the property of the KL–divergence $D_{KL}(d\|d') = H(d\|d') - H(d)$, where $H(d\|d')$ is the cross–entropy (likelihood when computed on finite samples) between d and d' and $H(d)$ is the entropy of d. When performing the projection we are minimizing the term $H(d\|d')$ since $H(d)$ does not depend on d'. We can state the following result.

Lemma 4.3.4 *Let $\widetilde{d}$ and $\widetilde{d}'$ be the solutions of the $\widetilde{OPT}_{P,\pi}(\epsilon)$ and $\widetilde{PROJ}_{\mathcal{D}_{\mathcal{P},\Pi}}(\widetilde{d})$ problems using N i.i.d. samples collected from $d^{P,\pi}$. Let $\epsilon > 0$ be the KL–constraint. Then, it holds that:*

$$J_{\widetilde{d}} - J_{\widetilde{d}'} \leq r_{\max}\sqrt{2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \inf_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} D_{KL}(d\|d') +} \tag{4.48}$$

$$+ r_{\max}\sqrt{2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \sup_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} \left|\widehat{H}(d\|d') - H(d\|d')\right|}. \tag{4.49}$$

Proof. Let us call:

$$\nu_1 = \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \sup_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} \left|\widehat{H}(d\|d') - H(d\|d')\right|. \tag{4.50}$$

Consider the best approximation of $\widetilde{d}$ contained in $\mathcal{D}_{\mathcal{P},\Pi}$, let us call it d^* ($d^* = \arg\min_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} H(\widetilde{d}\|d')$). Then we can state the following inequalities:

$$\begin{aligned}
J_{\widetilde{d}} - J_{\widetilde{d}'} &\leq r_{\max}\|\widetilde{d} - \widetilde{d}'\|_1 && (4.51)\\
&\leq r_{\max}\sqrt{2D_{KL}(\widetilde{d}\|\widetilde{d}')} && (4.52)\\
&\leq r_{\max}\sqrt{2H(\widetilde{d}\|\widetilde{d}') - 2H(\widetilde{d})} && (4.53)\\
&\leq r_{\max}\sqrt{2\widehat{H}(\widetilde{d}\|\widetilde{d}') - 2H(\widetilde{d}) + \nu_1} && (4.54)\\
&\leq r_{\max}\sqrt{2\left(\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i)\right)\widetilde{H}(\widetilde{d}\|\widetilde{d}') - 2H(\widetilde{d}) + \nu_1} && (4.55)\\
&\leq r_{\max}\sqrt{2\left(\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i)\right)\widetilde{H}(\widetilde{d}\|d^*) - 2H(\widetilde{d}) + \nu_1} && (4.56)\\
&\leq r_{\max}\sqrt{2\widehat{H}(\widetilde{d}\|d^*) - 2H(\widetilde{d}) + \nu_1} && (4.57)\\
&\leq r_{\max}\sqrt{2D_{KL}(\widetilde{d}\|d^*) + 2\nu_1} && (4.58)
\end{aligned}$$

$$\leq r_{\max}\sqrt{2D_{KL}(\widetilde{d}\|d^*)} + r_{\max}\sqrt{2\nu_1} \tag{4.59}$$

$$\leq r_{\max}\sqrt{2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \inf_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} D_{KL}(d\|d')} + r_{\max}\sqrt{2\nu_1}, \tag{4.60}$$

where line (4.52) follows from Pinsker inequality, lines (4.54) and (4.58) follow from the hypothesis, line (4.55) follows from the fact that $\widetilde{d'}$ is ERM, line (4.59) follows from the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ and lines (4.55) and (4.56) follows from the fact that $\left(\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i)\right)\widetilde{H}(\widetilde{d}\|\widetilde{d'}) = \widehat{H}(\widetilde{d}\|\widetilde{d'})$. □

Putting all together we get the following result.

Theorem 4.3.1 (Error Decomposition) *Let $\pi \in \Pi$ and $P \in \mathcal{P}$ be the current policy and transition model respectively. Let $\epsilon > 0$ be the KL–divergence constraint. Let $d' \in \mathcal{D}_{\mathcal{P},\Pi}$ be the solution of the REMPS$_{P,\pi}(\epsilon)$ problem and $\widetilde{d'} \in \mathcal{D}_{\mathcal{P},\Pi}$ be the solution of the $\widetilde{\text{REMPS}}_{P,\pi}(\epsilon)$ problem computed with $N > 0$ samples collected with $d^{P,\pi}$. Then, under Assumptions 4.3.1, it holds that:*

$$\begin{aligned} J_p - J_{\widetilde{d'}} \leq\ & 2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widetilde{J}_d| + \\ & + \frac{2r_{\max}}{\epsilon} \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \left|\widetilde{D}_{KL}(d\|d^{P,\pi}) - D_{KL}(d\|d^{P,\pi})\right| + \\ & + r_{\max}\sqrt{2 \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \inf_{\overline{d} \in \mathcal{D}_{\mathcal{P},\Pi}} D_{KL}(\overline{d}\|d') +} \\ & + r_{\max}\sqrt{2 \sup_{d \in \mathcal{D}^{d_{P,\pi}}} \sup_{d' \in \mathcal{D}_{\mathcal{P},\Pi}} \left|\widehat{H}(d\|d') - H(d\|d')\right|}. \end{aligned}$$

Proof. Put together Lemma 4.3.3 and Lemma 4.3.4. □

4.3.5 Analysis for bounded probability densities

In the following, we will provide a finite–sample analysis of REMPS under the following restrictive assumption.

Assumption 4.3.3 *(Finite* inf*, Non–zero* sup*) For every $\pi \in \Pi$ and a transition model $P \in \mathcal{P}$, for every $d \in \mathcal{D}_{d^{P,\pi}}$ and for every $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ it holds that $0 < m \leq d(s,a,s') \leq M < +\infty$ and $0 < m \leq d^{P,\pi}(s,a,s') \leq M < +\infty$.*

This assumption ensures that all loss function we are considering are uniformly bounded and allows us to state a sequence of useful facts.

Lemma 4.3.5 *For any $d \in \mathcal{D}_{d^{P,\pi}}$ and for any $d' \in \mathcal{D}_{\mathcal{P},\Pi}$. The following facts hold:*

1. *The importance weights are bounded above and below: $\frac{m}{M} \leq \widehat{w}(x) \leq \frac{M}{m}$.*

2. *The empirical KL divergence is bounded:* $\left|\widehat{D}_{KL}(d\|d^{P,\pi})\right| \leq \max\left\{\frac{1}{e}, \frac{M}{m}\log\frac{M}{m}\right\}$*;*
3. *The empirical cross–entropy is bounded:* $\left|\widehat{H}(d'\|d)\right| \leq \max\left\{-\frac{M}{m}\log m, \frac{M}{m}\log M\right\}$*;*
4. $\left|\widehat{D}_{KL}(d\|d^{P,\pi}) - \widetilde{D}_{KL}(d\|d^{P,\pi})\right| \leq \Lambda(M,m,N)\left|\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i) - 1\right|$*, where* $\Lambda(M,m,N) = \max\left\{\log\frac{M}{m}+1, -\log\frac{m}{M}-1, \log N + 1\right\}$*.*
5. $\left|\widehat{J}_d - \widetilde{J}_d\right| \leq r_{\max}\left|\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i) - 1\right|$*.*

Proof. 1. Immediate consequence of Assumption 4.3.3, just observing that $\widehat{w}(x) = d(x)/d^{P,\pi}(x)$.

2. $|\widehat{D}_{KL}(d\|d^{P,\pi})| \leq \frac{1}{N}\sum_{i=1}^{N}|\widehat{w}(x)\log\widehat{w}(x)|$. Now, we know that $\widehat{w}(x) \leq \frac{M}{m}$ and that the function $|y\log y|$ has a local maximum whose value is $1/e$. As a consequence, $|\widehat{w}(x)\log\widehat{w}(x)| \leq \max\{1/e, M/m\}$.
3. $|\widehat{H}(d\|d')| \leq \frac{1}{N}\sum_{i=1}^{N}|\widehat{w}(x)\log d'(x)|$. The maximum is attained when both $\widehat{w}(x)$ and $|\log d'(x)|$ are maximum. $\widehat{w}(x) \leq M/m$, while $|\log d'(x)| \leq \max\{-\log m, \log M\}$.
4. The absolute derivative of $y\log y$ is $|\log y + 1|$. Consider the term $\widehat{w}(x_i)\log\widehat{w}(x_i)$, we know that $m/M \leq \widehat{w}(x_i) \leq M/m$, therefore the maximum absolute derivative has value $\max\{\log(M/m)+1, -\log(m/M)-1\}$. Consider the term $N\widetilde{w}(x_i) = N\widehat{w}(x_i)/\sum_{i=1}^{N}\widehat{w}(x_i)$. We know that $m/M \leq N\widetilde{w}(x_i) \leq N$, thus the maximum absolute derivative has value $\max\{\log(N)+1, -\log(m/M)-1\}$. Since the Lipschitz constant of an average is smaller or equal to the Lipschitz constant of each term, we get the result.
5. Consider the inequalities:

$$\begin{aligned}
\left|\widehat{H}(d'\|d) - \widetilde{H}(d'\|d)\right| &= \left|\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i)r(x_i) - \frac{\sum_{i=1}^{N}\widehat{w}(x_i)r(x_i)}{\sum_{i=1}^{N}\widehat{w}(x_i)}\right| \\
&= \left|\frac{\sum_{i=1}^{N}\widehat{w}(x_i)r(x_i)}{\sum_{i=1}^{N}\widehat{w}(x_i)}\left(\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i) - 1\right)\right| \\
&\leq r_{\max}\left|\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i) - 1\right|
\end{aligned}$$

□

We report now a standard result of learning theory that we are going to use extensively throughout the analysis Mohri et al. (2012).

Theorem 4.3.2 *Let $\mathcal{H}$ be a family real-valued functions and let $\mathcal{G} = \{L_h(x) : h \in \mathcal{H}\}$ be the family of loss functions associated to $\mathcal{H}$. Assume that* $\mathrm{Pdim}(\mathcal{G}) = v$*. and that the loss function L is bounded by M.Then, for any $\delta \in (0,1)$, with probability at least* $1-\delta$*, the following holds for all $h \in \mathcal{H}$:*

$$\mathop{\mathbb{E}}_{X}[L_h(X)] \leq \frac{1}{N}\sum_{i=1}^{N}L_h(x_i) + M\sqrt{\frac{8v\log\frac{2eN}{v} + 8\log\frac{4}{\delta}}{N}}. \tag{4.61}$$

$$\frac{1}{N}\sum_{i=1}^{N}L_h(x_i) \leq \mathop{\mathbb{E}}_{X}[L_h(X)]\, M\sqrt{\frac{8v\log\frac{2eN}{v} + 8\log\frac{4}{\delta}}{N}}. \tag{4.62}$$

Using this result, we immediately derive the following.

Lemma 4.3.6 *Each of these events hold with probability at least* $1-\delta$*:*

$(\mathcal{E}_1)$ $\forall d \in \mathcal{D}_{d^{P,\pi}} : \left| \frac{1}{N}\sum_{i=1}^N \widehat{w}(x_i) - 1 \right| \le \frac{M}{m}\sqrt{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}}$;

$(\mathcal{E}_1)$ $\forall d \in \mathcal{D}_{d^{P,\pi}} : \left| \widehat{J}_d - J_d \right| \le r_{\max}\frac{M}{m}\sqrt{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}}$;

$(\mathcal{E}_3)$ $\forall d \in \mathcal{D}_{d^{P,\pi}} : \left| \widehat{D}_{KL}(d\|d^{P,\pi}) - D_{KL}(d\|d^{P,\pi}) \right| \le \bar{K}\sqrt{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}}$, *where* $\bar{K} = \max\left\{\frac{1}{e}, \frac{M}{m}\log\frac{M}{m}\right\}$;

$(\mathcal{E}_4)$ $\forall d \in \mathcal{D}_{d^{P,\pi}}, \forall d' \in \mathcal{D}_{\mathcal{P},\Pi} : \left| \widehat{H}(d'\|d) - H(d'\|d) \right| \le \frac{M}{m}\bar{M}\sqrt{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}}$, *where* $\bar{M} = \max\{-\log m, \log M\}$.

Proof. It is a trivial application of Theorem 4.3.2 after symmetrization, by carefully defining the max of each function involved and exploiting Assumption 4.3.2. □

We can now put all together.

Theorem 4.3.3 (Finite–Sample Bound under Assumption 4.3.3) *Let* $\pi \in \Pi$ *and* $P \in \mathcal{P}$ *be the current policy and transition model respectively. Let* $\epsilon > 0$ *be the KL–divergence constraint. Let* $d' \in \mathcal{D}_{\mathcal{P},\Pi}$ *be the solution of the REMPS*$_{P,\pi}(\epsilon)$ *problem and* $\widetilde{d}' \in \mathcal{D}_{\mathcal{P},\Pi}$ *be the solution of the* $\widetilde{\text{REMPS}}_{P,\pi}(\epsilon)$ *problem computed with* $N > 0$ *samples collected with* $d^{P,\pi}$*. Then, under Assumptions 4.3.1, 4.3.2 and 4.3.3, there exists a constant* ϕ *and function* $\psi(N) = \mathcal{O}(\log N)$*, such that for any* $\delta \in (0,1)$*, with probability at least* $1-4\delta$ *it holds that:*

$$\begin{aligned} J_d - J_{\widetilde{d}'} \le & \sqrt{2} r_{\max} \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \inf_{\bar{d} \in \mathcal{D}_{\mathcal{P},\Pi}} \sqrt{D_{KL}(d\|\bar{d})} + \\ & + r_{\max}\phi \sqrt[4]{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}} + \\ & + r_{\max}\psi(N)\sqrt{\frac{8v \log \frac{2eN}{v} + 8 \log \frac{8}{\delta}}{N}}. \end{aligned}$$

Proof. We start from Theorem 4.3.1 and we bound each term using Lemma 4.3.5 and Lemma 4.3.6. Let us start with $\sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widetilde{J}_d|$:

$$\begin{aligned} \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widetilde{J}_d| &= \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widetilde{J}_d \pm \widehat{J}_d| \\ &\le \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widehat{J}_d| + \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |\widehat{J}_d - \widetilde{J}_d| \\ &\le \sup_{d \in \mathcal{D}_{d^{P,\pi}}} |J_d - \widehat{J}_d| + r_{\max} \sup_{d \in \mathcal{D}_{d^{P,\pi}}} \left| \frac{1}{N}\sum_{i=1}^N \widehat{w}(x_i) - 1 \right| \end{aligned}$$

$$\leq 2r_{\max}\frac{M}{m}\sqrt{\frac{8v\log\frac{2eN}{v}+8\log\frac{8}{\delta}}{N}},$$

where we exploited events $(\mathcal{E}_1)$ and $(\mathcal{E}_2)$. Consider $\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi})-D_{KL}(d\|d^{P,\pi})\right|$:

$$\begin{aligned}
\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi})-D_{KL}(d\|d^{P,\pi})\right| &= \sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi})-D_{KL}(d\|d^{P,\pi})\pm\widehat{D}_{KL}(d\|d^{P,\pi})\right| \\
&\leq \sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|D_{KL}(d\|d^{P,\pi})-\widehat{D}_{KL}(d\|d^{P,\pi})\right|+ \\
&\quad+\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|\widetilde{D}_{KL}(d\|d^{P,\pi})-\widehat{D}_{KL}(d\|d^{P,\pi})\right| \\
&\leq \sup_{d\in\mathcal{D}_{d^{P,\pi}}}\left|D_{KL}(d\|d^{P,\pi})-\widehat{D}_{KL}(d\|d^{P,\pi})\right| \\
&\quad+\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\Upsilon(M,m,N)\left|\frac{1}{N}\sum_{i=1}^{N}\widehat{w}(x_i)-1\right| \\
&\leq\left(\max\left\{\frac{1}{e},\frac{M}{m}\log\frac{M}{m}\right\}+\Upsilon(M,m,N)\right)\sqrt{\frac{8v\log\frac{2eN}{v}+8\log\frac{8}{\delta}}{N}} \\
&\leq f(N)\sqrt{\frac{8v\log\frac{2eN}{v}+8\log\frac{8}{\delta}}{N}},
\end{aligned}$$

where we defined $\Upsilon(M,m,N)=\max\left\{\log\frac{M}{m}+1,-\log\frac{m}{M}-1,\log N+1\right\}$,
$f(N)=\left(\max\left\{\frac{1}{e},\frac{M}{m}\log\frac{M}{m}\right\}+\Upsilon(M,m,N)\right)$ and we exploited events $(\mathcal{E}_1)$ and $(\mathcal{E}_3)$. Finally, the term $\sup_{d\in\mathcal{D}^{d^{P,\pi}}}\sup_{d'\in\mathcal{D}_{\Pi,\mathcal{P}}}\left|\widehat{H}(d\|d')-H(d\|d')\right|$ can be bounded using Lemma 4.3.6. Let us define $c=\frac{M}{m}\max\{-\log m,\log M\}$ and $\nu=\sqrt{\frac{8v\log\frac{2eN}{v}+8\log\frac{8}{\delta}}{N}}$ and we put all together obtaining:

$$\begin{aligned}
J_{d'}-J_{\widetilde{d}'} &\leq 4r_{\max}\nu+\frac{2r_{\max}}{\epsilon}f(N)\nu+r_{\max}\sqrt{2}\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\inf_{\overline{d}\in\mathcal{D}_{\mathcal{P},\Pi}}\sqrt{D_{KL}(d\|\overline{d})}+r_{\max}\sqrt{2c\nu} \\
&= r_{\max}\sqrt{2}\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\inf_{\overline{d}\in\mathcal{D}_{\Pi,\mathcal{P}}}\sqrt{D_{KL}(d\|\overline{d})}+r_{\max}\sqrt{\nu}\left(\left(4+\frac{2}{\nu}f(N)\right)\sqrt{\nu}+\sqrt{2c}\right) \\
&= r_{\max}\sqrt{2}\sup_{d\in\mathcal{D}_{d^{P,\pi}}}\inf_{\overline{d}\in\mathcal{D}_{\Pi,\mathcal{P}}}\sqrt{D_{KL}(d\|\overline{d})}+r_{\max}\phi\sqrt{\nu}+r_{\max}\psi_\epsilon(N)\nu,
\end{aligned}$$

where we renamed $\psi_\epsilon(N)=4+\frac{2}{\epsilon}f(N)$ and $\phi=\sqrt{2c}$. Notice that $\psi_\epsilon(N)=\mathcal{O}(\log N)$. Since we made a union bound over the events $(\mathcal{E}_1)$, $(\mathcal{E}_2)$, $(\mathcal{E}_3)$ and $(\mathcal{E}_4)$, the statement holds with probability $1-4\delta$. $\square$

Chapter 5

Experimental Evaluation

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are, if it doesn't agree with experiment, it's wrong.

R.P. Feynman

In the previous chapter we presented the REMPS algorithm, able to optimize the model and the policy in the context of discretes and continuous CMDPs. In this chapter we perform an experimental evaluation of REMPS. In Section 5.1 we consider the chain environment, that is a toy problem for which we can visualize the return surface and we can easily understand the behaviour of our algorithm. In Section 5.2 we evaluate the performance of our algorithm on the discrete-actions, continuous-states, cart-pole problem, a standard RL benchmark. We compare REMPS with the G(PO)MDP (Baxter and Bartlett, 2001) extension for CMDPs (see Appendix B.2). In Section 5.3 we consider an autonomous driving and configuration problem in a complex environment: TORCS. The goal of the TORCS experiment is to show the benefit of environment configuration in a complex task, similar to real world tasks.

5.1 Chain Problem

We use the chain problem as a proof of concept to show the benefits of our algorithm. Using this environment we experimentally validate the ability of REMPS to overcome local minima in the return landscape.
The environment in illustrated in Fig. 5.1. There are two actions available to the agent denoted with a_0 and a_1. The policy is very simple and has only one parameter:

$$\pi_\theta(a_0|s) = \theta \; \forall s \in \mathcal{S}, \tag{5.1}$$

$$\pi_\theta(a_1|s) = 1 - \theta \; \forall s \in \mathcal{S} . \tag{5.2}$$

Thus the policy is uniquely defined by the parameter $\theta \in [0, 1]$, that is the probability of selecting action a_0 and it is the same on all states. The model has only one parameter, ω,

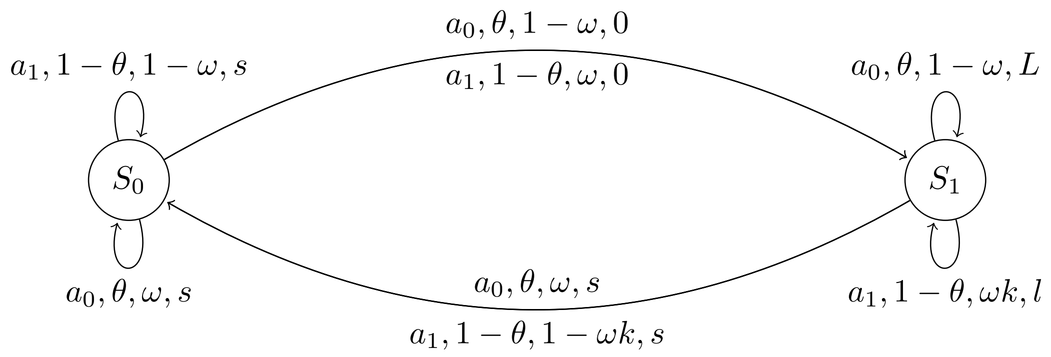
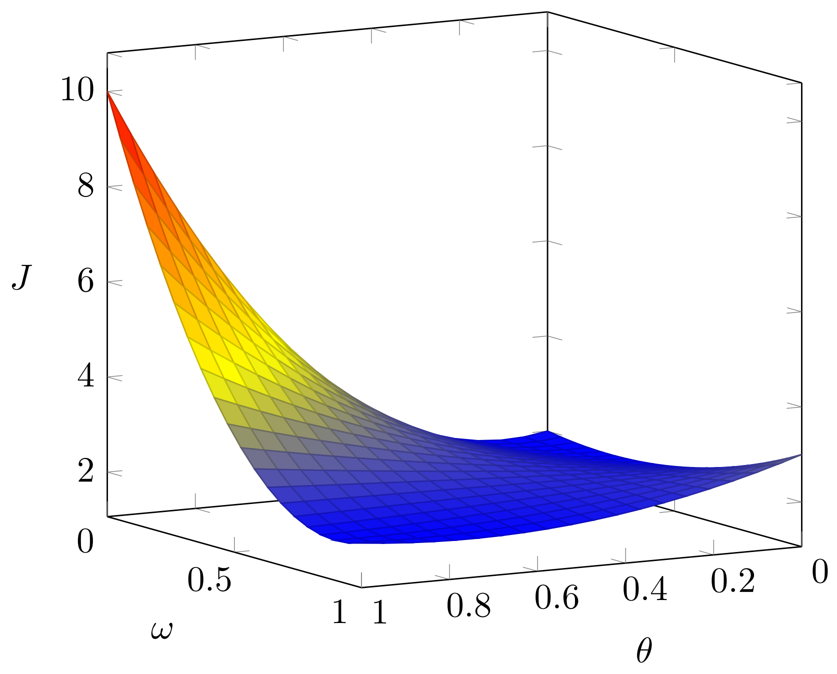

$a_0, \theta, 1-\omega, 0$
$a_1, 1-\theta, \omega, 0$
$a_1, 1-\theta, 1-\omega, s$
$a_0, \theta, 1-\omega, L$
S_0
S_1
a_0, θ, ω, s
a_0, θ, ω, s
$a_1, 1-\theta, 1-\omega k, s$
$a_1, 1-\theta, \omega k, l$

Figure 5.1: Chain problem. On the edges we have (action, action probability, transition probability, reward).

Parameter	Value
k	0.2
L	10
l	8
s	2
ω_0	0.8
θ_0	0.2
num of samples	$2 \cdot 10^4$
num of steps per episode	$5 \cdot 10^2$

Table 5.1: Hyper-parameters used in our chain experiments.

that is the probability of action failure. Action a_0 (forward), if successful, brings the agent to state s_1. Action a_1 (backward), if successful, brings the agent to state s_0. The agent gets an high reward, $L > 0$, if, starting from state s_1 executes successfully action a_0. The agent gets a smaller reward l, $(0 < l < L)$ if it lands in state s_1 starting from s_1 but executing action a_1. Agent gets an even smaller reward, s $(0 < s < l)$ when it lands in state s_0. The parameter k is not configurable and it has been added to avoid symmetries in the return surface. In the chain problem we are in the ideal case, the model is known, thus we can project the stationary distribution directly. We initialized the model parameters to ω_0 and the policy parameter to θ_0 such that the initial position in the return landscape is near a local minima. Starting from this position a gradient method should point toward the local maxima, while we empirically demonstrate the ability of our algorithm to reach the pair (P^*,π^*) yielding maximum performance in this environment. The average reward surface as function of the model and policy parameters is illustrated in Fig. 5.2.
Table 5.1 summarizes the parameters used in our experiments.
We compare the result of our algorithm with G(PO)MDP over model and policy parameters. It is possible to see that G(PO)MDP, being a gradient method follows the slope of the objective function and it is attracted by the *local* maximum in $\theta = 0, \omega = 1$. REMPS outperforms G(PO)MDP reaching the *global* maximum in $\theta = 1, \omega = 0$ after few iterations. We perform different runs varying the ϵ parameter to show the effect of the parameter. Results are shown in Fig. 5.3.



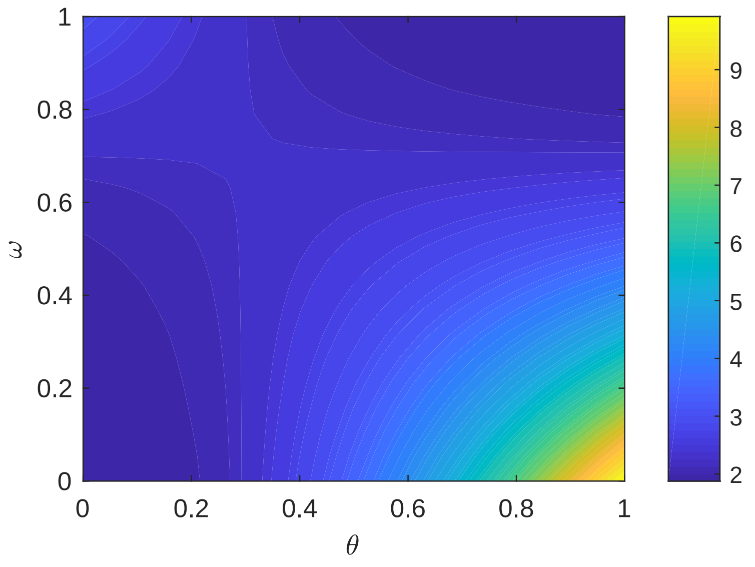


Figure 5.2: Different representations of the average reward surface of the chain experiment as function of model and policy parameters. There is a local maximum in $\theta = 0, \omega = 1$, a global maximum in $\theta = 1, \omega = 0$. We start our algorithm near the local minima in $\theta = 0.33, \omega = 0.66$.

5.1.1 Sensitivity to ϵ

The choice of ϵ is critical and problem-dependent. A too small ϵ causes a premature convergence of REMPS to local maximum. A too high ϵ yields imprecise estimation of the performance of the model-policy target and causes an imprecise projection on the space of possible probability distributions. Fig. 5.4 shows the performance of the best model-policy found as function of ϵ. We show the value of the primal that is the value of the objective

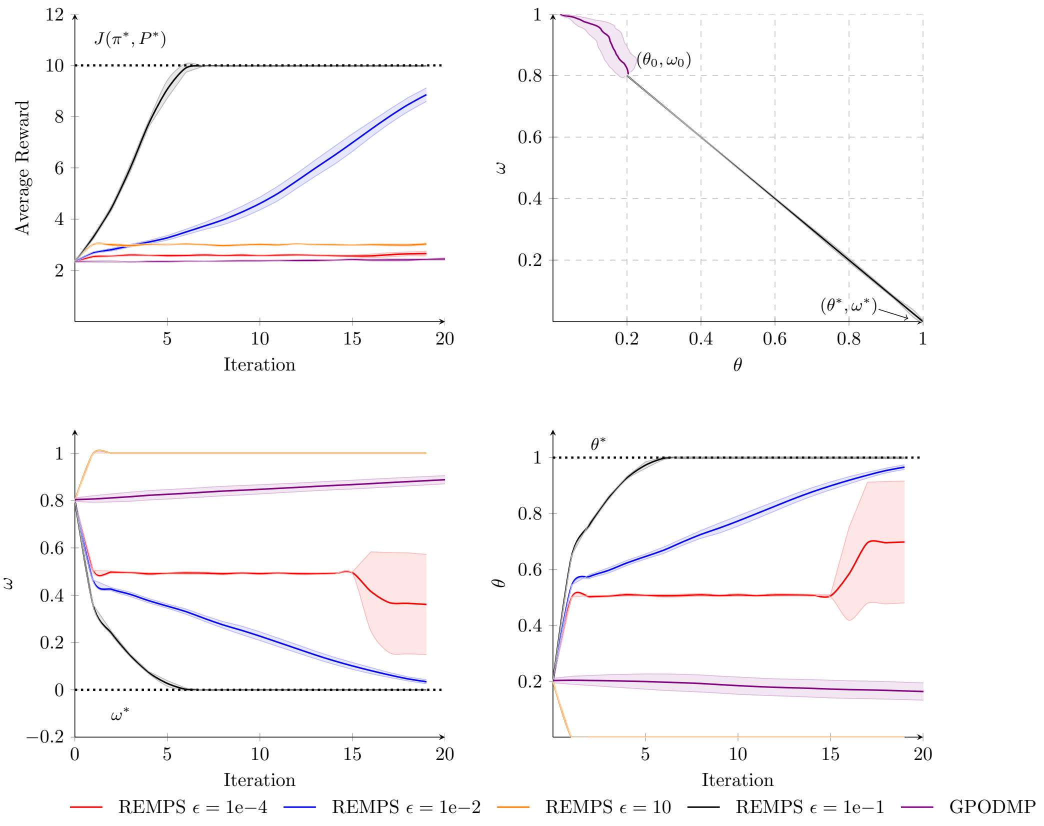


Figure 5.3: Chain experiment. Top left: average reward of REMPS and GPOMDP. Top right: updates of the model and policy parameter of GPOMDP and REMPS. Bottom left: shows the updates of the model parameter ω. Bottom right: updates of the policy parameter θ. Shaded areas represent the 95% confidence interval over 10 runs of the algorithm.

function without any constraint on the policy and model. We can see that the value of the primal is greater than the actual value of the objective after the projection, thus the projection yields a degradation of the performance. The performance degradation is due to the following reason. Moving too far away from the current stationary distribution the primal solution might not be representable using our hypothesis space. Performing a Moment Projection we obtain a a performance degradation with respect to the original solution.

5.1.2 Sensitivity to parameter initialization

REMPS behaves consistently with respect to a random initialization of model and policy parameters. In Fig. 5.5 we can see that REMPS updates the model and policy parameters toward the global maximum while GPOMDP updates vary with the initial parameters. In the GPOMDP learning curve it is possible to see clearly the two attractors. REMPS shows a stable behaviour in the case of few model parameters and complex return landscape.

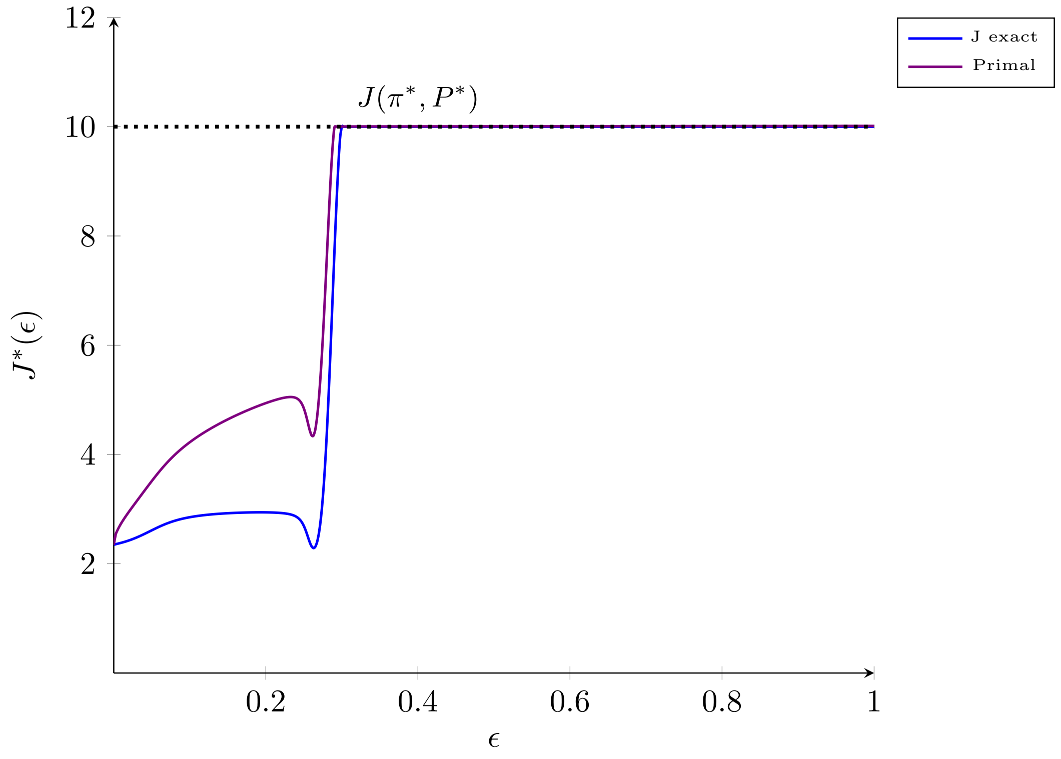


Figure 5.4: Average reward and primal of the best model-policy couple in the chain experiment as function of ϵ using the projection of the discounted stationary state distribution.

5.1.3 Comparison with SPMI

SPMI is, at the moment of writing, the unique algorithm proposed for CMDPs. We compare in this section the comparison between SMPI and REMPS. In Fig. 5.6 we show the behaviour of the variants of SPMI on the chain experiment. We can easily notice that SPMI requires a huge number of iterations before convergence. While REMPS converges approximately after 10 iterations, SPMI requires a number of iterations in the order of 10^3. This is due to the conservative step size of safe approaches. SPMI, SPMI-alt, SPMI-sup and SPI-SMI reach the global maximum while SMI-SPI goes (very slowly) to the local maximum. SMI-SPI is not able to reach the global maximum since it alternates a model improvement step to a policy improvement step considering the two components in a separate manner.
We recall that SPMI is applicable to the chain experiment since this environment has a discrete state space and a discrete action space, while the standard version of this algorithm cannot be applied to the environments presented later in this chapter.

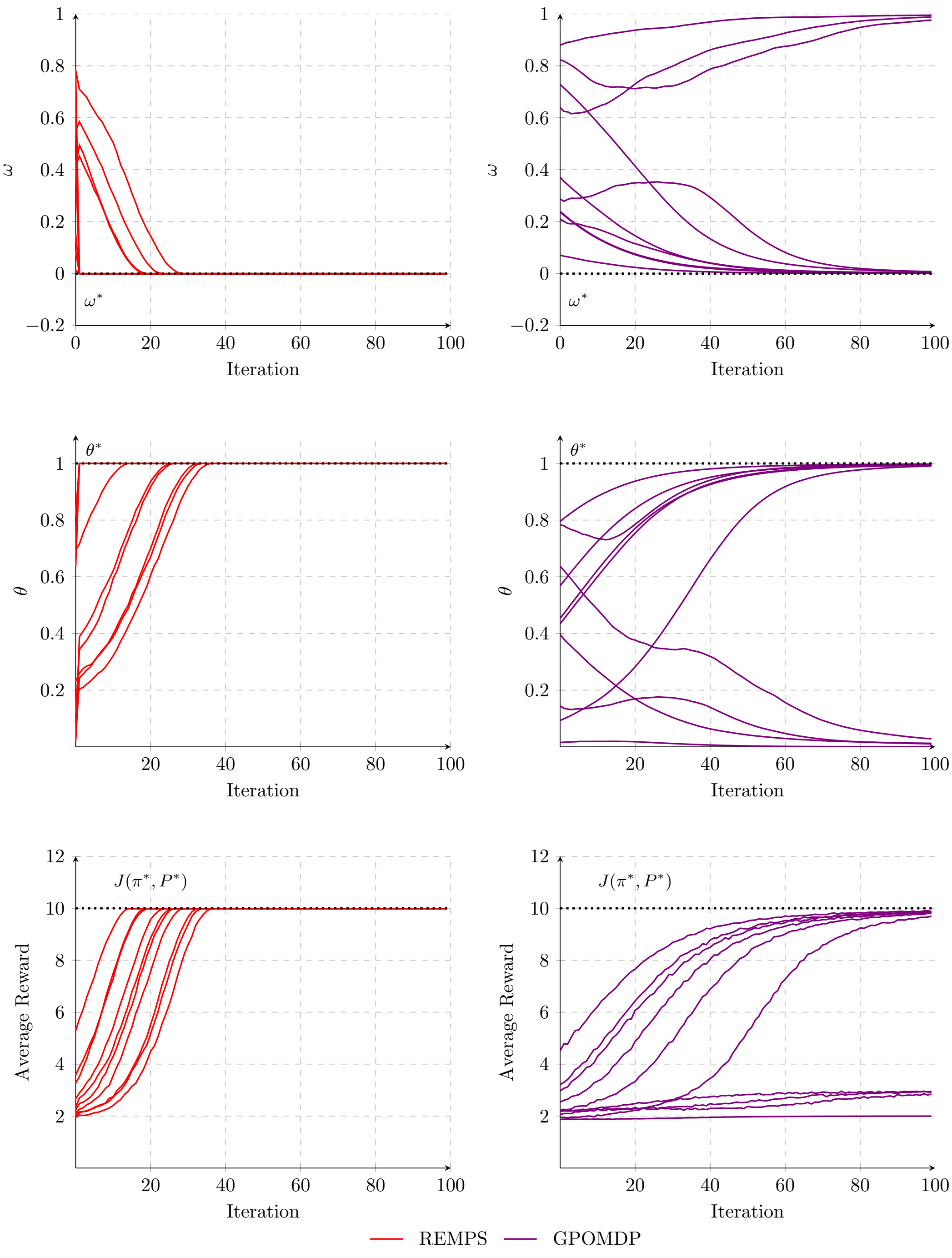


Figure 5.5: Chain experiment with random initialization of model and policy parameter. Comparison between GPOMDP and REMPS. Top: model parameter. Center: policy parameter. Bottom: average reward.

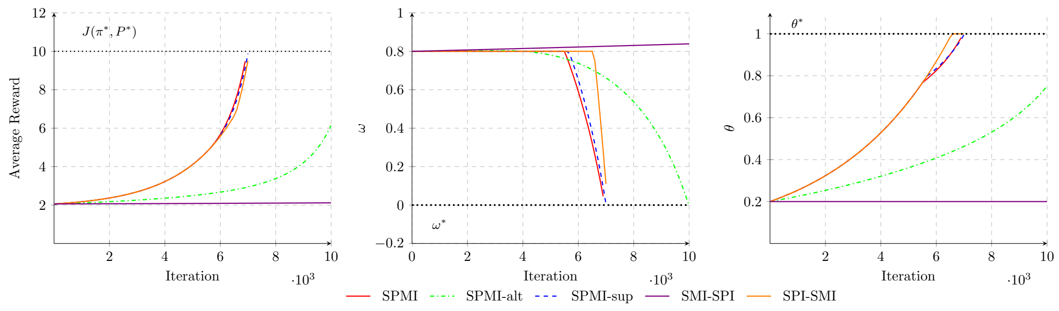


Figure 5.6: SPMI on the chain experiment. Left: average reward. Center: model parameter. Right: policy parameter.

5.2 Cart-Pole

The Cart-Pole domain is a standard RL benchmark. The Cart-Pole world consists of a cart that moves along the horizontal axis and a pole that is anchored on the cart. The state space is continuous and it is represented by the x position of the cart, the cart velocity $\dot{x}$, the pole angle γ with respect to the vertical, the pole angular velocity $\dot{\gamma}$. The action space is discrete and composed by two actions: left L or right R. The model parameter is represented by the force ω to be applied to the cart, that is the same for both actions. The range of the force is $[0, 30]$. The resulting force is $\pm\omega$ depending on the action. We add noise on the resulting action proportional to the force applied and independent noise to each state component. The Cart-Pole environment is represented in Fig. 5.7.

The goal is to keep the pole in the vertical position ($\gamma = 0$) as long as possible. The episode ends when the pole reaches a certain angle ($|\gamma| > \bar{\gamma}$) or after a predefined number of steps. We want to encourage smaller forces, to this end we use the following reward function:

$$R(s, a, s') = 10 - \frac{\omega^2}{20} - 20 * (1 - \cos(\gamma)).$$

The first part is a fixed reward for each for each timestep the pole is up and the pole angle is inside the range $[-\bar{\gamma}, +\bar{\gamma}]$. The second part is a penalty proportional to the force. The third part is a penalty proportional the pole angle, it is 0 when the pole is in the vertical position. Ideally the agent should learn to balance the pole with the smaller force possible, keeping it fixed in the vertical position.

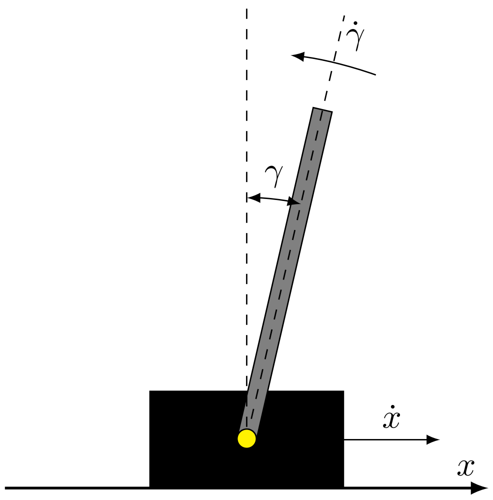


Figure 5.7: Cart-Pole environment representation.

Parameter	Value
Num of samples	10^5
Dual Regularization	0
Policy Regularization	0
ϵ	10^{-3}
Policy	Linear with softmax
ω_0	8

Table 5.2: Hyper-parameters used in the exact cartpole experiment.

Parameter	Value
Num of samples	$5 \cdot 10^4$
Dual Regularization	10^{-4}
Policy Regularization	0
ϵ	10^{-3}
Policy	Linear with softmax
ω_0	8

Table 5.3: Hyper-parameters used in the approximated cartpole experiment.

5.2.1 Results

We test the performance of REMPS both in the case of an exact model and in the case of an approximated (fitted) model. In Fig. 5.8 we show the performance of our algorithm starting from a fixed value of the model parameter, $\omega_0 = 8$. In the exact case, Fig. 5.8 (left), it is possible to see that the model parameter has a decreasing trend in the first iterations, then it converges to a minimum. The timesteps increase and then reach a maximum. The return increases since the agent is less penalized by the force term (since the model parameter is becoming smaller) and it is learning to balance the pole. In the approximated case, Fig. 5.8 (right), the cart-pole model is learned by a neural network (NN). The inputs of the NN are the state s_t, the action a_t and the model parameter ω. The output neurons represents the mean and the variance of a Gaussian distribution over the state space representing the probability of landing in a given state. We collect data with a fixed random policy before training. We fitt the dynamic model by maximizing the log-likelihood, obtaining a neural network predicting $P(s'|s, a, \omega)$. The model learns the effect of the model parameters on the dynamic. As baseline we use the G(PO)MDP algorithm over the model and policy parameters. Both in the exact version and in the approximated one, REMPS shows a more stable behaviour with respect to G(PO)MDP. In the exact case the two algorithms are comparable, the improvement of REMPS are slower but at the end it outperforms G(PO)MDP with precise selection of the model parameter.
In the approximated case (using the same approximation of the model) REMPS performs better, it is very stable compared to G(PO)MDP.

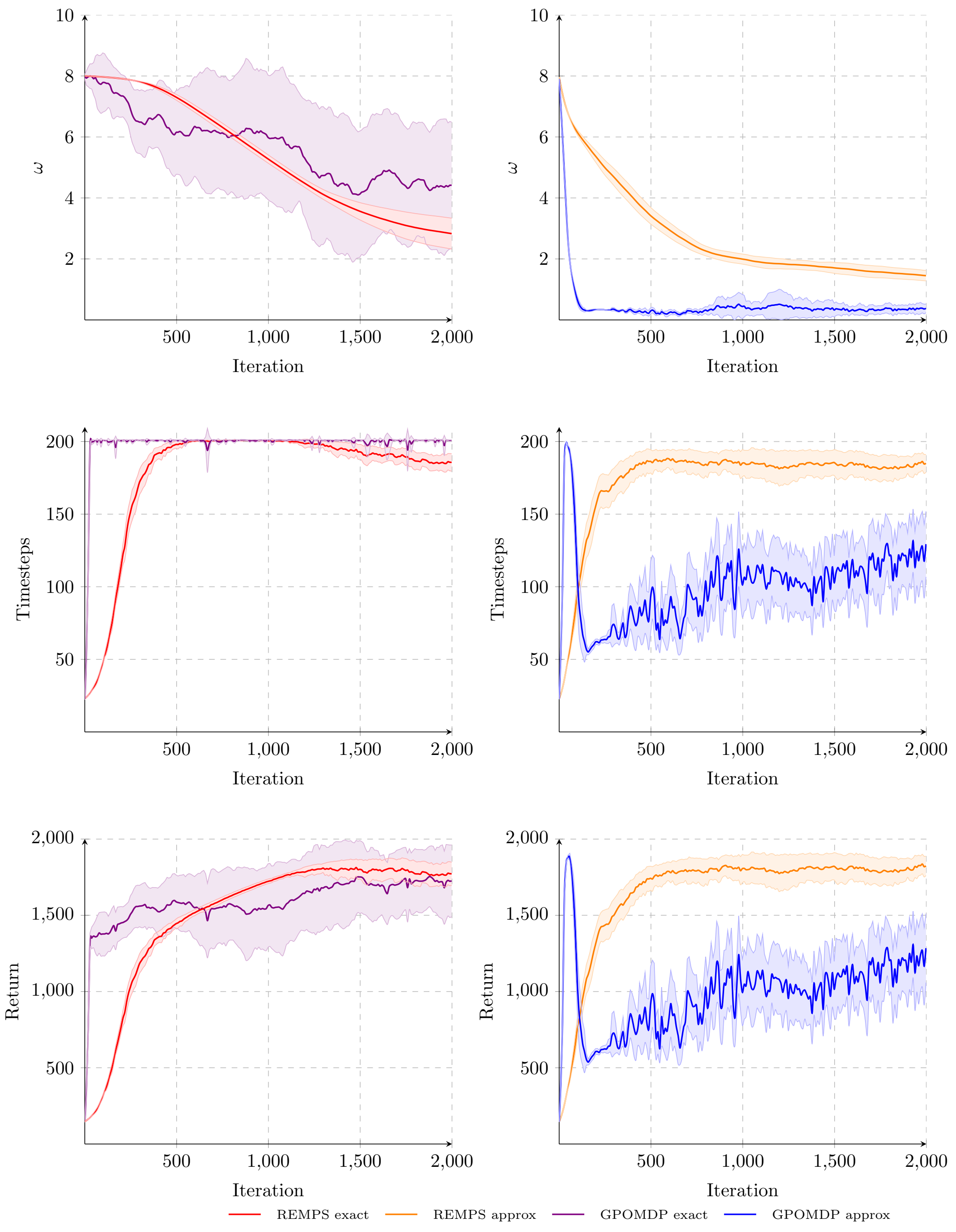


Figure 5.8: Cartpole experiment. Left: results of REMPS and GPOMDP using the exact model. Right: results of REMPS and GPOMDP using the approximated model. Top: model parameter ω. Middle: average timesteps per episode. Bottom: Average return. The shaded area represents the 95% confidence interval over twenty runs of the algorithm.

Parameter	Description
angle	Angle between the car direction and the direction of the track axis.
rpm	Number of rotation per minute of the car engine.
speedX	Speed of the car along the longitudinal axis of the car.
speedY	Speed of the car along the transverse axis of the car.
speedZ	Speed of the car along the Z axis of the car.
track	Vector of 19 range finder sensors: each sensors returns the distance between the track edge and the car within a range of 200 meters.
trackPos	Distance between the car and the track axis.
wheelSpinVel	Vector of 4 sensors representing the rotation speed of wheels.

Table 5.4: State space of the TORCS experiment.

5.3 Autonomous Driving and Configuration with TORCS

The Open Racing Car Simulator (Bernhard Wymann, 2013), TORCS, is a car racing simulation, which allows to simulate driving races. It is a complete 3D racing simulator offering a complete set of sensors and controls. TORCS has been used as RL environments in (Loiacono et al., 2010; Lillicrap et al., 2015b; Koutník et al., 2013; Mnih et al., 2016) and many others.
We modified the source code of TORCS adding the possibility to configure the car parameters following the "Car Setup Competition".[1] The goal of this experiment is to show the benefits of the environment configuration in the context of autonomous driving.

5.3.1 Environment Description

The state space of the TORCS environment is composed by 29 dimensions, $\mathcal{S} \subseteq \mathbb{R}^{29}$. The action space is composed by 2 dimensions, $\mathcal{A} \subseteq \mathbb{R}^2$. The first dimension of the action space is the acceleration/brake action, while the second dimension is the steering angle. The configuration space is very large, so we considered only a subset in our experiments. All configuration parameters are normalized in the range $[0, 1]$. The state space space is summarized in Table 5.4 and the configuration parameters in Table 5.5. We defined the reward function in the following way:

$$R(s, a, s') = \text{speedX}' \cdot \cos(\text{angle}'), \tag{5.3}$$

where speedX$'$ is the velocity on the longitudinal direction of the car in state s' and angle$'$ is the angle between the car direction and the direction of the track axis. We give a penalty of 1000 if the agent runs backward, if it goes out of track or if the progress in the race is too low. The rationale behind this reward is to encourage the agent to go at high speed and to stay centered with respect to the track.

[1]https://sourceforge.net/projects/cig

Parameter	Description
Rear Wing	Angle of the rear wing.
Front Wing	Angle of the front wing.
Front-Rear Brake Repartion	Repartition of the brake between the front and rear.
Front Anti-Roll Bar	Front Spring.
Rear Anti-Roll Bar	Rear Spring.
Front Left-Right Brake	Brake disk diameter of the front wheels.
Rear Left-Right Brake	Brake disk diameter of the rear wheels.

Table 5.5: Configuration space of the TORCS experiment.

The policy we used in our experiments is a Gaussian Policy parameterized by a fully connected neural network:

$$\pi(a|s) = \mathcal{N}(\boldsymbol{\phi_m}(s), \boldsymbol{\phi_v}), \tag{5.4}$$

where the mean $\boldsymbol{\phi_m}(s)$ is a non-linear function of the current state s and the variance $\boldsymbol{\phi_v}$ it is independent from the current state.
In the projection phase of REMPS we can only perform a disjoint projection of the policy and model (see Section 4.2.2) since the state and action spaces are continuous.

5.3.2 Results

In the first phase of our experiments we run a random policy in order to collect the dataset for model fitting. We fitted the model with a neural network using a predefined number of iterations as in Section 5.2. During the actual training phase we used the model network to approximate the system dynamics and we optimized the policy and the environment configuration. A comparison between the results obtained learning only the policy (REPS) and the results obtained learning the policy and the environment configuration (REMPS) are given in Fig. 5.9. Configuring the environment yields a performance boosting even in an environment with complex dynamic and even starting with a random policy, where the effect of the configuration is limited. We highlight the fact that we learn the model once using a fixed, random policy, while we could alternate model-policy optimization and model fitting for a more precise estimation.

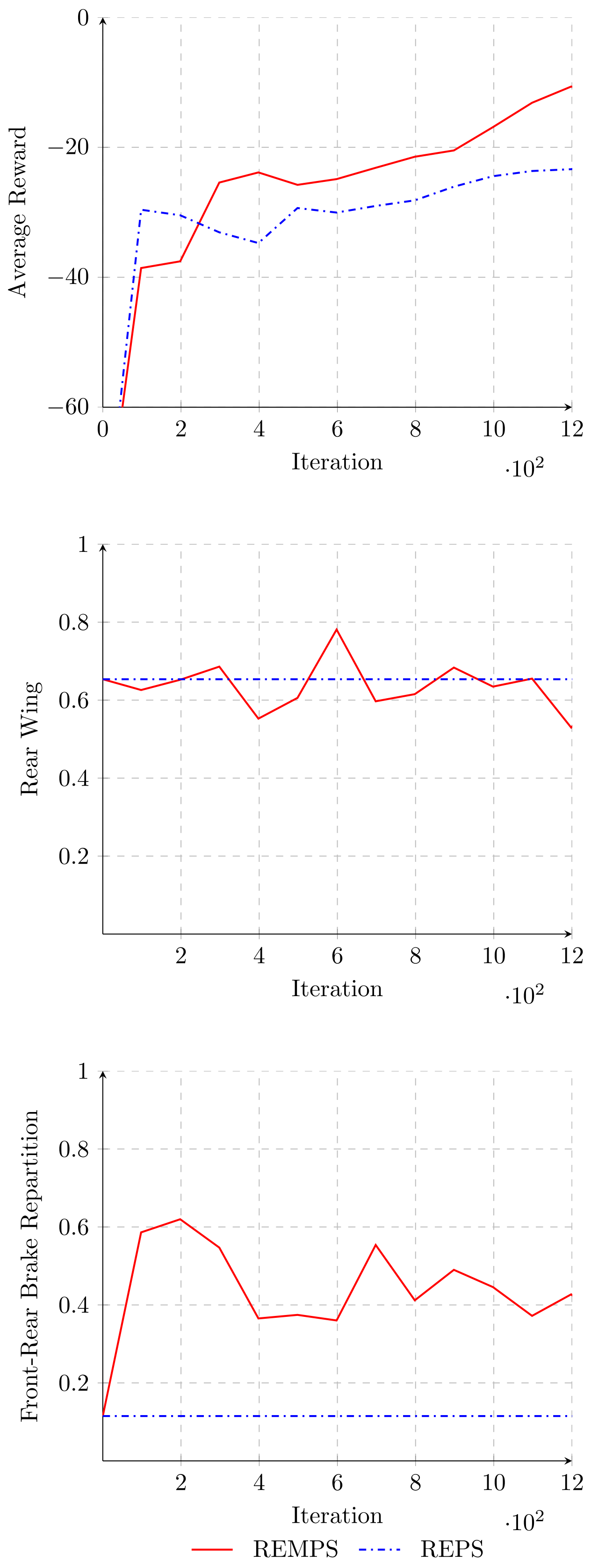


Figure 5.9: TORCS experiment. Comparison between policy learning and policy-configuration learning. Learning the configuration yields a performance improvement. Top: Average reward. Middle: Rear wing angle. Bottom: Front-Rear Brake repartition.

Chapter 6

Discussion and Conclusions

In this chapter, we discuss the main contribution of this thesis, and we propose some possible future extensions and refinement of the approaches presented in this work. The first contribution of this document is a new algorithm, namely **REMPS**, able to solve the model-policy learning problem in the context of CMDPs. REMPS is an extension of REPS (Peters et al., 2010) that considers also the model optimization. Along with REMPS we present three types of policy and model projection strategies in order to deal with limited representation power. REMPS is able to work with continuous state and action spaces, moreover it does not require the knowledge of the full transition model, requiring only an approximation.

We presented a theoretical study of the property of the algorithm, providing a lower bound to the difference between the performance of the solution found using an infinite number of sample and infinite representation power and the solution using a finite number of samples and finite representation power. We showed that this bound depends on the number of samples, on the representation power of the model and policy spaces and on the value of the KL–constraint ϵ.

Along with a theoretical study of the algorithm, we presented also an empirical evaluation. We tested our algorithm on three domains. The first domain (chain problem) is a proof of concept showing the ability of REMPS to overcome local minima. The second domain (cart-pole) is a standard RL benchmark with continuous action space, discrete action space. We used cart-pole to test the performances in both the exact model and in the approximated model scenarios. The last experiment is a complex experiment of autonomous driving and configuration using the TORCS environment. In our experiments we showed the benefits of model configuration and the benefits of using an information theoretical approach with respect to a gradient method such as G(PO)MDP.

We outline some possible extensions and research directions.

Adaptive ϵ The ϵ parameter in the KL constraint is critical since it is responsible for the magnitude of the update. A high value of ϵ results in too large model-policy updates, while a too small ϵ results in no updates at all. An optimal value for ϵ can be found using heuristics or using a principled way, optimizing some utility function.

Other divergences REMPS (and REPS) algorithm is based on the KL divergence. The usage of this kind of distance is justified by the closed form solution of the learning problem. However the KL distance has some problems. The first problem is that it is not symmetric, so after the model-policy projection we have no clue on the distance from the projected model and policy to the ones used for sampling. Moreover the use of KL distance requires the policy to be stochastic, since the KL between two deterministic policy is 0 or ∞. An interesting extension of our algorithm could use different types of distance, i.e. Rényi divergence, Hellinger or Weisserstein. We highlight the fact that using the Total Variation distance it is not possible to solve the REMPS optimization problem in closed form.

Policy Space Identification The presented approach to the CMDP learning problem is a joint approach, in which the supervisor and the agent are the same entity. In a realistic case the two entity can be different, i.e. an F1 pilot and a mechanical engineer. In these cases it can be beneficial to split the policy learning and the model learning. The agent should be freely allowed to learn using some strategy, the supervisor should study the agent behaviour proposing for each episode the best (according to some utility measure) environment configuration. In order to do this the supervisor should know the agent's policy space. However, in realistic cases the policy can be unknown. In these scenarios the supervisor should identify the policy space of the agent and perform optimizations using this approximation.

Finite–Time Analysis In our theoretical analysis we derived a bound valid for a single step of REMPS. An interesting theoretical extension would be an analysis for multiple steps of REMPS obtaining a bound on $J_{d^{(T)}} - J_{\widetilde{d}^{(T)}}$, where $d^{(T)}$ is the distribution obtained using infinite samples after T step of optimization and $\widetilde{d}^{(T)}$ is the solution obtained with N samples with limited capacity on the model and policy selection.

Bibliography

[Abdolmaleki et al. 2015] ABDOLMALEKI, Abbas ; LIOUTIKOV, Rudolf ; PETERS, Jan R. ; LAU, Nuno ; PUALO REIS, Luis ; NEUMANN, Gerhard: Model-Based Relative Entropy Stochastic Search. In: CORTES, C. (Ed.) ; LAWRENCE, N. D. (Ed.) ; LEE, D. D. (Ed.) ; SUGIYAMA, M. (Ed.) ; GARNETT, R. (Ed.): *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 3537–3545. pp. 3537–3545

[Amari 1998] AMARI, Shun-Ichi: Natural Gradient Works Efficiently in Learning. In: *Neural Comput.* 10 (1998), February, nr. 2, pp. 251–276. – 10 (1998), February, nr. 2, pp. 251–276

[Baxter and Bartlett 2001] BAXTER, Jonathan ; BARTLETT, Peter L.: Infinite-horizon policy-gradient estimation. In: *Journal of Artificial Intelligence Research* 15 (2001), pp. 319–350. 15 (2001), pp. 319–350

[Bellman 1957] BELLMAN, Richard: *Dynamic Programming*. Dover Publications, 1957 Dover Publications, 1957

[Bernhard Wymann 2013] BERNHARD WYMANN, Christophe Guionneau Christos Dimitrakakis Rémi Coulom Andrew S.: *TORCS, The Open Racing Car Simulator*. http://www.torcs.org. 2013

[Bowerman 1974] BOWERMAN, Bruce L.: *Nonstationary Markov decision processes and related topics in nonstationary Markov chains*, Iowa State University, PhD Thesis, 1974

[Cheevaprawatdomrong et al. 2007] CHEEVAPRAWATDOMRONG, Torpong ; SCHOCHETMAN, Irwin E. ; SMITH, Robert L. ; GARCIA, Alfredo: Solution and Forecast Horizons for Infinite-Horizon Nonhomogeneous Markov Decision Processes. In: *Math. Oper. Res.* 32 (2007), February, nr. 1, pp. 51–72. 32 (2007), February, nr. 1, pp. 51–72

[Cortes et al. 2010] CORTES, Corinna ; MANSOUR, Yishay ; MOHRI, Mehryar: Learning Bounds for Importance Weighting. In: LAFFERTY, J. D. (Ed.) ; WILLIAMS, C. K. I. (Ed.) ; SHAWE-TAYLOR, J. (Ed.) ; ZEMEL, R. S. (Ed.) ; CULOTTA, A. (Ed.): *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, pp. 442–450. – pp. 442–450

[Daniel et al. 2012] DANIEL, Christian ; NEUMANN, Gerhard ; PETERS, Jan: Hierarchical Relative Entropy Policy Search. In: LAWRENCE, Neil D. (Ed.) ; GIROLAMI, Mark (Ed.): *Proceedings of the Fifteenth International Conference on Artificial Intelligence* *and Statistics* vol. 22. La Palma, Canary Islands : PMLR, 21–23 Apr 2012, pp. 273–281. pp. 273–281

[Deisenroth 2011] DEISENROTH, Marc P.: A Survey on Policy Search for Robotics. In: *Foundations and Trends in Robotics* 2 (2011), nr. 1-2, pp. 1–142. 2 (2011), nr. 1-2, pp. 1–142

[d'Epenoux 1963] D'EPENOUX, F.: A Probabilistic Production and Inventory Problem. In: *Management Science* 10 (1963), nr. 1, pp. 98–108. 10 (1963), nr. 1, pp. 98–108

[Furmston and Barber 2012] FURMSTON, Thomas ; BARBER, David: A Unifying Perspective of Parametric Policy Search Methods for Markov Decision Processes. In: PEREIRA, F. (Ed.) ; BURGES, C. J. C. (Ed.) ; BOTTOU, L. (Ed.) ; WEINBERGER, K. Q. (Ed.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 2717–2725. pp. 2717–2725

[Garcia and Smith 2000] GARCIA, Alfredo ; SMITH, Robert L.: Solving Nonstationary Infinite Horizon Dynamic Optimization Problems. In: *Journal of Mathematical Analysis and Applications* 244 (2000), nr. 2, pp. 304 – 317. 244 (2000), nr. 2, pp. 304 – 317

[Ghate and L. Smith 2013] GHATE, Archis ; L. SMITH, Robert: A Linear Programming Approach to Nonstationary Infinite-Horizon Markov Decision Processes. 61 (2013), 04, pp. 413–425. 61 (2013), 04, pp. 413–425

[Givan et al. 2000] GIVAN, Robert ; LEACH, Sonia ; DEAN, Thomas: Bounded-parameter Markov decision processes. In: *Artificial Intelligence* (2000), pp. 39. (2000), pp. 39

[Harmanec 2002] HARMANEC, David: Generalizing Markov decision processes to imprecise probabilities. In: *Journal of Statistical Planning and Inference* 105 (2002), June, nr. 1, pp. 199–213. 105 (2002), June, nr. 1, pp. 199–213

[Hopp et al. 1987] HOPP, Wallace J. ; BEAN, James C. ; SMITH, Robert L.: A New Optimality Criterion for Nonhomogeneous Markov Decision Processes. In: *Operations Research* 35 (1987), nr. 6, pp. 875–883. 35 (1987), nr. 6, pp. 875–883

[Howard 1960] HOWARD, R. A.: *Dynamic Programming and Markov Processes*. Cambridge, MA : MIT Press, 1960 Cambridge, MA : MIT Press, 1960

[Kakade and Langford 2002] KAKADE, Sham ; LANGFORD, John: Approximately Optimal Approximate Reinforcement Learning. In: *Proceedings of the Nineteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2002 (ICML '02), pp. 267–274. – pp. 267–274

[Kakade 2002] KAKADE, Sham M.: A Natural Policy Gradient. In: DIETTERICH, T. G. (Ed.) ; BECKER, S. (Ed.) ; GHAHRAMANI, Z. (Ed.): *Advances in Neural Information Processing Systems 14*. MIT Press, 2002, pp. 1531–1538. pp. 1531–1538

[Kober et al. 2013] KOBER, Jens ; BAGNELL, J. A. ; PETERS, Jan: Reinforcement learning in robotics: A survey. In: *The International Journal of Robotics Research* 32 (2013), nr. 11, pp. 1238–1274. 32 (2013), nr. 11, pp. 1238–1274

[Koutník et al. 2013] KOUTNÍK, Jan ; CUCCU, Giuseppe ; SCHMIDHUBER, Jürgen ; GOMEZ, Faustino: Evolving Large-scale Neural Networks for Vision-based Reinforcement Learning. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA : ACM, 2013 (GECCO '13), pp. 1061–1068. – pp. 1061–1068

[Kupcsik et al. 2013] KUPCSIK, Andras G. ; DEISENROTH, Marc P. ; PETERS, Jan ; NEUMANN, Gerhard: Data-Efficient Generalization of Robot Skills with Contextual Policy Search. (2013), pp. 7. (2013), pp. 7

[Lillicrap et al. 2015a] LILLICRAP, Timothy P. ; HUNT, Jonathan J. ; PRITZEL, Alexander ; HEESS, Nicolas ; EREZ, Tom ; TASSA, Yuval ; SILVER, David ; WIERSTRA, Daan: Continuous control with deep reinforcement learning. In: *CoRR* abs/1509.02971 (2015). – abs/1509.02971 (2015)

[Lillicrap et al. 2015b] LILLICRAP, Timothy P. ; HUNT, Jonathan J. ; PRITZEL, Alexander ; HEESS, Nicolas ; EREZ, Tom ; TASSA, Yuval ; SILVER, David ; WIERSTRA, Daan: Continuous control with deep reinforcement learning. In: *CoRR* abs/1509.02971 (2015). – abs/1509.02971 (2015)

[Loiacono et al. 2010] LOIACONO, D. ; PRETE, A. ; LANZI, P. L. ; CARDAMONE, L.: Learning to overtake in TORCS using simple reinforcement learning. In: *IEEE Congress on Evolutionary Computation*, July 2010, pp. 1–8. – pp. 1–8

[Metelli et al. 2018a] METELLI, Alberto M. ; MUTTI, Mirco ; RESTELLI, Marcello: Configurable Markov Decision Processes. In: DY, Jennifer (Ed.) ; KRAUSE, Andreas (Ed.): *Proceedings of the 35th International Conference on Machine Learning* vol. 80. Stockholmsmässan, Stockholm Sweden : PMLR, 10–15 Jul 2018, pp. 3488–3497. pp. 3488–3497

[Metelli et al. 2018b] METELLI, Alberto M. ; PAPINI, Matteo ; FACCIO, Francesco ; RESTELLI, Marcello: Policy Optimization via Importance Sampling. In: *arXiv:1809.06098 [cs, stat]* (2018), September. (2018), September

[Mnih et al. 2016] MNIH, Volodymyr ; BADIA, Adria P. ; MIRZA, Mehdi ; GRAVES, Alex ; LILLICRAP, Timothy ; HARLEY, Tim ; SILVER, David ; KAVUKCUOGLU, Koray: Asynchronous Methods for Deep Reinforcement Learning. In: BALCAN, Maria F. (Ed.) ; WEINBERGER, Kilian Q. (Ed.): *Proceedings of The 33rd International Conference on Machine Learning* vol. 48. New York, New York, USA : PMLR, 20–22 Jun 2016, pp. 1928–1937. – pp. 1928–1937

[Mnih et al. 2013] MNIH, Volodymyr ; KAVUKCUOGLU, Koray ; SILVER, David ; GRAVES, Alex ; ANTONOGLOU, Ioannis ; WIERSTRA, Daan ; RIEDMILLER, Martin A.: Playing Atari with Deep Reinforcement Learning. In: *CoRR* abs/1312.5602 (2013). – abs/1312.5602 (2013)

[Mnih et al. 2015] MNIH, Volodymyr ; KAVUKCUOGLU, Koray ; SILVER, David ; RUSU, Andrei A. ; VENESS, Joel ; BELLEMARE, Marc G. ; GRAVES, Alex ; RIEDMILLER, Martin ; FIDJELAND, Andreas K. ; OSTROVSKI, Georg ; PETERSEN, Stig ; BEATTIE, Charles ; SADIK, Amir ; ANTONOGLOU, Ioannis ; KING, Helen ; KUMARAN, Dharshan ; WIERSTRA, Daan ; LEGG, Shane ; HASSABIS, Demis: Human-level control through deep reinforcement learning. In: *Nature* 518 (2015), 02, pp. 529 EP –. – 518 (2015), 02, pp. 529 EP –

[Mohri et al. 2012] MOHRI, M. ; ROSTAMIZADEH, A. ; TALWALKAR, A.: *Foundations of Machine Learning*. MIT Press, 2012 (Adaptive computation and machine learning series). – (Adaptive computation and machine learning series)

[Moody et al. 1998] MOODY, John ; WU, Lizhong ; LIAO, Yuansong ; SAFFELL, Matthew: Performance functions and reinforcement learning for trading systems and portfolios. In: *Appears in Journal of Forecasting* 17 (1998), 09, pp. 441–470. 17 (1998), 09, pp. 441–470

[Peters and Schaal 2008a] PETERS, J. ; SCHAAL, S.: Natural Actor-Critic. In: *Neurocomputing* 71 (2008), mar, nr. 7-9, pp. 1180–1190. 71 (2008), mar, nr. 7-9, pp. 1180–1190

[Peters et al. 2010] PETERS, Jan ; MULLING, Katharina ; ALTUN, Yasemin: *Relative Entropy Policy Search*. 2010

[Peters and Schaal 2008b] PETERS, Jan ; SCHAAL, Stefan: Reinforcement learning of motor skills with policy gradients. In: *Neural Networks* 21 (2008), nr. 4, pp. 682 – 697. – 21 (2008), nr. 4, pp. 682 – 697

[Pirotta et al. 2013] PIROTTA, Matteo ; RESTELLI, Marcello ; PECORINO, Alessio ; CALANDRIELLO, Daniele: Safe Policy Iteration. In: DASGUPTA, Sanjoy (Ed.) ; MCALLESTER, David (Ed.): *Proceedings of the 30th International Conference on Machine Learning* vol. 28. Atlanta, Georgia, USA : PMLR, 17–19 Jun 2013, pp. 307–315. – pp. 307–315

[Puterman 1994] PUTERMAN, Martin L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA : John Wiley & Sons, Inc., 1994 New York, NY, USA : John Wiley & Sons, Inc., 1994

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

[Rényi 1961] RÉNYI, Alfréd: On Measures of Entropy and Information. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif. : University of California Press, 1961, pp. 547–561. – pp. 547–561

[Schulman et al. 2015] SCHULMAN, John ; LEVINE, Sergey ; MORITZ, Philipp ; JORDAN, Michael I. ; ABBEEL, Pieter: Trust Region Policy Optimization. In: *arXiv:1502.05477 [cs]* (2015), February. (2015), February

[Schulman et al. 2017] SCHULMAN, John ; WOLSKI, Filip ; DHARIWAL, Prafulla ; RADFORD, Alec ; KLIMOV, Oleg: Proximal Policy Optimization Algorithms. In: *arXiv:1707.06347 [cs]* (2017), July. (2017), July

[Silver et al. 2016] SILVER, David ; HUANG, Aja ; MADDISON, Chris J. ; GUEZ, Arthur ; SIFRE, Laurent ; DRIESSCHE, George van den ; SCHRITTWIESER, Julian ; ANTONOGLOU, Ioannis ; PANNEERSHELVAM, Veda ; LANCTOT, Marc ; DIELEMAN, Sander ; GREWE, Dominik ; NHAM, John ; KALCHBRENNER, Nal ; SUTSKEVER, Ilya ; LILLICRAP, Timothy ; LEACH, Madeleine ; KAVUKCUOGLU, Koray ; GRAEPEL, Thore ; HASSABIS, Demis: Mastering the game of Go with deep neural networks and tree search. In: *Nature* 529 (2016), 01, pp. 484 EP –. – 529 (2016), 01, pp. 484 EP –

[Silver et al. 2017] SILVER, David ; SCHRITTWIESER, Julian ; SIMONYAN, Karen ; ANTONOGLOU, Ioannis ; HUANG, Aja ; GUEZ, Arthur ; HUBERT, Thomas ; BAKER, Lucas ; LAI, Matthew ; BOLTON, Adrian ; CHEN, Yutian ; LILLICRAP, Timothy ; HUI, Fan ; SIFRE, Laurent ; DRIESSCHE, George van den ; GRAEPEL, Thore ; HASSABIS, Demis: Mastering the game of Go without human knowledge. In: *Nature* 550 (2017), 10, pp. 354 EP –. – 550 (2017), 10, pp. 354 EP –

[Sutton and Barto 1998] SUTTON, Richard S. ; BARTO, Andrew G.: *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA : MIT Press, 1998 Cambridge, MA, USA : MIT Press, 1998

[Sutton et al. 1999] SUTTON, Richard S. ; MCALLESTER, David ; SINGH, Satinder ; MANSOUR, Yishay: Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. Cambridge, MA, USA : MIT Press, 1999 (NIPS'99), pp. 1057–1063. pp. 1057–1063

[Van Erven and Harremos 2012] VAN ERVEN, Tim ; HARREMOS, Peter: Rényi Divergence and Kullback-Leibler Divergence. In: *CoRR* abs/1206.2459 (2012). – abs/1206.2459 (2012)

[Williams 1992] WILLIAMS, Ronald J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: *Machine Learning* 8 (1992), May, nr. 3, pp. 229–256. 8 (1992), May, nr. 3, pp. 229–256

Appendix A

Proof of Linear Programming Formulation

This appendix provides the proof of Theorem 2.2 that we report here for completeness:

$$\begin{aligned} &\underset{v}{\text{minimize}} \sum_{s\in\mathcal{S}} \mu(s)v(s) \\ &\text{subject to } v(s) \geq r(s,a) + \sum_{s'\in\mathcal{S}} P(s' \mid s,a)v(s')\,, \forall a \in \mathcal{A}\,. \end{aligned}$$

Theorem (Linear Programming Solution) v^* *is the solution of the above linear program.*

Proof. Let T^* be the Bellman optimality operator, then the above LP can be rewritten as:

$$\begin{aligned} &\underset{v}{\text{minimize}}\ \mu^T v \\ &\text{subject to } v \geq T^*(v)\,. \end{aligned}$$

Using the *monotonicity property* if $v \geq T^*(v)$, then $T^*(v) \geq T^*(T^*(v))$, and by applying infinite times the operator we obtain: $v \geq T^{*\infty}(v) = v^*$. Any feasible solution of the LP must satisfy $v \geq T^*(v)$, thus it must satisfy $v \geq v^*$. Hence, assuming all entries μ are positive, v^* is the optimal solution to the LP. $\square$

Appendix B

Gradient Methods for CMDP

In this section we provide the straightforward extension of REINFORCE and G(PO)DMP gradient estimation optimizing the policy and the configuration parameters.

B.1 REINFORCE

Let us start by stating the expression of the gradient of the expected return with respect to a parametric transition model differentiable in its parameters $\boldsymbol{\omega}$.

Theorem B.1.1 (P-Gradient Theorem, from (Metelli et al., 2018a)) *Let P_ω be a class of parametric stochastic transition models differentiable in $\boldsymbol{\omega}$, π be the current policy, the gradient of the expected return with respect to $\boldsymbol{\omega}$ is given by:*

$$\nabla_{\boldsymbol{\omega}} J^{P,\pi} = \int_{\mathcal{S}} \int_{\mathcal{A}} d^{P,\pi}(s,a) \int_{\mathcal{S}} \nabla_{\boldsymbol{\omega}} P_{\boldsymbol{\omega}}(s'|s,a) u^{P,\pi}(s,a,s') \mathrm{d}s' \mathrm{d}a \mathrm{d}s$$

We can now derive in a straightforward manner the REINFORCE estimator for model learning:

$$\widehat{\nabla_{\boldsymbol{\omega}} J^{P,\pi}}_{RF} = \langle \left(\sum_{k=0}^{H} \nabla_{\boldsymbol{\omega}} \log P_{\boldsymbol{\omega}}(s_{k+1}|s_k,a_k) \right) \left(\sum_{k=0}^{H} \gamma^k R(s_k,a_k,s_{k+1}) \right) \rangle_N \,, \qquad \text{(B.1)}$$

where $\langle \cdot \rangle_N$ denotes the empirical average over a batch size of dimension N.

B.2 G(PO)DMP

In order to derive the G(PO)MDP estimator for model learning we start from a trajectory based perspective:

$$J^{P,\pi} = \int p_{\boldsymbol{\theta},\boldsymbol{\omega}}(\tau) G(\tau) \mathrm{d}\tau,$$

where $p_{\boldsymbol{\theta},\boldsymbol{\omega}}(\tau)$ is the probability of the trajectory τ under the distribution induced by the parameters $\boldsymbol{\theta}, \boldsymbol{\omega}$ and $G(\tau)$ is the return of the trajectory τ. Using the log-trick and taking the derivative with respect to the model parameters we obtain:

$$\nabla_{\boldsymbol{\omega}} J^{P,\pi} = \int p_{\boldsymbol{\theta},\boldsymbol{\omega}}(\tau) \nabla_{\boldsymbol{\omega}} \log p(\tau) G(\tau) \mathrm{d}\tau \tag{B.2}$$

$$= \int p_{\boldsymbol{\theta},\boldsymbol{\omega}}(\tau) \left(\sum_{k=0}^{H} \log P_{\boldsymbol{\omega}}(s_{k+1}|s_k, a_k) \right) G(\tau). \tag{B.3}$$

Now we are exactly in the G(PO)MDP settings and we can use the following approximation of the gradient:

$$\widehat{\nabla_{\boldsymbol{\omega}} J^{P\pi}}_{G(PO)MDP} = \langle \sum_{l=0}^{H} \left(\sum_{k=l}^{H} \nabla_{\boldsymbol{\omega}} \log P_{\boldsymbol{\omega}}(s_{k+1}|s_k, a_k) \right) \left(\gamma^l R(s_l, a_l, s_{l+1}) \right) \rangle_N \, , \tag{B.4}$$

where $\langle \cdot \rangle_N$ denotes the empirical average over a batch size of dimension N.

Appendix C

REMPS derivation

This appendix provides the derivation of the REMPS solution. We report here the formulation of the REMPS problem. For the sake of brevity we use $\mathcal{X} = \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ and $(s, a, s') = x \in \mathcal{X}$, moreover we indicate with $d(\cdot)$ the optimized distribution and with $d^{P,\pi}(\cdot)$ the sampling distribution.

$$\underset{d}{\text{maximize}} \int_{\mathcal{X}} d(x)R(x) \tag{C.1}$$

$$\text{subject to} \int_{\mathcal{X}} d(x) \log \frac{d(x)}{d^{P,\pi}(x)} \mathrm{d}x \leq \epsilon \tag{C.2}$$

$$\int_{\mathcal{X}} d(x) \mathrm{d}x = 1 \ . \tag{C.3}$$

We solve the problem with Lagrangian multipliers. We denote with η the langrangian multiplier associated with the KL constraint and with λ the multiplier associated with the constraint of being a valid distribution.

$$\mathcal{L}(d(\cdot), \eta, \lambda) = \int_{\mathcal{X}} d(x)R(x)\mathrm{d}x + \tag{C.4}$$

$$+ \eta \left(\epsilon - \int_{\mathcal{X}} d'(x) \log \frac{d(x)}{d^{P,\pi}(x)} \mathrm{d}x \right) + \tag{C.5}$$

$$+ \lambda \left(1 - \int_{\mathcal{X}} d(x) \mathrm{d}x \right) \ . \tag{C.6}$$

Observe that $\frac{\partial}{\partial f(x_0)} \int f(x)g(x)dx = g(x_0)$. So we take the derivative with respect to $d(x)$ to get:

$$R(x) - \eta \log \frac{d(x)}{d^{P,\pi}(x)} + \eta - \lambda = 0 \, , \tag{C.7}$$

from which we get, solving for $d(x)$:

$$d(x) = d^{P,\pi}(x) \exp\left(\frac{R(x)}{\eta}\right) \exp\left(1 - \frac{\lambda}{\eta}\right) \ . \tag{C.8}$$

By enforcing the constraint that d should be a valid distribution we obtain:

$$d(x) = \frac{d^{P,\pi}(x) \exp\left(\frac{R(x)}{\eta}\right)}{\int_{\mathcal{X}} d^{P,\pi}(x) \exp\left(\frac{R(x)}{\eta}\right) \mathrm{d}x} . \tag{C.9}$$

Substituting into the Lagrangian function (C.6), we obtain the dual function:

$$\begin{aligned} g(\eta, \lambda) &= -\eta + \eta\epsilon + \lambda && \text{(C.10)} \\ &= \eta \log \left(\int_{\mathcal{X}} d^{P,\pi}(x) \exp\left(\epsilon + \frac{R(x)}{\eta}\right) \mathrm{d}x \right) . && \text{(C.11)} \end{aligned}$$

From (C.8) we extract the policy and model inducing the distribution d. We return to the original formulation for the sake of clarity.

$$\begin{aligned} \pi'(a|s) &= \frac{\int_{\mathcal{S}} d(s,a,s')\mathrm{d}s'}{\int_{\mathcal{A}} \int_{\mathcal{S}} d(s,a,s')\mathrm{d}s'\mathrm{d}a} && \text{(C.12)} \\ &= \frac{\pi(a|s) \int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(x)}{\eta}\right) \mathrm{d}s'}{\int_{\mathcal{A}} \pi(a|s) \int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(x)}{\eta}\right) \mathrm{d}s'\mathrm{d}a}, && \text{(C.13)} \\ P'(s'|a,s) &= \frac{d(s,a,s')}{\int_{\mathcal{S}} d(s,a,s')\mathrm{d}s'} && \text{(C.14)} \\ &= \frac{P(s'|s,a) \exp\left(\epsilon + \frac{R(x)}{\eta}\right)}{\int_{\mathcal{S}} P(s'|s,a) \exp\left(\epsilon + \frac{R(x)}{\eta}\right) \mathrm{d}s'} . && \text{(C.15)} \end{aligned}$$