POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer science and engineering

Transfer Learning for Actor-Critic methods in Lipschitz Markov Decision Processes

Supervisor:

PROF. MARCELLO RESTELLI

Assistant Supervisors:

DOTT. MATTEO PIROTTA

ING. ANDREA TIRINZONI

Master Graduation Thesis by:

DANIEL FELIPE VACCA MANRIQUE

Student Id n. 852802

Academic Year 2016-2017

A Mis Increíbles

ACKNOWLEDGMENTS

Ringrazio il Prof. Marcello Restelli per avermi dato questa prima opportunità di contatto con il mondo della ricerca, e per la sua guida e il suo sostegno, costanti lungo questo percorso, senza i cui non avrei conseguito questo risultato. Ringrazio il Dott. Matteo Pirotta che, con la sua importante esperienza, ha anche contribuito in maniera significativa al successo di questo lavoro, nonostante le distanze fisiche che ci separavano. Voglio anche ringraziare Andrea Tirinzoni che, pur essendo presente solo negli ultimi mesi, è stato sempre disponibile a darmi una mano quando ne avevo bisogno. Grazie ancora a tutti e tre per quest'esperienza di crescita professionale.

También agradezco a Mis Increíbles, cuyo apoyo nunca faltó desde que empecé con este loco sueño de venir a Italia 5 años atrás; más que el soporte económico, el impulso moral que me dieron todo este tiempo fue lo que me trajo hasta el final. Gracias al resto de mi familia y mis amigos en Colombia, que siempre creyeron en mí para lograr esta meta.

Gracias, en fin, a todos los que, durante estos dos años en Milán, participaron en lo que ha sido la mejor experiencia de mi vida.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

RL	Reinforcement Learning
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
MRP	Markov Reward Process
MC	Monte Carlo
FIM	Fisher Information Matrix
TD	Temporal difference
MSE	Mean Squared Error
MSBE	Mean Squared Bellman Error
MSTDE	Mean Squared Temporal Difference Error
LSTD	Least Squares Temporal Difference
MSPBE	Mean Squared Projected Bellman Error
NEU	Norm of Expected TD Update
OPE	Operator Error
FPE	Fixed-Point Error
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
IS	Importance Sampling
ESS	Effective Sample Size
TL	Transfer Learning
PLC	Pointwise Lipschitz Continuous

ABSTRACT

Reinforcement Learning (RL) is one of the most prominent frameworks for designing artificial agents when the only source of knowledge is the interaction with an environment. Special difficulties arise when the state and action spaces are continuous since the classical RL algorithms are no longer feasible or ensured to converge. Actor-critic approaches emerge as a solution for these issues by combining the proven convergence of policy-gradient methods with the representation power of function approximators, all in a low variance estimation.
Transfer Learning (TL) is the paradigm that addresses the problem of transferring past experience from different tasks when the agent is facing a new, unknown one. Its purpose is to develop algorithms that speed-up the new learning process by leveraging the past knowledge. Various advances concerning TL in RL have been developed in the recent years, but most of the proposals focus on transfer higher level information like value functions, policies or feature maps.
The Lipschitz continuity property, a stronger notion of continuity that concerns the elements of the task, can provide enough information to create sample-level transfer mechanisms to extend the Actor-Critic methods. This thesis introduces two such mechanisms based on weighted estimators: one performs an optimistic selection of the weights and the other goes for a pessimistic perspective. Both techniques are compared with a transfer mechanism based on Importance Sampling (IS) estimators. The optimistic approach produces good results in most of the experimental cases, showing that it is a promising alternative over the IS-based one and the no transfer scenario. The pessimistic approach, instead, tends to be too conservative on the selection of the weights, and offers no special improvements with respect to the no transfer scenario.

ESTRATTO

L'Apprendimento per rinforzo (reinforcement learning) è uno dei più noti paradigmi per la progettazione di agenti intelligenti in contesti dove l'unica fonte di informazione è l'interazione con l'ambiente. Particolari difficoltà emergono quando gli spazi di stato e azione sono continui, dato che gli algoritmi classici di RL non sono più applicabili o non hanno nessuna garanzia di convergenza. Gli approcci Actor-Critic nascono come una soluzione a questi problema, combinando la convergenza dei metodi policy-gradient con le abilità rappresentative dei function approximators, avendo come risultato uno stimatore a bassa varianza.
L'Apprendimento per trasferimento d'informazione (transfer learning) è il paradigma che studia il problema della trasmissione dell'esperienza ottenuta dall'interazione con diversi ambienti nell'apprendimento di un nuovo compito. Il principale obiettivo è lo sviluppo di algoritmi che velocizzino il nuovo processo d'apprendimento utilizzando le conoscenze passate. Varie proposte nell'ambito di TL in RL sono state presentate negli anni recenti, ma quasi tutte si concentrano sul trasferimento d'informazioni di alto livello come value functions, policies o feature maps.
La proprietà di continuità Lipschitziana, una nozione più forte di continuità che riguarda gli elementi dell'ambiente, può dare sufficiente informazione per creare meccanismi di trasferimento a livello di singoli campioni per i metodi Actor-Critic. Questa tesi propone due di tali meccanismi basati su stimatori pesati: uno fa una selezione ottimistica dei pesi mentre l'altro sceglie una prospettiva pessimistica. I due metodi sono confrontati con un meccanismo di trasferimento basato su stima tramite Importance Sampling. L'approccio ottimistico produce buoni risultati in quasi tutti gli scenari sperimentali, rivelandosi un'alternativa promettente ad IS e all'apprendimento senza trasferimento. L'approccio pessimistico risulta invece essere troppo conservativo nella scelta dei pesi, e non offre speciali vantaggi rispetto all'apprendimento classico.

1 INTRODUCTION

The purpose of Artificial Intelligence is to develop techniques that make computers to resemble human's mind as close as possible. While trying to clear the fuzziness of this goal, various different paradigms have been proposed. One of the most popular ones is based on the concept of *goal-oriented artificial agents* existing within a specific environment (Russell and Norvig, 2003). This paradigm is based on three important elements:

- The *agent*, the human-like behaving entity. It relies on *actions* that can be executed on trying to reach the goal;
- The *environment*, the world the agent exists in. It refers to everything that is not part of the agent and that is, therefore, out of its control. The *state* of the environment describes the conditions of the relevant elements of the world;
- The *goal* of the agent, a description of the situation(s) involving the agent and the environment that the agent expects to reach.

The agent is supposed to find the appropriate way to behave such that the interaction with the environment leads to the fulfillment of the goal.
The interaction between the agent and the environment is governed by the following mechanism: the agent executes an action on the environment, the environment reacts to the action by possibly changing its state, the agent observes the new state and executes a new action, and so on; this produces a chain of states and actions that represents an *instance* of the interaction process. When the states observed by the agent along the process coincide entirely with the real states of the environment, there is *full observability*; if the state observed by the agent contains only a fraction of the information of the state of the environment, there is *partial observability*. In order to achieve the goal, the agent will try to find a *strategy* that tells it what actions are to be performed based on the previous history of interaction and the current observed state of the environment, so that such goal is reached. Figure 1.1 shows a graphical representation of this process.
This strategy-finding problem (referred to as the *agent problem* from now on) has been approached from several different perspectives along time (Russell and Norvig, 2003). Most of them rely on having prior knowledge about the environment that can help in designing the agent. However, there is a number of real-life problems in which such knowledge is insufficient or even absent at all, making those approaches unsuitable. Within this context, the Reinforcement Learning

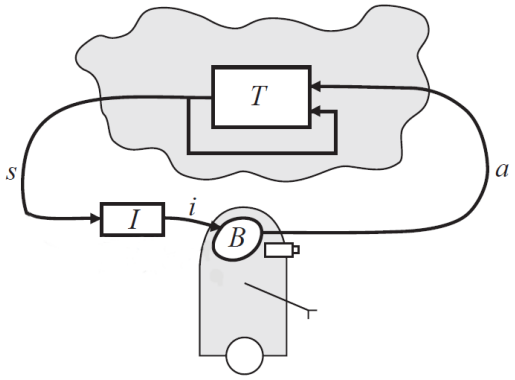


Figure 1.1: The agent-environment interaction model. Through actions, the agent produces changes in the environment's state, which are can be observed either fully or partially (adapted from (Kaelbling, Littman, and Moore, 1996)).

(RL) (Sutton and Barto, 1998) framework has emerged as a successful attempt to solve the agent problem by leveraging the data coming from experience as the only source of knowledge.

RL solves the agent problem by adding assumptions to simplify the modeling of the environment and the agent, and by leveraging the idea of utility function as the modeling tool for the agent's goal. As a consequence, the problem acquires a recursive optimal structure which allows to develop particularly interesting solutions. A family of them, the value based techniques (Kober, Bagnell, and Peters, 2013), strongly exploits this property and derives the optimal policy (strategy) directly from the utility functions; the core of these techniques is the policy evaluation step, which consists of assessing the current strategy of the agent. On the other hand, policy based methods (Deisenroth, Neumann, and Peters, 2013) directly search the space of policies when trying to find a solution; the most widely spread subfamily of these is the policy gradient methods (Peters and Schaal, 2008b), which assume a parametric family of policies and optimize the utility by following its gradient with respect to those parameters. Actor-critic methods (Konda and Tsitsiklis, 2003) are yet another family of approaches that try to merge value based and policy based algorithms on an attempt to overcome their drawbacks; their power lies on the separation of the policy assessment (performed by the so called critic) and the policy optimization (performed by the so called actor) steps.

Various real-life contexts where the agent problem arises have yet another particular source of complexity regarding implementation details: the set of states of the environment and the set of actions of the agent have a continuous nature. RL tackles this issue by resorting to function approximation for the utility functions and parametric spa-

ces of functions for the policies. Under these conditions, actor-critic approaches result in a very suitable solution.

While pursuing the goal of building human-behaving computers, the agent problem can be further extended to scenarios in which there is more than one environment to interact with. In these cases, the agent is not only expected to learn how to behave optimally in all the environments, but also to identify how they are related so that the experience can be leveraged across them.
Within the RL framework, these types of problems are classified as Transfer Learning (TL) problems (Taylor and Stone, 2009). An important instance of TL-RL problems considers an agent that has deeply interacted with a set of environments (called source tasks) and is now facing a new unknown environment (called target task). Under the assumption that all the tasks share some similarities, the goal of the agent is to transfer relevant knowledge from the sources to speed-up learning in the target.
The field of robotics is an example that can find TL-RL solutions to be particularly useful. Actor-critic algorithms with policy-gradient actors and function-approximators based critics are suitable due to the inherent continuous nature of these problems; in fact, successful results have been obtained in practice (Grondman et al., 2012). Because of the costs related to the collection of experience in these environments, TL-based techniques raise as a promising alternative to obtain highly performing agents at lower interaction requirements, so that there is no need to redo all the training when the environment changes.

1.1 MOTIVATION

The design and implementation of the different TL-RL techniques is strongly influenced by the assumptions regarding the task similarities; these dictate what type of knowledge can be transferred (e. g.utility functions, policy parameters, feature maps, etc.) and how it is transferred. An important example is the family of continuous tasks that satisfy stronger smoothness properties, namely the Lipschitz continuity, not only within the task elements but also across different tasks. Such similarity property is a good source of information for the agent during the transfer process.
One of the most important open issues in TL is negative transfer, i. e., the negative impact on the learning process caused by the transfer of useless source knowledge into the target task. Most techniques do not directly address this issue because they consider sufficiently safe similarity assumptions. The Lipschitz continuity property, however, might offer sufficient information to the agent not only for transferring but also for adequately selecting what to transfer.

1.2 GOAL

This thesis considers a Transfer Learning-Reinforcement Learning scenario where the transition and rewards models of the tasks satisfy the Lipschitz continuity property. The purpose is to deeply exploit this property for the design of a transfer mechanism that provides not only a speed-up for learning in the target task, but also robustness to negative transfer. Furthermore, the continuity nature of the problem and the local similarity information provided by the Lipschitz conditions justify mechanisms that extend actor-critic algorithms based on the transfer of the most basic pieces of experience, i. e., on samples.

1.3 CONTRIBUTION

The result of the work developed along this thesis is an algorithm that implements an actor-critic approach based on weighted estimators for the transfer scenario. The purpose of the weights is to correct the source samples' contribution to the target so that they indeed represent an improvement in the estimation.
Two alternatives are proposed for the selection of the weights: an optimistic one, which aims to reduce the estimated error induced by the transfer, and a pessimistic one, which aims to reduce an overestimation of the estimated error. Both approaches are compared to a well-known family of weighted estimators, the IS estimators, which are known to be unbiased, but suffer from a high variance.
The pessimistic approach offers no significant improvements for the transfer scenario, and can be easily affected by negative transfer. The optimistic approach, instead, happens to offer the same performance than the IS-based one, while reducing its variance and dropping the need for the target task model.

1.4 OUTLINE

The work in this thesis is organized along well-defined chapters.
Chapter 2 formally introduces the RL framework and the most important theoretical elements that it is based on. It also emphasizes actor-critic techniques for the continuous scenario, by presenting part of the state-of-the-art algorithms for policy evaluation and policy gradient. At last, it formalizes the Lipschitz continuity property within the element of a single task.
Chapter 3 concerns TL in the RL context. It presents the specific issues that transfer algorithms must face when dealing with RL problems, like the precise definition of the similarity assumptions, the type of knowledge that is transferred, or the metric that is supposed to be optimized. Some previous works on transfer of samples are also described in this chapter.

Chapter 4 introduces a proposal for modeling the Lipschitz continuity across different task, formalizes the problem tackled by this thesis, and uses them both to introduce the three considered estimators: IS-based, optimistic, and pessimistic.
Chapter 5 presents the experimental setting that the estimators have been evaluated on, as well as the obtained results and a discussion about them.
Chapter 6 finishes this work by presenting the conclusions that can be derived from the obtained results, and discusses future work to further improve them.

2

REINFORCEMENT LEARNING

The purpose of this chapter is to introduce the RL framework. Section 2.1 starts with the formalization of the agent problem formulation within this framework, and then Section 2.2 gives a brief taxonomy concerning the algorithms that intend to solve it. The chapter goes on with more detailed descriptions of important families of algorithms: policy gradient, in Section 2.3, and policy evaluation, in Section 2.4. Section 2.5 continues by presenting the actor-critic approach, a popular paradigm that combines policy gradient and policy evaluation into a more powerful technique. Finally, Section 2.6 introduces a formal approach for modeling the Lipschitz continuity properties within the elements of a single environment.

2.1 THEORETICAL FRAMEWORK: MARKOV DECISION PROCESSES

Reinforcement Learning is the subarea of Machine Learning that studies the strategy-finding problem related to goal-oriented artificial agents from an experience-based perspective (Sutton and Barto, 1998). The agent is assumed to have zero (or very poor) prior knowledge about the dynamics of the environment it will act upon in, so the only source of information that can be leveraged is the experience coming from interaction with it.
On its most general form, the problem tackled by RL is too complex; hence, there are a couple of assumptions that are usually accepted in order to simplify it and make it easier to be solved (Sutton and Barto, 1998).

Definition 2.1 (Reinforcement Learning Assumptions)**.** The RL paradigm strengthens the agent-environment paradigm (Chapter 1) by adding the following assumptions:

- *Discrete time transitions:* the interaction between the agent and the environment happens at discrete time steps. Thus, any instance of such an interaction process can be expressed by a sequence

  $$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots ,$$

  where s_i are the states and a_i are the actions;

- *Markovian transitions:* given the current state of the environment and the action just executed by the agent, the transition to the next state (or set of candidate next states) does not depend on

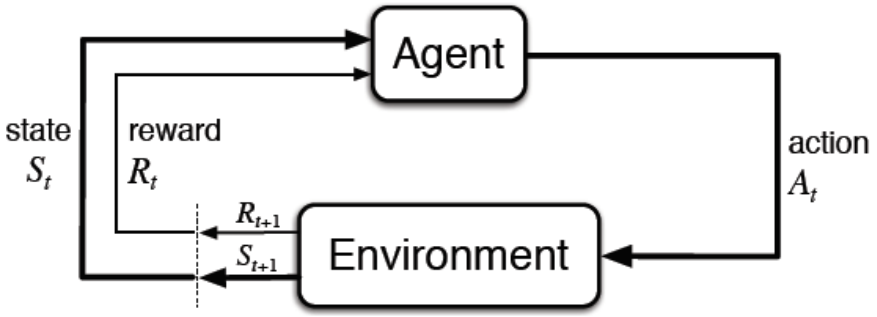


Figure 2.1: In addition to the resulting state, the agent also observes a reward inherent to the just performed transition (Sutton and Barto, 1998).

the history of previously visited states and actions, but only on the current state and action;

– *Stationary transition model:* the model that describes the transition between states does not depend on the current time step, it is invariant along the whole interaction between the agent and the environment;

– *Reward-based assessed transitions:* each transition produces a scalar reward for the agent to assess the goodness of executing the just applied action while being in the just abandoned state. The interaction instance is thus enriched to account for the rewards r_i:

$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots .$$

Figure 2.1 shows the enriched interaction process between the agent and the environment under the RL framework.
Thanks to these assumptions, RL can resort to well-defined mathematical entities to model and study the agent problem. In fact, the whole framework is based on the concept of Markov Decision Process (MDP).
MDPs are the corner stone of RL. They are used to formalize the environment, making it easier to model later on the agent, the goal and the corresponding strategy-finding problem. Although they are all founded on the same rationale, different formal definitions of MDPs exist. This thesis focuses on the case in which both the set of actions of the agent and the set of states of the environment are continuous, and the reward observed by the agent on each transition is deterministic.

Definition 2.2 (Markov Decision Process)**.** An MDP is a 6-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu)$ composed of:

– The set of the states the environment, called *state space*, denoted by $\mathcal{S}$. It is a measurable space with its Borel σ-algebra $\mathcal{B}(\mathcal{S})$, and

a Polish space with metric $d_{\mathcal{S}}$; furthermore, $\mathcal{S} \subset \mathbb{R}^{n_{\mathcal{S}}}$. The space $\Delta(\mathcal{S})$ is composed of all the measures defined on $\mathcal{B}(\mathcal{S})$;

– The set of actions available to the agent for applying to the environment, called *action space*, denoted by $\mathcal{A}$. It is a measurable space with its Borel σ-algebra $\mathcal{B}(\mathcal{A})$, and a Polish space with metric $d_{\mathcal{A}}$; furthermore, $\mathcal{A} \subset \mathbb{R}^{n_{\mathcal{A}}}$. The space $\Delta(\mathcal{A})$ is composed of all the measures defined on $\mathcal{B}(\mathcal{A})$;

– The *transition model*, that defines the dynamics of the transition between different states of the environment induced through an action applied by the agent. It is modeled with a function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, $(s, a, s') \mapsto \mathcal{P}(s'|s, a)$, such that $\mathcal{P}(\cdot|s, a)$ is a distribution for a probability measure on $\mathcal{B}(\mathcal{S})$ with respect to the Lebesgue measure; abusing notation, $\mathcal{P}(\cdot|s, a)$ is used both for the density function and the corresponding probability measure. $\mathcal{P}$ defines the dynamics of the environment in the sense that $\mathcal{P}(\cdot|s, a)$ is the distribution over the states that can be reached by the environment given that it was in state s and the agent performed action a;

– The *reward model*, that defines the reward obtained by the agent after executing a certain action in a certain state. It is modeled by a function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [R_1, R_2] \subset \mathbb{R}$, where $r(s, a, s')$ is the reward for the transition from s to s' as caused by action a. The expected immediate reward $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [R_1, R_2]$ is the expected value of the reward function r with respect to the distribution $\mathcal{P}(\cdot|s, a)$ induced by the corresponding state and action:

$$\mathcal{R}(s, a) = \int_{\mathcal{S}} r(s, a, s')\mathcal{P}(s'|s, a)\, ds';$$

– The *discount factor*, that measures the interest of the agent in longer term rewards. It is represented by a real number $\gamma \in [0, 1]$;

– The *initial state distribution*, that defines the initial condition of the environment with respect to the state space. It is modeled by function $\mu : \mathcal{S} \to \mathbb{R}$ that is a distribution for a probability measure on $\mathcal{B}(\mathcal{S})$ with respect to the Lebesgue measure; abusing notation, μ is used both for the density function and the probability measure. μ defines the initial conditions of the environment in the sense that it is the distribution over the states in which the environment can be found at the beginning of the interaction process.

MDPs can be enriched to account for partial observability (Chapter 1) by extending this formulation and producing Partially Observable Markov Decision Processes (POMDPs). For the scope of this thesis, full

observability is assumed and, therefore, the above presented MDPs are enough.
The instances of the agent-environment interaction process i. e.the chains of state-action-rewards produced by such interaction are given a special name in the RL framework.

Definition 2.3 (Trajectory)**.** Any sequence of states, actions and rewards produced by the interaction between the agent and the environment is called trajectory. A trajectory is denoted by

$$\begin{aligned}\tau &= \langle s_0, a_0, r_0\rangle, \langle s_1, a_1, r_1\rangle, \langle s_2, a_2, r_2\rangle ... \\ &= (\langle s_i, a_i, r_i\rangle)_{i=0}^{\infty}.\end{aligned}$$

The resulting state from the transition at each time step can be explicitly specified:

$$\begin{aligned}\tau &= \langle s_0, a_0, r_0, s_0'\rangle, \langle s_1, a_1, r_1, s_1'\rangle, \langle s_2, a_2, r_2, s_2'\rangle ... \\ &= (\langle s_i, a_i, r_i, s_i'\rangle)_{i=0}^{\infty},\end{aligned}$$

where $s_i' = s_{i+1}$ for all $i \in \mathbb{N}$. The set of all possible trajectories for an MDP is denoted by $\mathcal{T}$.

2.1.1 *The agent: Policies and Markov Reward Processes*

In the Reinforcement Learning framework, the agent (or, better, its strategy) is modeled by the concept of *policy* (Vlassis, 2007). Intuitively, a policy is a mechanism that, given the current state of the environment, produces an action (or set of actions) to be performed by the agent; the policy requires to know only the current state in order to select an action because of the Markov property assumption (see Definition 2.1). In addition, policies are usually assumed to be stationary: the output for a given state does not depend on the current time step, it is invariant along the whole interaction process for the same policy. All in all, the formalization of the concept of policy can be provided.

Definition 2.4 (Policy)**.** A policy is any function $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $(s, a) \mapsto \pi(a|s)$, such that $\pi(\cdot|s)$ is a distribution for a probability measure on $\mathcal{B}(\mathcal{A})$ with respect to the Lebesgue measure; abusing notation, $\pi(\cdot|s)$ refers both to the density function and the probability measure. π defines the agent's strategy in the sense that $\pi(\cdot|s)$ is the distribution over the actions that can be taken by the agent given that the environment's state is s. The space of all policies (as density functions) is denoted by Π.

Observation 2.5. *Thanks to full observability, it is possible to use the same state space for the environment and the agent.*

A policy π defines entirely the behavior of the agent and, consequently, the complete dynamics of the interaction process. In particular, the policy induces a well defined behavior on the rewards that the MDP is now producing (Szepesvari, 2010). This phenomenom is formalized by means of a Markov Reward Process (MRP):

Definition 2.6 (Markov Reward Process)**.** Given an MDP $\mathcal{M}$ and a policy π, the MRP induced on $\mathcal{M}$ by π is a 5-tuple $\mathcal{M}^\pi = (\mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma, \mu)$ where

- $\mathcal{P}^\pi$ is the state-to-state transition model induced by the policy and represents a distribution obtained by

$$\mathcal{P}^\pi(s'|s) = \int_{\mathcal{A}} \mathcal{P}(s'|s,a)\pi(a|s)\,da;$$

- $\mathcal{R}^\pi$ is the expected immediate reward function induced by the policy, obtained by

$$\mathcal{R}^\pi(s) = \int_{\mathcal{A}} \mathcal{R}(s,a)\pi(a|s)\,da.$$

Observation 2.7. *The concept of action somehow disappears within an MRP. In fact, it is just a Markov Chain augmented with rewards related to the transitions (Szepesvari, 2010).*

A policy also induces a probability distribution over the trajectory space $\mathcal{T}$ of the MDP; such distribution is denoted by ρ^π_μ.
Starting from the distribution over trajectories, it is possible to define a distribution over states that accounts for the probability of finding each state along the trajectory, discounting the occurrences according to the time step in which they are observed. This is the so called *discounted future state distribution* (Sutton et al., 1999).

Definition 2.8 (Discounted future state distribution)**.** Let $\mathcal{M}$ be an MDP and π be a policy. The discounted future state distribution is defined as

$$\begin{aligned}\delta^\pi_\mu(s_0) = (1-\gamma) \quad &\Big(\mu(s_0) + \gamma \int_{\mathcal{S}} \mathcal{P}^{\pi(1)}(s_0|s)\mu(s)\,ds \\ &+ \gamma^2 \int_{\mathcal{S}} \mathcal{P}^{\pi(2)}(s_0|s)\mu(s)\,ds + \ldots\Big)\end{aligned}$$

where $\mathcal{P}^{\pi(t)}(\cdot|s)$ is the distribution over states obtained by following the dynamics of the (reward) process for t time steps when starting from state s. For $t = 1$, $\mathcal{P}^{\pi(1)}(\cdot|s) = \mathcal{P}^\pi(\cdot|s)$. The coefficient $1-\gamma$ is a normalization factor. The distribution has a recursive form:

$$\delta^\pi_\mu(s_0) = (1-\gamma)\mu(s_0) + \gamma \int_{\mathcal{S}} \mathcal{P}^\pi(s_0|s)\delta^\pi_\mu(s)\,ds.$$

Starting from the future state distribution, the idea can be extended to define a future state-action distribution too (Pirotta, Restelli, and Bascetta, 2015).

Definition 2.9 (Discounted future state-action distribution)**.** Let $\mathcal{M}$ be an MDP and π be a policy. The discounted future state distribution is defined as

$$\zeta(s, a)^{\pi}_{\mu} = \delta^{\pi}_{\mu}(s)\pi(a|s).$$

Further distributions over the transitions can be introduced by following the same idea (these are handy for simplifying notation later on).

Definition 2.10 (Transition tuples distribution)**.** Let $\mathcal{M}$ be an MDP and π be a policy. The transition tuples distribution is defined by

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

Definition 2.11 (Extended transition tuples distribution)**.** Let $\mathcal{M}$ be an MDP and π be a policy. The extended transition tuples distribution is defined by

$$\overline{d}^{\pi}_{\mu}(s, a, s', a') = \delta^{\pi}_{\mu}(s)\pi(a|s)\mathcal{P}(s'|s, a)\pi(a'|s').$$

2.1.2 *The goal: Cumulative rewards*

The goal of a RL agent is to maximize the cumulative reward obtained along the whole interaction process with the environment by selecting an appropriate policy (Sutton and Barto, 1998). In order to guide the selection of the best policy, a utility function can be used to assess each one of them with respect to the cumulative reward it produces. There are various options for such function proposed by the RL framework (see e. g.Kaelbling, Littman, and Moore (1996)); the work on this thesis uses the *expected discounted reward*.

Definition 2.12 (Expected discounted reward)**.** Given an MDP $\mathcal{M}$ and a policy π, the expected discounted reward is defined as

$$J^{\pi}_{\mu} = \mathop{\mathbb{E}}_{\tau\sim\rho^{\pi}_{\mu}}\left[\sum_{i=0}^{\infty}\gamma^{i}r_{i}\right].$$

The utility value of a policy is referred to as its *performance*. Inspired by the above definition, trajectories can be given associated a *return* value.

Definition 2.13 (Return of a trajectory)**.** Given a trajectory τ, its (discounted) return is given by

$$\mathcal{R}(\tau) = \sum_{i=0}^{\infty}\gamma^{i}r_{i}.$$

The discount factor γ gets now a mathematical motivation: provided that $\gamma < 1$, it ensures that the cumulative reward always converges and makes the utility function to be well-defined. It is also clear now how its value expresses the interest of the agent in shorter or longer term rewards.

Infinite or finite but unbounded length trajectories are studied to get insights on the theoretical properties of MDPs, but they are rather uncommon in practice. What usually happens is that the length is large enough to consider the expected discounted reward as utility function, but it is still taken as finite to ensure that the trajectory distribution is well-defined.

Definition 2.14 (Distribution over trajectories)**.** The distribution over finite-length trajectories is given by

$$\begin{aligned}\rho_\mu^\pi(\tau) = \; & \mu(s_0)\pi(a_0|s_0)\mathcal{P}(s_1|s_0,a_0)\pi(a_1|s_1)\mathcal{P}(s_2|s_1,a_1)... \\ & \pi(a_{T-1}|s_{T-1})\mathcal{P}(s_T|s_{T-1},a_{T-1}) \\ = \; & \mu(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t)\mathcal{P}(s_{t+1}|s_t,a_t),\end{aligned}$$

where T is the length of the trajectories.

Under these definitions, the goal of the agent formally consists on searching the policy space to find a policy that is optimal with respect to the expected discounted reward, i. e., the policy with the best performance.

Definition 2.15 (Optimal policy)**.** Given an MDP $\mathcal{M}$, an optimal policy π^* is any policy such that

$$\pi^* \in \operatorname*{argmax}_{\pi\in\Pi} J_\mu^\pi.$$

2.1.3 *Value functions*

The expected discounted reward (Definition 2.12) is good for intuitively understanding the goal of the agent, but using it to solve the agent problem can be quite complex. One alternative is to take the performance of the policy and break it down to utility values associated to states and state-action pairs. The functions giving such values are known as *value functions* (Sutton and Barto, 1998).

Definition 2.16 (Value functions)**.** Given an MDP $\mathcal{M}$ and a policy π, the state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ outputs, for each state s, the expected discounted reward obtained by starting at such state and following π along the rest of the trajectory. That is,

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau\in\mathcal{T}}\left[\sum_{i=0}^{\infty}\gamma^i r_i|s_0 = s\right].$$

The state value function is usually referred to as just the value function. Similarly, the state-action value function $Q^\pi : \mathcal{S} \to \mathbb{R}$ outputs, for each state-action pair (s, a), the expected discounted reward obtained by starting at such state, applying such action and following π during the rest of the trajectory. That is,

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{\tau \in \mathcal{T}} \left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a \right].$$

Observation 2.17. *The performance of a policy π can be written in terms of the state value function:*

$$J_\mu^\pi = \mathop{\mathbb{E}}_{s \sim \mu} [V^\pi(s)] \tag{2.1}$$

The state value function and the state-action value function are also referred to as *V-function* and *Q-function*, respectively. An important feature of the value functions is their recursive nature. In fact,

$$\begin{aligned} V^\pi(s) &= \int_{\mathcal{A}} \left(\mathcal{R}(s, a) + \gamma \int_{\mathcal{S}} V^\pi(s') \mathcal{P}(s'|s, a) \, ds' \right) \pi(a|s) \, da; \\ Q^\pi(s, a) &= \mathcal{R}(s, a) + \gamma \int_{\mathcal{S}} \int_{\mathcal{A}} Q^\pi(s', a') \mathcal{P}(s'|s, a) \pi(a'|s') \, da' \, ds'. \end{aligned} \tag{2.2}$$

The *advantage function* is also used to understand the utility value of the policy (Baird, 1994).

Definition 2.18 (Advantage function)**.** Let $\mathcal{M}$ be an MDP and π be a policy. Given a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$, the advantage function measures how much utility is obtained by applying action a and then following π with respect to the average utility produced by immediately following π in the state s:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s).$$

The *optimal value functions* are those that, at each point, have the maximum possible value attainable by any policy in the policy space at such point.

Definition 2.19 (Optimal value functions)**.** Given an MDP $\mathcal{M}$, the optimal state value function is defined as:

$$V^*(s) = \max_{\pi \in \Pi} V^\pi(s);$$

similarly, the optimal state-action value function is defined as:

$$Q^*(s, a) = \max_{\pi \in \Pi} Q^\pi(s, a).$$

The optimal value functions provide a simpler perspective of the optimality criteria for the policies: a policy that realizes the optimal value functions is also optimal with respect to the performance. An important result ensures the existence of policies that realize such optimal value functions (Puterman, 1994).

Theorem 2.1. For any MDP there exists a policy $\pi^* \in \Pi$ such that $V^\pi = V^*$ and $Q^\pi = Q^*$.

The optimal value functions have a recursive form too:

$$\begin{aligned} V^*(s) &= \max_{a \in \mathcal{A}} \left(\mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} V^*(s') \mathcal{P}(s'|s,a)\, ds' \right); \\ Q^*(s,a) &= \mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} \left(\max_{a' \in \mathcal{A}} Q^*(s',a') \right) \mathcal{P}(s'|s,a)\, ds' \end{aligned} \tag{2.3}$$

2.1.4 *Bellman operators and Bellman equations*

The recursive form of the value functions (Equation (2.2)) inspired the formulation of the *Bellman expectation operators*.

Definition 2.20 (Bellman expectation operators)**.** Given an MDP $\mathcal{M}$ and a policy π, the Bellman expectation operator for state functions is an operator that maps state functions (real-valued functions of the state space) into state functions, $T^\pi_{\mathcal{S}} : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$, defined by:

$$(T^\pi_{\mathcal{S}} V)(s) = \int_{\mathcal{A}} \left(\mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} V(s') \mathcal{P}(s'|s,a)\, ds' \right) \pi(a|s)\, da.$$

It is also referred to just as Bellman state operator.
Similarly, the Bellman expectation operator for state-action functions is an operator that maps state-action functions into state-action functions, $T^\pi_{\mathcal{SA}} : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, defined by:

$$(T^\pi_{\mathcal{SA}} Q)(s,a) = \mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} \int_{\mathcal{A}} Q(s',a') \mathcal{P}(s'|s,a) \pi(a'|s')\, da'\, ds'.$$

It is also referred to just as Bellman state-action operator.

For the policy π, the value functions V^π and Q^π are a fixed point of its Bellman state operator and its Bellman state-action operator, respectively. If $0 < \gamma < 1$, the operators are a contraction (with respect to the maximum norm) and, therefore, V^π and Q^π are the unique fixed points of the respective operators (Szepesvari, 2010).
By using the Bellman operators, it is possible to formulate the so called *Bellman expectation equations*:

$$\begin{aligned} T^\pi_{\mathcal{S}} V &= V; \\ T^\pi_{\mathcal{SA}} Q &= Q. \end{aligned} \tag{2.4}$$

These equations aim to find functions that have the same recursive structure of the value functions. Having $0 < \gamma < 1$ implies that the value functions are the unique solutions of the Bellman expectation equations.
The recursive form of the optimal value functions (Equation (2.3)) inspired the formulation of the *Bellman optimality operators*.

Definition 2.21 (Bellman optimality operators)**.** Given an MDP $\mathcal{M}$, the Bellman optimality operator for state functions is an operator that maps state functions into state functions, $T_S^* : \mathbb{R}^S \to \mathbb{R}^S$, defined by:

$$(T_S^* V)(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}(s,a) + \gamma \int_S V(s') \mathcal{P}(s'|s,a)\, ds' \right).$$

It is also referred to just as Bellman optimality state operator.
Similarly, the Bellman optimality operator for sate-action functions is an operator that maps state-action functions into state-action functions, $T_{S\mathcal{A}}^* : \mathbb{R}^{S\times\mathcal{A}} \to \mathbb{R}^{S\times\mathcal{A}}$, defined by:

$$(T_{S\mathcal{A}}^* Q)(s,a) = \mathcal{R}(s,a) + \gamma \int_S \left(\max_{a' \in \mathcal{A}} Q(s',a') \right) \mathcal{P}(s'|s,a)\, ds'.$$

It is also referred to just as Bellman optimality state-action operator.

The optimal value functions V^* and Q^* are a fixed point of the Bellman optimality state operator and the Bellman optimality state-action operator, respectively. If $0 < \gamma < 1$, the operators are a contraction (with respect to the maximum norm) and, therefore, V^* and Q^* are the unique fixed points of the respective operators (Szepesvari, 2010). By using the Bellman optimality operators, it is possible to formulate the so called *Bellman optimality equations*:

$$\begin{aligned} T_S^* V &= V; \\ T_{S\mathcal{A}}^* Q &= Q. \end{aligned} \tag{2.5}$$

These equations aim to find functions that have the same recursive structure of the optimal value functions. Having $0 < \gamma < 1$ implies that the optimal value functions are the unique solutions of the Bellman optimality equations.

2.2 BRIEF TAXONOMY OF REINFORCEMENT LEARNING ALGORITHMS

Based on the formulation of the agent problem (Chapter 1) proposed by the RL framework (Section 2.1.3 and Section 2.1.2), many algorithms have come out with possible solutions using different approaches along time (some examples can be found in Kaelbling, Littman, and Moore (1996), Sutton and Barto (1998), Bertsekas (2007)). This section intends to provide a brief taxonomy that allows to better understand the rationale motivating each solution. For such matter, four different dimensions are considered: model requirements (Section 2.2.1), policy-based sampling strategy (Section 2.2.2), solution strategy (Section 2.2.3), and sample usage (Section 2.2.4).

2.2.1 *Model requirements: Model-based vs. Model-free*

The first dimension refers to the requirements of the algorithm with respect to the model of the MDP. The two extreme points along this

MODEL-FREE	MODEL-BASED
Q-learning (Watkins, 1989), SARSA (Rummery and Niranjan, 1994), REINFORCE (Williams, 1992)	RL-DT (Hester and Stone, 2009), DYNA (Sutton, 1991), Contextual policy search (Kupcsik et al., 2017)

Table 2.1: Some examples of model-free and model-based algorithms.

dimension are *model-based* algorithms and *model-free* algorithms (Kaelbling, Littman, and Moore, 1996).
Model-based algorithms are those that require an explicit approximation of the model of the MDP. Not every model-based algorithm is equally demanding on this matter: some of them might need information about every element of the MDP (transition model, reward model, initial state distribution, etc.) while others might require approximations only for a fraction of them (e. g.the reward model only).
On the other hand, model-free algorithms are those that do not require any explicit approximation of the model of the MDP in their implementation.
Model-free algorithms usually have low computational demands but require lots of data in order to get good results; they are well-suited for problems in which computation is costly but sampling is cheap. Instead, model-based algorithms usually have a higher time complexity but a lower data complexity, so they fit well in contexts where computation is cheap but gathering data is costly. In addition, they provide the benefit of producing an approximation of the model of the environment which can be useful for studying its behavior (Kaelbling, Littman, and Moore (1996), Atkeson and Santamaria (1997)).
Table 2.1 lists some examples of both model-free and model-based algorithms.

2.2.2 *Policy-based sampling strategy: On-policy vs. Off-policy*

The second dimension of the taxonomy refers to the relation between the policy that is being used to interact with the environment (*behavioral* policy) and the policy that is being learned by the agent (*target* policy). The two extreme points along this dimension are *on-policy* algorithms and *off-policy* algorithms (Sutton and Barto, 1998).
On-policy algorithms are those that need the behavioral and the target policies to be the same. Off-policy algorithms, on the contrary, admit a behavioral policy that is different from the target one (although it is permitted that they coincide).
Given that the only source of knowledge in RL is the interaction experience, the agent is constantly facing a dilemma while searching the optimal policy: stick to the current safe but possibly sub-optimal policy, or try novel actions to try to improve the policy although without

ON-POLICY	OFF-POLICY
SARSA (Rummery and Niranjan, 1994), TD(λ) (Sutton, 1988), eNAC (Peters and Schaal, 2008a)	Q-learning (Watkins, 1989), Off-policy FPKF(λ) (Geist and Scherer, 2014), Off-policy actor-critic (Degris, White, and Sutton, 2012)

Table 2.2: Some examples of on-policy and off-policy algorithms.

any guarantee about it. This issue is referred to as the *exploration-exploitation* dilemma (Kober, Bagnell, and Peters, 2013). It is closely related to the trade-off between on-policy and off-policy algorithms: an on-line algorithm will converge to a solution if and only if it is no longer exploring the policy space.
On-policy algorithms are intuitively easier to understand so that studying their theoretical properties (e. g.convergence) is usually a simpler task; however, they have problems overcoming the exploration-exploitation dilemma. Off-policy algorithms, instead, easily deal with such dilemma, but their theoretical properties are more difficult to study; in fact, they usually require to modify the standard assumptions so that resulting model is compatible with the approach. Off-policy is also useful when exploring new policies is costly, because it allows the agent to learn multiple policies while actually executing only one (Degris, White, and Sutton, 2012).
Table 2.2 lists some examples of on-policy and off-policy algorithms.

2.2.3 *Solution strategy: Policy-based vs. Value-based*

The third dimension of the taxonomy has to do with the nature of the strategy that the algorithm adopts with respect to elements of the model that it exploits. In particular, the considered elements are the policy and the value functions. The two extreme points along this dimension are *policy-based* algorithms and *value-based* algorithms (Kober, Bagnell, and Peters, 2013).
Policy-based algorithms are those that explicitly exploit the policy but do not use the value functions (at least in a direct way); hence, they need to maintain an explicit representation for it. Value-based algorithms, on the opposite, explicitly exploit the value functions but do not do so with the policy; therefore, an explicit representation of them is to be kept.
Large state and action spaces represent a potential issue for both policy-based and value-based algorithms, as storing the complete value function or policy is unfeasible or even impossible. Function approximators can be used in order to overcome this problem.
Value-based techniques have good convergence properties under small state and action spaces and turn out very suitable when there is insufficient or none prior knowledge about the problem; however, the

POLICY-BASED	VALUE-BASED
REINFORCE (Williams, 1992), G(PO)MDP (Baxter and Bartlett, 2001)	Value-iteration (Sutton and Barto, 1998), FQI (Antos, Munos, and Szepesvári, 2007)

Table 2.3: Some examples of policy-based and value-based algorithms.

changes to the underlying policy during the learning process can be unstable, and the convergence properties disappear when large state and action spaces demand the use of function approximators. Policy-based methods are well-suited for large state or action spaces, produce smooth changes in the policy during the learning process, and allow to introduce prior expert knowledge in the policy; unfortunately, they suffer from a high variance and have issues with local minima (Kober, Bagnell, and Peters (2013), Beitelspacher et al. (2006)).
Unlike the other dimensions of the taxonomy, this has a rather fuzzy nature. The reason for this is that combining techniques from both extremes has overcome their individual issues and has produced very successful approaches (like actor-critic algorithms, Section 2.5).
Table 2.3 lists some examples of (pure) policy-based and value-based algorithms.

2.2.4 *Sample usage: Online vs. Offline*

The fourth dimension of the taxonomy refers to the frequency with which the agent commits the effects of the knowledge acquired from the collected samples. The two extreme points along this dimension are *online* algorithms and *offline* algorithms (Lange, Gabel, and Riedmiller, 2012).
Online algorithms are those that take the knowledge from new samples and use it to update whatever information is kept about the problem right in the moment when the samples are observed; the sampling and the learning processes are carried out simultaneously. Instead, offline algorithms accumulate the knowledge from a number of samples and apply the update with some given periodicity; the sampling and the update processes are executed sequentially, one after the other, and each one stops while the other is taking place.
Online algorithms are the best choice for agents that are constantly interacting with the environment for long periods of time; they have low storage requirements but might be slower to converge because of the frequent updates. Offline algorithms are good for agents that have short interactions with their environment; their memory complexity is higher but the stability of the learning process is improved.
Lange, Gabel, and Riedmiller (2012), although presenting a finer classification of this dimension, remark that the distinctions between the categories depend on the implementation rather than on the actual

formulation; thus, almost any algorithm can be classified as online and offline. Because of this, instead of giving a list of examples in each class, the reader is referred there for some insights on how the algorithms can be implemented in either way.

2.3 POLICY GRADIENT

Among the alternatives offered by policy-based methods (Section 2.2.3, Deisenroth, Neumann, and Peters, 2013), the work presented in this thesis focuses on those implementing the concept of *policy gradient*. Policy gradient assumes a parametric policy space so that each one of the considered policies is identified by a parameter taken from a parameter space. Formally,

$$\Pi_\Theta = \{\pi_\theta \mid \theta \in \Theta \subset \mathbb{R}^d\}.$$

This assumption restricts the search of the optimal policy only to those belonging to the selected parametric space; depending on the nature of the problem and of this space, such optimal policy might or not be actually contained in there. Thus, the goal of the agent is to maximize the performance of the policy but constrained to the set Π_Θ, which is equivalent to finding the parameter that realizes the maximum performance:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, J_\mu^{\pi_\theta}.$$

To simplify the notation along this section, dependence on the policy π_θ is expressed by specifying only the parameter θ; thus, for example, $J_\mu^{\pi_\theta}$ is simply denoted by J_μ^θ. Policy gradient proposes to solve this optimization problem with a *gradient ascent* approach: the gradient of the performance with respect to the parameter space is followed until a maximum is reached (Peters, Vijayakumar, and Schaal, 2003). When the actual direction of the gradient is followed, the method is said to be *"vanilla"* or *standard gradient*; if it follows a direction that results from linearly transforming the gradient according to the underlying topological structure of the policy space, the approach is known as *natural gradient* (Peters and Schaal, 2008b).
Three alternatives for calculating the gradient are presented in Section 2.3.1, Section 2.3.2 and Section 2.3.3. Natural gradient is detailed in Section 2.3.4.

2.3.1 *Finite differences*

Finite differences methods aim to approximate the gradient of the performance at some point in the parameter space based on the values of the performance given by policies whose parameters lie close to it (Deisenroth, Neumann, and Peters, 2013). This idea is justified by

the Taylor theorem which states that, in small neighborhoods of a given point, a (smooth enough) function behaves almost as a linear one with the same gradient as the original function at such point. That is, given a continuously differentiable function $f : X \subset \mathbb{R}^n \to \mathbb{R}$, a point $x \in X$ and a neighborhood $U \ni x$, it holds that, for every $y \in U$,

$$f(y) \approx f_{ref} + \nabla_x f(x)^\top \cdot (x - y),$$

where f_{ref} is an appropriate value. This can be rewritten into

$$\nabla_x f(x)^\top \cdot \Delta x \approx f(x + \Delta x) - f_{ref} \tag{2.6}$$

where Δx represents a small change in X.
One way to approximate the gradient is to evaluate the function at points $x + \Delta^{(1)}x, ..., x + \Delta^{(n)}x$, where $\Delta^{(i)}x$ is zero in all but the i-th component, so that the values $f(x + \Delta^{(1)}x), ..., f(x + \Delta^{(n)}x)$ can be used to solve for $\nabla_{x_i} f(x)$ at each dimension. Common choices for f_{ref} are $f(x)$, which produce forward-difference estimators, and $f(x - \Delta x)$, which produce central-difference estimators (Peters and Schaal, 2008b).
A more general approach comes from writing Equation (2.6) as a regression problem

$$\nabla_x f(x)^\top \cdot \Delta x + f_{ref} \approx f(x + \Delta x)$$

where the input is Δx, the output is $f(x + \Delta x)$ and the weights correspond to the values of the gradient and the reference value f_{ref}. The estimation is performed by collecting the values $f(x + \Delta^{(1)}x), ..., f(x + \Delta^{(n)}x)$, $m \geqslant n$, and solving the regression problem with any suitable technique. In this approach, the changes $\Delta^{(i)}x$ are no longer constrained to be null in all but one dimension (Peters and Schaal, 2008b).

In the context of policy gradient estimation, the parameter θ is slightly perturbed to produce the $\Delta^{(i)}\theta$ changes and the corresponding performance values $J_\mu^{\theta+\Delta^{(i)}\theta}$ are estimated with samples produced using each one of the policies (Kohl and Stone, 2004).

2.3.2 *Trajectory-based policy gradient*

Starting directly from the definition of the performance of a policy (Section 2.1.2), it is possible to express the gradient in terms of the distribution over trajectories and their observed return (Definition 2.13). Doing so gives place to another group of approaches.

Definition 2.22 (Policy gradient)**.** Let $\mathcal{M}$ be an MDP and Π_Θ a parametric family of policies. The gradient of the performance of a policy π_θ with respect to the parameters of the policy is given by (Deisenroth, Neumann, and Peters, 2013)

$$\nabla_\theta J_\mu^\theta = \mathop{\mathbb{E}}_{\tau \sim \rho_\mu^\theta} [\nabla_\theta \log \rho_\mu^\theta(\tau) \mathcal{R}(\tau)].$$

Observation 2.23. *The gradient of the logarithm of the distribution over trajectories does not depend on the transition model of the MDP:*

$$\begin{aligned}\nabla_\theta \log \rho^\theta_\mu(\tau) &= \nabla_\theta \log \left(\mu(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) \mathcal{P}(s_{t+1}|s_t, a_t) \right) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t).\end{aligned} \tag{2.7}$$

so the gradient can be written as:

$$\begin{aligned}\nabla_\theta J^\theta_\mu &= \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \right) \mathcal{R}(\tau) \right] \\ &= \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \right) \left(\sum_{t=0}^{T-1} \gamma^t r_t \right) \right].\end{aligned} \tag{2.8}$$

Observation 2.24 (REINFORCE policy gradient)**.** *For any* $b \in \mathbb{R}$*, the gradient can also be written as (Williams, 1992)*

$$\begin{aligned}\nabla_\theta J^\theta_\mu &= \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \right) (\mathcal{R}(\tau) - b) \right] \\ &= \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \right) \left(\sum_{t=0}^{T-1} \gamma^t r_t - b \right) \right].\end{aligned}$$

The constant b *is then called a* baseline*.*

Observation 2.25 (PGT/G(PO)MD)**.** *For any sequence* $(b_t)_{t=0}^{T-1} \subset \mathbb{R}$*, the gradient can also be written as (Peters and Schaal, 2008b)*

$$\nabla_\theta J^\theta_\mu = \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{l=t}^{T-1} \gamma^{l-t} r_l - b_t \right) \right],$$

or, equivalently (Baxter and Bartlett, 2001),

$$\nabla_\theta J^\theta_\mu = \mathop{\mathbb{E}}_{\tau\sim\rho^\theta_\mu} \left[\sum_{t=0}^{T-1} \left(\sum_{l=0}^{t} \nabla_\theta \log \pi_\theta(a_l|s_l) \right) (\gamma^t r_t - b_t) \right].$$

The constants $(b_t)_{t=0}^{T-1}$ *are called* baselines*.*

The gradient can then be approximated by doing a Monte Carlo (MC) estimate using samples drawn according to ρ^θ_μ, either with the simplified form of the gradient proposed by Equation (2.8) or some of the modified ones that introduce a baseline (Observation 2.24 or Observation 2.25).
Using the original form of the gradient for the estimate produces an estimator with a huge variance that can be decreased with an appropriate baseline. It is actually possible to find a value that is optimal

with respect to the variance reduction for either of the two baseline alternatives (Peters and Schaal, 2008b).
The REINFORCE formulation admits a different baseline for each component of the gradient. The optimal values can be proven to be

$$b_k = \frac{\mathop{\mathbb{E}}_{\tau\sim\rho_\mu^\theta}\left[\left(\sum_{t=0}^{T-1}\nabla_{\theta_k}\log\pi_\theta(a_t|s_t)\right)^2\sum_{t=0}^{T-1}\gamma^t r_t\right]}{\mathop{\mathbb{E}}_{\tau\sim\rho_\mu^\theta}\left[\left(\sum_{t=0}^{T-1}\nabla_{\theta_k}\log\pi_\theta(a_t|s_t)\right)^2\right]};$$

in practice, they are approximated by MC estimation.
The PGT/G(PO)MD formulations actually correspond to equivalent results that were produced independently. They are equivalent in the sense that their MC estimates produce the same result. Like REINFORCE, they admit a different baseline for each component of the gradient. The optimal baseline values in this case are

$$b_k^t = \frac{\mathop{\mathbb{E}}_{\tau\sim\rho_\mu^\theta}\left[\left(\sum_{l=0}^{t}\nabla_{\theta_k}\log\pi_\theta(a_l|s_l)\right)^2\gamma^t r_t\right]}{\mathop{\mathbb{E}}_{\tau\sim\rho_\mu^\theta}\left[\left(\sum_{l=0}^{t}\nabla_{\theta_k}\log\pi_\theta(a_l|s_l)\right)^2\right]};$$

again, they can be approximated in practice by means of MC estimates.
The PGT/G(PO)MD formulation can be more efficient than REINFORCE when it comes to variance reduction. This is caused by the fact that the former exploits the Markovian property (Definition 2.1) to make future actions independent of past rewards. Of course, in non-Markovian environments, such formulation is no longer valid (Peters and Schaal, 2008b).

2.3.3 *State-action-based policy gradient*

Yet another group of policy gradient methods is based on the *policy gradient theorem*.

Theorem 2.2 (Policy gradient theorem)**.** (Sutton et al., 1999) Let $\mathcal{M}$ be an MDP and Π_Θ a parametric family of policies. The gradient of the performance of a policy π_θ with respect to its parameters is given by

$$\nabla_\theta J_\mu^\theta = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta}[Q^\theta(s,a)\nabla_\theta\log\pi_\theta(a|s)].$$

Observation 2.26. *For any* $b : \mathcal{S}\to\mathbb{R}$*, the gradient can also be written as*

$$\nabla_\theta J_\mu^\theta = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta}[(Q^\theta(s,a)-b(s))\nabla_\theta\log\pi_\theta(a|s)].$$

The function b *is then called a* baseline.

The gradient can then be approximated by doing a MC estimate using samples drawn according to ζ_μ^θ, either with the original form of the gradient proposed by the theorem or the modified one with a baseline.
When using the theorem directly, i. e., the estimate is done by averaging only over the values $Q^\theta(s,a)\nabla_\theta \log \pi_\theta(a|s)$, the Q-function (or a good estimate of it) is required for the calculation; in addition, the resulting estimate is known to suffer from high variance. This can be overcome by choosing an appropriate baseline that reduces the variance while keeping a zero bias. Inspired by the definition of the advantage function (Definition 2.18), a common choice for such baseline is the V-function (Bhatnagar et al. (2009), Grondman et al. (2012)). This results in an estimator that averages values of the form $A^\theta(s,a)\nabla_\theta \log \pi_\theta(a|s)$. In this case, it is necessary to know either the advantage function or both the Q-function and the V-function.
When unknown, the value functions can be approximated with policy evaluation techniques (Section 2.4), usually supported with function approximators (Section 2.4.3). If the feature space is not selected properly, it can introduce a bias that might harm the gradient estimation. Sutton et al. (1999) and Konda and Tsitsiklis (2003) give conditions under which the gradient estimation with function approximation is actually unbiased.

Theorem 2.3 (Policy gradient with compatible function approximation)**.** Let $\mathcal{M}$ be an MDP, $\mathcal{F}_\mathcal{W} = \{f_w : \mathcal{S}\times\mathcal{A} \to \mathbb{R} \mid w \in \mathcal{W}\}$ a parametric family of linear function approximators and Π_Θ a parametric family of policies. If w^* is optimal in the sense that:

$$\mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta}\left[(\widehat{Q_\mu^\theta}(s,a) - f_{w^*}(s,a))\nabla_w f_{w^*}(s,a)\right] = 0,$$

being $\widehat{Q_\mu^\theta}$ an unbiased estimator of Q_μ^θ, and f_w is compatible with the policy parametrization in the sense that:

$$\nabla_w f_w(s,a) = \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} = \nabla_\theta \log \pi_\theta(a|s),$$

then

$$\nabla_\theta J_\mu^\theta = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta}[f_{w^*}(s,a)\nabla_\theta \log \pi_\theta(a|s)].$$

Observation 2.27. *The fact that the function approximators for the Q-function are linear with respect to their parameters implies that actually*

$$f_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^\top w,$$

i. e., the feature map for the Q-function is given by the log-gradient of the policy. From this it follows that (Peters, Vijayakumar, and Schaal, 2003)

$$\begin{aligned}\nabla_\theta J_\mu^\theta &= \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta}[\nabla_\theta \log \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)^\top]w^* \\ &= \frac{1}{1-\gamma}F(\theta)w^*,\end{aligned}$$

where $F(\theta)$ *is the Fisher Information Matrix (FIM) of the probability distribution over trajectories* ρ_μ^θ *with respect to the parameter* θ*. This matrix can also be obtained by*

$$F(\theta) = \mathop{\mathbb{E}}_{s\sim\delta_\mu^\theta}[F(\theta, s)] = \int_{\mathcal{S}} \delta_\mu^\theta(s) F(\theta, s)\, ds,$$

where $F(\theta, s)$ *is the FIM of the distribution over actions defined by* π_θ*:*

$$\begin{aligned} F(\theta, s) &= \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top\right] \\ &= \int_{\mathcal{A}} \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top \pi(a|s)\, da. \end{aligned}$$

2.3.4 *Natural gradient*

Following the standard gradient direction while trying to find the optimal policy in the policy space can lead to sub-optimal or even unfeasible solutions (Peters and Schaal, 2008a). This can occur when the shape of the parameter space does not capture properly that of the subset of policies defined by the parametric family in the complete policy space. Natural gradient methods intend to overcome this by appropriately modifying the gradient to get the "right" direction.
When being at the point θ, gradient ascent approaches select as ascent direction the steepest one with respect to some metric, i. e., the direction $\Delta\theta$ which maximizes $J_\mu^{\theta+\Delta\theta}$ while keeping $\|\Delta\theta\|^2_{G(\theta)}$ equal to a small constant value; the metric is defined through a positive-definite matrix $G(\theta)$ by the expression $\|\theta_0\|^2_{G(\theta)} = \theta_0^\top G(\theta)\theta_0$. The (approximate) solution of such maximization problem gives as ascent direction $G(\theta)^{-1}\nabla_\theta J_\mu^\theta$ (Kakade, 2001).
Vanilla gradient methods (Section 2.3) select as metric the Euclidean norm defined by the matrix $G(\theta) = I$, which results in taking the gradient as ascent direction. Natural gradient, instead, defines the metric through a *Riemannian tensor* that relates the topology of the manifold defined by the parametric family of policies and the topology of the parameter space (Amari, 1998). The resulting Riemannian metric is desired to be *covariant*, i. e., it gives the same direction in the policy space regardless of the parameter space used to parametrize the policies.
The FIM (Observation 2.27) is proved (Peters and Schaal, 2008a) to define a covariant metric suitable for the space of parametric policies. Hence, the natural gradient direction is obtained through

$$\widetilde{\nabla_\theta} J_\mu^\theta = F(\theta)^{-1}\nabla_\theta J_\mu^\theta.$$

The FIM produces a linear transformation that rotates vectors with an angle of no more that $\pi/2$; thus, the performance is still ensured to improve and convergence to a maximum can be guaranteed. If com-

patible function approximators (Theorem 2.3) are used, the natural gradient of the performance reduces to

$$\widetilde{\nabla_\theta} J_\mu^\theta = \frac{1}{1-\gamma} w^*,$$

where w^* is the optimal parameter as defined by Theorem 2.3.

2.4 POLICY EVALUATION

Value-based algorithms (Section 2.2.3) keep an explicit representation of the value functions relative to the current policy of the agent and, therefore, require the ability to calculate such functions. The process of finding the V-function and Q-function of a policy is called *policy evaluation*.
Among the large number of policy evaluation methods (see e. g.Polydoros and Nalpantidis (2017)), the work presented in this thesis is based on *Temporal difference* approaches; the scope of this section is to introduce the most widely spread ones. It starts presenting the naïve Monte Carlo algorithms in Section 2.4.1, to motivate and present basic TD algorithms in Section 2.4.2, and concludes with the more sophisticated alternatives in Section 2.4.3.

2.4.1 *Monte Carlo estimation*

Starting from the very definition of the value functions (Section 2.16), Monte Carlo methods propose to estimate the value functions by sampling trajectories and averaging the corresponding obtained returns (Definition 2.13) (Sutton and Barto, 1998).
For the state value function, a separate average $\widehat{V_\mu^\pi}(s)$ is calculated for each state s by using the return of the observed trajectories (or portions of them) that start at s. This idea raises two alternatives for the estimation of the utility value of the state s: the *first visit* option accounts for the returns obtained by starting only from the first occurrence of s in each observed trajectory, while *every visit* will average over the returns obtained by starting at any occurrence of s in the observed trajectories (Singh and Sutton, 1996).
First visit Monte Carlo is clearly unbiased but has a large variance produced by the cumulative randomness of all the transitions in the trajectory. Every visit reduces the variance by using more data in the estimation although such data introduce a bias.
To prevent the need for storing all the observed trajectories when updating the estimation $\widehat{V_\mu^\pi}(s)$ upon arrival of a new sample, these approaches resort to an *incremental mean* that allows a recursive form

for the calculation (Sutton and Barto, 1998). That is, when a new trajectory is observed, the estimate of the state s_t is updated by

$$\widehat{V^\pi}(s_t) \leftarrow \widehat{V^\pi}(s_t) + \frac{1}{N}\left(\sum_{l=t}^{T-1} \gamma^{l-t} r_l - \widehat{V^\pi}(s_t)\right),$$

where $N-1$ is the number of updates that had been performed on the estimate. Given that the older estimates are less reliable (as they were calculated on less data), the above formula is modified to reduce the contribution of such samples in the estimation as new observations are obtained

$$\widehat{V^\pi}(s_t) \leftarrow \widehat{V^\pi}(s_t) + \alpha_N\left(\sum_{l=t}^{T-1} \gamma^{l-t} r_l - \widehat{V^\pi}(s_t)\right),$$

where the *learning rate* α_N is a value between 0 and 1 that measures exactly such contribution at the N-th observation. This estimator of the mean is known as the *exponential average*, which turns out to be consistent provided that $\sum_{i=1}^{\infty} \alpha_i = \infty$ and $\sum_{i=1}^{\infty} \alpha_i^2 < \infty$.

The term $\sum_{l=t}^{T-1} \gamma^{l-t} r_l$ is called the *state MC target* and $\sum_{l=t}^{T-1} \gamma^{l-t} r_l - \widehat{V^\pi}(s_t)$ is the *state MC error*.
The formulation of the method for the Q-function is analogous (Sutton and Barto, 1998): separate estimates $\widehat{Q^\pi}(s,a)$ are maintained for each state-action pair and, upon arrival of a new trajectory, updated according to the expression

$$\widehat{Q^\pi}(s_t,a_t) \leftarrow \widehat{Q^\pi}(s_t,a_t) + \alpha\left(\sum_{l=t}^{T-1} \gamma^{l-t} r_l - \widehat{Q^\pi}(s_t,a_t)\right).$$

The term $\sum_{l=t}^{T-1} \gamma^{l-t} r_l$ is the *state-action MC target* and $\sum_{l=t}^{T-1} \gamma^{l-t} r_l - \widehat{Q^\pi}(s_t,a_t)$ is the *state-action MC error*.

2.4.2 *Temporal Difference estimation*

Temporal difference (TD) methods (Sutton, 1988) are inspired by the recursive form of the value functions (Equation (2.2)). For the state value function, given that

$$\begin{aligned} V^\pi(s) &= \int_{\mathcal{A}} \left(\mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} V^\pi(s')\mathcal{P}(s'|s,a)\,ds'\right)\pi(a|s)\,da \\ &= \mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}} [\mathcal{R}(s,a) + \gamma V^\pi(s')], \end{aligned}$$

the proposal of TD is not to use a simple MC estimate but to iteratively update the estimation of the utility of each state towards the fixed point defined by the above equation. That is, when the transition $\langle s_t, a_t, s_{t+1}, r_t\rangle$ is observed, the value is updated by:

$$\widehat{V^\pi_\mu}(s_t) \leftarrow \widehat{V^\pi_\mu}(s_t) + \alpha(r_t + \gamma\widehat{V^\pi_\mu}(s_{t+1}) - \widehat{V^\pi_\mu}(s_t)).$$

The quantity $r_t + \gamma \widehat{V}^{\pi}_{\mu}(s_{t+1})$ is known as *state TD target*, and $r_t + \gamma \widehat{V}^{\pi}_{\mu}(s_{t+1}) - \widehat{V}^{\pi}_{\mu}(s_t)$ is the *state TD error*. Samples can be taken from trajectories distributed as ρ^{π}_{μ} or transitions drawn from d^{π}_{μ}.
The equivalent version of this method for the state-action value function is the SARSA algorithm (Rummery and Niranjan, 1994). Based on the fact that (Equation (2.2))

$$\begin{aligned} Q^{\pi}(s,a) &= \mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} \int_{\mathcal{A}} Q^{\pi}(s',a')\mathcal{P}(s'|s,a)\pi(a'|s')\,da'\,ds' \\ &= \mathop{\mathbb{E}}_{\substack{s'\sim\mathcal{P}(\cdot|s,a) \\ a'\sim\pi(\cdot|s')}} [\mathcal{R}(s,a) + \gamma Q^{\pi}(s',a')], \end{aligned}$$

the update rule for the estimate $\widehat{Q}^{\pi}_{\mu}(s_t, a_t)$ proposed by SARSA upon observation of the transition $\langle s_t, a_t, s_{t+1}, , a_{t+1}, r_t \rangle$ is

$$\widehat{Q}^{\pi}_{\mu}(s_t,a_t) \leftarrow \widehat{Q}^{\pi}_{\mu}(s_t,a_t) + \alpha(r_t + \gamma \widehat{Q}^{\pi}_{\mu}(s_{t+1},a_{t+1}) - \widehat{Q}^{\pi}_{\mu}(s_t,a_t)).$$

As before, the *state-action TD target* is defined by $r_t + \gamma \widehat{Q}^{\pi}_{\mu}(s_{t+1}, a_{t+1})$ and the *state-action TD error* is $r_t + \gamma \widehat{Q}^{\pi}_{\mu}(s_{t+1}, a_{t+1}) - \widehat{Q}^{\pi}_{\mu}(s_t, a_t)$.
Estimation of the Q-function needs "longer" samples than that of the V-function in the sense that, for each transition, it requires not only the arrival state but also an action performed in there. Thus, samples can be obtained from trajectories distributed as ρ^{π}_{μ} or transitions drawn from $\overline{d}^{\pi}_{\mu}$.

2.4.3 *Policy evaluation with function approximators*

MC (Section 2.4.1) and TD (Section 2.4.2) methods for policy evaluation require to explicitly maintain separate estimates of the value functions for every state and state-action pair. This is unfeasible when the state and action spaces are too big, and even impossible when they are continuous. This issue can be overcome by resorting to linear function approximators, but doing so makes it impossible to apply MC and TD approaches as presented before. However, they serve as a good inspiration for policy evaluation techniques with function approximators.
Although an important number of TD methods based on function approximation has been proposed along time (see e. g.Geist and Pietquin (2013), Dann, Neumann, and Peters (2014), Geist and Scherrer (2014)), the majority can be reduced to an optimization problem defined on the parameter space; they differ on the objective function and the mechanism used to optimize it. The most popular optimization functions are presented in Section 2.4.3.1, and the most common optimization techniques in Section 2.4.3.2; Section 2.4.3.3 concludes by presenting a specific example of a TD algorithm based on function approximation. The content of this section is mostly based on (Dann, Neumann, and Peters, 2014).

2.4.3.1 *The objective functions*

For the sake of simplicity, the objective functions presented here concern only the state value function; their equivalent versions for the state-action value function are easily derived from these by accordingly replacing the distributions in the expectations, and taking the appropriate feature map $\boldsymbol{\varphi}$ for the state-action space.
Let $\mathcal{V}_{\boldsymbol{\phi}}^{\mathcal{B}}$ the space of the linear function approximators for the V-function with feature map $\boldsymbol{\phi} = (\phi_1, ..., \phi_d)^\top$ and parameter space $\mathcal{B} \subset \mathbb{R}^d$. Therefore, for any $\beta \in \mathcal{B}$, the function $V_\beta \in \mathcal{V}_{\boldsymbol{\phi}}^{\mathcal{B}}$ is such that $V_\beta(s) = \boldsymbol{\phi}(s)^\top \beta$, for all $s \in \mathcal{S}$.
Given a function $p : \mathcal{S} \to \mathbb{R}^*$, $\mathbb{R}^* = \mathbb{R}^+ \cup \{0\}$, define the seminorm $\|\cdot\|_p$ on the space $\mathbb{R}^{\mathcal{S}}$ of real-valued functions with domain $\mathcal{S}$ by

$$\|V\|_p = \sqrt{\int_{\mathcal{S}} V(s)^2 p(s)\, ds}.$$

All the objective functions presented in this section are plugged into a minimization problem for the purpose of policy evaluation.

Mean Squared Error
A first proposal for the objective function, based on the original motivation of the function approximators, is the Mean Squared Error (MSE) measured with respect to the discounted future state distribution (Definition 2.9):

$$\mathrm{MSE}(\beta) = \|V^\pi - V_\beta\|^2_{\delta_\mu^\pi} \tag{2.9}$$

The main problem with this proposal is that it requires to know the real state value function, which is exactly what we want to approximate. A possible solution is to optimize towards Monte Carlo estimations of the returns at some points, but they are known to suffer from a high variance.

Mean Squared Bellman Error
In order to reduce this variance, the second objective function aims to optimize towards the result of one application of the Bellman operator (Definition 2.20), inspired by the fact that true value function is the unique fixed point of such operator. This gives the Mean Squared Bellman Error (MSBE):

$$\begin{aligned}\mathrm{MSBE}(\beta) &= \|V_\beta - T^\pi V_\beta\|^2_{\delta_\mu^\pi} \\ &= \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}\left[(V_\beta(s) - \mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}}[r(s,a,s') + \gamma V_\beta(s')])^2\right].\end{aligned} \tag{2.10}$$

By defining the state TD error $\delta(s,a,s') = r(s,a,s') + \gamma V_\theta(s') - V_\theta(s) = r(s,a,s') + (\gamma\boldsymbol{\phi}(s') - \boldsymbol{\phi}(s))^\top \beta$, the MSBE can be rewritten as:

$$\begin{aligned}\mathrm{MSBE}(\beta) = \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}\Bigg[\Big(\big(\mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}}[\gamma\boldsymbol{\phi}(s')] - \boldsymbol{\phi}(s)\big)^\top\beta \\ + \mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}}[r(s,a,s')]\Big)^2\Bigg].\end{aligned} \tag{2.11}$$

This formula shows that optimizing the MSBE is equivalent to solving a linear least-squares regression problem that has

- $\mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}}[\gamma\boldsymbol{\phi}(s')] - \boldsymbol{\phi}(s)$ as inputs,
- $-\mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{P}(\cdot|s,a)}}[r(s,a,s')]$ as outputs, and
- β as parameter.

However, the traditional formulation of the least-squares regression accounts for noise only in the outputs, while here the observed input is also affected by noise. The most problematic consequence of this issue is that solving the optimization problem usually requires, for the same sample starting at s, two independent observations of the successor state s': one for the input term and one for the output term. This is known as the *double-sampling problem*.

Mean Squared Temporal Difference Error

Ignoring the double-sampling problem and taking the same successor observation introduces a bias that results in optimizing the Mean Squared Temporal Difference Error (MSTDE) instead of the MSBE. The MSTDE is given by

$$\mathrm{MSTDE}(\beta) = \mathop{\mathbb{E}}_{(s,a,s')\sim d_\mu^\pi}[\delta(s,a,s')^2] \tag{2.12}$$

Mean Squared Projected Bellman Error

To avoid the difficulties related to the optimization of the MSBE, the Mean Squared Projected Bellman Error (MSPBE) proposes to perform the optimization towards the projection onto the space of representable functions $\mathcal{V}_{\boldsymbol{\phi}}^{\mathcal{B}}$ of one application of the Bellman operator:

$$\mathrm{MSPBE}(\beta) = \|V_\beta - \Pi T^\pi V_\beta\|_{\delta_\mu^\pi}^2, \tag{2.13}$$

where Π is the projection operator that maps a state function to the closest function in $\mathcal{V}_{\boldsymbol{\phi}}^{\mathcal{B}}$. For linearly parametrized families of functions, this has a closed-form solution

$$\begin{aligned}&\Pi V = \mathop{\mathrm{argmin}}_{V_\beta\in\mathcal{V}_{\boldsymbol{\phi}}^{\mathcal{B}}}\|V_\beta - V\|_{\delta_\mu^\pi}^2 \Rightarrow \\ &\Pi V(s) = \boldsymbol{\phi}(s)^\top\big(\mathop{\mathbb{E}}_{s_0\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s_0)\boldsymbol{\phi}(s_0)^\top]\big)^{-1}\mathop{\mathbb{E}}_{s'\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s')V(s')] \\ &\Pi V(s) = \boldsymbol{\phi}(s)^\top(M_\mu^{\pi,\Phi})^{-1}\mathop{\mathbb{E}}_{s'\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s')V(s')].\end{aligned} \tag{2.14}$$

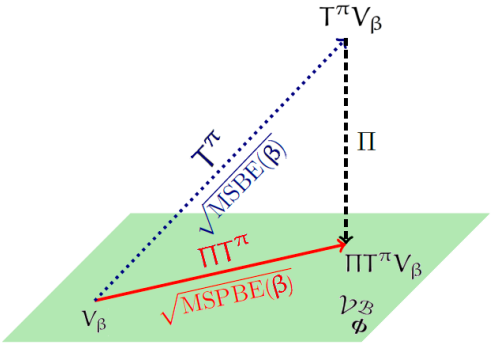


Figure 2.2: The MSBE minimizes the distance from the function approximator to its Bellman-operator-image, while MSPBE minimizes the distance to the Bellman-operator-image projected to the hypothesis space. (Adapted from Dann, Neumann, and Peters (2014))

The original form of the MSPBE seems to lose the connection to the MSE, which is the ideal objective function. However, Sutton et al. (2009) showed that

$$\begin{aligned}\text{MSPBE}(\beta) &= \|V_\beta - T^\pi V_\beta\|^2_{U_\mu^{\pi,\Phi}} \\ &= \left\| \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s)(V_\beta(s) - T^\pi V_\beta(s))] \right\|^2_{(M_\mu^{\pi,\Phi})^{-1}} \\ &= \left\| \mathop{\mathbb{E}}_{(s,a,s')\sim d_\mu^\pi}[\boldsymbol{\phi}(s)\delta(s,a,s')] \right\|^2_{(M_\mu^{\pi,\Phi})^{-1}},\end{aligned} \tag{2.15}$$

where the seminorm $\|\cdot\|^2_{U_\mu^{\pi,\Phi}}$ on $\mathbb{R}^S$ is defined by (the subscript $U_\mu^{\pi,\Phi}$ is just for notation)

$$\|V\|^2_{U_\mu^{\pi,\Phi}} = \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s)^\top V(s)](M_\mu^{\pi,\Phi})^{-1} \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s)V(s)],$$

and $\|\cdot\|^2_{(M_\mu^{\pi,\Phi})^{-1}}$ is a norm in $\mathbb{R}^d$ defined by

$$\|v\|^2_{(M_\mu^{\pi,\Phi})^{-1}} = v^\top (M_\mu^{\pi,\Phi})^{-1} v.$$

Thus, the MSPBE still measures the MSBE but does so with a different metric.
The MSPBE is actually obtained by neglecting the component of the MSBE that is orthogonal to the space $\mathcal{V}_{\boldsymbol{\Phi}}^{\mathcal{B}}$ (Figure 2.2). In fact,

$$\text{MSBE}(\beta) = \text{MSPBE}(\beta) + \|T^\pi V_\beta - \Pi T^\pi V_\beta\|^2_{\delta_\mu^\pi}.$$

Norm of Expected TD Update
The last line Equation (2.15) shows that the MSPBE is minimized if and only if

$$\mathop{\mathbb{E}}_{(s,a,s')\sim d_\mu^\pi}[\boldsymbol{\phi}(s)\delta(s,a,s')] = 0,$$

which means that there is no correlation between the feature vector and the state TD error. Inspired this, the Norm of Expected TD Update (NEU) proposes to optimize

$$\mathrm{NEU}(\beta) = \left\| \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[\boldsymbol{\phi}(s)(V_\beta(s) - \mathrm{T}^\pi V_\beta(s))] \right\|_2^2. \tag{2.16}$$

It has the same minimum of the MSPBE although the function might have a different shape.

Operator Error and Fixed-Point Error
The MSPBE can be optimized by solving two nested optimization problems, namely the Operator Error (OPE) and the Fixed-Point Error (FPE). They are defined by

$$\begin{aligned} \mathrm{OPE}(\beta', \omega) &= \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[(V_{\beta'}(s) - \mathrm{T}^\pi V_\omega(s))^2]; \\ \mathrm{FPE}(\beta, \omega') &= \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[(V_\beta(s) - V_{\omega'}(s))^2] \\ &= \mathop{\mathbb{E}}_{s\sim\delta_\mu^\pi}[(\boldsymbol{\phi}(s)^\top(\beta - \omega'))^2]. \end{aligned} \tag{2.17}$$

In the OPE, the parameter ω is fixed and the function is optimized with respect to β'; this intends to get the best approximation of the application of the Bellman operator to the function V_ω. In the FPE, β is fixed and the optimization is performed with respect to ω'; this aims to bring the parameters ω and β as close as possible.

2.4.3.2 *Optimization mechanisms*

Most of TD algorithms that rely on function approximators and aim to minimize one of the objective functions presented in Section 2.4.3.1 resort either to gradient-based, least-squares or probabilistic techniques in order to perform the optimization.
Gradient-based methods follow a gradient descent approach: the gradient of the selected objective function with respect to the approximator parameters is somehow estimated, and the parameters are updated towards the opposite direction of such gradient until some convergence conditions are satisfied. In particular, Stochastic Gradient Descent (SGD) is a very popular procedure for the calculation of the (approximate) gradient because it can work online giving estimations based on one sample at a time (Robbins and Monro, 1951).
Least-squares approaches tackle the optimization problem by formulating it as a linear-regression problem. This is then solved either directly with the closed-form solution or iteratively by performing incremental updates to the parameters. A special advantage of using the closed form solution is that there is no need to tune the step size used for the updates of the parameter.
Probabilistic techniques rely on a probabilistic perspective for solving

ALGORITHM	OBJECTIVE FUNCTION	MINIMIZATION PROCEDURE
GTD (Sutton, Szepesvári, and Maei, 2008)	MSPBE	SGD on NEU
RG (Leemon, 1995)	MSBE/MSTDE	SGD on MSBE
LSTD (Bradtke, 1996)	MSPBE	Closed-form least-squares
LSPE (Nedić and Bertsekas, 2003)	MSE	Iterative least-squares
KTD (Geist and Pietquin, 2014)	MSPBE	Probabilistic - Parameter tracking by Kalman Filtering

Table 2.4: Some examples of temporal difference algorithms based on function approximation (based on Dann, Neumann, and Peters (2014)).

the optimization problem: the most likely parameters for the available data are selected as the optimal ones. The main advantage of these approaches is that, in addition to the approximate minimum, they provide a measure of the uncertainty related to the found solution.

2.4.3.3 *Least Squares Temporal Difference*

Most of TD algorithms compatible with function approximators result from combining one of the objective functions presented in Section 2.4.3.1 with one of the optimization mechanisms shown in Section 2.4.3.2 and defining a couple implementation details. Table 2.4 shows some examples of such algorithms that have been proposed in the literature, pointing out the selected objective function and optimization procedure. This thesis focuses on the Least Squares Temporal Difference (LSTD) algorithm, which is explained in this section. For details on the other algorithms, the reader might refer to the surveys by Geist and Pietquin (2013), Dann, Neumann, and Peters (2014), Geist and Scherrer (2014).

LSTD (Nedić and Bertsekas, 2003) takes as objective function the MSPBE (Equation (2.13)) and tries to minimize it with the closed-form of its least-squares formulation. For the state value function, such formulation is evident in

$$\begin{aligned} \text{MSPBE}(\beta) = & \Big\| \mathop{\mathbb{E}}_{(s,a,s')\sim d_\mu^\pi} [\boldsymbol{\phi}(s)(\boldsymbol{\phi}(s) - \gamma\boldsymbol{\phi}(s'))^\top]\beta \\ & - \mathop{\mathbb{E}}_{(s,a,s')\sim d_\mu^\pi} [\boldsymbol{\phi}(s) r(s,a,s')] \Big\|^2_{(M^V_{\pi,\boldsymbol{\phi},\mu})^{-1}} \\ = & \; \|A^V_{\pi,\boldsymbol{\phi},\mu}\beta - b^V_{\pi,\boldsymbol{\phi},\mu}\|^2_{(M^V_{\pi,\boldsymbol{\phi},\mu})^{-1}}, \end{aligned} \tag{2.18}$$

so that the optimal parameter is given by

$$\beta_V^* = (A^V_{\pi,\boldsymbol{\phi},\mu})^{-1} b^V_{\pi,\boldsymbol{\phi},\mu}.$$

Since the above expression cannot be exactly calculated in practice, LSTD outputs an approximated value of β_V^* based on approximations of $A^V_{\pi,\boldsymbol{\phi},\mu}$ and $b^V_{\pi,\boldsymbol{\phi},\mu}$. In fact, from a dataset $\mathcal{D} = (\langle s_i, a_i, s'_i, r_i\rangle)_{i=1}^n$ drawn according to d^π_μ, the algorithm calculates the estimates:

$$\widehat{A^V_{\boldsymbol{\phi}}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\phi}(s_i)(\boldsymbol{\phi}(s_i) - \gamma\boldsymbol{\phi}(s'_i))^\top,$$

$$\widehat{b^V_{\boldsymbol{\phi}}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\phi}(s_i) r_i,$$

which are used to calculate the (approximate) optimal parameter:

$$\widehat{\beta_V^*}(\mathcal{D}) = (\widehat{A^V_{\boldsymbol{\phi}}}(\mathcal{D}))^{-1}\widehat{b^V_{\boldsymbol{\phi}}}(\mathcal{D}).$$

Robust inversion e. g.with Singular Value Decomposition (SVD) is used when inverting $\widehat{A^V_{\boldsymbol{\phi}}}$.

LSTD-Q (Lagoudakis and Parr, 2003) is the equivalent version of LSTD for the action-value function. By defining the seminorm

$$\|Q\|^2_{\zeta^\pi_\mu} = \mathop{\mathbb{E}}_{(s,a)\sim\zeta^\pi_\mu}[Q(s,a)^2]$$

on the space $\mathbb{R}^{\mathcal{S}\mathcal{A}}$ of real-valued functions with domain $\mathcal{S}\times\mathcal{A}$, LSTD-Q aims to minimize the MSPBE

$$\mathrm{MSPBE}(\beta) = \|Q_\beta - \Pi T^\pi Q_\beta\|^2_{\zeta^\pi_\mu},$$

where Π is the projection operator onto the space of linear function approximators for the Q-function, $\mathcal{Q}^{\mathcal{B}_Q}_{\boldsymbol{\varphi}}$, and T^π is the state-action Bellman expectation operator (Definition 2.20). The least-squares linear regression formulation of this problem is given by

$$\begin{aligned}\mathrm{MSPBE}(\beta) = &\Big\| \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^\pi_\mu}[\boldsymbol{\varphi}(s,a)(\boldsymbol{\varphi}(s,a) - \gamma\boldsymbol{\phi}(s',a'))^\top]\beta \\ &- \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^\pi_\mu}[\boldsymbol{\varphi}(s,a) r(s,a,s')]\Big\|^2_{(M^Q_{\pi,\boldsymbol{\varphi},\mu})^{-1}} \\ = &\|A^Q_{\pi,\boldsymbol{\varphi},\mu}\beta - b^Q_{\pi,\boldsymbol{\varphi},\mu}\|^2_{(M^Q_{\pi,\boldsymbol{\varphi},\mu})^{-1}},\end{aligned} \tag{2.19}$$

so that the optimal parameter is

$$\beta_Q^* = (A^Q_{\pi,\boldsymbol{\varphi},\mu})^{-1} b^Q_{\pi,\boldsymbol{\varphi},\mu}.$$

From a dataset $\mathcal{D} = (\langle s_i, a_i, s'_i, a'_i, r_i\rangle)_{i=1}^n$ drawn according to $\overline{d}^\pi_\mu$, the matrices are estimated through:

$$\widehat{A^Q_{\boldsymbol{\varphi}}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\varphi}(s_i,a_i)(\boldsymbol{\varphi}(s_i,a_i) - \gamma\boldsymbol{\varphi}(s'_i,a'_i))^\top,$$

$$\widehat{b^Q_{\boldsymbol{\varphi}}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\varphi}(s_i,a_i) r_i,$$

leading to the approximate optimal parameter

$$\widehat{\beta^*_Q}(\mathcal{D}) = (\widehat{A^Q_\varphi}(\mathcal{D}))^{-1}\widehat{b^Q_\varphi}(\mathcal{D}).$$

Again, robust inversion is used when inverting $\widehat{A^Q_\varphi}$.

2.5 THE ACTOR-CRITIC APPROACH

Policy-based and value-based algorithms, as well as their advantages and drawbacks, were briefly discussed in Section 2.2.3. Several attempts to combine the two techniques to leverage their benefits while trying to mitigate their problems have been proposed along time, being the *actor-critic* paradigm one of the most prominent ones (Barto, Sutton, and Anderson (1983), Konda and Tsitsiklis (2003), Peters and Schaal (2008)). It is actually inspired by the *policy iteration* algorithms (Howard, 1960) that greedily improve the policy based on the information of the utility values that it produces. The most important advantage of actor-critic over policy iteration is the ability to easily deal with continuous state and action spaces. It also provides the possibility to introduce prior knowledge of both the policy space and the value functions, and proceeds with smooth changes on the policy along the learning process (Grondman et al. (2012), Beitelspacher et al. (2006), Kober, Bagnell, and Peters (2013)).

In actor-critic algorithms, the search for the optimal policy is an iterative process composed by two phases: the first one aims to estimate the utility values produced by the current policy, and the second one decides how to update such policy based on the just computed estimations. The iteration goes on until some convergence conditions are met. Under this paradigm, the agent can be seen as the composition of two entities: the *critic*, in charge of executing the first phase, and the *actor*, that implements the second phase (Konda and Tsitsiklis, 2003). Figure 2.3 shows a graphical description of a typical actor-critic architecture.

Most of the actor-critic work is developed for agents that use parametric policies (Section 2.3) and function approximators for the value functions (Section 2.2.3). The work of the critic is actually a policy evaluation task (Section 2.4), so it can be implemented with any of the algorithms proposed in Section 2.4.3. The work of the actor is a policy improvement task based on the utility values of the policy; the Policy gradient theorem (Theorem 2.2) sets the justification and theoretical foundations of this idea, hence the actor is usually implemented with some policy gradient algorithm based on such result (see Section 2.3.3 or Section 2.3.2). Thus, actor-critic algorithms combine the good convergence properties of actor-only methods (i. e.policy gradient) with the low variance of value-based techniques (Grondman et al., 2012).

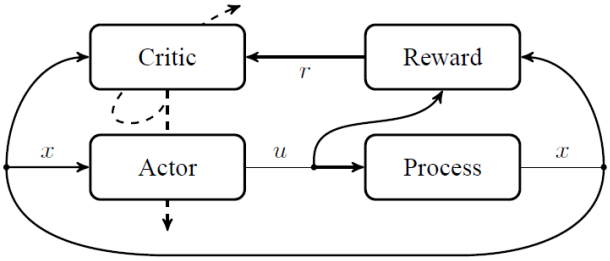


Figure 2.3: The actor takes an action u to produce a new state x and reward r. The critic then evaluates resulting in an update (dashed line) of itself and the actor. The update of the actor depends as well on the observed process' response (Grondman et al., 2012).

Many different implementations of actor-critic algorithms can be found in the literature.
Konda and Tsitsiklis (2003) selects an actor-critic implementation that uses a gradient-based policy evaluation approach (Section 2.4.3.2) for the critic to minimize the MSBE (Section 2.4.3.1) version with eligibility traces, and a state-action based policy gradient algorithm for the actor (Section 2.3.3), while performing online updates of the parameters of both the entities.
Another group of popular actor-critic implementations choose actors that follow natural gradient approaches (Section 2.3.4). Peters and Schaal (2008) used an LSTD-Q(λ)-based critic with compatible features (Theorem 2.3) so that the update rule of the actor turns out to be very simple. Some implementations with other approximation architectures are proposed by Bhatnagar et al. (2009).
Grondman et al. (2012) present a survey on actor-critic algorithms that perform online updates based on the observed TD error.

2.6 LIPSCHITZ MARKOV DECISION PROCESSES

This thesis focuses on environments with nice continuity properties that can be exploited during the design of the agent. The formal setting that aims to capture such behavior is exposed along this section, and serves as theoretical inspiration for the extension of the continuity properties across different MDPs proposed later on in this document.
The upcoming formulation has already been introduced by Pirotta, Restelli, and Bascetta (2015), who used it to find a step size in policy-gradient algorithms with improvement guarantees. They model the continuous properties of these problems by resorting to the concept of *Lipschitz continuous* functions and *pointwise Lipschitz continuous* functions.

Definition 2.28 (Lipschitz continuous function)**.** A function $f : X \to Y$, where (X, d_X) and (Y, d_Y) are metric spaces, is called Lipschitz continuous whenever

$$\forall x_1, x_2 \in X, d_Y(f(x_1), f(x_2)) \leqslant L_f d_X(x_1, x_2),$$

where $L_f \in \mathbb{R}$ is called Lipschitz constant. It is also said that f is L_f-Lipschitz continuous, or L_f-LC for short.

Definition 2.29 (Pointwise Lipschitz continuous function)**.** A function $f : X \to Y$, where (X, d_X) and (Y, d_Y) are metric spaces, is called pointiwse Lipschitz continuous whenever there exists a function $L_f : X \to \mathbb{R}$ such that, for any $x_1 \in X$, $L_f(x_1)$ satisfies

$$\forall x_2 \in X, d_Y(f(x_1), f(x_2)) \leqslant L_f(x_1) d_X(x_1, x_2).$$

The function L_f is required to be bounded, which implies that f is also Lipschitz continuous. It is also said that f is L_f-pointwise Lipschitz continuous, or L_f-PLC for short.

Euclidean spaces are given the metric structure with their natural Euclidean norm. Product spaces (e. g. $\mathcal{S} \times \mathcal{A}$) are endowed with the taxicab norm composed of the sum of the metrics on each factor space (e. g. $d_{\mathcal{S}\mathcal{A}} = d_{\mathcal{S}} + d_{\mathcal{A}}$). Spaces of probability distributions are provided with the Kantorovich or L^1-Wasserstein distance (Dudley, 2002) as the metric function.

Definition 2.30 (Kantorovich metric)**.** If p and q are two probability measures on a separable metric space X, the function

$$\mathcal{K}(p, q) = \sup_f \left\{ \left| \int f \, d(p - q) \right| \mid \|f\|_L \leqslant 1 \right\}$$

is a metric on the space of probability measures on X. $\|\cdot\|_L$ is the semi-norm defined by $\|f\|_L = \sup_{x_1 \neq x_2} \left\{ \frac{d_Y(f(x_1), f_(x_2))}{d_X(x_1, x_2)} \right\}$ on the space of real-valued functions whose domain is X.

With this in mind, it is possible to define *Lipschitz MDPs*.

Definition 2.31 (Lipschitz MDP)**.** An MDP $\mathcal{M}$ is said to be Lipschitz when its transition model and its reward model are Pointwise Lipschitz Continuous (PLC). For the transition model, it means that the function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, $(s, a) \mapsto \mathcal{P}(\cdot|s, a)$ is $L_{\mathcal{P}-\mathcal{S}\mathcal{A}}$-PLC. For the reward model, it means that the function $\mathcal{R}(\cdot, \cdot)$ is $L_{\mathcal{R}-\mathcal{S}\mathcal{A}}$-PLC.

Policies can be similarly defined to be Lipschitz, although this time the definition also includes the continuity with respect to the parameter space.

Definition 2.32 (Lipschitz Policy)**.** A parametric policy π_θ is said Lipschitz if it is PLC with respect both to the state space and the parameter space. For the state space, it means that when $\theta \in \Theta$ is fixed, the

function $\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A})$, $s \mapsto \pi_\theta(\cdot|s)$ is $L^\theta_{\pi-\mathcal{S}}$-PLC. For the parameter space, it means that when $s \in \mathcal{S}$ is fixed, for any $a \in \mathcal{A}$, the function $\pi_{(\cdot)}(a|s)$ is $L^{s,a}_{\pi-\Theta}$-PLC.

Note that the above definition differs from the one by Pirotta, Restelli, and Bascetta (2015) when it comes to the continuity with respect to the parameter space. The motivation for this is exposed in Section 4.1. However, under mild assumptions on the distribution $\pi_\theta(\cdot|s)$, it is possible to show that this definition implies the original one (Section D.1).
Pirotta, Restelli, and Bascetta (2015) also consider some assumptions on the gradient of the log-policy.

Definition 2.33 (Lipschitz gradient of policy logarithm)**.** A policy π_θ, $\theta \in \Theta \subset \mathbb{R}^d$, is said to have a Lipschitz logarithm gradient if it satisfies the following conditions:

- It is uniformly bounded with respect to the state-action space. That is, for fixed $\theta \in \Theta$ and $i = 1, ..., d$, there exists a constant M^i_θ such that, for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$

$$|\nabla_{\theta_i} \log \pi_\theta(a|s)| \leqslant M^i_\theta.$$

- When $\theta \in \Theta$ and $i = 1, ..., d$ are fixed, for any $a \in \mathcal{A}$, the function $\nabla_{\theta_i} \log \pi_\theta(a|\cdot)$ is $L^{\theta,i,a}_{\nabla \log \pi-\mathcal{S}}$-PLC.
- When $s \in \mathcal{S}$ and $i = 1, ..., d$ are fixed, for any $a \in \mathcal{A}$, the function $\nabla_{\theta_i} \log \pi_{(\cdot)}(a|s)$ is $L^{s,i,a}_{\nabla \log \pi-\Theta}$-PLC.

3

TRANSFER LEARNING

Machine Learning emerged as a subarea of Artificial Intelligence with the objective of studying and creating computer programs with the ability of autonomously learning how to execute a given task (Mitchell, 1997); for such purpose, human learning theory itself has been used as a source of inspiration for a couple of approaches. A particular example is the concept of TL, the ability of humans to enhance learning on a new task by exploiting previously acquired knowledge from similar tasks; the effectiveness of this approach has been demonstrated in practice (see e. g.Brown and Kane (1988)).
In the context of Machine Learning, the purpose of TL is to develop mechanisms that are able to produce knowledge that generalizes not only within the task but also across tasks (Taylor and Stone, 2009). An important number of TL algorithms for general Machine Learning already exist. For example, Caruana (1994) studies how backdrop neural networks can use the weights learned from one task as inductive bias for improving generalization in a group in similar tasks. Thrun and Pratt (1998) also present a detailed survey about transfer across various Machine Learning tasks.
RL has also been target for the TL paradigm. The purpose of this chapter is to describe how the general ideas of TL can be modified to fit in a RL context. Section 3.1 presents a series of relevant concepts that arise from bringing TL into RL, some of which allow to define a taxonomy for TL-RL algorithms. Section 3.2 lists and briefly describes some interesting TL-RL approaches.

3.1 TRANSFER LEARNING CONCEPTS FOR REINFORCEMENT LEARNING

The purpose of TL in the context of Machine Learning is to create algorithms that are capable not only of learning how to execute a certain task but to retain the acquired knowledge and to reuse it while learning new tasks, so that the performance therein is hopefully increased (Taylor and Stone (2009), Lazaric (2012)). This definition, although intuitively clear, is not enough to carefully study TL algorithms in the RL framework. There are two key concepts that should be further defined: the knowledge that is eligible to be reused, and the performance measure that is expected to be enhanced.
This concepts are elaborated on throughout this section. Section 3.1.1 defines what is considered as transferable knowledge in RL, and provides a brief taxonomy for TL-RL algorithms that can be derived from

such definition, while Section 3.1.2 presents the alternatives of metrics that can be used to assess a TL-RL algorithm.
When introduced into a transfer scenario, RL agents are added a new component in charge of implementing the transfer process, namely the *transfer algorithm*. Thus, it is important to remark that the term "agent" still refers to the collection of algorithms and procedures used during the learning process, and that the transfer algorithm is just a part of it. The work presented here focuses on the TL-RL scenario in which the agent has already interacted with a set of tasks, called *source tasks*, and expects to leverage such source experience to face a new unknown task, referred to as *target task*. Naturally, tasks are represented by MDPs within the TL-RL framework.

3.1.1 *Transferable knowledge and a Transfer Learning-Reinforcement Learning taxonomy*

The design of a TL algorithm for a RL problem needs to address the matter of what is to be considered as transferable knowledge. Taylor and Stone (2009) propose, in addition, to define

- the assumptions, if any, that are being made about the similarity between the tasks;
- the actual mechanism that identifies and transfers knowledge (source task selection);
- the specific way in which tasks are related (inter-task mappings); and
- the restrictions, if any, of the specific RL algorithms that are allowed.

These five items (including the type of transferable knowledge) correspond to the five dimensions of the taxonomy for TL-RL approaches that they present, and also define completely the transfer setting assumed by the algorithm.

With respect to the **assumed task differences**, a TL algorithm might encounter tasks that differ in any component of the MDP. When the tasks share the same state and action spaces, the setting is referred to as *transfer of task with fixed domains*; when the spaces are different, it is called *transfer of task with different domains* (Lazaric, 2012).

The **identification and transfer of source knowledge** concerns the selection of relevant source tasks and the subsequent transfer of knowledge instances. Regarding the task selection, four scenarios can be considered:

- a single source task has been selected by a human and thus the algorithm can rely on such selection;

- all the source tasks can be safely considered as relevant so that there is no need for a selection procedure;
- the algorithm maintains a "library" of source tasks and has the ability to select the most relevant ones when facing the target task;
- the algorithm has the ability to modify a single source task so to make its knowledge more useful on learning.

Concerning the specific **knowledge that can be transferred**, it can be mainly classified by its specificity:

- *Low-level* knowledge, that refers to sets of samples (either single transitions or complete trajectories), value functions, policies, full task models, or prior distributions. This can be directly introduced to the agent before facing the target task.
- *High-level* knowledge, that refers to suggested actions to execute under some situations (in the form of subsets of the action space), partial policies or options [1], rules or advices, important feature maps, proto-value functions [2], shaping rewards [3], and subtask definitions. These are harder to exploit directly to initialize the agent although knowing them can help the learning process.

When the transfer setting establishes that the tasks differ in any of the elements of the MDP, something else can be said about how they are different. The entities capturing such information are called **inter-task mappings**. In general, regarding inter-task mappings, the considered scenarios are:

- no mappings are necessary because the MDPs coincide entirely i. e.it is a fixed domain setting;
- the necessary mappings are provided by a human so the algorithm does not have to learn them; and
- the algorithm can leverage some knowledge to learn the mappings.

In the particular case that the algorithm is to learn the task mappings, a nested dimension can emerge depending on the knowledge used for such purpose. The authors consider three specific examples:

- the algorithm exploits source and target experience for learning the mappings;

1 Refer to (Sutton, Precup, and Singh, 1999) for a definition of *option*.
2 Refer to (Mahadevan, 2005) for a definition of *proto-value functions*.
3 Refer to (Ng, Harada, and Russell, 1999) for a definition of *reward shaping*.

- the algorithm is provided a qualitative description of the transition model of the MDPs so that the mappings can be inferred from there; and
- the algorithm counts with mappings for some of the differing components and will use them to learn the missing ones.

The type of knowledge that is being transferred directly **restricts the specific RL algorithms** that the agent can implement if it wants to leverage the source information. For instance, if the transfer algorithm tells the agent to reuse a source policy, the agent cannot implement a pure value-based method (Section 2.2.3).

Note that these dimensions are applicable to the transfer algorithm only and thus do not define the complete RL agent. In fact, a TL-RL agent is completely defined both when it decides how to implement the transfer and how to execute the learning process in the target task.

Lazaric (2012) provides another simpler taxonomy for TL-RL algorithms, which is mostly a coarser version of the one presented above. The most noteworthy difference concerns the categories in which the transferred knowledge is classified, for which he identifies three approaches: *instance transfer*, where samples (either trajectories or transitions) are the transferred elements; *representation transfer*, in which the transferred information is some higher level representation of the task components (e. g.the basis functions for the value functions); and *parameter transfer*, that transfer the parameters the target RL algorithm needs as inputs (e. g.steps size or starting point of a gradient-based algorithm).
Another important contribution is that the introduction of a formal framework for the transfer scenario in RL. Let $\mathfrak{T}$ be a space of tasks (i. e.MDPs) and Ω a probability distribution on such space. A (transfer) task environment is then defined by the tuple $\mathfrak{E} = \langle \mathfrak{T}, \Omega \rangle$. The idea inspiring transfer in RL (and in Machine Learning in general) is that a learner will probably perform well on a target task sampled from Ω, provided that it has an average good performance on an already known set of souce tasks sampled from there too. Transfer algorithms are hence designed to try to achieve such goal.
A TL-RL agent is then composed by two main components: the transfer algorithm and the RL algorithm. The first one can be seen as a function that takes knowledge from source and target tasks, as sampled from Ω, to produce transferred knowledge; the second one will then be a function with inputs the transferred and target knowledge and output a solution in the hypothesis space. It is possible, however, that the transfer algorithm does not have actual access to any target knowledge. Anyway, this shows again how the design of the transfer

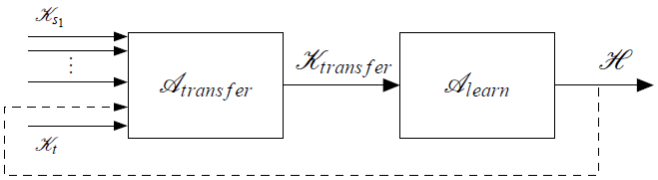


Figure 3.1: A TL-RL agent relies on the transfer algorithm to create the transfer knowledge based on source knowledge to improve the learning algorithm (Lazaric, 2012).

algorithm is closely related to the RL algorithm. Figure 3.1 depicts this scenario.

3.1.2 *Performance measures for Transfer Learning-Reinforcement Learning algorithms*

The design of the transfer algorithm is strongly driven by the performance measure that the agent wants to optimize. Taylor and Stone (2009) consider five different metrics:

- *Jumpstart*: given by the difference between the agent's initial performance in the target task with and without transfer. It provides information about the initial effect of the transfer algorithm but does not really capture the learning behavior produced later on.
- *Asymptotic performance*: given by the difference between the final performance of the agent in the target task with and without transfer. It gives a good insight on how transfer can affect learning on the long term, although it says nothing about the actual sample requirements necessary to get there.
- *Total reward*: given by the difference between the total reward accumulated by the agent during the learning process with and without transfer. It accounts both for benefit in the learning rate (sample demands) and in the asymptotic performance induced by transfer on the learning process.
- *Transfer ratio*: given by the ratio of the area below the learning curve when transfer takes place to that obtained when no transfer is performed. It can be calculated through

$$r = \frac{\textit{area under curve with transfer - area under curve without transfer}}{\textit{area under curve without transfer}}.$$

  This measures the relative improvement of transfer with respect to standard learning, and is more appropriate when both scenarios have the same final performance. The value is strictly related to the reward structure of the tasks; in fact, it is scale

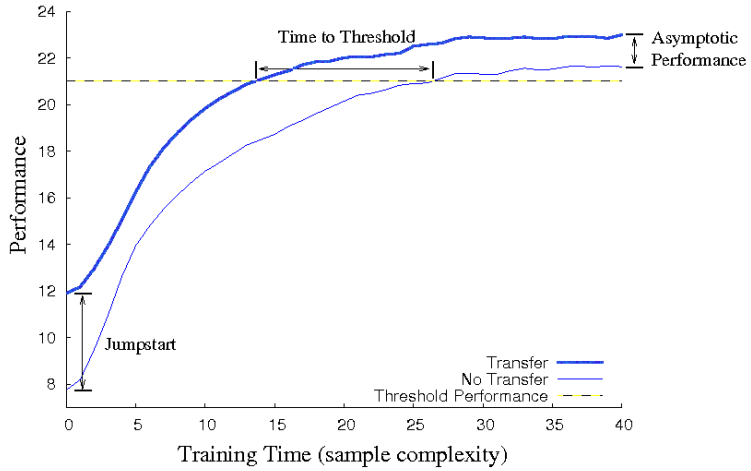


Figure 3.2: The graph shows a case in which learning with transfer outperforms standard learning with respect to jumpstart, asymptotic performance, time to threshold, and total reward (area below each curve) (Taylor and Stone, 2009).

invariant (the value is the same if the reward is multiplied by a constant factor) but not translation invariant (the value changes if the reward is added a constant term).

– *Time to threshold*: given by the difference between the time (in number of samples) required for the agent to achieve a given performance value with and without transfer. The main drawback of this metric is that the threshold is mainly arbitrarily chosen and so will be the perceived benefit of transfer. The threshold value is actually domain dependent and thus there are no standard methods to define it; nonetheless, in (Taylor, Stone, and Liu, 2007), there are a couple of suggested techniques for the selection of such value.

Figure 3.2 shows a graphical explanation for the metrics.

Lazaric (2012) refers all of these metrics in his work (with exception of the total reward) although presented in different categories. He considers that a transfer algorithm can set three types of objectives:

– *Learning speed improvement*, that refers to the reduction in the knowledge demands of the agent induced by the transfer algorithm. It can be measured with the time to threshold, the area ratio (transfer ratio), and with a *finite-sample analysis*. The latter, not considered in the above presented metrics, has a rather formal motivation with respect to the others (which are more empirically inspired); it aims to bound the error between the produced solution and the optimal one as a function of (among others) the number of samples needed to find it. The idea is that a good transfer algorithm will be able to get tighter bounds for

the same number of target samples with respect to the no transfer scenario.
Regardless of how it is measured, this objective is usually pursued by instance transfer-based algorithms.

- *Asymptotic improvement*, that refers to the gain in the performance of the found optimal policy caused by introducing transfer. It can be measured by comparing the respective asymptotic performances of the standard and transfer learning scenarios. A finite-sample analysis can take place here as well, by doing the bound on the error a function of the bias of the agent. The idea is that a good transfer algorithm will bias the agent towards a better hypothesis space with respect to the no transfer scenario. This objective is usually pursued by representation transfer-based algorithms.

- *Jumpstart improvement*, that refers to the gain in the initial performance caused by the initialization suggested by the transfer algorithm. It can be measured by comparing the respective initial performances of the standard and transfer learning scenarios. This objective is usually pursued by parameter transfer-based algorithms.

All the previously described metrics care only about the relative improvement caused by the transfer algorithm, but ignore the cost of acquiring the knowledge in the source tasks. Taylor and Stone (2009) talk about two different scenarios in this regard:

- *Total time scenario*, where the objective is to reduce the overall time necessary both for creating the source knowledge and reusing it in the target task. It is more suitable for evaluating a human-guided agent that is presented a sequence of tasks to learn from, expecting that it leverages on previous experience for learning each new task. The value of such an agent is in being able to optimize the complete learning process.

- *Target task time scenario*, where the objective is to reduce the learning time in the target task only. It is more appropriate for evaluating a fully autonomous agent that is already provided with source knowledge, and that must have the ability to select the most convenient source for a given target task, understand how they are related, and use it for learning. The power of such an agent can be summarized by its ability of effectively reusing past experience.

Figure 3.3 introduces these two scenarios in a graphical way.
In the ideal case, all the just described metrics should give results that demonstrate the benefit of transfer for the learning process; however,

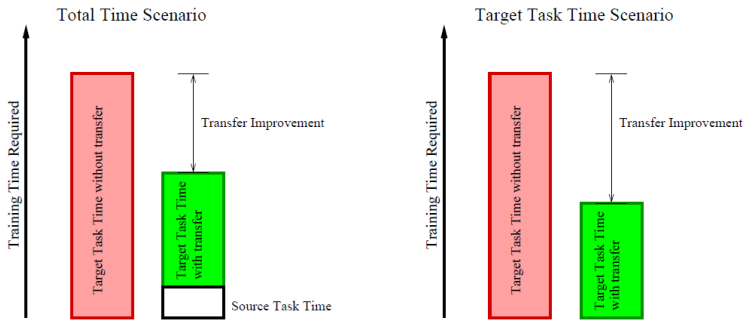


Figure 3.3: The figure shows the difference of measuring improvement in each time scenario. Which one to select depends on the assumptions about the agent and the specific objective of the problem (Taylor and Stone, 2009).

this might not always be the case. One of the most important open issues of TL in RL is the so called *negative transfer*, i. e., the negative impact on the learning process provoked by the introduction of useless source knowledge, which is reflected in negative results as suggested by the metrics. An important step towards fully autonomous TL-RL agents is for them to have the ability not only of identifying relevant source knowledge, but of recognizing and ignoring the information that could harm the learning process.

3.2 TRANSFER LEARNING ALGORITHMS IN REINFORCEMENT LEARNING

The area of TL in RL has been quite active during the last years. Many different works trying various approaches have been published, each one providing novel and interesting insights on this field. This section briefly recalls some of those proposals that are relevant for the scope of this thesis.

Lazaric, Restelli, and Bonarini (2008) present an algorithm that selectively transfers samples (transitions) from the source tasks with the goal of augmenting the dataset to be given to a Fitted Q Iteration algorithm (Antos, Munos, and Szepesvári, 2007). The selection is based on some metrics that account for the local similarity of the transition and reward models of each source task with respect to the target task. Source tasks are sorted according to their *compliance* with the target task, and samples from each one are drawn according to their *relevance*; the number of samples taken from each task is proportional to their rank after the previous sorting process.

Lazaric and Restelli (2011) introduce three algorithms whose objective is to use transfer to augment a dataset for a Fitted Q Iteration algorithm as well. However, this approach relies on the minimization of some approximation error for creating the sampling strategy rather than on the per-sample similarity of the tasks. The three algorithms

are based on the proportion used to draw samples from each source task, but they differ on the way such proportion is obtained: the first one, All Samples Transfer, assumes that the proportion is fixed so that the algorithm only has to perform the sampling; the second one, Best Average Transfer, finds the proportion that minimizes the *estimated transfer error*; the third one, Best Trade-off Transfer, finds the proportion that optimizes the trade-off between the *transfer error* and the *estimation error*.

Laroche and Barlier (2017) give another example of sample transfer for Fitted Q Iteration algorithms. They assume that the tasks differ only in the reward models (so it is a *shared dynamics* transfer setting), which implies that the transferred samples introduce uncertainty only on the reward and not on the transition model. Thus, the algorithm estimates the reward model on the target samples only and uses it to cast the source samples directly into the dataset. Exploration is guided with *optimism in the face of uncertainty* heuristic, based on confidence intervals from UCRL.

The algorithm proposed by Barrett, Taylor, and Stone (2010) transfers the Q-function from the source task for executing SARSA(λ). They assume a transfer setting within different domains where the inter-task mapping for the state-action spaces are given to the agent. Thus, the target value function is modeled as the sum of the source value function (appropriately composed with the task mappings) and some function approximator whose parameters are learned by means of the SARSA updates.

4

TRANSFER LEARNING APPROACHES FOR ACTOR-CRITIC ALGORITHMS

The goal of this section is to present the novel approach for transfer in Lipschitz continuous task environments proposed by this thesis. The algorithm solves an optimization problem for the transfer part whose output are the weights to be used by the actor and the critic on calculating their estimates. The idea of weighted estimates was inspired by IS approaches and attempts to overcome the variance issues inherent to them (the reader who is not familiar with Importance Sampling can refer to Chapter A for a review). Section 4.1 introduces the formal setting of Lipschitz task environments that sets the theoretical basis for the design of the transfer algorithm. Section 4.3 describes the details of the implementation of the actor-critic approach used by RL component of the agent. Section 4.4 presents an IS-based approach for transfer into this actor-critic algorithm. Finally, Section 4.5 and Section 4.6 describe the optimistic and pessimistic approaches proposed by this thesis for transferring, both inspired by the ideas of the IS formulation.

4.1 THE SETTING: LIPSCHITZ CONTINUOUS TASK ENVIRONMENTS

Section 2.6 already introduced how smooth continuous properties within the elements of a single task can be appropriately modeled. For the purpose of designing TL algorithms in this context, such notion has to be extended to elements between different tasks.
Unlike in the formulation for a single MDP, the Kantorovich distance can no longer be used here for measuring distance between probability distributions. Although frequently used in the context of MDPs (Ferns, Panangaden, and Precup (2005), Hinderer (2005), Rachelson and Lagoudakis (2010)), this metric provides a rather global information about the similarity of the distributions which is not suitable for the purpose of the approach presented in this thesis (Appendix B). Thus, Lipschitz continuity across tasks is modeled with stronger conditions that explicitly account for the pointwise similarity of the distributions.

The space of tasks has to be given a metric structure before introducing any Lipschitz condition. This can be achieved by modeling such space as a parametric one, and directly inheriting the metric nature provided by the parameter space. Hence, the concept of *parametric task environment* is born.

Definition 4.1 (Parametric task environment)**.** A parametric task environment is a task environment $\mathfrak{E} = \langle \mathfrak{T}, \Omega \rangle$ such that all the tasks in the task space are identified by a parameter vector, $\mathfrak{T} = \{T_\upsilon \mid \upsilon \in \Upsilon \subset \mathbb{R}^{n_\mathfrak{T}}\}$. In particular, the tasks are parametrized by 2-dimensional vectors $\upsilon = (\epsilon, \xi)^\top$ where ϵ is the parameter of the initial state distribution and the transition model of the task, and ξ is that of the reward model.
The parameter space of the initial state distribution and of the transition model is denoted by $\mathcal{E} \subset \mathbb{R}$, and that of the reward model by $\Xi \subset \mathbb{R}$. Therefore, the task space will be of the form $\mathfrak{T} = \{T_{\epsilon,\xi} \mid (\epsilon, \xi) \in \mathcal{E} \times \Xi\}$.

With this in mind, and using the same terminology of the framework by Lazaric (2012), the formal setting can be finally described.

Definition 4.2 (Lipschitz task environment)**.** A parametric task environment $\mathfrak{E} = \langle \mathfrak{T}, \Omega \rangle$ is said to be Lipschitz if each task is Lipschitz in the sense of Definition 2.31, and if the initial state distribution, transition model and reward model are Lipschitz with respect to their parameter spaces. For the initial state distribution, it means that for any $s \in \mathcal{S}$, the function $\mu_{(\cdot)}(s)$ is $L^s_{\mu-\mathcal{E}}$-PLC. For the transition model, it means that when $s \in \mathcal{S}$ and $a \in \mathcal{A}$ are fixed, for any $s' \in \mathcal{S}$, the function $\mathcal{P}_{(\cdot)}(s'|s,a)$ is $L^{s,a,s'}_{\mathcal{P}-\mathcal{E}}$-PLC. For the reward model, it means that when $s \in \mathcal{S}$ and $a \in \mathcal{A}$ are fixed, for any $s' \in \mathcal{S}$, the function $r_{(\cdot)}(s,a,s')$ is $L^{s,a,s'}_{r-\Xi}$-PLC.

4.2 THE PROBLEM

This section formalizes the transfer problem tackled by this thesis.
Let $\mathfrak{E} = \langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment in the sense of Definition 4.2, and $(T_j)_{j=1}^m$ be tasks with parameters $(\epsilon_j, \xi_j)_{j=1}^m$ sampled from such environment. Let Π_Θ be a parametric policy space whose policies are Lipschitz in the sense of Definition 2.32 and whose logarithm gradients are Lipschitz in the sense of Definition 2.33. Let $(\pi_j)_{j=2}^m$ be fixed policies from Π_Θ with parameters $(\theta_j)_{j=2}^m$.
Taking T_1 as the target task, and assuming that the agent has access to experience from the source tasks $(T_j)_{j=2}^m$ obtained through the policies $(\pi_j)_{j=2}^m$ and to the Lipschitz constants that govern the continuity across tasks, the goal of the TL-RL agent is to leverage such knowledge so to possibly speed-up the learning process in T_1.
The agent is assumed to have very few knowledge about the target task: it only knows the parameters and has access to some experience from there. In addition to the source experience, the agent is assumed to know the source tasks' models and parameters. Nothing is said about the source policies, but, as shown in Chapter 5, the transfer gives better results when they correspond to the source optimal policies.

Note that the assumption of knowing the target parameters is essential to be able to use the Lipschitz conditions, because otherwise the agent lacks of information about the similarity between the tasks.
The fact that knowing the target parameters does not mean knowing the actual models is an abstraction for the cases in which the models are too complex for really knowing how the parameters define them. In the case of the source tasks, it would mean that they have some black-box mechanism that can reliably replicate the model.
In the remaining of this chapter, to make notation simpler, scripts denoting dependence on the task parameters (ϵ_j, ξ_j) and on the policy parameters θ_j are replaced by just the index j.

4.3 THE ACTOR-CRITIC IMPLEMENTATION

The TL-RL problem formalized in Section 4.2 is approached in this thesis from an actor-critic perspective (Section 2.5). This section describes the specific implementation of such technique that is used by the agent in its RL component; therefore, only one task is considered here.
The continuous nature of the problem makes function approximation based critics and policy-gradient based actors a reasonable choice. In particular, the critic implements LSTD (Section 2.4.3.3) as the policy evaluation mechanism for both the V-function and the Q-function, and the actor calculates the gradient based on the Policy Gradient Theorem (Theorem 2.2) with baselines for a lower-variance estimate. Thus, the learning process of the RL agent is performed according to the following outline: starting from an initial randomly selected policy π_{θ_0}, the agent collects experience by interaction with the task through π_{θ_0}, that is used by the critic to execute LSTD; based on the resulting evaluation of the policy, the actor applies the Policy Gradient Theorem to find the gradient of the performance with respect to the parameter of the policy, and updates such parameter accordingly; then, new experience is collected by executing the new policy, and the process is iterated until some convergence conditions are met. A pseudo-code describing this is given in Algorithm 4.1. The critic and the actor are described with more details in the remaining of the section.

4.3.1 *The critic*

Regarding the critic, the case of the V-function is explained first and the one of the Q-function is described right after that.
Let $\mathcal{V}_{\boldsymbol{\Phi}}^{\mathcal{B}_V}$ be the space of parametric linear function approximators for the V-function assumed by the critic, where $\boldsymbol{\Phi}$ is the feature map and $\mathcal{B}_V$ is the parameter space. Recall that, when evaluating a policy π_θ, the purpose of LSTD is to minimize the MSPBE (Equation (2.18)); such

Algorithm 4.1 Actor-critic algorithm in the no-transfer scenario

function ACTOR-CRITIC(T) ▷ T is task to be learned
  $\theta \leftarrow$ randomly chosen initial policy parameter
  while has not converged **do**
    $\mathcal{D} \leftarrow$ experience from T with π_θ
    $V^\theta, Q^\theta \leftarrow \text{LSTD}(\mathcal{D})$
    $\nabla_\theta J \leftarrow \text{CALCULATE-GRADIENT}(\mathcal{D}, \pi_\theta, V^\theta, Q^\theta)$
    $\theta \leftarrow \theta + \alpha \nabla_\theta J$
  end while
end function

minimum is attained at (note that the dependence on π_θ is denoted by just writing θ)

$$\beta^*_{(V)} = (A^{(V)}_{\mu,\theta,\boldsymbol{\phi}})^{-1} b^{(V)}_{\mu,\theta,\boldsymbol{\phi}},$$

where

$$\begin{aligned} A^{(V)}_{\mu,\theta,\boldsymbol{\phi}} &= \mathop{\mathbb{E}}_{(s,a,s')\sim d^\theta_\mu}[\boldsymbol{\phi}(s)(\boldsymbol{\phi}(s) - \gamma\boldsymbol{\phi}(s'))^\top] \\ &= \mathop{\mathbb{E}}_{(s,a,s')\sim d^\theta_\mu}[\Delta_{\boldsymbol{\phi}}(s,a,s')], \\ b^{(V)}_{\mu,\theta,\boldsymbol{\phi}} &= \mathop{\mathbb{E}}_{(s,a,s')\sim d^\theta_\mu}[\boldsymbol{\phi}(s) r(s,a,s')] \\ &= \mathop{\mathbb{E}}_{(s,a,s')\sim d^\theta_\mu}[\rho_{\boldsymbol{\phi}}(s,a,s')]. \end{aligned} \tag{4.1}$$

Given the experience $\mathcal{D} = (\langle s_i, a_i, s'_i, r_i\rangle)_{i=1}^n$ sampled from d^θ_μ, the above matrices are approximated with

$$\begin{aligned} \widehat{A^{(V)}_{\boldsymbol{\phi}}}(\mathcal{D}) &= \frac{1}{n}\sum_{i=1}^n \Delta_{\boldsymbol{\phi}}(s_i, a_i, s'_i), \\ \widehat{b^{(V)}_{\boldsymbol{\phi}}}(\mathcal{D}) &= \frac{1}{n}\sum_{i=1}^n \rho_{\boldsymbol{\phi}}(s_i, a_i, s'_i), \end{aligned} \tag{4.2}$$

leading to the approximate optimal parameter for the V-function approximator

$$\widehat{\beta^*_{(V)}}(\mathcal{D}) = (\widehat{A^{(V)}_{\boldsymbol{\phi}}}(\mathcal{D}))^{-1} \widehat{b^{(V)}_{\boldsymbol{\phi}}}(\mathcal{D}).$$

Let $\mathcal{Q}^{\mathcal{B}_Q}_{\boldsymbol{\varphi}}$ be the space of parametric linear function approximators for the Q-function assumed by the critic, where $\boldsymbol{\varphi}$ is the feature map and $\mathcal{B}_Q$ is the parameter space. Recall that, when evaluating a policy π_θ, the purpose of LSTD is to minimize the MSPBE (Equation (2.19)); such minimum is attained by (note that the dependence on π_θ is denoted by just writing θ)

$$\beta^*_{(Q)} = (A^{(Q)}_{\mu,\theta,\boldsymbol{\varphi}})^{-1} b^{(Q)}_{\mu,\theta,\boldsymbol{\varphi}},$$

Algorithm 4.2 LSTD in the no-transfer scenario

function LSTD($\mathcal{D}$)
$A^V \leftarrow \frac{1}{n}\sum_{(s,a,s')\in\mathcal{D}} \boldsymbol{\phi}(s)(\boldsymbol{\phi}(s)-\gamma\boldsymbol{\phi}(s'))^\top$
$b^V \leftarrow \frac{1}{n}\sum_{(s,a,s')\in\mathcal{D}} \boldsymbol{\phi}(s)r(s,a,s')$
$\beta^V \leftarrow (A^V)^{-1}b^V$
$A^Q \leftarrow \frac{1}{n}\sum_{(s,a,s',a')\in\mathcal{D}} \boldsymbol{\varphi}(s,a)(\boldsymbol{\varphi}(s,a)-\gamma\boldsymbol{\varphi}(s',a'))^\top$
$b^Q \leftarrow \frac{1}{n}\sum_{(s,a,s',a')\in\mathcal{D}} \boldsymbol{\varphi}(s,a)r(s,a,s')$
$\beta^Q \leftarrow (A^Q)^{-1}b^Q$
return V_{β^V}, Q_{β^Q} ▷ Structures for value function evaluation
end function

where

$$\begin{aligned} A^{(Q)}_{\mu,\theta,\boldsymbol{\varphi}} &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^{\theta}_{\mu}}[\boldsymbol{\varphi}(s,a)(\boldsymbol{\varphi}(s,a)-\gamma\boldsymbol{\varphi}(s',a'))^\top] \\ &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^{\theta}_{\mu}}[\Delta_{\boldsymbol{\varphi}}(s,a,s',a')], \\ b^{(Q)}_{\mu,\theta,\boldsymbol{\varphi}} &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^{\theta}_{\mu}}[\boldsymbol{\varphi}(s,a)r(s,a,s')] \\ &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}^{\theta}_{\mu}}[\rho_{\boldsymbol{\varphi}}(s,a,s',a')]. \end{aligned} \tag{4.3}$$

Given the experience $\mathcal{D} = (\langle s_i, a_i, s'_i, a'_i, r_i\rangle)_{i=1}^n$ sampled from $\overline{d}^{\theta}_{\mu}$, the above matrices are approximated with

$$\begin{aligned} \widehat{A^{(Q)}_{\boldsymbol{\varphi}}}(\mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n}\Delta_{\boldsymbol{\varphi}}(s_i,a_i,s'_i,a'_i), \\ \widehat{b^{(Q)}_{\boldsymbol{\varphi}}}(\mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n}\rho_{\boldsymbol{\varphi}}(s_i,a_i,s'_i,a'_i), \end{aligned} \tag{4.4}$$

leading to the approximate optimal parameter for the Q-function approximator

$$\widehat{\beta^*_{(Q)}}(\mathcal{D}) = (\widehat{A^{(Q)}_{\boldsymbol{\varphi}}}(\mathcal{D}))^{-1}\widehat{b^{(Q)}_{\boldsymbol{\varphi}}}(\mathcal{D}).$$

The whole critic is summarized in Algorithm 4.2. Note that the output is not the complete representation of the value functions but rather any object that can evaluate such functions (e. g.a wrapped procedure with the feature map and the parameter vector).

4.3.2 *The actor*

Let Π_Θ be the parametric family of policies assumed by the actor. Recall that the Policy Gradient Theorem with baseline (Theorem 2.2) states that the gradient of the performance with respect to the para-

Algorithm 4.3 Gradient estimation in the no-transfer scenario

function CALCULATE-GRADIENT($\mathcal{D}, \pi_\theta, V^\theta, Q^\theta$)
  $\nabla_\theta J \leftarrow \frac{1}{n(1-\gamma)} \sum_{(s,a)\in\mathcal{D}} (Q^\theta(s,a) - V^\theta(s)) \nabla_\theta \log \pi_\theta(a|s)$
  return $\nabla_\theta J$
end function

meters at the policy π_θ is given by (note that the dependence on π_θ is denoted by just writing θ):

$$\nabla_\theta J_\mu^\theta = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta} [(Q^\theta(s,a) - b(s)) \nabla_\theta \log \pi_\theta(a|s)],$$

where b is any real-valued function of the state space. In particular, the actor is allowed to use the V-function as a baseline, which leads to the expression

$$\begin{aligned} \nabla_\theta J_\mu^\theta &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta} [(Q^\theta(s,a) - V^\theta(s)) \nabla_\theta \log \pi_\theta(a|s)] \\ &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_\mu^\theta} [\eta_\theta(s,a)]. \end{aligned} \tag{4.5}$$

Thus, given the experience $\mathcal{D} = (\langle s_i, a_i, Q^\theta(s_i,a_i), V^\theta(s_i)\rangle)_{i=1}^n$ sampled from ζ_μ^θ, the actor can estimate the gradient through

$$\widehat{\nabla_\theta J}(\mathcal{D}) = \frac{1}{n(1-\gamma)} \sum_{i=1}^n \eta_\theta(s_i, a_i).$$

However, the actor does not have access to the actual value functions but only to the approximates provided by the critic. Therefore, it is given experience in the form $\mathcal{D} = (\langle s_i, a_i, \widehat{Q^\theta}(s_i,a_i), \widehat{V^\theta}(s_i)\rangle)_{i=1}^n$ so that the resulting estimator is

$$\begin{aligned} \widehat{\nabla_\theta J}(\mathcal{D}) &= \frac{1}{n(1-\gamma)} \sum_{i=1}^n \nabla_\theta \log \pi_\theta(a_i|s_i) (\widehat{Q^\theta}(s_i,a_i) - \widehat{V^\theta}(s_i)) \\ &= \frac{1}{n(1-\gamma)} \sum_{i=1}^n \widehat{\eta_\theta}(s_i,a_i). \end{aligned} \tag{4.6}$$

The actor is summarized in Algorithm 4.3.

4.4 TRANSFER WITH IMPORTANCE SAMPLING

Having in mind the TL-RL problem presented in Section 4.2 and the actor-critic implementation proposed in Section 4.3, an IS based method for approaching such problem is introduced in this section (the reader can refer to Chapter A for a review of IS). The estimators used by the critic and the actor are extended to consider the IS scenario. Those of the critic is studied first, and they are followed by those of the actor.

4.4.1 *The critic*

For the case of the V-function's critic, from Equation (4.1) it follows that

$$\begin{aligned} A^{(V)}_{1,\phi} &= \mathop{\mathbb{E}}_{(s,a,s')\sim d_1}[\Delta_\phi(s,a,s')] \\ &= \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}\left[\Delta_\phi(s,a,s')\frac{d_1(s,a,s')}{d_j(s,a,s')}\right] \end{aligned}$$

and that

$$\begin{aligned} b^{(V)}_{1,\phi} &= \mathop{\mathbb{E}}_{(s,a,s')\sim d_1}[\rho_{1,\phi}(s,a,s')] \\ &= \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}\left[\rho_{1,\phi}(s,a,s')\frac{d_1(s,a,s')}{d_j(s,a,s')}\right] \end{aligned}$$

for any source task T_j; therefore, given the experience $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, s'_{ij}, r_{ij}\rangle)_{j=1}^m{}_{i=1}^{n_j}$, $n = \sum_{j=1}^m n_j$, where each $\mathcal{D}_j$ is sampled according to d_j, the estimates for the matrices in Equation (4.2) can be extended to:

$$\begin{aligned} \widehat{A^{(V)}_\phi}(\mathcal{D}) &= \frac{1}{n}\sum_{j=1}^m\sum_{i=1}^{n_j} w^V_{ij}\Delta_\phi(s_{ij},a_{ij},s'_{ij}), \\ \widehat{b^{(V)}_\phi}(\mathcal{D}) &= \frac{1}{n}\sum_{j=1}^m\sum_{i=1}^{n_j} w^V_{ij}\rho_{1,\phi}(s_{ij},a_{ij},a'_{ij}), \end{aligned} \tag{4.7}$$

where

$$w^V_{ij} = \frac{d_1(s_{ij},a_{ij},s'_{ij})}{d_j(s_{ij},a_{ij},s'_{ij})}.$$

This leads to the approximate optimal parameter for the V-function approximator

$$\widehat{\beta^*_{(V)}}(\mathcal{D}) = (\widehat{A^{(V)}_\phi}(\mathcal{D}))^{-1}\widehat{b^{(V)}_\phi}(\mathcal{D}).$$

For the case of the Q-function's critic, from Equation (4.3) it follows that

$$\begin{aligned} A^{(Q)}_{1,\varphi} &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim \overline{d}_1}[\Delta_\varphi(s,a,s',a')] \\ &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim \overline{d}_j}\left[\Delta_\varphi(s,a,s',a')\frac{\overline{d}_1(s,a,s',a')}{\overline{d}_j(s,a,s',a')}\right] \end{aligned}$$

and that

$$\begin{aligned} b^{(Q)}_{1,\varphi} &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim \overline{d}_1}[\rho_{1,\varphi}(s,a,s',a')] \\ &= \mathop{\mathbb{E}}_{(s,a,s',a')\sim \overline{d}_j}\left[\rho_{1,\varphi}(s,a,s',a')\frac{\overline{d}_1(s,a,s',a')}{\overline{d}_j(s,a,s',a')}\right] \end{aligned}$$

Algorithm 4.4 LSTD in the Importance Sampling scenario

function WEIGHTED-LSTD($\mathcal{D}, w^Q, w^V$)
$A^V \leftarrow \frac{1}{n}\sum_{(s,a,s')\in\mathcal{D}} w^V(s,a,s')\boldsymbol{\phi}(s)(\boldsymbol{\phi}(s)-\gamma\boldsymbol{\phi}(s'))^\top$
$b^V \leftarrow \frac{1}{n}\sum_{(s,a,s')\in\mathcal{D}} w^V(s,a,s')\boldsymbol{\phi}(s)r_1(s,a,s')$
$\beta^V \leftarrow (A^V)^{-1}b^V$
$A^Q \leftarrow \frac{1}{n}\sum_{(s,a,s',a')\in\mathcal{D}} w^Q(s,a,s',a')\boldsymbol{\varphi}(s,a)(\boldsymbol{\varphi}(s,a) - \gamma\boldsymbol{\varphi}(s',a'))^\top$
$b^Q \leftarrow \frac{1}{n}\sum_{(s,a,s',a')\in\mathcal{D}} w^Q(s,a,s',a')\boldsymbol{\varphi}(s,a)r_1(s,a,s')$
$\beta^Q \leftarrow (A^Q)^{-1}b^Q$
return V_{β^V}, Q_{β^Q}
end function

for any source task T_j; therefore, given the experience $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, s'_{ij}, a'_{ij}, r_{ij}\rangle)_{j=1}^{m}{}_{i=1}^{n_j}$, $n = \sum_{j=1}^m n_j$, where each $\mathcal{D}_j$ is sampled according to $\overline{d}_j$, the estimates for the matrices in Equation (4.4) can be extended to:

$$\begin{aligned}\widehat{A_{\boldsymbol{\varphi}}^{(Q)}}(\mathcal{D}) &= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j} w_{ij}^Q \Delta_{\boldsymbol{\varphi}}(s_{ij}, a_{ij}, s'_{ij}, a'_{ij}),\\ \widehat{b_{\boldsymbol{\varphi}}^{(Q)}}(\mathcal{D}) &= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j} w_{ij}^Q \rho_{1,\boldsymbol{\varphi}}(s_{ij}, a_{ij}, s'_{ij}, a'_{ij}),\end{aligned} \tag{4.8}$$

where

$$w_{ij}^Q = \frac{\overline{d}_1(s_{ij}, a_{ij}, s'_{ij}, a'_{ij})}{\overline{d}_j(s_{ij}, a_{ij}, s'_{ij}, a'_{ij})}.$$

This leads to the approximate optimal parameter for the Q-function approximator

$$\widehat{\beta^*_{(Q)}}(\mathcal{D}) = (\widehat{A_{\boldsymbol{\varphi}}^{(Q)}}(\mathcal{D}))^{-1}\widehat{b_{\boldsymbol{\varphi}}^{(Q)}}(\mathcal{D}).$$

Algorithm 4.4 extends Algorithm 4.2 to the IS-based scenario.
Note that $w_{ij}^Q = w_{ij}^V \frac{\pi_1(a'_{ij}|s'_{ij})}{\pi_j(a'_{ij}|s'_{ij})}$.
Furthermore, notice that the estimators in Equation (4.7) and in Equation (4.8) are using reward function from the target task even in the state-action pairs taken from the source tasks. In practice, since the reward function is usually selected by the designer of the agent, this is not a problem; however, whenever it is not possible to access the target reward function, any method can be used to approximate it (possibly exploiting the source samples as well).

Algorithm 4.5 Gradient estimation in the Importance Sampling scenario

function WEIGHTED-CALCULATE-GRADIENT($\mathcal{D}, \pi_1, V_1, Q_1, w_{\nabla J}$)
$\nabla_\theta J \leftarrow \frac{1}{n(1-\gamma)} \sum_{(s,a)\in\mathcal{D}} w^{\nabla J}(s,a)(Q_1(s,a) - V_1(s))\nabla_\theta \log \pi_1(a|s)$
return $\nabla_\theta J$
end function

4.4.2 *The actor*

For the actor, from Equation (4.5) it follows that

$$\begin{aligned}\nabla_\theta J_1 &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_1} [\eta_1(s,a)] \\ &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} \left[\eta_1(s,a)\frac{\zeta_1(s,a)}{\zeta_j(s,a)}\right]\end{aligned}$$

for any source task T_j; therefore, given the experience $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, \widehat{\eta_1}(s_{ij}, a_{ij})\rangle)_{j=1}^m{}_{i=1}^{n_j}$, $n = \sum_{j=1}^m n_j$, where each $\mathcal{D}_j$ is sampled according to ζ_j, the estimate for the gradient in Equation (4.6) can be extended to:

$$\widehat{\nabla_\theta J_1}(\mathcal{D}) = \frac{1}{n(1-\gamma)} \sum_{j=1}^{m} \sum_{i=1}^{n_j} w_{ij}^{\nabla J} \widehat{\eta_1}(s_i, a_i), \tag{4.9}$$

where

$$w_{ij}^{\nabla J} = \frac{\zeta_1(s_{ij}, a_{ij})}{\zeta_j(s_{ij}, a_{ij})}.$$

Algorithm 4.5 extends Algorithm 4.3 to the IS-based scenario.

Algorithm 4.6 presents the extension of Algorithm 4.1 to the IS-based transfer scenario. The IS-WEIGHTS functions just calculate the importance sampling weights (for each part of the critic and for the actor) based on the transition models of the tasks and on the policies; for simplicity, their implementation is omitted. Note that, to reduce the potential impact of the variance induced by the differences between the densities, the target policy is actually initialized based on the parameters of the source policies.

With respect to the taxonomy presented in Section 2.2, the RL component of the IS actor-critic algorithm is model-based, both policy based and value based, and offline. It might be seen as off-policy in the sense that information from a different policy is being used during the learning of the target one, although such an information is actually sampled from a different task.

For what concerns the taxonomy presented in Section 3.1.1, the TL component of the IS actor-critic algorithm satisfies:

Algorithm 4.6 Actor-critic algorithm in the Importance Sampling scenario

function IS-ACTOR-CRITIC($(T_j)_{j=1}^m, (\pi_j)_{j=2}^m$)
  $\mathcal{D} \leftarrow \emptyset$
  for $j = 2...m$ **do**
    $\mathcal{D}_j \leftarrow$ experience from T_j with π_j
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$
  end for
  $\theta_1 \leftarrow$ initial policy parameter based on $(\pi_j)_{j=2}^m$
  while has not converged **do**
    $\mathcal{D}_1 \leftarrow$ experience from T_1 with π_1
    $w^Q \leftarrow$ IS-WEIGHTS-Q($\mathcal{D}_1 \cup \mathcal{D}, (T_j)_{j=1}^m, (\pi_j)_{j=1}^m$)
    $w^V \leftarrow$ IS-WEIGHTS-V($\mathcal{D}_1 \cup \mathcal{D}, (T_j)_{j=1}^m, (\pi_j)_{j=1}^m$)
    $V_1, Q_1 \leftarrow$ WEIGHTED-LSTD($\mathcal{D}_1 \cup \mathcal{D}, w^Q, w^V$)
    $w^{\nabla J} \leftarrow$ IS-WEIGHTS-GRADIENT($\mathcal{D}_1 \cup \mathcal{D}, (T_j)_{j=1}^m, (\pi_j)_{j=1}^m$)
    $\nabla_\theta J \leftarrow$ WEIGHTED-CALCULATE-GRADIENT($\mathcal{D}_1 \cup \mathcal{D}, \pi_1, V_1, Q_1, w^{\nabla J}$)
    $\theta_1 \leftarrow \theta_1 + \alpha \nabla_\theta J$
  end while
end function

- Task similarity assumptions: defined in the formulation in Section 4.2 in terms of the Lipschitz continuity.
- Source task selection: the agent selects the relevant knowledge not at task level, but at sample level by means of the importance weights assigned to each one of them.
- Type of transferred knowledge: the algorithm transfers low-level knowledge in the form of individual transition samples and policy parameters for the initialization.
- Inter-task mappings: the state and action spaces coincide so that no mappings for these components are necessary. Some information about the mapping for the transition and reward models and for the initial state distribution is given with the Lipschitz conditions of the problem.
- Restrictions of RL algorithms: none, because the transferred knowledge corresponds to individual samples that can be exploited by any algorithm.

Note that the transfer component of the IS actor-critic algorithm corresponds to the calculation of the importance weights, and the RL component corresponds to a weighted version of the standard actor-critic approach. Therefore, the transfer algorithm is directly exploiting the target knowledge to produce the transferred one as it uses the target model for the calculation of the weights.

4.5 TRANSFER WITH AN OPTIMISTIC APPROACH

The IS-based implementation of the actor-critic algorithm presented in Section 4.4 is applicable when the transition models of target and source tasks are known, but this is not the case for the formulation presented in Section 4.2. This section introduces a proposal for solving such issue when the agent has access to some target samples. The idea is to use a weighted estimator to introduce the source samples, and select the weights that minimize the error caused by the transfer process. As usual, the critic and the actor are studied separately.

4.5.1 *The critic*

Consider first the critic for the V-function.
Let $W^V = \{\omega : [1, ..., m] \times \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^*\}$, $\mathbb{R}^* = \mathbb{R} \cup \{0\}$, be the space of all *weightening strategies*, i. e., the space of all functions that assign weights to samples when transferred into the target task. The IS weightening strategy $w_j^V(s, a, s') = d_1(s, a, s')/d_j(s, a, s')$ is a member of W^V.
Under the Lipschitz conditions of the problem in Section 4.2, it is possible to prove that the tuples distribution $d_{(\cdot,\cdot)}(s, a, s')$ is $L_{d-\Theta\mathcal{E}}^{s,a,s'}$-PLC for every $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ (Section C.1). Define, then, the following functions:

$$
\begin{aligned}
L_{d_j}(s, a, s') &= \max(0, d_j(s, a, s') \\
&\qquad - L_{d-\Theta\mathcal{E}}^{s,a,s'}(\theta_j, \epsilon_j) d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_j, \epsilon_j))), \\
U_{d_j}(s, a, s') &= d_j(s, a, s') + L_{d-\Theta\mathcal{E}}^{s,a,s'}(\theta_j, \epsilon_j) d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_j, \epsilon_j)), \\
l_{d_j}(s, a, s') &= \frac{L_{d_j}(s, a, s')}{d_j(s, a, s')} \\
&= \max\left(0, 1 - \frac{L_{d-\Theta\mathcal{E}}^{s,a,s'}(\theta_j, \epsilon_j)}{d_j(s, a, s')} d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_j, \epsilon_j))\right), \\
u_{d_j}(s, a, s') &= \frac{U_{d_j}(s, a, s')}{d_j(s, a, s')} \\
&= 1 + \frac{L_{d-\Theta\mathcal{E}}^{s,a,s'}(\theta_j, \epsilon_j)}{d_j(s, a, s')} d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_j, \epsilon_j)).
\end{aligned}
\tag{4.10}
$$

Thanks to the Lipschitz conditions, for any $j = 1...m$ and $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the target model satisfies:

$$L_{d_j}(s, a, s') \leqslant d_1(s, a, s') \leqslant U_{d_j}(s, a, s');$$

so a weightening strategy $\omega \in W^V$ is said to be *admissible* when, for any $j = 1...m$ and $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, it satisfies:

$$l_{d_j}(s, a, s') \leqslant \omega_j(s, a, s') \leqslant u_{d_j}(s, a, s').$$

Denote by W_1^V the space of admissible weightening strategies. Of course, any $\omega \in W_1^V$ satisfies $\omega_1(s,a,s') = 1$ for any $(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$; in particular, the IS strategy w^V is a member of W_1^V.
Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, s'_{ij}, r_{ij}\rangle)_{j=1\ i=1}^{m\ n_j}$, let

$$\widehat{A_\omega^{(V)}}(\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j} \omega_j(s_{ij}, a_{ij}, s_{ij})\Delta_\phi(s_{ij}, a_{ij}, s'_{ij}),$$

$$\widehat{b_\omega^{(V)}}(\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j} \omega_j(s_{ij}, a_{ij}, s_{ij})\rho_{1,\phi}(s_{ij}, a_{ij}, s'_{ij})$$

be the estimators for the matrices $A_{1,\phi}^{(V)}$, $b_{1,\phi}^{(V)}$ induced by ω given $\mathcal{D}$. For a fixed strategy $\omega \in W_1^V$, define:

$$\begin{aligned} g(\omega) = \ & \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\phi}^{(V)} - \widehat{A_\omega^{(V)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\phi}^{(V)} - \widehat{A_\omega^{(V)}}(\mathcal{D}_1)\|_2^2\right] \\ & + \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\phi}^{(V)} - \widehat{b_\omega^{(V)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\phi}^{(V)} - \widehat{b_\omega^{(V)}}(\mathcal{D}_1)\|_2^2\right] \end{aligned}$$

which represents the change in the expected square ℓ_2-norm loss of the matrices estimation when using transfer with the strategy ω with respect to not using transfer at all. The *optimal strategy* ω^* is the arg-solution of:

$$\underset{\omega \in W_1^V}{\text{minimize}} \quad g(\omega),$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned} \underset{\omega \in W^V}{\text{minimize}} \quad & g(\omega) \\ \text{s. t.} \quad & l_{d_j}(s,a,s') \leqslant \omega_j(s,a,s') \leqslant u_{d_j}(s,a,s'), \\ & \forall j = 1...m, (s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}. \end{aligned}$$

To be solved in practice, g can be expressed as (Section D.2)

$$\begin{aligned} & g(\omega) = \\ & \sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j\big(A_{1,\phi}^{(V)-(k,l)} \\ & \qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\Delta_\phi^{(k,l)}(s,a,s')]\big)\Big)^2 \\ & + \frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\Delta_\phi^{(k,l)}(s,a,s')] \\ & + \Big(\frac{1}{n^2} - \frac{1}{n_1^2}\Big) n_1 \mathop{\mathrm{Var}}_{(s,a,s')\sim d_1}[\Delta_\phi^{(k,l)}(s,a,s')]\Bigg) \\ & + \sum_{k=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j\big(b_{1,\phi}^{(V)-(k)} \\ & \qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\rho_{1,\phi}^{(k)}(s,a,s')]\big)\Big)^2 \end{aligned}$$

$$+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\rho^{(k)}_{1,\Phi}(s,a,s')]$$
$$+\left(\frac{1}{n^2}-\frac{1}{n_1^2}\right)n_1 \operatorname*{Var}_{(s,a,s')\sim d_1}[\rho^{(k)}_{1,\Phi}(s,a,s')]\Big),$$

so that it can be estimated directly from a dataset. That is, the matrices $A^{(V)}_{1,\Phi}$, $b^{(V)}_{1,\Phi}$ are estimated from the available target samples, and the terms with $\mathrm{Var}_{(s,a,s')\sim d_1}$ are dropped as they do not depend on ω.

Consider now the critic for the Q-function.
Let $W^Q = \{\omega : [1,...,m] \times \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A} \to \mathbb{R}^*\}$ be the space of all *weightening strategies*, i. e., the space of all functions that assign weights to samples when transferred into the target task. The IS weightening strategy $w^Q_j(s,a,s',a') = \overline{d}_1(s,a,s',a')/\overline{d}_j(s,a,s',a')$ is a member of W^Q.
Under the Lipschitz conditions of the problem in Section 4.2, it is possible to prove that the extended tuples distribution $\overline{d}_{(\cdot,\cdot)}(s,a,s',a')$ is $L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}$-PLC for every $(s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}$ (Section C.1). Define, then, the following functions:

$$\begin{aligned}
L_{\overline{d}_j}(s,a,s',a') &= \max(0,\overline{d}_j(s,a,s',a') \\
&\qquad - L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta_j,\epsilon_j)\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j))),\\
U_{\overline{d}_j}(s,a,s',a') &= \overline{d}_j(s,a,s',a') \\
&\qquad + L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta_j,\epsilon_j)\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)),\\
l_{\overline{d}_j}(s,a,s',a') &= \frac{L_{\overline{d}_j}(s,a,s',a')}{\overline{d}_j(s,a,s',a')} \\
&= \max\left(0, 1-\frac{L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta_j,\epsilon_j)}{\overline{d}_j(s,a,s',a')}\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j))\right),\\
u_{\overline{d}_j}(s,a,s',a') &= \frac{U_{\overline{d}_j}(s,a,s',a')}{\overline{d}_j(s,a,s',a')} \\
&= 1+\frac{L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta_j,\epsilon_j)}{\overline{d}_j(s,a,s',a')}\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)).
\end{aligned} \tag{4.11}$$

Thanks to the Lipschitz conditions, for any $j = 1...m$ and $(s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}$, the target model satisfies:

$$L_{\overline{d}_j}(s,a,s',a') \leqslant \overline{d}_1(s,a,s',a') \leqslant U_{\overline{d}_j}(s,a,s',a');$$

so a weightening strategy $\omega \in W^Q$ is said to be *admissible* when, for any $j = 1...m$ and $(s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}$, it satisfies:

$$l_{\overline{d}_j}(s,a,s',a') \leqslant \omega_j(s,a,s',a') \leqslant u_{\overline{d}_j}(s,a,s',a').$$

Denote by W_1^Q the space of admissible weightening strategies. Of course, any $\omega \in W_1^Q$ satisfies $\omega_1(s,a,s',a') = 1$ for any $(s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}$; in particular, the IS strategy w^Q is a member of W_1^Q. Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, s'_{ij}, a'_{ij}, r_{ij}\rangle)_{j=1\ i=1}^{m\ n_j}$, let

$$\widehat{A_\omega^{(Q)}}(\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j}\omega_j(s_{ij},a_{ij},s_{ij},a'_{ij})\Delta_{\boldsymbol{\varphi}}(s_{ij},a_{ij},s'_{ij},a'_{ij}),$$

$$\widehat{b_\omega^{(Q)}}(\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j}\omega_j(s_{ij},a_{ij},s_{ij},a'_{ij})\rho_{1,\boldsymbol{\varphi}}(s_{ij},a_{ij},s'_{ij},a'_{ij})$$

be the estimators for the matrices $A_{1,\boldsymbol{\varphi}}^{(Q)}$, $b_{1,\boldsymbol{\varphi}}^{(Q)}$ induced by ω given $\mathcal{D}$. For a fixed strategy $\omega \in W_1^Q$, define:

$$\begin{aligned} g(\omega) = \ & \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\boldsymbol{\varphi}}^{(Q)} - \widehat{A_\omega^{(Q)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\boldsymbol{\varphi}}^{(Q)} - \widehat{A_\omega^{(Q)}}(\mathcal{D}_1)\|_2^2\right] \\ & + \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\boldsymbol{\varphi}}^{(Q)} - \widehat{b_\omega^{(Q)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\boldsymbol{\varphi}}^{(Q)} - \widehat{b_\omega^{(Q)}}(\mathcal{D}_1)\|_2^2\right] \end{aligned}$$

which represents the change in the expected square ℓ_2-norm loss of the matrices estimation when using transfer with the strategy ω with respect to not using transfer at all. The *optimal strategy* ω^* is the arg-solution of:

$$\underset{\omega\in W_1^Q}{\text{minimize}} \quad g(\omega),$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned} \underset{\omega\in W^Q}{\text{minimize}} \quad & g(\omega) \\ \text{s. t.} \quad & l_{\overline{d}_j}(s,a,s',a') \leqslant \omega_j(s,a,s',a') \leqslant u_{\overline{d}_j}(s,a,s',a'), \\ & \forall j = 1...m, (s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}. \end{aligned}$$

To be solved in practice, g can be expressed as (Section D.2)

$$\begin{aligned} & g(\omega) = \\ & \sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j (A_{1,\boldsymbol{\varphi}}^{(Q)-(k,l)} \\ & \qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta_{\boldsymbol{\varphi}}^{(k,l)}(s,a,s',a')]\Big)^2 \\ & + \frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta_{\boldsymbol{\varphi}}^{(k,l)}(s,a,s',a')] \\ & + \Big(\frac{1}{n^2} - \frac{1}{n_1^2}\Big) n_1 \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_1}[\Delta_{\boldsymbol{\varphi}}^{(k,l)}(s,a,s',a')]\Bigg) \end{aligned}$$

$$+\sum_{k=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j\big(b_{1,\varphi}^{(Q)-(k)} - \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho_{1,\Phi}^{(k)}(s,a,s',a')]\big)\Big)^2$$
$$+\frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho_{1,\Phi}^{(k)}(s,a,s',a')]$$
$$+\Big(\frac{1}{n^2}-\frac{1}{n_1^2}\Big)n_1 \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_1}[\rho_{1,\Phi}^{(k)}(s,a,s',a')]\Bigg),$$

so that it can be estimated directly from a dataset. That is, the matrices $A_{1,\varphi'}^{(V)}$ $b_{1,\varphi}^{(V)}$ are estimated from the available target samples, and the terms with $\mathrm{Var}_{(s,a,s',a')\sim\overline{d}_1}$ are dropped as they do not depend on ω.

No pseudo-code for the optimistic critic is provided as it would mainly consist on some minimization procedure for the above function, and plugging the resulting optimal strategy in Algorithm 4.4.

4.5.2 *The actor*

Let $W^{\nabla J} = \{\omega : [1,...,m] \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^*\}$ be the space of all *weightening strategies*, i. e., the space of all functions that assign weights to samples when transferred into the target task. The IS weightening strategy $w_j^{\nabla J}(s,a) = \zeta_1(s,a)/\zeta_j(s,a)$ is a member of $W^{\nabla J}$.
Under the Lipschitz conditions of the problem in Section 4.2, it is possible to prove that the state-action distribution $\zeta_{(\cdot,\cdot)}(s,a)$ is $L^{s,a}_{\zeta-\Theta\mathcal{E}}$-PLC for every $(s,a) \in \mathcal{S}\times\mathcal{A}$ (Section C.1). Define, then, the following functions:

$$\begin{aligned}
L_{\zeta_j}(s,a) &= \max(0,\zeta_j(s,a) - L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta_j,\epsilon_j)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j))),\\
U_{\zeta_j}(s,a) &= \zeta_j(s,a) + L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta_j,\epsilon_j)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)),\\
l_{\zeta_j}(s,a) &= \frac{L_{\zeta_j}(s,a)}{\zeta_j(s,a)}\\
&= \max\Big(0, 1-\frac{L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta_j,\epsilon_j)}{\zeta_j(s,a)}d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j))\Big),\\
u_{\zeta_j}(s,a) &= \frac{U_{\zeta_j}(s,a)}{\zeta_j(s,a)}\\
&= 1+\frac{L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta_j,\epsilon_j)}{\zeta_j(s,a)}d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)).
\end{aligned} \tag{4.12}$$

Thanks to the Lipschitz conditions, for any $j = 1...m$ and $(s,a) \in \mathcal{S}\times\mathcal{A}$, the target model satisfies:

$$L_{\zeta_j}(s,a) \leqslant \zeta_1(s,a) \leqslant U_{\zeta_j}(s,a);$$

so a weightening strategy $\omega \in W^{\nabla J}$ is said to be *admissible* when, for any $j = 1...m$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$, it satisfies:

$$l_{\zeta_j}(s, a) \leqslant \omega_j(s, a) \leqslant u_{\zeta_j}(s, a).$$

Denote by $W_1^{\nabla J}$ the space of admissible weightening strategies. Of course, any $\omega \in W_1^{\nabla J}$ satisfies $\omega_1(s, a) = 1$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$; in particular, the IS strategy $w^{\nabla J}$ is a member of $W_1^{\nabla J}$.
Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, \widehat{\eta_1}(s_{ij}, a_{ij})\rangle)_{j=1\ i=1}^{m\ n_j}$, let

$$\widehat{\nabla_\theta J_\omega}(\mathcal{D}) = \frac{1}{n(1-\gamma)} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \omega_j(s_{ij}, a_{ij}) \widehat{\eta_1}(s_{ij}, a_{ij})$$

be the estimator for the gradient $\nabla_\theta J_1$ induced by ω given $\mathcal{D}$.
For a fixed strategy $\omega \in W_1^{\nabla J}$, define:

$$g(\omega) = \mathop{\mathbb{E}}_{\mathcal{D}} \left[\|\nabla_\theta J_1 - \widehat{\nabla_\theta J_\omega}(\mathcal{D})\|_2^2 \right] - \mathop{\mathbb{E}}_{\mathcal{D}} \left[\|\nabla_\theta J_1 - \widehat{\nabla_\theta J_\omega}(\mathcal{D}_1)\|_2^2 \right]$$

which represents the change in the expected square ℓ_2-norm loss of the gradient estimation when using transfer with the strategy ω with respect to not using transfer at all. The *optimal strategy* ω^* is the arg-solution of:

$$\begin{aligned} \underset{\omega \in W_1^{\nabla J}}{\text{minimize}} \quad & g(\omega), \end{aligned}$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned} \underset{\omega \in W^{\nabla J}}{\text{minimize}} \quad & g(\omega) \\ \text{s. t.} \quad & l_{\zeta_j}(s, a) \leqslant \omega_j(s, a) \leqslant u_{\zeta_j}(s, a), \\ & \forall j = 1...m, (s, a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

To be solved in practice, g can be expressed as (Section D.2)

$$\begin{aligned} & g(\omega) = \\ & \sum_{k=1}^{d_a} \Bigg(\Big(\frac{1}{n} \sum_{j=2}^{m} n_j (\nabla_{\theta_k} J_1 \\ & \qquad - \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} [\omega_j(s, a) \widehat{\eta_1^{(k)}}(s, a)]) \Big)^2 \\ & + \frac{1}{n^2(1-\gamma)^2} \sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j} [\omega_j(s, a) \widehat{\eta_1}^{(k)}(s, a)] \\ & + \Big(\frac{1}{n^2} - \frac{1}{n_1^2} \Big) \frac{n_1}{(1-\gamma)^2} \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_1} [\widehat{\eta_1}^{(k)}(s, a)] \Bigg) \end{aligned}$$

so that it can be estimated directly from a dataset. That is, the gradient $\nabla_{\theta_k} J$ is estimated from the available target samples, and the

term with $\mathrm{Var}_{(s,a)\sim\zeta_1}$ are dropped as they do not depend on ω.

No pseudo-code for the optimistic actor is provided as it would mainly consist on some minimization procedure for the above function, and plugging the resulting optimal strategy in Algorithm 4.5.
The optimal strategies of the critics and the actor are used within the optimistic actor-critic algorithm for defining the weighted estimators, just like in the IS formulation. Thus, the only difference with respect to Algorithm 4.6 would concern the calculation of the weights; that is, the weights given to WEIGHTED-LSTD and WEIGHTED-CALCULATE-GRADIENT are no longer calculated with the IS-WEIGHTS procedures but by solving the corresponding minimization problems. For this reason, no pseudo-code is provided for the optimistic actor-critic transfer algorithm as it would mostly be the same as the IS-based actor-critic algorithm.

With respect to the taxonomy presented in Section 2.2, the RL component of the optimistic actor-critic algorithm is model-free, both policy based and value based, and offline. It might be seen as off-policy in the sense that information from a different policy is being used during the learning of the target one, although such an information is actually sampled from a different task.
For what concerns the taxonomy presented in Section 3.1.1, the TL component of the optimistic actor-critic algorithm satisfies:

- Task similarity assumptions: defined in the formulation in Section 4.2 in terms of the Lipschitz continuity.
- Source task selection: the agent selects the relevant knowledge not at task level, but at sample level by means of the importance weights assigned to each one of them.
- Type of transferred knowledge: the algorithm transfers low-level knowledge in the form of individual transition samples and policy parameters for the initialization.
- Inter-task mappings: the state and action spaces coincide so that no mappings for these components are necessary. Some information about the mapping for the transition and reward models and for the initial state distribution is given with the Lipschitz conditions of the problem.
- Restrictions of RL algorithms: none, because the transferred knowledge corresponds to individual samples that can be exploited by any algorithm.

Note that the transfer component of the optimistic actor-critic algorithm corresponds to the calculation of the optimal strategy, and the RL component corresponds to a weighted version of the standard

actor-critic approach. Therefore, the transfer algorithm is directly exploiting the target knowledge to produce the transferred one as the estimates of the matrices and the gradient, obtained with the target samples for, are plugged into the objective functions.

4.6 TRANSFER WITH A PESSIMISTIC APPROACH

The optimistic approach presented in Section 4.5 for solving the problem formulated in Section 4.2 is applicable whenever there are at least some target samples to obtain the initial estimates. This section introduces a proposal for the scenario in which no target experience is available. The idea is to use a weighted estimator to introduce the source samples, and select the weights through an adversarial-like approach, i. e., select the weights that work best in the worst case scenario. As usual, the critic and the actor are studied separately.

4.6.1 *The critic*

Consider first the critic for the V-function.
Consider W^V, the space of all weightening strategies, and W_1^V, the space of admissible weightening strategies, as introduced in Section 4.5.1. Recalling the functions defined in Equation (4.10), define D_1 as the space of *admissible target tuple distributions* by

$$D_1 = \{d \in \Delta(\mathcal{SAS}) \mid L_{d_j}(s,a,s') \leqslant d(s,a,s') \leqslant U_{d_j}(s,a,s')\},$$

where $\Delta(\mathcal{SAS})$ is the space of probability distributions over $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$.
Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, s'_{ij}, r_{ij}\rangle)_{j=1\ i=1}^{m\ n_j}$, recall from Section 4.5.1 the estimators $\widehat{A_\omega^{(V)}}(\mathcal{D})$ and $\widehat{b_\omega^{(V)}}(\mathcal{D})$ for the matrices $A_{1,\Phi}^{(V)}$ and $b_{1,\Phi}^{(V)}$, respectively.
Under the Lipschitz conditions in Section 4.2, it is possible to prove that $A_{(\cdot),\Phi}^{(V)-(k,l)}$ is $L_{A-\Theta\mathcal{E}}^{k,l}$-PLC, and that $b_{(\cdot),\Phi}^{(V)-(k)}$ is $L_{b-\Theta\mathcal{E}}^{k}$-PLC (Section C.2). Define $A_{1,\Phi}^{(V)}(d)$ and $b_{1,\Phi}^{(V)}(d)$ as the corresponding matrices if d was the target tuples distribution.
For a fixed strategy $\omega \in W_1^V$, define:

$$\begin{aligned} g_\omega(d) = \ & \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\Phi}^{(V)}(d) - \widehat{A_\omega^{(V)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\Phi}^{(V)}(d) - \widehat{A_\omega^{(V)}}(\mathcal{D}_1)\|_2^2\right] \\ & + \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\Phi}^{(V)}(d) - \widehat{b_\omega^{(V)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\Phi}^{(V)}(d) - \widehat{b_\omega^{(V)}}(\mathcal{D}_1)\|_2^2\right], \end{aligned}$$

where the expectations are taken over the datasets obtained from the target task with d as tuple distribution, and the source tasks. The function g represents the change in the expected square ℓ_2-norm loss of the matrices estimation when using transfer with the strategy ω

with respect to not using transfer at all, if d corresponded to the target task. The *optimal strategy* ω^* is the arg-solution of:

$$\underset{\omega \in W_1^{(V)}}{\text{minimize}} \quad \underset{d \in D_1}{\text{maximize}}\ g_\omega(d),$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned}
\underset{\omega \in W^{(V)}}{\text{minimize}} \quad & \underset{d \in \Delta(\mathcal{SAS})}{\text{maximize}}\ g_\omega(d) \\
\text{s. t.} \quad & L_{d_j}(s,a,s') \leqslant d(s,a,s') \leqslant U_{d_j}(s,a,s'), \\
& l_{d_j}(s,a,s') \leqslant \omega_j(s,a,s') \leqslant u_{d_j}(s,a,s'), \\
& \forall j = 1...m, (s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}.
\end{aligned}$$

To be solved in practice, g can be upper bounded by the function g_1 (Section D.2)

$$\begin{aligned}
& g_\omega(\overline{d}) \leqslant g_1(\omega) = \\
& \sum_{k=1}^{d_c} \sum_{l=1}^{d_c} \Bigg(\Bigg(\frac{1}{n} \sum_{j=2}^{m} n_j L^{k,l}_{A-\Theta\mathcal{E}}(\theta_j, \epsilon_j) d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)) \\
& \qquad + \Big| \frac{1}{n} \sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s')\sim d_j} [(1-\omega_j(s,a,s'))\Delta^{(k,l)}_{\Phi}(s,a,s')] \Big| \Bigg)^2 \\
& + \frac{1}{n^2} \sum_{j=2}^{m} n_j \mathop{\text{Var}}_{(s,a,s')\sim d_j} [\omega_j(s,a,s')\Delta^{(k,l)}_{\Phi}(s,a,s')] \Bigg) \\
& + \sum_{k=1}^{d_c} \Bigg(\Bigg(\frac{1}{n} \sum_{j=2}^{m} n_j L^{k}_{b-\Theta\mathcal{E}\Xi}(\theta_j, \epsilon_j, \xi_j) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j)) \\
& \qquad + \Big| \frac{1}{n} \sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s')\sim d_j} [\rho^{(k)}_{j,\Phi}(s,a,s') - \\
& \qquad\qquad \omega_j(s,a,s')\rho^{(k)}_{1,\Phi}(s,a,s')] \Big| \Bigg)^2 \\
& + \frac{1}{n^2} \sum_{j=2}^{m} n_j \mathop{\text{Var}}_{(s,a,s')\sim d_j} [\omega_j(s,a,s')\rho^{(k)}_{1,\Phi}(s,a,s')] \Bigg),
\end{aligned}$$

so that it can be estimated directly from a dataset and the problem reduces to a minimization.

Consider now the critic for the Q-function.
Consider W^Q, the space of all weightening strategies, and W_1^Q, the space of admissible weightening strategies, as introduced in Section 4.5.1. Recalling the functions defined in Equation (4.10), define $\overline{D}_1$ as the space of *admissible target extended tuple distributions* by

$$\overline{D}_1 = \{\overline{d} \in \Delta(\mathcal{SASA}) \mid L_{\overline{d}_j}(s,a,s',a') \leqslant \overline{d}(s,a,s',a') \leqslant U_{\overline{d}_j}(s,a,s',a')\},$$

where $\Delta(\mathcal{SASA})$ is the space of probability distributions over $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A}$.

Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^{m} = (\langle s_{ij}, a_{ij}, s'_{ij}, a'_{ij}, r_{ij}\rangle)_{j=1}^{m}{}_{i=1}^{n_j}$, recall from Section 4.5.1 the estimators $\widehat{A_{\omega}^{(Q)}}(\mathcal{D})$ and $\widehat{b_{\omega}^{(Q)}}(\mathcal{D})$ for the matrices $A_{1,\boldsymbol{\varphi}}^{(Q)}$ and $b_{1,\boldsymbol{\varphi}}^{(Q)}$, respectively.
Under the Lipschitz conditions in Section 4.2, it is possible to prove that $A_{(\cdot),\boldsymbol{\varphi}}^{(Q)-(k,l)}$ is $L_{A-\Theta\mathcal{E}}^{k,l}$-PLC, and that $b_{(\cdot),\boldsymbol{\varphi}}^{(Q)-(k)}$ is $L_{b-\Theta\mathcal{E}}^{k}$-PLC (Section C.2). Define $A_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d})$ and $b_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d})$ as the corresponding matrices if $\overline{d}$ was the target extended tuples distribution.
For a fixed strategy $\omega \in W_1^Q$, define:

$$\begin{aligned} g_{\omega}(\overline{d}) = \; & \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d}) - \widehat{A_{\omega}^{(Q)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|A_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d}) - \widehat{A_{\omega}^{(Q)}}(\mathcal{D}_1)\|_2^2\right] \\ & + \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d}) - \widehat{b_{\omega}^{(Q)}}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|b_{1,\boldsymbol{\varphi}}^{(Q)}(\overline{d}) - \widehat{b_{\omega}^{(Q)}}(\mathcal{D}_1)\|_2^2\right], \end{aligned}$$

where the expectations are taken over the datasets obtained from the target task with $\overline{d}$ as extended tuple distribution, and the source tasks. The function g represents the change in the expected square ℓ_2-norm loss of the matrices estimation when using transfer with the strategy ω with respect to not using transfer at all, if $\overline{d}$ corresponded to the target task. The *optimal strategy* ω^* is the arg-solution of:

$$\underset{\omega \in W_1^{(Q)}}{\text{minimize}} \quad \underset{\overline{d} \in \overline{D}}{\text{maximize}}\; g_{\omega}(\overline{d}),$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned} \underset{\omega \in W^{(Q)}}{\text{minimize}} \quad & \underset{\overline{d} \in \Delta(\mathcal{S}\mathcal{A}\mathcal{S})}{\text{maximize}}\; g_{\omega}(\overline{d}) \\ \text{s. t.} \quad & L_{\overline{d}_j}(s,a,s',a') \leqslant \overline{d}(s,a,s',a') \leqslant U_{\overline{d}_j}(s,a,s',a'), \\ & l_{d_j}(s,a,s',a') \leqslant \omega_j(s,a,s',a') \leqslant u_{d_j}(s,a,s',a'), \\ & \forall j = 1...m, (s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}. \end{aligned}$$

To be solved in practice, g can be upper bounded by the function g_1 (Section D.2)

$$\begin{aligned} & g_{\omega}(\overline{d}) \leqslant g_1(\omega) = \\ & \sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Bigg(\Bigg(\frac{1}{n}\sum_{j=2}^{m} n_j L_{A-\Theta\mathcal{E}}^{k,l}(\theta_j,\epsilon_j)\, d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)) \\ & \qquad + \left|\frac{1}{n}\sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[(1-\omega_j(s,a,s',a'))\Delta_{\boldsymbol{\varphi}}^{(k,l)}(s,a,s',a')]\right|\Bigg)^2 \\ & + \frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta_{\boldsymbol{\varphi}}^{(k,l)}(s,a,s',a')]\Bigg) \end{aligned}$$

$$+\sum_{k=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j L^k_{b-\Theta\mathcal{E}\Xi}(\theta_j,\epsilon_j,\xi_j)\, d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j))$$
$$+\Big|\frac{1}{n}\sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\rho^{(k)}_{j,\varphi}(s,a,s',a')-$$
$$\omega_j(s,a,s',a')\rho^{(k)}_{1,\varphi}(s,a,s',a')]\Big|\Big)^2$$
$$+\frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho^{(k)}_{1,\varphi}(s,a,s',a')]\Bigg),$$

so that it can be estimated directly from a dataset and the problem reduces to a minimization.

No pseudo-code for the pessimistic critic is provided as it would mainly consist on some minimization procedure for the above function, and plugging the resulting optimal strategy in Algorithm 4.4.

4.6.2 *The actor*

Consider $W^{\nabla J}$, the space of all weightening strategies, and $W_1^{\nabla J}$, the space of admissible weightening strategies, as introduced in Section 4.5.2. Recalling the functions defined in Equation (4.12), define Z_1 as the space of *admissible target state-action distributions* by

$$Z_1 = \{\zeta \in \Delta(\mathcal{SA}) \mid L_{\zeta_j}(s,a) \leqslant \zeta(s,a) \leqslant U_{\zeta_j}(s,a)\},$$

where $\Delta(\mathcal{SA})$ is the space of probability distributions over $\mathcal{S}\times\mathcal{A}$.
Given a strategy ω and a dataset $\mathcal{D} = (\mathcal{D}_j)_{j=1}^m = (\langle s_{ij}, a_{ij}, \widehat{\eta_1}(s_{ij},a_{ij})\rangle)_{j=1\ i=1}^{m\ n_j}$, recall from Section 4.5.2 the estimator $\widehat{\nabla_\theta J_\omega}(\mathcal{D})$ for the gradient $\nabla_\theta J_1$.
Under the Lipschitz conditions in Section 4.2, it is possible to prove that $\nabla_{\theta_k} J_{(\cdot)}$ is $L^k_{\nabla J-\Theta\mathcal{E}}$-PLC. Define $\nabla_\theta J_1(\zeta)$ the corresponding gradient if ζ was the target state-action distribution.
For a fixed strategy $\omega \in W_1^{\nabla J}$, define:

$$g_\omega(\zeta) = \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\nabla_\theta J_1(\zeta) - \widehat{\nabla_\theta J_\omega}(\mathcal{D})\|_2^2\right] - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\nabla_\theta J_1(\zeta) - \widehat{\nabla_\theta J_\omega}(\mathcal{D}_1)\|_2^2\right],$$

where the expectations are taken over the datasets obtained from the target task with ζ as state-action distribution, and the source tasks. The function g represents the change in the expected square ℓ_2-norm loss of the gradient estimation when using transfer with the strategy ω with respect to not using transfer at all, if ζ corresponded to the target task. The *optimal strategy* ω^* is the arg-solution of:

$$\mathop{\text{minimize}}_{\omega\in W_1^{(\nabla J)}} \quad \mathop{\text{maximize}}_{\zeta\in Z_1} g_\omega(\zeta),$$

which can be formulated as a constrained optimization problem:

$$\begin{aligned}
&\underset{\omega \in W^{(\nabla J)}}{\text{minimize}} \quad \underset{\zeta \in \Delta(\mathcal{S}\mathcal{A}\mathcal{S})}{\text{maximize}}\ g_\omega(\zeta)\\
&\text{s. t.} \qquad L_{\zeta_j}(s,a) \leqslant \zeta(s,a) \leqslant U_{\zeta_j}(s,a),\\
&\qquad\qquad l_{d_j}(s,a) \leqslant \omega_j(s,a) \leqslant u_{d_j}(s,a),\\
&\qquad\qquad \forall j = 1...m, (s,a) \in \mathcal{S} \times \mathcal{A}.
\end{aligned}$$

To be solved in practice, g can be upper bounded by the function g_1 (Section D.2)

$$\begin{aligned}
&g_\omega(\overline{d}) \leqslant g_1(\omega) =\\
&\sum_{k=1}^{d_a} \Bigg(\Big(\frac{1}{n} \sum_{j=2}^{m} n_j L^k_{\nabla J - \Theta\mathcal{E}\Xi}(\theta_j, \epsilon_j, \xi_j) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j))\\
&\qquad + \Big| \frac{1}{n(1-\gamma)} \sum_{j=2}^{m} n_j \underset{(s,a)\sim\zeta_j}{\mathbb{E}} [\widehat{\eta_j^{(k)}}(s,a) - \omega_j(s,a)\widehat{\eta_1^{(k)}}(s,a)] \Big| \Big)^2\\
&+ \frac{1}{n^2(1-\gamma)^2} \sum_{j=2}^{m} n_j \underset{(s,a)\sim\zeta_j}{\text{Var}} [\omega_j(s,a)\widehat{\eta_1^{(k)}}(s,a)] \Bigg),
\end{aligned}$$

so that it can be estimated directly from a dataset and the problem reduces to a minimization.

No pseudo-code for the pessimistic actor is provided as it would mainly consist on some minimization procedure for the above function, and plugging the resulting optimal strategy in Algorithm 4.5.
The optimal strategies of the critics and the actor are used within the pessimistic actor-critic algorithm for defining the weighted estimators, just like in the IS formulation. Thus, the only difference with respect to Algorithm 4.6 would concern the calculation of the weights; that is, the weights given to WEIGHTED-LSTD and WEIGHTED-CALCULATE-GRADIENT are no longer calculated with the IS-WEIGHTS procedures but by solving the corresponding minimization problems. For this reason, no pseudo-code is provided for the pessimistic actor-critic transfer algorithm as it would mostly be the same as the IS-based actor-critic algorithm.

With respect to the taxonomy presented in Section 2.2, the RL component of the pessimistic actor-critic algorithm is model-free, both policy based and value based, and offline. It might be seen as off-policy in the sense that information from a different policy is being used during the learning of the target one, although such an information is actually sampled from a different task.
For what concerns the taxonomy presented in Section 3.1.1, the TL component of the pessimistic actor-critic algorithm satisfies:

– Task similarity assumptions: defined in the formulation in Section 4.2 in terms of the Lipschitz continuity.

- Source task selection: the agent selects the relevant knowledge not at task level, but at sample level by means of the importance weights assigned to each one of them.
- Type of transferred knowledge: the algorithm transfers low-level knowledge in the form of individual transition samples and policy parameters for the initialization.
- Inter-task mappings: the state and action spaces coincide so that no mappings for these components are necessary. Some information about the mapping for the transition and reward models and for the initial state distribution is given with the Lipschitz conditions of the problem.
- Restrictions of RL algorithms: none, because the transferred knowledge corresponds to individual samples that can be exploited by any algorithm.

Note that the transfer component of the pessimistic actor-critic algorithm corresponds to the calculation of the optimal strategy, and the RL component corresponds to a weighted version of the standard actor-critic approach. Therefore, the transfer algorithm is directly exploiting the target knowledge to produce the transferred one as the samples an be used for estimates into the objective functions.

5

EXPERIMENTS

This chapter presents the experimental environment the algorithms introduced in Chapter 4 are tested in, as well as their results and corresponding analysis. Section 5.1 describes with details the general task environment that the agent is facing, and Section 5.2 talks about the specific task instances and the experiments. Finally, Section 5.3 presents the obtained results and gives an analysis of such outcomes.

5.1 TASK ENVIRONMENT: MOUNTAIN CAR

The task environment selected to develop the experiments is *Continuous Mountain Car* [1].
The agent is driving a car at a valley in the middle of two mountains, and its goal is to reach the top of one of them as soon as possible while using the least amount of energy. The car can apply force forward or backward to gain or lose speed, but does not have enough power to reach the top on single shot; thus, the agent has to learn how to use the forces to complete the task. Figure 5.1 illustrates this setting.
The state space is the set $\mathcal{S} = [-10, 10] \times [-0.78, 0.78]$, where the first component accounts for the horizontal position of the car and the second one for the speed; the goal position is at 8.34. The action space is the interval $\mathcal{A} = [-1, 1]$, and it represents the force applied to the car (positive means forward, negative means backward). The transition model parameter space is the interval $\mathcal{E} = [0, 0.1]$, which represents the power of the car; it does not affect the initial state distribution though, only the transition dynamics. The reward model is the same for all the tasks in the environment.
Denote by ϵ the power of the car, by p the horizontal position of the car, by v the velocity of the car, and by a the action applied by the agent. The transition model is given by

$$\mathcal{P}(\cdot|(p, v), a) \sim \mathcal{N}(s_\epsilon(p, v, a), \Sigma^2),$$

where

$$s_\epsilon(p, v, a) = \begin{pmatrix} p + v + a\epsilon - 0.028\cos(0.27p + 0.9) \\ v + a\epsilon - 0.028\cos(0.27p + 0.9), \end{pmatrix}$$

and

$$\Sigma^2 = \text{diag}(\sigma_p^2 = 0.5^2, \sigma_v^2 = 0.04^2)$$

1 Sutton and Barto (1998) introduce this example in a discrete version; the one presented in this chapter is a modified one for the continuous case

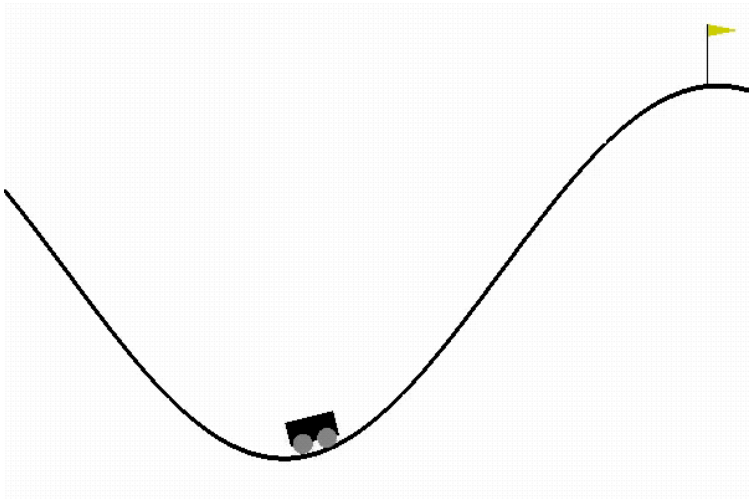


Figure 5.1: The car is supposed to gain speed to reach the goal position, marked with a flag (taken from the OpenAI [2] implementation).

Transitions out of the bounds of the state space are taken as transitions to the closest edge. The velocity is zeroed when a transition results in the position on the left boundary with a negative velocity. The initial state distribution defines the initial position of the car:

$$\mu = U(-2.5, -1.95);$$

it starts always with zero speed. The reward model corresponds to

$$r((p,v),a,(p',v')) = \begin{cases} -0.1 \cdot 2^{4|a|}, & p < 8.34 \text{ and } p' < 8.34 \\ -0.1 \cdot 2^{4|a|} + 10, & p < 8.34 \text{ and } p' \geqslant 8.34 \\ 0, & p >= 8.34. \end{cases}$$

The policy parameter space is the set $\Theta = [0,1] \times [0,1]$. For a parameter $\theta \in \Theta$, the associated policy is defined by

$$\pi_\theta(\cdot|(p,v)) \sim \mathcal{N}(s_\theta(p,v), \sigma_a^2 = 0.4^2),$$

where

$$s_\theta(p,v) = \big(\mathbb{1}(v \geqslant 0)\theta_1 + \mathbb{1}(v < 0)\theta_2\big)v^*,$$

being $v* = v/0.78$ the signed normalized speed. Actions out of the bounds of the action space are taken as the closest edge.

5.2 EXPERIMENTAL INSTANCES

To satisfy the assumption of completely knowing the source models, the tasks from the environment presented in Section 5.1 are discretized.

Each component of the state space is evenly divided into 20 intervals so that there are 20×20 blocks, each one represented by its midpoint. For the transition model, the probability of each discrete state is taken as the density covered by its associated block in the continuous distribution.
Likewise, the action space is evenly divided into 10 intervals, each one represented by its midpoint. The model of the policy is discretized in a similar way to the transition model.
The task and policy parameters are kept as continuous spaces though.

The target task corresponds to the parameter $\epsilon_1 = 0.0222$, and the two source tasks to the parameters $\epsilon_2 = 0.0278$ and $\epsilon_3 = 0.0778$.
The formulation of the problem (Section 4.2) says nothing about the source policies used to obtain the source samples. However, the Lipschitz assumptions allow to show that a local maximum for the target task lies in a neighborhood of the source optimal parameter (Section D.3); thus, it is reasonable to think that transferring from the optimal source policies would give good results in practice.
With this configuration and the above observation in mind, the selected experiments are listed in Table 5.1. The table is composed of the following columns:

- *Name*, a shorthand identifier for the experiment.
- *Used algorithm*, the algorithm used during the experiment. SAC stands for Standard Actor-Critic; it refers standard learning without any transfer. ISAC stands for Importance Sampling Actor Critic; it refers to the IS-based actor-critic algorithm presented in Section 4.4. MinAC stands for Minimization Actor-Critic; it refers to the optimistic approach introduced in Section 4.5. MinMaxAC stands for Min-Max Actor-Critic; it refers to the pessimistic approach proposed in Section 4.6.
- *Source tasks*, the source tasks used for transfer. T_2 refers to the source task with parameter $\epsilon_2 = 0.0278$, and T_3 refers to the source task with parameter $\epsilon_3 = 0.0778$.
- *Source policy*, the nature of the source policy used to get the source samples. *o* means the optimal source policy is used, and *r* means that the policy with the highest distance from the optimal target one is used (for the purpose of checking robustness against negative transfer). When there is only one source task, the initial target policy is made to coincide with the source policy.

For the transfer scenarios, the source dataset consists of 25.000 samples.

NAME	USED ALGORITHM	SOURCE TASKS	SOURCE POLICY
NoTransfer	SAC	N. A.	N. A.
IS-2-os	ISAC	T_2	o
IS-3-os	ISAC	T_3	o
IS-2-rs	ISAC	T_2	r
IS-3-rs	ISAC	T_3	r
Min-2-os	MinAC	T_2	o
Min-3-os	MinAC	T_3	o
Min-2-rs	MinAC	T_2	r
Min-3-rs	MinAC	T_3	r
MinMax-2-os	MinMaxAC	T_2	o
MinMax-3-os	MinMaxAC	T_3	o
MinMax-2-rs	MinMaxAC	T_2	r
MinMax-3-rs	MinMaxAC	T_3	r

Table 5.1: The list of performed experiments, along with their name and configuration.

5.3 ANLYSIS OF THE RESULTS

Figure 5.2, Figure 5.3, Figure 5.4, and Figure 5.5 present the learning curves of all the experiments. Each graph shows how the performance of the obtained policy changes with respect to the size of the dataset available for executing the actor-critic steps. The learning curve from NoTransfer is replicated in the other graphs so that the impact of transfer is easier to perceive.

Figure 5.2 shows the learning curve for the NoTransfer experiment. This is used as the baseline for assessing the impact of the different transfer algorithms. Note that optimal policies are obtained with around 3.000 samples, and that the variance is consequently decreased after this point.

Figure 5.3 shows the learning curves for the IS actor-critic algorithm in the different configurations.
Figure 5.3a corresponds to transfer from the most similar task using its optimal policy. The jumpstart, the asymptotic performance and the variance of the learning process show all a positive impact due to transfer.
Figure 5.3b corresponds to transfer from the most different task using its optimal policy. The jumpstart and asymptotic performance reflect the benefit of transfer, although the variance shows no significant im-

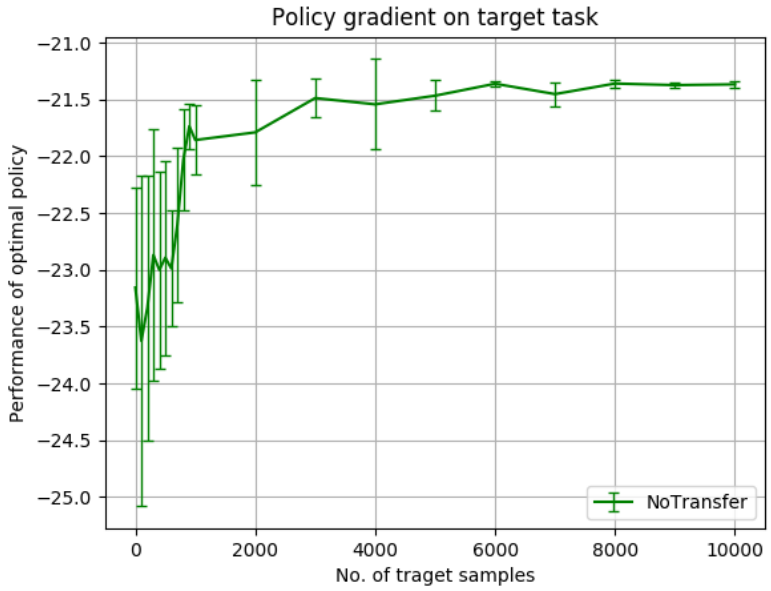
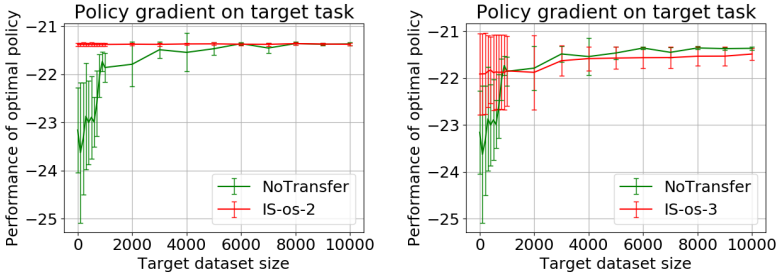


Figure 5.2: Learning curve of the NoTransfer experiment

provement.
Figure 5.3c corresponds to transfer from the most similar task using the most different policy. The jumpstart, the asymptotic performance and the variance of the learning process show all a positive impact due to transfer.
Figure 5.3d corresponds to transfer from the most different task using the most different policy. The most evident benefit is given by the jumpstart, while the variance and the asymptotic performance seem to be negatively affected. This is an example of negative transfer.
The results of IS-os-2 are completely expected, given the similarity between the tasks and the fact that the optimal source policy is being used. The higher variance of IS-rs-2 is explained by the difference between the policies, but it still reflects robustness to negative transfer at least in the case where the tasks' dynamics are not very different.
As shown by IS-os-3 and IS-rs-3, the change in the task parameters has a strong impact on the variance of the learning process, regardless of the nature of the source policy. However, the transfer of the optimal policy in the case of IS-os-3 produces a good jumpstart and asymptotic performance. This is not the case for IS-rs-3, where the only benefit is in the jumpstart.
Note that, in all cases, the performance does not evolve significantly along the learning process. The reason for this might be the proportion of the sizes of the source and target datasets, that reduces the contribution of the target samples to the estimation; however, decreasing the number of source samples is not an option because it would further increase the variance of the estimators.



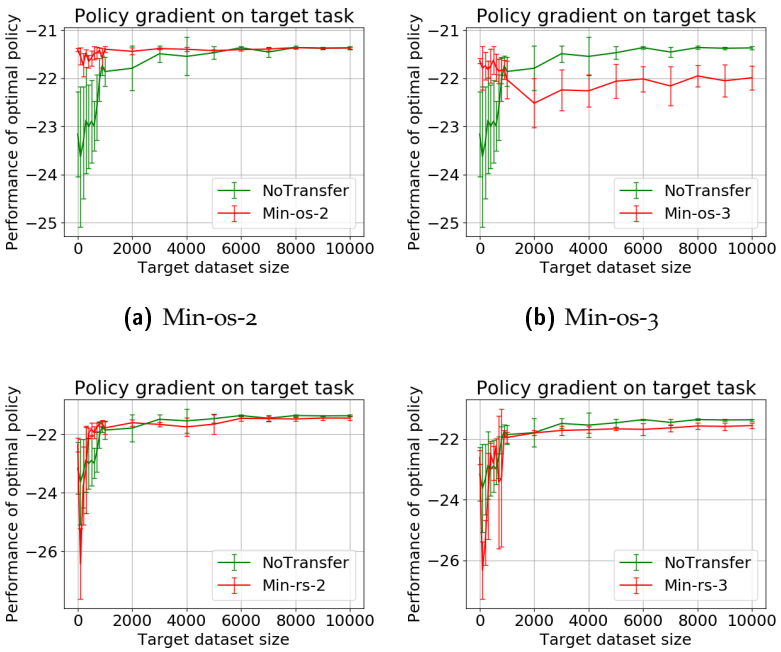
(a) IS-os-2 (b) IS-os-3

(c) IS-rs-2 (d) IS-rs-3

Figure 5.3: Learning curves for the IS experiments

Figure 5.4 shows the learning curves for the optimistic actor-critic algorithm in the different configurations.
Figure 5.4a corresponds to transfer from the most similar task using its optimal policy. The jumpstart, the asymptotic performance and the variance of the learning process show all a positive impact due to transfer.
Figure 5.4b corresponds to transfer from the most different task using its optimal policy. The most evident benefit is given by the jumpstart, while the asymptotic performance seem to be negatively affected. No impact on the variance is clear in this case.
Figure 5.4c corresponds to transfer from the most similar task using the most different policy. Besides a small reduction of variance asymptotically, no significant effects are observed as a consequence of transfer.
Figure 5.4d corresponds to transfer from the most different task using the most different policy. No significant effects are observed as a consequence of transfer.
The results of Min-os-2 are very satisfactory and show the effectiveness of the optimistic actor-critic algorithm at least in this configuration. The results of Min-rs-2 are explained by the very nature of the algorithm: it selects the weights that bring the final estimate as close as possible to the estimations obtained directly from target samples only (which are very noisy in the early learning phases), while keeping a low variance. The difference between the policies enlarges

the range of values available for the weights, making it easier for the algorithm to follow such noisy estimations with a lower variance. An explanation for the results of Min-os-3 is presented later in this section. A similar analysis to that of Min-rs-2 can be followed for Min-rs-3. Note, however, that the variance is higher in this case because the range of values for the weights is larger, which gives them more freedom to follow the target-only estimations. Anyway, the algorithm succeeds on avoiding the negative transfer by making the learning process to at least behave as in the no transfer scenario.



(a) Min-os-2

(b) Min-os-3

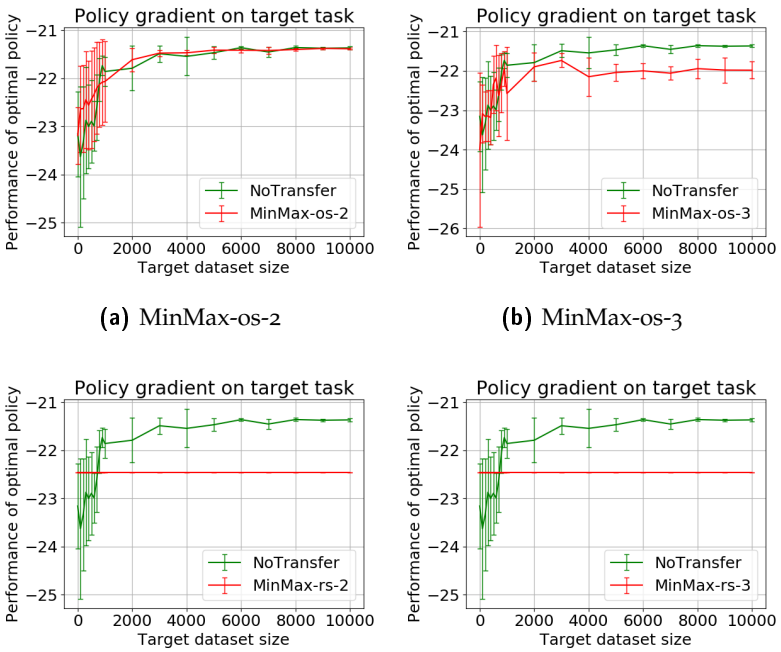
(c) Min-rs-2

(d) Min-rs-3

Figure 5.4: Learning curves for the Min experiments

Figure 5.5 shows the learning curves for the pessimistic actor-critic algorithm in the different configurations.
Figure 5.5a corresponds to transfer from the most similar task using its optimal policy. No significant effects are observed as a consequence of transfer.
Figure 5.5b corresponds to transfer from the most different task using its optimal policy. Besides the negative impact on the asymptotic performance, no other significant effects are observed as a consequence of transfer.
Figure 5.5c corresponds to transfer from the most similar task using the most different policy. Although there is a small jumpstart and a significant variance reduction, the asymptotic performance is strongly harmed.
Figure 5.5d corresponds to transfer from the most different task using

the most different policy. Although there is a small jumpstart and a significant variance reduction, the asymptotic performance is strongly harmed.
The reason for these results is the overestimation of the bias introduced by this algorithm, that results in a very pessimistic scenario. Therefore, there are no clear insights on how the weights can affect the estimation, making the selection of a good strategy very difficult. In particular, for the MinMax-rs cases, the algorithm is so pessimistic that it chooses the same strategy along the whole learning process.



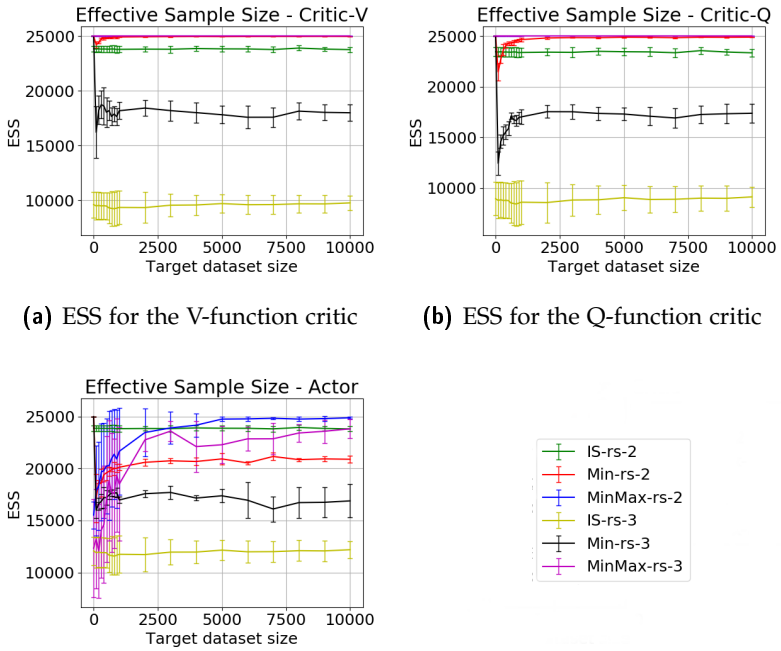
(a) MinMax-os-2 (b) MinMax-os-3

(c) MinMax-rs-2 (d) MinMax-rs-3

Figure 5.5: Learning curves for the MinMax experiments

Figure 5.6 and Figure 5.7 present the Effective Sample Size (ESS) produced by all the experiments. Each graph shows its evolution with respect to the size of the target dataset used to execute the actor-critic steps.

Figure 5.6 shows the ESS for the cases where transfer is done from the optimal source policy.
The IS actor-critic algorithm keeps a constant ESS along all the learning process for the critics and the actor. This is caused by the fact that the weights depend only on the tasks' models and these do not change during the process.
The optimistic actor-critic algorithm shows a small growth on the ESS during the beginning of the learning process, but it becomes constant after some point. With exception of the actor for Min-os-2, it decides



(a) ESS for the V-function critic

(b) ESS for the Q-function critic

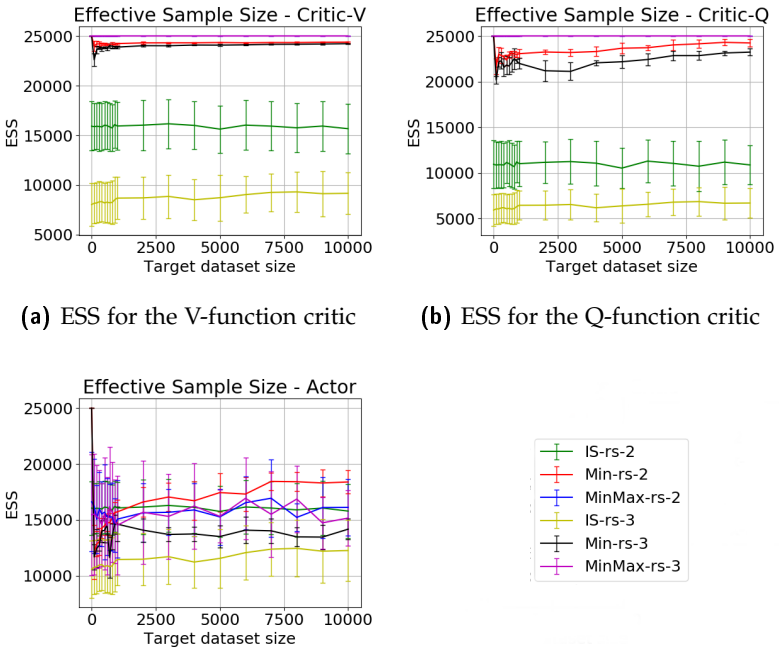
(c) ESS for the actor

Figure 5.6: Effective sample size for transfer from optimal policy

to introduce more samples than the IS algorithm.
The pessimistic actor-critic algorithm shows a constant ESS for the critics, and a small growth that later stabilizes for the actor. It decides to introduces more samples than the IS and optimistic algorithms. This reflects the fact that it mainly focuses on reducing the variance because the part of the effect of the weightening strategy on the bias has been replaced by the constant upper-bound.

Figure 5.6 shows the ESS for the cases where transfer is done from the most different policy.
The observed behavior is basically the same as that for transfer from the optimal source policy. This time, the gap between the ESS of the IS algorithm and of the optimistic and pessimistic ones is larger, which means that these two algorithms are now focusing more on reducing the variance.

The IS actor-critic algorithm gives the most solid jumpstart for all of the studied settings. However, it requires the model of the target task and has a slow convergence. In addition, suffers from a high variance and lacks of a way to modify the impact of the source samples in the estimation as the target dataset grows; this makes it prone to negative transfer.
The optimistic actor-critic algorithm tries to overcome these issues by finding a compromise between the bias and variance induced by



(a) ESS for the V-function critic

(b) ESS for the Q-function critic

(c) ESS for the actor

Figure 5.7: Effective sample size for transfer from worst policy

transfer. Although a jumpstart is not obtained in all scenarios, the initial performance of the policies is not significantly harmed. It also succeeds on not worsening the variance with respect to the IS algorithm, and is able to modify the transfer strategy as more target samples are available. Note that it achieves all this without requiring a model for the target task. Unfortunately, it is still vulnerable to negative transfer. The reason for this is that the bias and the variance might not lie within the same numerical ranges, and it causes the terms not to be given the appropriate relative relevance. For example, the algorithm can decide to introduce many source samples to cause a bias increase that is not significant with respect to the variance reduction, but that has a big effect in the bias space. Anyway, when this does not represent a problem, the algorithm turns out to be more resilient to negative transfer than IS actor-critic.

The pessimistic actor-critic algorithm turns out to be too conservative. Its decision of upper-bounding part of the bias is too dangerous because the real effect of a high bias is not perceived correctly. It basically offers no benefit with respect to the no transfer scenario, if it is not actually performing worse.

Note that all the algorithms decide to transfer less samples from the most different task than from the most similar one, and to transfer less samples from the most different policy than from the optimal ones. Thus, they are all able to correctly identify informative tasks and policies.

6

CONCLUSIONS AND FUTURE WORK

The Lipschitz continuity property, although very demanding with respect to the smoothness of the changes across the task space, represents a very powerful source of information for the design of Transfer Learning algorithms. In particular, when such information gets to describe pointwise similarity across tasks, it can be used for transferring low level knowledge such as individual samples.
This thesis has focused on the extension of actor-critic algorithms to transfer scenarios by studying how the Lipschitz conditions can be leveraged for an effective sample-level transfer. In particular, policy-gradient actors and function approximation based critics have been considered.
Importance Sampling estimators come as the first idea for a transfer mechanism. Although they are indeed unbiased and produce positive results in most cases, they suffer from high variance and are only applicable when a model for the target task is available. However, the idea of weighted estimators is taken from there and sets the basis of the two approaches proposed in this thesis. Both of them select the weights of their estimators by minimizing some approximation of the error that would result from the weighted transfer.
The first approach goes for the minimization of an optimistic approximation of the error calculated directly from the available target samples. As a result, it is capable of overcoming the variance and model requirement issues of the Importance Sampling estimation, while still producing good results in most cases. The main drawback of this approach is the unbalanced attention it pays to the bias and variance terms while selecting the optimal set of weights, which might even translate into negative transfer.
The second approach considers an overestimation of the approximated error for the minimization problem that defines the weights. The consequence of such overestimation is a blurry perception of the effects of the weights in the estimation, which makes it difficult for the algorithm to adequately select them. Thus, the transfer mechanism of this approach offers no significant improvements with respect to learning without transfer, and can even be easily affected by negative transfer.

Even if the optimistic algorithm is still not completely robust to negative transfer, the results it has produced make it a promising alternative for sample-based transfer. The following alternatives can be used as a starting point for later work trying to improve this approach:

- The objective function can be modified to include coefficients that correct the balance between the bias and variance terms. Note that they would correspond to hyper-parameters that depend on the specific task environment, so their selection can be itself a difficult work.

- It is possible that, even with the optimal strategy, the transfer of source samples represents a negative impact in the estimation. Given that the objective function is supposed to detect exactly this, the random nature of the estimation can be studied so that, based on confidence intervals, the algorithm can decide whether or not to perform transfer.

- The normalized versions of the Importance Sampling estimators offer lower variance estimates even if they are not unbiased. Thus, the approach can be rederived but based on normalized weighted estimators.

- In terms of implementation, the approximation of the objective function is linear in the size of the source dataset, linear in the dimension of the policy parameter space, and quadratic in the dimension of the feature space for the value function approximators. This calculation is performed several times at each single step of the actor-critic loop while solving the minimization problem, so that the full algorithm can be very slow. Any work for speeding-up the algorithm in this regard can be very useful.

BIBLIOGRAPHY

Amari, Shun-Ichi (1998). "Natural Gradient Works Efficiently in Learning." In: *Neural Comput.* 10.2, pp. 251–276. ISSN: 0899-7667 (cit. on p. 25).

Antos, András, Rémi Munos, and Csaba Szepesvári (2007). "Fitted Q-iteration in Continuous Action-space MDPs." In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NIPS'07. USA: Curran Associates Inc., pp. 9–16. ISBN: 978-1-60560-352-0 (cit. on pp. 19, 46).

Atkeson, C. G. and J. C. Santamaria (1997). "A comparison of direct and model-based reinforcement learning." In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4, pp. 3557–3564 (cit. on p. 17).

Baird, Leemon C. (1994). "Reinforcement learning in continuous time: advantage updating." In: *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*. Vol. 4, 2448–2453 vol.4 (cit. on p. 14).

Barrett, Samuel, Matthew E. Taylor, and Peter Stone (2010). "Transfer Learning for Reinforcement Learning on a Physical Robot." In: *Ninth International Conference on Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop (AAMAS - ALA)* (cit. on p. 47).

Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson (1983). "Neuronlike adaptive elements that can solve difficult learning control problems." In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5, pp. 834–846. ISSN: 0018-9472 (cit. on p. 35).

Baxter, Jonathan and Peter L. Bartlett (2001). "Infinite-horizon Policy-gradient Estimation." In: *J. Artif. Int. Res.* 15.1, pp. 319–350. ISSN: 1076-9757 (cit. on pp. 19, 22).

Beitelspacher, Josh, Jason Fager, Greg Henriques, and Amy McGovern (2006). "Policy gradient vs. value function approximation: A reinforcement learning shootout. Technical Report No. CS-TR-06-001." In: (cit. on pp. 19, 35).

Bertsekas, Dimitri P. (2007). *Dynamic Programming and Optimal Control, Vol. II*. 3rd. Athena Scientific. ISBN: 1886529302, 9781886529304 (cit. on p. 16).

Bhatnagar, Shalabh, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee (2009). "Natural Actor-critic Algorithms." In: *Automatica* 45.11, pp. 2471–2482. ISSN: 0005-1098 (cit. on pp. 24, 36).

Bradtke Steven J.and Barto, Andrew G. (1996). "Linear Least-Squares algorithms for temporal difference learning." In: *Machine Learning* 22.1, pp. 33–57. ISSN: 1573-0565 (cit. on p. 33).

Brown, Ann L. and Mary J. Kane (1988). "Preschool children can learn to transfer: Learning to learn and learning by example." In: 20, pp. 493–523 (cit. on p. 39).

Cantelli, Francesco P. (1933). "Sulla determinazione empirica della leggi di probabilita." In: *Giornale dell'Istituto Italiano degli Attuari* 4, pp. 421–424 (cit. on p. 93).

Caruana, Rich (1994). "Learning Many Related Tasks at the Same Time with Backpropagation." In: *Proceedings of the 7th International Conference on Neural Information Processing Systems*. NIPS'94. Cambridge, MA, USA: MIT Press, pp. 657–664 (cit. on p. 39).

Dann, Christoph, Gerhard Neumann, and Jan Peters (2014). "Policy Evaluation with Temporal Differences: A Survey and Comparison." In: *Journal of Machine Learning Research* 15, pp. 809–883 (cit. on pp. 28, 31, 33).

Degris, Thomas, Martha White, and Richard S. Sutton (2012). "Off-Policy Actor-Critic." In: *CoRR* abs/1205.4839. arXiv: 1205.4839 (cit. on p. 18).

Deisenroth, Marc P., Gerhard Neumann, and Jan Peters (2013). "A Survey on Policy Search for Robotics." In: *Found. Trends Robot* 2.1–2, pp. 1–142. ISSN: 1935-8253 (cit. on pp. 2, 20, 21).

Dudley, R.M. (2002). *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press. ISBN: 9780521007542 (cit. on p. 37).

Ferns, Norm, Prakash Panangaden, and Doina Precup (2005). "Metrics for Markov Decision Processes with Infinite State Spaces." In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. UAI'05. Arlington, Virginia, United States: AUAI Press, pp. 201–208. ISBN: 0-9749039-1-4 (cit. on p. 49).

Geist, Matthieu and Olivier Pietquin (2013). "Algorithmic Survey of Parametric Value Function Approximation." In: *IEEE Transactions on Neural Networks and Learning Systems* 24.6, pp. 845–867. ISSN: 2162-237X (cit. on pp. 28, 33).

— (2014). "Kalman Temporal Differences." In: *CoRR* abs/1406.3270. arXiv: 1406.3270 (cit. on p. 33).

Geist, Matthieu and Bruno Scherrer (2014). "Off-policy Learning with Eligibility Traces: A Survey." In: *J. Mach. Learn. Res.* 15.1, pp. 289–333. ISSN: 1532-4435 (cit. on pp. 18, 28, 33).

Gibbs, Alison L. and Francis Edward Su (2002). "On Choosing and Bounding Probability Metrics." In: *International Statistical Review* 70.3, pp. 419–435. ISSN: 1751-5823 (cit. on p. 115).

Glivenko, Valery I. (1933). "Sulla determinazione empirica della leggi di probabilita." In: *Giornale dell'Istituto Italiano degli Attuari* 4, pp. 92–99 (cit. on p. 93).

Grondman, Ivo, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska (2012). "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients." In: *Trans. Sys. Man Cy-*

ber Part C 42.6, pp. 1291–1307. ISSN: 1094-6977 (cit. on pp. 3, 24, 35, 36).

Hammersley, J.M. and K.W. Morton (1954). "Poor man's Monte Carlo." In: 16, pp. 23–38 (cit. on p. 95).

Hester, Todd and Peter Stone (2009). "Generalized Model Learning for Reinforcement Learning in Factored Domains." In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 717–724. ISBN: 978-0-9817381-7-8 (cit. on p. 17).

Hinderer, K. (2005). "Lipschitz Continuity of Value Functions in Markovian Decision Processes." In: 62, pp. 3–22 (cit. on p. 49).

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press (cit. on p. 35).

Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore (1996). "Reinforcement Learning: A Survey." In: *J. Artif. Int. Res.* 4.1, pp. 237–285. ISSN: 1076-9757 (cit. on pp. 2, 12, 16, 17).

Kakade, Sham (2001). "A Natural Policy Gradient." In: *Advances in Neural Information Processing Systems 14 (NIPS 2001)*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. MIT Press, pp. 1531–1538 (cit. on p. 25).

Kober, Jens, J. Andrew Bagnell, and Jan Peters (2013). "Reinforcement Learning in Robotics: A Survey." In: *Int. J. Rob. Res.* 32.11, pp. 1238–1274. ISSN: 0278-3649 (cit. on pp. 2, 18, 19, 35).

Kohl, Nate and Peter Stone (2004). "Policy gradient reinforcement learning for fast quadrupedal locomotion." In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 3, 2619–2624 Vol.3 (cit. on p. 21).

Konda, Vijay R. and John N. Tsitsiklis (2003). "On Actor-Critic Algorithms." In: *SIAM J. Control Optim.* 42.4, pp. 1143–1166. ISSN: 0363-0129 (cit. on pp. 2, 24, 35, 36).

Kong, Augustine (1992). "A note on importance sampling using standardized weights." In: *University of Chicago, Dept. of Statistics, Tech. Rep* 348 (cit. on pp. 94, 95).

Kupcsik, Andras, Marc P. Deisenroth, Jan Peters, Ai P. Loh, Prahlad Vadakkepat, and Gerhard Neumann (2017). "Model-based contextual policy search for data-efficient generalization of robot skills." In: *Artificial Intelligence* 247.Supplement C. Special Issue on AI and Robotics, pp. 415 –439. ISSN: 0004-3702 (cit. on p. 17).

Lagoudakis, Michail G. and Ronald Parr (2003). "Least-squares Policy Iteration." In: *J. Mach. Learn. Res.* 4, pp. 1107–1149. ISSN: 1532-4435 (cit. on p. 34).

Lange, Sascha, Thomas Gabel, and Martin Riedmiller (2012). "Batch Reinforcement Learning." In: *Reinforcement Learning: State-of-the-Art*. Ed. by Marco Wiering and Martijn van Otterlo. Berlin, Hei-

delberg: Springer Berlin Heidelberg, pp. 45–73. ISBN: 978-3-642-27645-3 (cit. on p. 19).

Laroche, Romain and Merwan Barlier (2017). "Transfer Reinforcement Learning with Shared Dynamics." In: *AAAI-17 - Thirty-First AAAI Conference on Artificial Intelligence*. San Francisco, United States, p. 7 (cit. on p. 47).

Lazaric, Alessandro (2012). "Transfer in Reinforcement Learning: a Framework and a Survey." In: *Reinforcement Learning - State of the art*. Ed. by Martijn van Otterlo Marco Wiering. Vol. 12. Springer, pp. 143–173 (cit. on pp. 39, 40, 42–44, 50).

Lazaric, Alessandro and Marcello Restelli (2011). "Transfer from Multiple MDPs." In: *CoRR* abs/1108.6211. arXiv: 1108.6211 (cit. on p. 46).

Lazaric, Alessandro, Marcello Restelli, and Andrea Bonarini (2008). "Transfer of Samples in Batch Reinforcement Learning." In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: ACM, pp. 544–551. ISBN: 978-1-60558-205-4 (cit. on p. 46).

Leemon, Baird (1995). "Residual Algorithms: Reinforcement Learning with Function Approximation." In: *In Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, pp. 30–37 (cit. on p. 33).

Levine, Sergey and Vladlen Koltun (2013). "Guided Policy Search." In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, pp. III–1–III–9 (cit. on p. 96).

Mahadevan, Sridhar (2005). "Proto-value Functions: Developmental Reinforcement Learning." In: *Proceedings of the 22Nd International Conference on Machine Learning*. ICML '05. New York, NY, USA: ACM, pp. 553–560. ISBN: 1-59593-180-5 (cit. on p. 41).

Mahmood, A. Rupam, Hado van Hasselt, and Richard S. Sutton (2014). "Weighted Importance Sampling for Off-policy Learning with Linear Function Approximation." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Cambridge, MA, USA: MIT Press, pp. 3014–3022 (cit. on p. 96).

Mahmood, A. Rupam and Richard S. Sutton (2015). "Off-policy Learning Based on Weighted Importance Sampling with Linear Computational Complexity." In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. UAI'15. Arlington, Virginia, United States: AUAI Press, pp. 552–561. ISBN: 978-0-9966431-0-8 (cit. on p. 96).

Mitchell, Thomas M. (1997). *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc. ISBN: 0070428077, 9780070428072 (cit. on p. 39).

Nedić, A. and D. P. Bertsekas (2003). "Least Squares Policy Evaluation Algorithms with Linear Function Approximation." In: *Discrete Event Dynamic Systems* 13.1, pp. 79–110. ISSN: 1573-7594 (cit. on p. 33).

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell (1999). "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping." In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 278–287. ISBN: 1-55860-612-2 (cit. on p. 41).

Owen, Art B. (2013). *Monte Carlo theory, methods and examples* (cit. on p. 95).

Peshkin, Leonid and Christian R. Shelton (2002). "Learning from Scarce Experience." In: *Proceedings of the Nineteenth International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 498–505. ISBN: 1-55860-873-7 (cit. on p. 96).

Peters, J., S. Vijayakumar, and S. Schaal (2003). "Reinforcement learning for humanoid robotics." In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids2003)*. clmc. Karlsruhe, Germany, Sept.29-30 (cit. on pp. 20, 24).

Peters, Jan and Stefan Schaal (2008a). "Natural Actor-Critic." In: *Neurocomput.* 71.7-9, pp. 1180–1190. ISSN: 0925-2312 (cit. on pp. 18, 25, 35, 36).

— (2008b). "Reinforcement learning of motor skills with policy gradients." In: *Neural Networks* 21.4. Robotics and Neuroscience, pp. 682–697. ISSN: 0893-6080 (cit. on pp. 2, 20–23).

Pirotta, Matteo, Marcello Restelli, and Luca Bascetta (2015). "Policy gradient in Lipschitz Markov Decision Processes." In: *Machine Learning* 100.2, pp. 255–283. ISSN: 1573-0565 (cit. on pp. 12, 36, 38, 97, 99, 115).

Polydoros, Athanasios S. and Lazaros Nalpantidis (2017). "Survey of Model-Based Reinforcement Learning: Applications on Robotics." In: *J. Intell. Robotics Syst.* 86.2, pp. 153–173. ISSN: 0921-0296 (cit. on p. 26).

Precup, Doina, Richard S. Sutton, and Satinder P. Singh (2000). "Eligibility Traces for Off-Policy Policy Evaluation." In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 759–766. ISBN: 1-55860-707-2 (cit. on p. 96).

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471619779 (cit. on p. 14).

Rachelson, Emmanuel and Michail G. Lagoudakis (2010). "On the locality of action domination in sequential decision making." In: *International Symposium on Artificial Intelligence and Mathematics,*

ISAIM 2010, Fort Lauderdale, Florida, USA, January 6-8, 2010 (cit. on p. 49).

Robbins, Herbert and Sutton Monro (1951). "A Stochastic Approximation Method." In: *Ann. Math. Statist.* 22.3, pp. 400–407 (cit. on p. 32).

Rosenbluth, Marshall N. and Arianna W. Rosenbluth (1955). "Monte Carlo Calculation of the Average Extension of Molecular Chains." In: *The Journal of Chemical Physics* 23.2, pp. 356–359. eprint: https://doi.org/10.1063/1.1741967 (cit. on p. 95).

Rubinstein, Reuven Y. (1981). *Simulation and the Monte Carlo Method*. 1st. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471089176 (cit. on pp. 93, 95, 96).

Rummery, G. A. and M. Niranjan (1994). *On-Line Q-Learning Using Connectionist Systems*. Tech. rep. TR 166. Cambridge, England: Cambridge University Engineering Department (cit. on pp. 17, 18, 28).

Russell, Stuart J. and Peter Norvig (2003). *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education. ISBN: 0137903952 (cit. on p. 1).

Singh, Satinder P. and Richard S. Sutton (1996). "Reinforcement Learning with Replacing Eligibility Traces." In: *Machine Learning* 22.1, pp. 123–158. ISSN: 1573-0565 (cit. on p. 26).

Sutton, Richard S. (1988). "Learning to Predict by the Methods of Temporal Differences." In: *Mach. Learn.* 3.1, pp. 9–44. ISSN: 0885-6125 (cit. on pp. 18, 27).

— (1991). "Dyna, an Integrated Architecture for Learning, Planning, and Reacting." In: *SIGART Bull.* 2.4, pp. 160–163. ISSN: 0163-5719 (cit. on p. 17).

Sutton, Richard S. and Andrew G. Barto (1998). *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press. ISBN: 0262193981 (cit. on pp. 2, 7, 8, 12, 13, 16, 17, 19, 26, 27, 73).

Sutton, Richard S., Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning." In: *Artif. Intell.* 112.1-2, pp. 181–211. ISSN: 0004-3702 (cit. on p. 41).

Sutton, Richard S., Csaba Szepesvári, and Hamid Reza Maei (2008). "A Convergent O(N) Algorithm for Off-policy Temporal-difference Learning with Linear Function Approximation." In: *Proceedings of the 21st International Conference on Neural Information Processing Systems*. NIPS'08. USA: Curran Associates Inc., pp. 1609–1616. ISBN: 978-1-6056-0-949-2 (cit. on p. 33).

Sutton, Richard S., David McAllester, Satinder Singh, and Yishay Mansour (1999). "Policy Gradient Methods for Reinforcement Learning with Function Approximation." In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Cambridge, MA, USA: MIT Press, pp. 1057–1063 (cit. on pp. 11, 23, 24).

Sutton, Richard S., Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora (2009). "Fast Gradient-descent Methods for Temporal-difference Learning with Linear Function Approximation." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, pp. 993–1000. ISBN: 978-1-60558-516-1 (cit. on p. 31).

Szepesvari, Csaba (2010). *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers. ISBN: 1608454924, 9781608454921 (cit. on pp. 11, 15, 16).

Taylor, Matthew E. and Peter Stone (2009). "Transfer Learning for Reinforcement Learning Domains: A Survey." In: *J. Mach. Learn. Res.* 10, pp. 1633–1685. ISSN: 1532-4435 (cit. on pp. 3, 39, 40, 43–46).

Taylor, Matthew E., Peter Stone, and Yaxin Liu (2007). "Transfer Learning via Inter-Task Mappings for Temporal Difference Learning." In: *J. Mach. Learn. Res.* 8, pp. 2125–2167. ISSN: 1532-4435 (cit. on p. 44).

Theodoridis, S. (2015). *Machine Learning: A Bayesian and Optimization Perspective*. .NET Developers Series. Elsevier Science. ISBN: 9780128017227 (cit. on p. 94).

Thrun, Sebastian and Lorien Pratt, eds. (1998). *Learning to Learn*. Norwell, MA, USA: Kluwer Academic Publishers. ISBN: 0-7923-8047-9 (cit. on p. 39).

Tokdar, Surya T. and Robert E. Kass (2010). "Importance sampling: a review." In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.1, pp. 54–60. ISSN: 1939-0068 (cit. on pp. 93, 94, 96).

Vershik, A. M. (2013). "Long History of the Monge-Kantorovich Transportation Problem." English. In: *The Mathematical Intelligencer* 35.4, pp. 1–9 (cit. on p. 97).

Vlassis, Nikos (2007). *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. 1st. Morgan and Claypool Publishers. ISBN: 1598295268, 9781598295269 (cit. on p. 10).

Watkins, Christopher J. (1989). "Learning from Delayed Rewards." PhD thesis. Cambridge, UK: King's College (cit. on pp. 17, 18).

Wawrzynski, Pawel and Andrzej Pacut (2008). "Truncated Importance Sampling for Reinforcement Learning with Experience Replay." In: pp. 305–315 (cit. on p. 96).

Williams, Ronald J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: *Mach. Learn.* 8.3-4, pp. 229–256. ISSN: 0885-6125 (cit. on pp. 17, 19, 22).

Zhao, Qiang, Guo Liu, and Guiding Gu (2013). "Variance Reduction Techniques of Importance Sampling Monte Carlo Methods for Pricing Options." In: *Journal of Mathematical Finance* 3.4, p. 6 (cit. on p. 94).

A

IMPORTANCE SAMPLING

A recurrent problem found in applied statistics (and, in particular, in RL) is the calculation of the expected value of a function with respect to some probability distribution. Often, this value cannot be exactly computed because it is analytically complex or because the distribution is unknown.
If it possible to access samples from the distribution, the target expectation can still be approximated with the empirical mean due to the almost sure convergence of the empirical distribution to the real one (Glivenko-Cantelli theorem, Glivenko (1933) and Cantelli (1933)). This justifies the creation of MC-based estimations.
When sampling the target distribution is costly (or even impossible), MC-based estimators are no longer applicable. However, IS techniques provide a solution for this issue given that some information about the target distribution is available and that sampling from another well-known distribution is still possible (Rubinstein, 1981). IS is also a powerful tool when MC estimators have high variance issues.
The scope of this chapter is to introduce the IS paradigm for expectation estimation and its role in the RL framework. Section A.1 describes the formal mathematical formulation of IS as well as some interesting properties. Section A.2 is concerned with the specific case of IS applied to RL.

A.1 MATHEMATICAL FORMULATION AND PROPERTIES

IS methods are very well suited to estimate the expectation of a function when the original distribution is known and costly to sample from or when it has an inherent high variance, but there is either way another well-known distribution available for cheap sampling. Along this section, the function is referred to as *target function*, the original distribution as *target distribution*, and the cheap sampling distribution as *source distribution*.
The IS paradigm is based on the following simple yet powerful observation (Tokdar and Kass, 2010). Let $X \subset \mathbb{R}^d$ and consider a real valued function $f : X \to \mathbb{R}$. For the probability distribution $p : X \to \mathbb{R}$, define

$$\mu_f = \int_X f(x)p(x)\,dx = \mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_p[f(x)],$$

the expected value of f with respect to the distribution p. For any other probability distribution q such that $q(x) > 0$ whenever $f(x)p(x) \neq 0$, it holds that

$$\mu_f = \mathbb{E}_q\left[\frac{p(x)}{q(x)}f(x)\right] = \mathbb{E}_q\left[w(x)f(x)\right],$$

where $w(x) = p(x)/q(x)$ is called *importance weight*. Taking f, p and q as target function, target distribution and source distribution, respectively, sets the rationale for the IS estimator.

Definition A.1 (Importance Sampling estimator)**.** Let f be the target function, p the target distribution and q any (compatible) source distribution. Given a set of i.i.d. samples $(x_i)_{i=1}^n$ drawn from q, the Importance Sampling estimator for μ_f is

$$\begin{aligned}\hat{\mu}_f &= \frac{1}{n}\sum_{i=1}^{n}\frac{p(x_i)}{q_(x_i)}f(x_i)\\ &= \frac{1}{n}\sum_{i=1}^{n}w(x_i)f(x_i),\end{aligned}$$

where $(w(x_i))_{i=1}^n$ are the importance weights.

This estimator happens to be both unbiased and consistent, and its variance is given by (Zhao, Liu, and Gu (2013), Tokdar and Kass (2010)):

$$\begin{aligned}\mathrm{Var}[\hat{\mu}_f] &= \frac{1}{n}\mathbb{E}_p[f(x)^2w(s)] - \mu f^2\\ &= \frac{1}{n}\int_X \frac{(f(x)p(x) - \mu_f q(x))^2}{q(x)}\,dx\end{aligned} \tag{A.1}$$

IS also works under the weaker scenario where the target distribution is known only up to a constant factor. In this case, the normalized IS estimator is to be used.

Definition A.2 (Normalized Importance Sampling estimator)**.** Let f be the target function, p the target distribution and q any (compatible) source distribution. If $(x_i)_{i=1}^n$ are i.i.d. samples from q, the normalized Importance Sampling estimator for μ_f is

$$\tilde{\mu}_f = \frac{\sum_{i=1}^n \frac{p_(x_i)}{q_(x_i)}f(x_i)}{\sum_{i=1}^n \frac{p_(x_i)}{q_(x_i)}} = \frac{\sum_{i=1}^n w_0(x_i)f(x_i)}{\sum_{i=1}^n w_0(x_i)},$$

where $(w(x_i))_{i=1}^n$ are the importance weights.

This estimator is consistent but biased (Theodoridis, 2015). Its variance is harder to study analytically, so approximate expression are often used. Kong (1992) proposed the formula

$$\mathrm{Var}[\tilde{\mu_f}] \approx \mathrm{Var}[\dot{\mu}_f](1 + \mathrm{Var}_p[w(x)]), \tag{A.2}$$

where $\dot{\mu}_f$ is the standard MC estimate for μ_f. Another alternative was proposed by Owen (2013):

$$\mathrm{Var}[\tilde{\mu_f}] \approx \frac{1}{n}\mathbb{E}_q[w(x)^2(f(x)-\mu_f)^2],$$

that he suggests to approximate in practice as:

$$\widehat{\mathrm{Var}[\tilde{\mu_f}]} \approx \sum_{i=1}^{n} w'(x_i)^2(f(x_i)-\tilde{\mu_f})^2,$$

where $w'(x_i) = w(x_i)/\sum_{i=1}^{n} w(x_i)$.

For variance reduction purposes, IS needs the source distribution to be chosen properly (Rubinstein, 1981). In fact, for the case of the IS estimator, last line in Equation (A.1) implies that an optimal choice for q is fp/μ_f as it would induce a zero-variance estimator. Although not applicable in practice, this observation suggests that good source distributions are those that are nearly proportional to fp. More precisely, the same equation leads to

$$\mathrm{Var}[\dot{\mu}_f] - \mathrm{Var}[\hat{\mu}_f] = \frac{1}{n}\mathbb{E}_p[(1-w(x))f(x)^2].$$

Therefore, the variance of the standard MC estimate can be decreased if $q > p$ in the regions that are likely for the target distribution and with high values of f, and $q < p$ for the areas that are unlikely for the target and with a low value of f (Owen, 2013).
For the case of the normalized IS estimator, studying the variance reduction is a challenging task because its variance is analytically complex itself. However, from Equation (A.2), Kong (1992) defined the *effective sample size*

$$\mathrm{ESS} = \frac{n}{1+\mathrm{Var}_f[w(x)]}$$

as a measure of how many samples, in average, have an effective contribution to the estimation. Given the observed samples $(x_i)_{i=1}^n$, this value is usually approximated through

$$\mathrm{ESS} \approx \frac{\|\boldsymbol{w}\|_1^2}{\|\boldsymbol{w}\|_2^2},$$

where $\boldsymbol{w} = (w(x_1), ..., w(x_n))^\top$. A high value for the effective sample size means that more samples were actually taken into account for the calculation, possibly resulting in a low variance for the estimator.

A.2 IMPORTANCE SAMPLING IN REINFORCEMENT LEARNING

One of the first successful applications of the idea of IS was achieved by Hammersley and Morton (1954), and Rosenbluth and Rosenbluth

(1955) in the field of physical statistics. Since then, it has been broadly studied and applied into a variety of other areas, and RL has not been the exception.
Among all the variants of IS methods that have been proposed along time (see e. g.Rubinstein (1981), Tokdar and Kass (2010)), the one presented in Section A.1 is the most commonly applied in the RL area. It is mainly introduced to derive algorithms that can reuse samples coming from different policies along the learning process.
One group of IS-based algorithms in RL focuses on policy evaluation (Section 2.4) within an off-policy context (Section 2.2.2). Precup, Sutton, and Singh (2000) develop an off-policy variant of the TD(λ) algorithm (Section 2.4.2) that introduces the importance weights in a transition-by-transition fashion into the eligibility traces; they provide formulations based on both the standard and the normalized IS estimators. Mahmood, Hasselt, and Sutton (2014) go one step further and present an off-policy variant of LSTD(λ) by seeing the IS estimators as solutions for weighted versions of linear-regression problems; they focus in particular on the on the formulation for the normalized IS estimator. Mahmood and Sutton (2015) go for a an approach that enriches SGD with normalized IS and combines it with eligibility traces to solve for the optimal parameters of the value function approximator; this algorithm provides linear complexity on the size of the feature space.
IS has also been used for policy search purposes in RL. Levine and Koltun (2013) apply Differential Dynamic Programming to build guiding distributions that favor sampling from high reward regions, and use samples from them to optimize a regularized version of the normalized IS estimator for the policy performance. Peshkin and Shelton (2002) work under POMDP and propose the concept of a proxy between the agent and the environment that keeps samples from different policies and, when queried, provides normalized IS estimations of both the performance and gradient of a new policy; this information is directly used to search for the optimal policy.
Actor-critic approaches have also been modified to account for IS. Wawrzynski and Pacut (2008), seeing actor-critic algorithm as an instance of single-adjustment methods, combine experience replay with truncated IS at each step of the iteration in order to create stochastic gradient estimations that guide the parameter updates. They also provide a study of the bias introduced by the truncation and prove that it vanishes asymptotically.

B

KANTOROVICH DISTANCE AND LOCAL INFORMATION

The Kantorovich metric for probability measures was introduced by Leonid Vitálievich Kantorórovich as a result of his studies on the transportation problem (Vershik, 2013).

Definition B.1 (Kantorovich metric)**.** Let (X, r) be a metric compact space, and p, q be two probability Borel measures on X. The function

$$\mathcal{K}(p, q) = \inf_{\gamma} \int r(x_1, x_2)\, d\gamma,$$

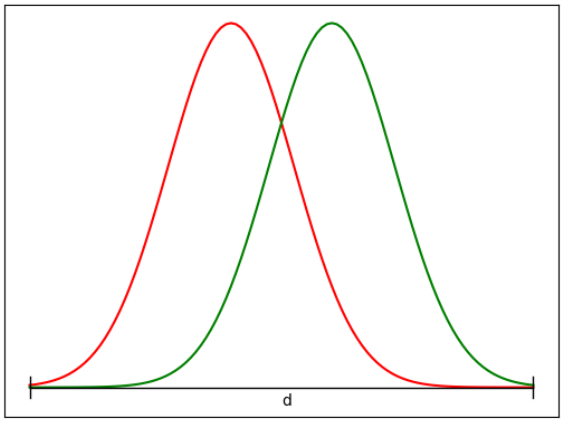
where the infimum is taken over all the probability Borel measures on $X \times X$ such that it has p and q as marginals, defines a metric on the space of probability measures on X.

The optimization problem that defines the metric is exactly the transportation problem with cost is r. Kantorórovich proved later on that

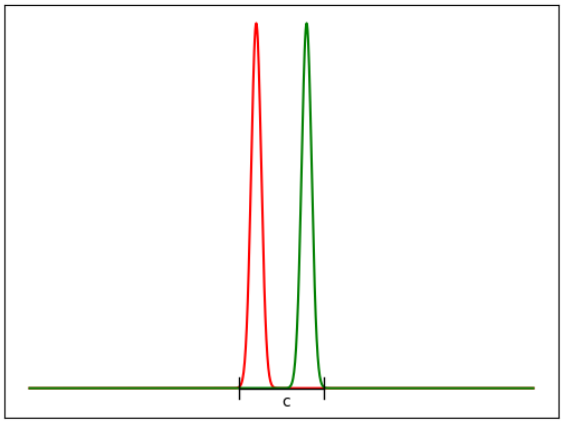
$$\mathcal{K}(p, q) = \sup \left\{ \left| \int f\, d(p - q) \right| \right\},$$

where the supremum is taken over all the 1-Lipschitz functions on X with respect to the metric r.
Intuitively, from this result, the metric measures the maximum difference between calculating the expected value of a Lipschitz function with each one of the distributions. This justifies the decision of Pirotta, Restelli, and Bascetta (2015) of modeling the continuity in robotic environments with this metric.
However, from the first definition, the intuitive meaning of the metric is that it measures the optimal cost of moving the densities to turn one distribution into the other, and, without any further assumptions, this gives no information about how the distributions differ from each other locally. Therefore, it is meaningless to build a transfer mechanism that can work at a sample level based only on the pointwise differences between the densities of the source and the target task.
To exemplify this, consider the two pairs of distributions in Figure B.1, all defined on the same space X. The Kantorovich metric between the pair of distributions in Figure B.1b can be made to coincide with that between the distributions in Figure B.1a: even if almost all the density has to be moved, c can be adjusted such that the overall cost of moving it coincides with the desired value. However, the pointwise distance between the pairs of distributions is completely different. Thus, the fact that two distributions are away from each

other by some value according to the Kantorovich distance gives no information of how the change locally.



(a) Locally similar



(b) Locally different

Figure B.1: The two pairs of probability distributions can have the same Kantorovich metric but the pointwise distances are very different.

C

LIPSCHITZ CONTINUITY

This chapter details the proofs regarding Lipschitz continuity that were not presented along the document. Section C.1 presents the proof for the extended tuples distribution, as well as the intermediate results on the continuity of the other distributions that are used later during the chapter. Section C.2 proves the continuity regarding the matrices used by the LSTD algorithm, both for the V-function and the Q-function. Section C.3 introduces the proofs for the continuity of the policy performance, which includes results on the continuity of the immediate reward. Section C.4 proves the continuity of the gradient of the policy performance, together with the intermediate results on the continuity of the V-function and the Q-function.
Most of the proofs can be redone by proving Lipschitz continuity with respect to the task parameters only, and then resorting to the results by Pirotta, Restelli, and Bascetta (2015) on the continuity with respect to the policy parameters to complete.

C.1 LISPCHITZ CONTINUITY OF THE TUPLES DISTRIBUTION

This section presents the proof of the Lipschitz continuity of the tuples distributions with respect to the task and policy parameters. It proceeds by stating the necessary assumptions and incrementally proving the result in a sequence of lemmata.

Assumption C.1. The Lipschitz constant of the transition model with respect to the task parameter space is bounded with respect to the action space. That is,

$$L^{s,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon) = \sup_{a\in\mathcal{A}} L^{s,a,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon) < \infty,$$

for any $\epsilon \in \mathcal{E}$. Further more, $L^{s,s'}_{\mathcal{P}-\mathcal{E}}(\cdot)$ is also bounded.

Assumption C.2. The Lipschitz constant of the policy with respect to the parameter space is bounded with respect to the action space. That is,

$$L^{s}_{\pi-\Theta}(\theta) = \sup_{a\in\mathcal{A}} L^{s,a}_{\pi-\Theta}(\theta) < \infty,$$

for any $\theta \in \Theta$. Furthermore, $L^{s}_{\pi-\Theta}(\cdot)$ is also bounded.

Lemma C.1 (Lipschitz continuity of state-to-state distribution)**.** *Let* $\langle \mathfrak{T}, \Omega \rangle$ *be a Lipschitz task environment (Definition 4.2), and* Π_Θ *be a para-*

metric space of Lipschitz policies (Definition 2.32). For any $(s, s') \in \mathcal{S} \times \mathcal{S}$*, the function* $\mathcal{P}_{(\cdot,\cdot)}(s'|s)$ *is* $L^{s,s'}_{\mathcal{P}-\Theta\mathcal{E}}$*-PLC, with*

$$L^{s,s'}_{\mathcal{P}-\Theta\mathcal{E}}(\theta, \epsilon) = M^{s,s'}_{\mathcal{P}-\mathcal{E}} L^{s}_{\pi-\Theta}(\theta)\lambda(\mathcal{A}) + L^{s,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon),$$

where $\lambda(\mathcal{A})$ *is the volume of* $\mathcal{A}$*.*

Proof. Let $(s, s') \in \mathcal{S} \times \mathcal{S}$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$\begin{aligned}
|\mathcal{P}_1(s'|s) - \mathcal{P}_2(s'|s)| = & \left| \int_{\mathcal{A}} (\mathcal{P}_1(s'|s,a)\pi_1(a|s) - \mathcal{P}_2(s'|s,a)\pi_2(a|s))\, da \right| \\
\leqslant & \int_{\mathcal{A}} \mathcal{P}_1(s'|s,a)|\pi_1(a|s) - \pi_2(a|s)|\, da \\
& + \int_{\mathcal{A}} \pi_2(a|s)|\mathcal{P}_1(s'|s,a) - \mathcal{P}_2(s'|s,a)|\, da \\
\leqslant & \int_{\mathcal{A}} \mathcal{P}_1(s'|s,a) L^{s,a}_{\pi-\Theta}(\theta_1) d_\Theta(\theta_1, \theta_2)\, da \\
& + \int_{\mathcal{A}} \pi_2(a|s) L^{s,a,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon_1) d_{\mathcal{E}}(\epsilon_1, \epsilon_2)\, da \\
\leqslant & M^{s,s'}_{\mathcal{P}-\mathcal{E}} L^{s}_{\pi-\Theta}(\theta_1) d_\Theta(\theta_1, \theta_2)\lambda(\mathcal{A}) \\
& + L^{s,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon_1) d_{\mathcal{E}}(\epsilon_1, \epsilon_2) \\
\leqslant & \left(M^{s,s'}_{\mathcal{P}-\mathcal{E}} L^{s}_{\pi-\Theta}(\theta_1)\lambda(\mathcal{A}) + \right. \\
& \left. L^{s,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon_1)\right) d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_1, \epsilon_2)) \\
= & L^{s,s'}_{\mathcal{P}-\Theta\mathcal{E}}(\theta_1, \epsilon_1) d_{\Theta\mathcal{E}}((\theta_1, \epsilon_1), (\theta_1, \epsilon_2)).
\end{aligned}$$

■

Assumption C.3. The Lipschitz constant of the state-to-state distribution with respect to the parameter space is bounded with respect to the starting state. That is,

$$L^{s'}_{\mathcal{P}-\Theta\mathcal{E}}(\theta, \epsilon) = \sup_{s \in \mathcal{A}} L^{s,s'}_{\mathcal{P}-\Theta\mathcal{E}}(\theta, \epsilon) < \infty,$$

for any $(\theta, \epsilon) \in \Theta \times \mathcal{E}$. Furthermore, $L^{s'}_{\mathcal{P}-\Theta\mathcal{E}}(\cdot, \cdot)$ is also bounded.

Assumption C.4. The function

$$\begin{aligned}
L^{s_0}_{\delta-\Theta\mathcal{E}}(\theta, \epsilon) = & K^{s_0}(\theta, \epsilon) + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1) K^{s_1}(\theta, \epsilon)\, ds_1 \\
& + \gamma^2 \int_{\mathcal{S}^2} \mathcal{P}_1(s_0|s_1)\mathcal{P}_1(s_1|s_2)\, K^{s_2}(\theta, \epsilon) + \ldots,
\end{aligned}$$

where

$$K^s(\theta, \epsilon) = (1-\gamma) L^{s}_{\mu-\mathcal{E}}(\epsilon) + \gamma L^{s}_{\mathcal{P}-\Theta\mathcal{E}}(\theta, \epsilon),$$

is well-defined for every $s_0 \in \mathcal{S}$, $\theta \in \Theta$ and $\epsilon \in \mathcal{E}$.

Lemma C.2 (Lipschitz continuity of the future state distribution)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $s \in \mathcal{S}$, the function $\delta_{(\cdot,\cdot)}(s)$ is $L^s_{\delta-\Theta\mathcal{E}}$-PLC, where $L^s_{\delta-\Theta\mathcal{E}}$ is the function defined in Assumption C.4.*

Proof. Let $s \in \mathcal{S}$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$
\begin{aligned}
&|\delta_1(s_0) - \delta_2(s_0)| \\
&= \Big|(1-\gamma)\mu_1(s_0) + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)\delta_1(s_1)\,ds_1 \\
&\quad - (1-\gamma)\mu_2(s_0) - \gamma \int_{\mathcal{S}} \mathcal{P}_2(s_0|s_1)\delta_2(s_1)\,ds_1\Big| \\
&\leqslant (1-\gamma)|\mu_1(s_0) - \mu_2(s_0)| \\
&\quad + \gamma \int_{\mathcal{S}} |\mathcal{P}_1(s_0|s_1)\delta_1(s_1) - \mathcal{P}_2(s_0|s_1)\delta_2(s_1))|\,ds_1 \\
&\leqslant (1-\gamma)L^{s_0}_{\mu-\mathcal{E}}(\epsilon_1)d_{\mathcal{E}}(\epsilon_1,\epsilon_2) \\
&\quad + \gamma \int_{\mathcal{S}} \delta_2(s_1)|\mathcal{P}_1(s_0|s_1) - \mathcal{P}_2(s_0|s_1)|\,ds_1 \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)|\delta_1(s_1) - \delta_2(s_1)|\,ds_1 \\
&\leqslant (1-\gamma)L^{s_0}_{\mu-\mathcal{E}}(\epsilon_1)d_{\mathcal{E}}(\epsilon_1,\epsilon_2) \\
&\quad + \gamma L^{s_0}_{\mathcal{P}-\Theta\mathcal{E}}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)|\delta_1(s_1) - \delta_2(s_1)|\,ds_1 \\
&\leqslant \big((1-\gamma)L^{s_0}_{\mu-\mathcal{E}}(\epsilon_1) + \gamma L^{s_0}_{\mathcal{P}-\Theta\mathcal{E}}(\theta_1,\epsilon_1)\big)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)|\delta_1(s_1) - \delta_2(s_1)|\,ds_1 \\
&= K^{s_0}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)|\delta_1(s_1) - \delta_2(s_1)|\,ds_1 \\
&\leqslant K^{s_0}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)\ K^{s_1}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\,ds_1 \\
&\quad + \gamma^2 \int_{\mathcal{S}^2} \mathcal{P}_1(s_0|s_1)\mathcal{P}_1(s_1,s_2)|\delta(\epsilon_1,s_2) - \delta(\epsilon_2,s_2)|\,ds_2\,ds_1 \\
&\leqslant K^{s_0}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \\
&\quad + \gamma \int_{\mathcal{S}} \mathcal{P}_1(s_0|s_1)K^{s_1}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\,ds_1 \\
&\quad + \gamma^2 \int_{\mathcal{S}^2} \mathcal{P}_1(s_0|s_1)\mathcal{P}_1(s_1|s_2)\ K^{s_2}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\,ds_2\,ds_1 \\
&\quad + \dots \\
&= L^{s_0}_{\delta\Theta\mathcal{E}}(\theta_1,\epsilon_1)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)).
\end{aligned}
$$

■

Lemma C.3. *Let* $(X, d_X), (Y, d_Y)$ *be metric spaces, and make* $X \times Y$ *a metric space with the metric* $d_{XY} = d_X + d_Y$*).*
Let $g : X \times Y \to \mathbb{R}$ *be a* L_g*-PLC function and* $h : Y \to \mathbb{R}$ *be a* L_h*-PLC function. Assume that* $|g|$ *is bounded by* M_g *and that* $|h|$ *is bounded by* M_h*. Then the function* $f : X \times Y \to \mathbb{R}$*,* $f(x, y) = g(x, y)h(y)$ *is* L_f*-PLC, where*

$$L_f(x, y) = M_g L_f(y) + M_f L_g(x, y).$$

Proof. Let $(x_1, y_1), (x_2, y_2) \in X \times Y$. Thus,

$$\begin{aligned}
&|f(x_1, y_1) - f(x_2, y_2)| \\
&= |g(x_1, y_1)h(y_1) - g(x_2, y_2)h(y_2)| \\
&\leqslant |g(x_1, y_1)|\,|h(y_1) - h(y_2)| \\
&\quad + |h(y_2)|\,|g(x_1, y_1) - g(x_2, y_2)| \\
&\leqslant M_g L_h(y_1) d_Y(y_1, y_2) \\
&\quad + M_h L_g(x_1, y_1) d_{XY}((x_1, y_1), (x_2, y_2)) \\
&\leqslant \big(M_g L_h(y_1) \\
&\quad + M_h L_g(x_1, y_1)\big) d_{XY}((x_1, y_1), (x_2, y_2)) \\
&= L_f(x_1, y_1) d_{XY}((x_1, y_1), (x_2, y_2)).
\end{aligned}$$

■

Assumption C.5. Policy is uniformly bounded with respect to the parameter space. That is, for every $(s, a) \in \mathcal{S} \times \mathcal{A}$, the constant $M^{s,a}_{\pi-\Theta}$ satisfies

$$\pi_\theta(a|s) \leqslant M^{s,a}_{\pi-\Theta},$$

for all $\theta \in \Theta$.

Assumption C.6. Future state distribution if uniformly bounded with respect to the parameter space. That is, for every $s \in \mathcal{S}$, the constant $M^s_{\delta-\Theta\mathcal{E}}$ satisfies

$$\delta_{\theta,\epsilon}(s) \leqslant M^s_{\delta-\Theta\mathcal{E}},$$

for all $(\theta, \epsilon) \in \Theta \times \mathcal{E}$.

Corollary C.4 (Lipschitz continuity of state-action distribution)**.** *Let* $\langle \mathfrak{T}, \Omega \rangle$ *be a Lipschitz task environment (Definition 4.2), and* Π_Θ *be a parametric space of Lipschitz policies (Definition 2.32). For any* $(s, a) \in \mathcal{S} \times \mathcal{A}$*, the function* $\zeta_{(\cdot,\cdot)}(s, a)$ *is* $L^{s,a}_{\zeta-\Theta\mathcal{E}}$*-Pointwise-Lipschitz-Continuous, where*

$$L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta_1, \epsilon_1) = M^s_{\delta-\Theta\mathcal{E}} L^{s,a}_{\pi-\Theta}(\theta_1, \epsilon_1) + M^{s,a}_{\pi-\Theta} L^s_{\delta-\Theta\mathcal{E}}(\theta_1, \epsilon_1).$$

Proof. Since $\zeta_{\theta,\epsilon}(s, a) = \delta_{\theta,\epsilon}(s)\pi_\theta(a|s)$, the proof follows from Lemma C.3, Lemma C.2, Assumption C.6, and Assumption C.5. ■

Assumption C.7. Transition model is uniformly bounded with respect to the parameter space. That is, for every $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the constant $M^{s,a,s'}_{\mathcal{P}-\mathcal{E}}$ satisfies

$$\mathcal{P}_\epsilon(s'|s, a) \leqslant M^{s,a,s'}_{\mathcal{P}-\mathcal{E}},$$

for all $\epsilon \in \mathcal{E}$. It also holds that $M^{s,s'}_{\pi-\mathcal{E}} = \sup_{a\in\mathcal{A}} M^{s,a,s'}_{\pi-\mathcal{E}}$ is finite.

Assumption C.8. State-action distribution is uniformly bounded with respect to the parameter space. That is, for every $(s, a) \in \mathcal{S} \times \mathcal{A}$, the constant $M^{s,a}_{\zeta-\Theta\mathcal{E}}$ satisfies

$$\zeta_{\theta,\epsilon}(s, a) \leqslant M^{s,a}_{\zeta-\Theta\mathcal{E}},$$

for all $(\theta, \epsilon) \in \Theta \times \mathcal{E}$.

Corollary C.5 (Lipschitz continuity of tuples distribution)**.** *Let $\langle \mathfrak{T}, \Omega\rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the function $d_{(\cdot,\cdot)}(s, a, s')$ is $L^{s,a,s'}_{d-\Theta\mathcal{E}}$-Pointwise-Lipschitz-Continuous, where*

$$L^{s,a,s'}_{d-\Theta\mathcal{E}}(\theta, \epsilon) = M^{s,a}_{\zeta-\Theta\mathcal{E}} L^{s,a,s'}_{\mathcal{P}-\mathcal{E}}(\epsilon) + M^{s,a,s'}_{\mathcal{P}-\mathcal{E}} L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta, \epsilon).$$

Proof. Since $d_{\theta,\epsilon}(s, a, s') = \zeta_{\theta,\epsilon}(s, a)\mathcal{P}_\epsilon(s'|s, a)$, the proof follows from Lemma C.3, Corollary C.4, Assumption C.8, and Assumption C.7. ■

Assumption C.9. Tuples distribution is uniformly bounded with respect to the parameter space. That is, for every $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the constant $M^{s,a,s'}_{d-\Theta\mathcal{E}}$ satisfies

$$d_{\theta,\epsilon}(a|s) \leqslant M^{s,a,s'}_{d-\Theta\mathcal{E}},$$

for all $(\theta, \epsilon) \in \Theta \times \mathcal{E}$.

Corollary C.6 (Lipschitz continuity of extended tuples distribution)**.** *Let $\langle \mathfrak{T}, \Omega\rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $(s, a, s', a') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A}$, the function $\overline{d}_{(\cdot,\cdot)}(s, a, s', a')$ is $L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}$-Pointwise-Lipschitz-Continuous, where*

$$L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta_1, \epsilon_1) = M^{s',a'}_{\pi-\Theta} L^{s,a,s'}_{d-\Theta\mathcal{E}}(\theta_1, \epsilon_1) + M^{s,a,s'}_{d-\Theta\mathcal{E}} L^{s',a'}_{\pi-\Theta}(\theta_1).$$

Proof. Since $\overline{d}_{\theta,\epsilon}(s, a, s', a') = d_{\theta,\epsilon}(s, a, s')\pi_\theta(a'|s')$, the proof follows from Lemma C.3, Corollary C.5, Assumption C.9, and Assumption C.5. ■

Observation C.1. *The above defined functions representing the Lipschitz constants are bounded with respect to the parameter spaces. This detail actually completes the proofs.*

C.2 LIPSCHITZ CONTINUITY OF THE MATRICES

This section provides a proof for the Lipschitz continuity of the matrices used by the LSTD algorithm with respect to the task and policy parameters. It proceeds by stating the necessary assumptions and proving the corresponding results.
Only the case of the V-function is proved; the results for the Q-function are easily obtained based on these ones.

Assumption C.10. The Lipschitz constant of the tuple distribution with respect to the task parameter space is bounded with respect to the state-action-sate space. That is,

$$L_{d-\Theta\mathcal{E}}(\theta,\epsilon) = \sup_{(s,a,s')\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}} L_{d-\Theta\mathcal{E}}^{s,a,s'}(\theta,\epsilon) < \infty,$$

for any $(\theta,\epsilon)\in\Theta\times\mathcal{E}$. Furthermore, $L_{d-\Theta\mathcal{E}}(\cdot,\cdot)$ is bounded.

Assumption C.11. The feature map $\boldsymbol{\phi}$ satisfies

$$\int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)(\phi_l(s)-\gamma\phi_l(s'))\,ds\,da\,s' < \infty$$

and

$$\int_{\mathcal{S}} |\phi_k(s)|(\delta_{\theta,\epsilon}(s)+1)\,ds < \infty$$

for every $k,l=1,...,d_c$ and $(\theta,\epsilon)\in\Theta\times\mathcal{E}$.

Assumption C.12. The Lipschitz constant of the reward model with respect to the task parameter space is bounded with respect to the state-action-sate space. That is,

$$L_{r-\Xi}(\xi) = \sup_{(s,a,s')\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}} L_{d-\Xi}^{s,a,s'}(\xi) < \infty,$$

for any $\xi\in\Xi$. Furthermore, $L_{r-\Xi}(\cdot)$ is bounded.

Lemma C.7 (Lipschitz continuity of the matrix $A_{(\cdot),\boldsymbol{\phi}}^{(V)}$)**.** *Let $\langle\mathfrak{T},\Omega\rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k,l=1,...,d_c$, the function $A_{(\cdot),\boldsymbol{\phi}}^{(V)-(k,l)}$ is $L_{A-\Theta\mathcal{E}}^{k,l}$-Pointwise-Lipschitz-Continuous, where*

$$L_{A-\Theta\mathcal{E}}^{k,l}(\epsilon,\theta) = L_{d-\Theta\mathcal{E}}(\epsilon,\theta)\int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)(\phi_l(s)-\gamma\phi_l(s'))|\,ds\,da\,s'.$$

Proof. Let $k,l=1,...,d_c$ be fixed, T_1,T_2 be tasks with parameters $(\epsilon_1,\xi_1),(\epsilon_2,\xi_2)\in\mathcal{E}\times\Xi$, respectively, and π_1,π_2 be policies with parameters $\theta_1,\theta_2\in\Theta$, respectively, for such tasks. Thus,

$$\left|A_{1,\boldsymbol{\phi}}^{(V)-(k,l)} - A_{2,\boldsymbol{\phi}}^{(V)-(k,l)}\right|$$

$$
\begin{aligned}
&= \Big| \int_{\mathcal{S}\mathcal{A}\mathcal{S}} (\phi_k(s)(\phi_l(s) - \gamma\phi_l(s')) d_1(s,a,s') \\
&- \phi_k(s)(\phi_l(s) - \gamma\phi_l(s')) d_2(s,a,s'))\, ds a s' \Big| \\
&= \Big| \int_{\mathcal{S}\mathcal{A}\mathcal{S}} \phi_k(s)(\phi_l(s) - \gamma\phi_l(s'))(d_1(s,a,s') - d_2(s,a,s'))\, ds a s' \Big| \\
&\leqslant \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)(\phi_l(s) - \gamma\phi_l(s'))|\, |d_1(s,a,s') - d_2(s,a,s')|\, ds a s' \\
&\leqslant \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)(\phi_l(s) - \gamma\phi_l(s'))| L_{d-\Theta\mathcal{E}}^{s,a,s'}(\epsilon_1,\theta_1) d_{\Theta,\mathcal{E}}((\epsilon_1,\theta_1),(\epsilon_2,\theta_2))\, ds a s' \\
&\leqslant L_{d-\Theta\mathcal{E}}(\epsilon_1,\theta_1) d_{\Theta,\mathcal{E}}((\epsilon_1,\theta_1),(\epsilon_2,\theta_2)) \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)(\phi_l(s) - \gamma\phi_l(s'))|\, ds a s' \\
&= L_{A-\Theta\mathcal{E}}^{k,l}(\epsilon_1,\theta_1) d_{\Theta,\mathcal{E}}((\epsilon_1,\theta_1),(\epsilon_2,\theta_2)).
\end{aligned}
$$

■

Lemma C.8 (Lipschitz continuity of the vector $b_{(\cdot),\boldsymbol{\Phi}}^{(V)}$)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k = 1, ..., d_c$, the function $b_{(\cdot),\boldsymbol{\Phi}}^{(V)-(k)}$ is $L_{b-\Theta\mathcal{E}}^k$-Pointwise-Lipschitz-Continuous, with*

$$
\begin{aligned}
L_{b-\Theta,\mathcal{E},\Xi}^k(\epsilon,\theta,\xi) = &\Big(R\, L_{d-\Theta\mathcal{E}}(\theta_1,\epsilon_1)\lambda(\mathcal{A})\lambda(\mathcal{S}) \\
&+ L_{r-\xi}(\xi_1) \Big) \int_{\mathcal{S}} |\phi_k(s)|(\delta_2(s) + 1)\, ds,
\end{aligned}
$$

where $R = \max\{|R_1|, |R_2|\}$ is a bound for the reward, and $\lambda(\mathcal{S})$, $\lambda(\mathcal{A})$ are the volumes of $\mathcal{S}$, $\mathcal{A}$, respectively.

Proof. Let $k = 1, ..., d_c$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$
\begin{aligned}
&\big| b_{1,\boldsymbol{\Phi}}^{(V)-(k)} - b_{2,\boldsymbol{\Phi}}^{(V)-(k)} \big| \\
&= \Big| \int_{\mathcal{S}\mathcal{A}\mathcal{S}} (\phi_k(s) r_1(s,a,s') d_1(s,a,s') - \phi_k(s) r_2(s,a,s') d_2(s,a,s'))\, ds a s' \Big| \\
&\leqslant \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)|\, |r_1(s,a,s') d_1(s,a,s') - r_2(s,a,s') d_2(s,a,s')|\, ds a s' \\
&\leqslant \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)|\, |r_1(s,a,s')|\, |d_1(s,a,s') - d_2(s,a,s')|\, ds a s' \\
&+ \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)| d_2(s,a,s') |r_1(s,a,s') - r_2(s,a,s')|\, ds a s' \\
&\leqslant R\, L_{d-\Theta\mathcal{E}}(\theta_1,\epsilon_1) d_{\Theta,\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)|\, ds a s' \\
&+ \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)| d_2(s,a,s') L_{r-\xi}^{s,a,s'}(\xi_1) d_\Xi(\xi_1,\xi_2)\, ds a s' \\
&\leqslant R\, L_{d-\Theta\mathcal{E}}(\theta_1,\epsilon_1) d_{\Theta,\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2)) \lambda(\mathcal{A})\lambda(\mathcal{S}) \int_{\mathcal{S}} |\phi_k(s)|\, ds
\end{aligned}
$$

$$
\begin{aligned}
&+ L_{r-\xi}(\xi_1) d_{\Xi}(\xi_1, \xi_2) \int_{\mathcal{S}\mathcal{A}\mathcal{S}} |\phi_k(s)| d_2(s, a, s')\, ds a s' \\
&\leqslant \Bigg(R\, L_{d-\Theta\mathcal{E}}(\theta_1, \epsilon_1) \lambda(\mathcal{A}) \lambda(\mathcal{S}) \\
&\qquad + L_{r-\xi}(\xi_1) \Bigg) \int_{\mathcal{S}} |\phi_k(s)| (\delta_2(s) + 1)\, ds d_{\Theta,\mathcal{E},\Xi}((\theta_1, \epsilon_1, \xi_1), (\theta_2, \epsilon_2, \xi_1)) \\
&= L^k_{b-\Theta,\mathcal{E},\Xi}(\theta_1, \epsilon_1, \xi_1) d_{\Theta,\mathcal{E},\Xi}((\theta_1, \epsilon_1, \xi_1), (\theta_2, \epsilon_2, \xi_1)).
\end{aligned}
$$

■

Assumption C.13. The Lipschitz constant of the extended tuple distribution with respect to the task parameter space is bounded with respect to the state-action-sate-action space. That is,

$$
L_{\overline{d}-\Theta\mathcal{E}}(\theta, \epsilon) = \sup_{(s,a,s',a') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A}} L^{s,a,s',a'}_{\overline{d}-\Theta\mathcal{E}}(\theta, \epsilon) < \infty,
$$

for any $(\theta, \epsilon) \in \Theta \times \mathcal{E}$. Furthermore, $L_{\overline{d}-\Theta\mathcal{E}}(\cdot, \cdot)$ is bounded.

Assumption C.14. The feature map $\boldsymbol{\varphi}$ satisfies

$$
\int_{\mathcal{S}\mathcal{A}\mathcal{S}\mathcal{A}} |\varphi_k(s, a)(\phi_l(s, a) - \gamma \varphi_l(s', a'))\, ds a s' a' < \infty
$$

and

$$
\int_{\mathcal{S}\mathcal{A}} |\varphi_k(s, a)| (\zeta_{\theta,\epsilon}(s, a) + 1)\, ds a < \infty
$$

for every $k, l = 1, ..., d_c$ and $(\theta, \epsilon) \in \Theta \times \mathcal{E}$.

Lemma C.9 (Lipschitz continuity of the matrix $A^{(Q)}_{(\cdot),\boldsymbol{\varphi}}$)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k, l = 1, ..., d_c$, the function $A^{(Q)-(k,l)}_{(\cdot),\boldsymbol{\varphi}}$ is $L^{k,l}_{A-\Theta\mathcal{E}}$-Pointwise-Lipschitz-Continuous, where*

$$
L^{k,l}_{A-\Theta\mathcal{E}}(\epsilon, \theta) = L_{d-\Theta\mathcal{E}}(\epsilon, \theta) \int_{\mathcal{S}\mathcal{A}\mathcal{S}\mathcal{A}} |\varphi_k(s, a)(\varphi_l(s, a) - \gamma \varphi_l(s', a'))|\, ds a s'.
$$

Lemma C.10 (Lipschitz continuity of the vector $b^{(Q)}_{(\cdot),\boldsymbol{\varphi}}$)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k = 1, ..., d_c$, the function $b^{(Q)-(k)}_{(\cdot),\boldsymbol{\varphi}}$ is $L^k_{b-\Theta\mathcal{E}}$-Pointwise-Lipschitz-Continuous, with*

$$
\begin{aligned}
L^k_{b-\Theta,\mathcal{E},\Xi}(\epsilon, \theta, \xi) = &\Bigg(R\, L_{\overline{d}-\Theta\mathcal{E}}(\theta_1, \epsilon_1) \lambda(\mathcal{S}) \lambda(\mathcal{A}) \\
&+ L_{r-\xi}(\xi_1) \Bigg) \int_{\mathcal{S}\mathcal{A}} |\varphi_k(s, a)| (\zeta_2(s, a) + 1)\, ds a,
\end{aligned}
$$

where $R = \max\{|R_1|, |R_2|\}$ is a bound for the rewrad, and $\lambda(\mathcal{S})$, $\lambda(\mathcal{A})$ are the volumes of $\mathcal{S}$, $\mathcal{A}$, respectively.

Observation C.2. *The above defined functions representing the Lipschitz constants are bounded with respect to the parameter spaces. This detail actually completes the proofs.*

C.3 LIPSCHITZ CONTINUITY OF THE POLICY PERFORMANCE

This section presents the proof of the Lipschitz continuity of the policy performance with respect to the task and policy parameters. It proceeds by stating the necessary assumptions and then proving the result.

Assumption C.15. The Lipschitz constant of the transition model with respect to the parameter space is bounded with respect to the ending state. That is,

$$L_{\mathcal{P}-\mathcal{E}}^{s,a}(\epsilon) = \sup_{s' \mathcal{S}} L_{\mathcal{P}-\mathcal{E}}(\epsilon)^{s,a,s'} < \infty,$$

for any $\epsilon \in \mathcal{E}$. Furthermore, $L_{\mathcal{P}-\mathcal{E}}(\dot{})^{s,a}$ is bounded.

Assumption C.16. The Lipschitz constant of the reward model with respect to the parameter space is bounded with respect to the ending state. That is,

$$L_{r-\Xi}^{s,a}(\xi) = \sup_{s' \mathcal{S}} L_{r-\Xi}(\xi)^{s,a,s'} < \infty,$$

for any $\xi \in \Xi$. Furthermore, $L_{r-\Xi}(\cdot)^{s,a}$ is bounded.

Lemma C.11 (Lipschitz continuity of the immediate reward)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). Then the function $\mathcal{R}_{(\cdot,\cdot)}(s,a)$ is $L_{\mathcal{R}-\mathcal{E}\Xi}^{s,a}$-PLC, where*

$$L_{\mathcal{R}-\mathcal{E}\Xi}^{s,a}(\epsilon,\xi) = R\, L_{\mathcal{P}-\mathcal{E}}^{s,a}(\epsilon)\lambda(\mathcal{S}) + L_{r-\Xi}^{s,a}(\xi).$$

Proof. Let T_1, T_2 be tasks with parameters $(\epsilon_1,\xi_1),(\epsilon_2,\xi_2) \in \mathcal{E}\times\Xi$, respectively. Let $(s,a) \in \mathcal{S}\times\mathcal{A}$ be fixed. Thus,

$$\begin{aligned}
&|\mathcal{R}_1(s,a) - \mathcal{R}_2(s,a)| \\
&= \left| \int_{\mathcal{S}} (r_1(s,a,s')\mathcal{P}_1(s'|s,a) - r_2(s,a,s')\mathcal{P}_2(s'|s,a))\, ds' \right| \\
&\leqslant \int_{\mathcal{S}} |r_1(s,a,s')|\, |\mathcal{P}_1(s'|s,a) - \mathcal{P}_2(s'|s,a)|\, ds' \\
&+ \int_{\mathcal{S}} \mathcal{P}_2(s'|s,a)|r_1(s,a,s') - r_2(s,a,s')|\, ds' \\
&\leqslant R \int_{\mathcal{S}} L_{\mathcal{P}-\mathcal{E}}^{s,a,s'}(\epsilon_1) d_{\mathcal{E}}(\epsilon_1,\epsilon_2)\, ds' \\
&+ \int_{\mathcal{S}} |\mathcal{P}_2(s'|s,a) L_{r-\Xi}^{s,a,s'}(\xi_1) d_{\Xi}(\xi_1,\xi_2)\, ds' \\
&\leqslant R\, L_{\mathcal{P}-\mathcal{E}}^{s,a}(\epsilon_1) d_{\mathcal{E}}(\epsilon_1,\epsilon_2)\lambda(\mathcal{S}) \\
&+ L_{r-\Xi}^{s,a}(\xi_1) d_{\Xi}(\xi_1,\xi_2) \\
&\leqslant L_{\mathcal{R}-\mathcal{E}\Xi}^{s,a}(\epsilon_1,\xi_1) d_{\mathcal{E},\Xi}((\epsilon_1,\xi_1),(\epsilon_2,\xi_2)).
\end{aligned}$$

■

Assumption C.17. The Lipschitz constant of the state-actions distribution with respect to the parameter space is bounded with respect to the state-action space. That is,

$$L_{\zeta-\Theta\mathcal{E}}(\theta,\epsilon) = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta,\epsilon) < \infty,$$

for any $(\theta,\epsilon)\in(\Theta\times\mathcal{E})$. Furthermore, $L_{\zeta-\Theta\mathcal{E}}(\cdot,\cdot)$ is bounded.

Assumption C.18. The Lipschitz constant of the expected immediate reward with respect to the parameter space is bounded with respect to the state-action space. That is,

$$L_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi) = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi) < \infty,$$

for any $(\epsilon,\xi)\in\mathcal{E}\times\Xi$. Furthermore, $L_{\mathcal{R}-\mathcal{E}\Xi}(\cdot,\cdot)$ is bounded.

Lemma C.12 (Lipschitz continuity of the policy performance)**.** *Let $\langle\mathfrak{T},\Omega\rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). Then for every the policy performance $J_{(\cdot,\cdot,\cdot)}$ is $L_{J-\Theta\mathcal{E}\Xi}$-PLC, with*

$$\begin{aligned}L_{J-\Theta\mathcal{E}\Xi}(\theta,\epsilon,\xi) =& \frac{1}{1-\gamma} R\, L_{\zeta-\Theta\mathcal{E}}(\theta,\epsilon)\lambda(\mathcal{S})\lambda(\mathcal{A}) \\ &+ \frac{1}{1-\gamma} L_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi),\end{aligned}$$

where $R = \max\{|R_1|,|R_2|\}$, and $\lambda(\mathcal{S}),\lambda(\mathcal{A})$ are the volumes of $\mathcal{S},\mathcal{A}$, respectively.

Proof. Let T_1, T_2 be tasks with parameters $(\epsilon_1,\xi_1),(\epsilon_2,\xi_2)\in\mathcal{E}\times\Xi$, respectively, and π_1,π_2 be policies with parameters $\theta_1,\theta_2\in\Theta$, respectively, for such tasks. Thus,

$$\begin{aligned}&|J_1 - J_2| \\ &= \left|\frac{1}{1-\gamma}\int_{\mathcal{S}\mathcal{A}} \big(\mathcal{R}_1(s,a)\zeta_1(s,a) - \mathcal{R}_2(s,a)\zeta_2(s,a)\big)\,ds da\right| \\ &\leqslant \frac{1}{1-\gamma}\int_{\mathcal{S}\mathcal{A}} |\mathcal{R}_1(s,a)|\,|\zeta_1(s,a)-\zeta_2(s,a)|\,ds da \\ &+ \frac{1}{1-\gamma}\int_{\mathcal{S}\mathcal{A}} \zeta_2(s,a)\,|\mathcal{R}_1(s,a)-\mathcal{R}_2(s,a)|\,ds da \\ &\leqslant \frac{1}{1-\gamma} R\int_{\mathcal{S}\mathcal{A}} L^{s,a}_{\zeta-\Theta\mathcal{E}}(\theta,\epsilon_1)\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\,ds da \\ &+ \frac{1}{1-\gamma}\int_{\mathcal{S}\mathcal{A}} \zeta_1(s,a) L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon_1,\xi_1)\,d_{\mathcal{E}\Xi}((\epsilon_1,\xi_1),(\epsilon_2,\xi_2))\,ds da \\ &\leqslant \frac{1}{1-\gamma} R\, L_{\zeta-\Theta\mathcal{E}}(\theta_1,\epsilon_1)\,d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\lambda(\mathcal{S})\lambda(\mathcal{A}) \\ &+ \frac{1}{1-\gamma} L_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon_1,\xi_1)\,d_{\mathcal{E}\Xi}((\epsilon_1,\xi_1),(\epsilon_2,\xi_2)) \\ &\leqslant L_{J-\Theta\mathcal{E}\Xi}(\theta_1,\epsilon_1,\xi_1)\,d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2)).\end{aligned}$$

■

Observation C.3. *The above defined functions representing the Lipschitz constants are bounded with respect to the parameter spaces. This detail actually completes the proofs.*

C.4 LIPSCHITZ CONTINUITY OF THE PERFORMANCE GRADIENT

This section presents the proof of the Lipschitz continuity of the gradient with respect to the task and policy parameters. It proceeds by stating the necessary assumptions and incrementally proving the result in a sequence of lemmata.
Recall Assumption C.2, and further assume:

Assumption C.19. The Lipschitz constant of the expected immediate reward with respect to the parameter space is bounded with respect to the action space. That is,

$$L^s_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi) = \sup_{a\in\mathcal{A}} L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi) < \infty,$$

for any $(\epsilon,\xi) \in \mathcal{E}\times\Xi$. Furthermore, $L^s_{\mathcal{R}-\mathcal{E}\Xi}(\cdot,\cdot)$ is bounded.

Assumption C.20. The Lipschitz constant of the state-to-state distribution with respect to the parameter space is bounded with respect to the ending state. That is,

$$L^s_{\mathcal{P}-\Theta\mathcal{E}}(\theta,\epsilon) = \sup_{s'\in\mathcal{A}} L^{s,s'}_{\mathcal{P}-\Theta\mathcal{E}}(\theta,\epsilon) < \infty,$$

for any $(\theta,\epsilon) \in \Theta\times\mathcal{E}$. Furthermore, $L^s_{\mathcal{P}-\Theta\mathcal{E}}(\cdot,\cdot)$ is also bounded.

Assumption C.21. The function

$$\begin{aligned} L^s_{V-\Theta\mathcal{E}\Xi}(\theta,\epsilon,\xi) = \; & K^s(\theta,\epsilon) + \gamma\int_{\mathcal{S}} \mathcal{P}_1(s'|s)K^s(\theta,\epsilon,\xi)\,ds \\ & + \gamma^2\int_{\mathcal{S}^2} \mathcal{P}_1(s'|s)\mathcal{P}_1(s''|s')\; K^{s'}(\theta,\epsilon,\xi) + \ldots, \end{aligned}$$

where

$$\begin{aligned} K^s(\theta,\epsilon,\xi) = & R\; L^s_{\pi-\Theta}(\theta)\lambda(\mathcal{A}) + L^s_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon,\xi) \\ & + \gamma\frac{R}{1-\gamma}L^s_{\mathcal{P}-\Theta\mathcal{E}}(\theta,\epsilon), \end{aligned}$$

is well-defined for every $s\in\mathcal{S}$, $\theta\in\Theta$, $\epsilon\in\mathcal{E}$ and $\xi\in\Xi$.

Lemma C.13 (Lipschitz continuity of the V-function)**.** *Let $\langle\mathfrak{T},\Omega\rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $s\in\mathcal{S}$, the function $V_{(\cdot)}(s)$ is $L^s_{V-\Theta\mathcal{E}\Xi}$-Pointwise-Lipschitz-Continuous, where $L^s_{V-\Theta\mathcal{E}\Xi}$ is the function defined in Assumption C.21.*

Proof. Let $s \in \mathcal{S} \times \mathcal{A}$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$
\begin{aligned}
& |V_1(s) - V_2(s)| \\
= \; & \Big| \int_{\mathcal{A}} \mathcal{R}_1(s,a)\pi_1(a|s)\,da + \gamma \int_{\mathcal{S}} V_1(s')\mathcal{P}_1(s'|s)\,ds' \\
& - \int_{\mathcal{A}} \mathcal{R}_2(s,a)\pi_2(a|s)\,da - \gamma \int_{\mathcal{S}} V_2(s')\mathcal{P}_2(s'|s)\,ds' \Big| \\
\leqslant \; & \int_{\mathcal{A}} |\mathcal{R}_1(s,a)\pi_1(a|s) - \mathcal{R}_2(s,a)\pi_2(a|s)|\,da \\
& + \gamma \int_{\mathcal{S}} |V_1(s')\mathcal{P}_1(s'|s) - V_2(s')\mathcal{P}_2(s'|s)|\,ds' \\
\leqslant \; & \int_{\mathcal{A}} |\mathcal{R}_1(s,a)|\,|\pi_1(a|s) - \pi_2(a|s)|\,da \\
& + \int_{\mathcal{A}} \pi_2(a|s)|\mathcal{R}_1(s,a) - \mathcal{R}_2(s,a)|\,da \\
& + \gamma \int_{\mathcal{S}} |V_1(s')|\,|\mathcal{P}_1(s'|s) - \mathcal{P}_2(s'|s)|\,ds' \\
& + \gamma \int_{\mathcal{S}} \mathcal{P}_2(s'|s)|V_1(s') - V_2(s')|\,ds' \\
\leqslant \; & R\, L^s_{\pi-\Theta}(\theta_1) d_\Theta(\theta_1,\theta_2)\lambda(\mathcal{A}) \\
& + L^s_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon_1,\xi_1) d_{\mathcal{E}\Xi}((\epsilon_1,\xi_1),(\epsilon_2,\xi_2)) \\
& + \gamma \frac{R}{1-\gamma} L^s_{\mathcal{P}-\Theta\mathcal{E}}(\theta_1,\epsilon_1) d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\lambda(\mathcal{S}) \\
& + \gamma \int_{\mathcal{S}} \mathcal{P}_2(s'|s)|V_1(s') - V_2(s')|\,ds' \\
\leqslant \; & K^s(\theta_1,\epsilon_1,\xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2)) \\
& + \gamma \int_{\mathcal{S}} \mathcal{P}_2(s'|s)|V_1(s') - V_2(s')|\,ds' \\
\leqslant \; & K^s(\theta_1,\epsilon_1,\xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2)) \\
& + \gamma \int_{\mathcal{S}} \mathcal{P}_2(s'|s) K^{s'}(\theta_1,\epsilon_1,\xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2))\,ds' \\
& + \gamma^2 \int_{\mathcal{S}^2} \mathcal{P}_2(s'|s)\mathcal{P}_2(s''|s') K^{s''}(\theta_1,\epsilon_1,\xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2))\,ds'' \\
& + \dots \\
= \; & L_{V-\Theta\mathcal{E}\Xi}(\theta_1,\epsilon_1,\xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2)).
\end{aligned}
$$

■

Recall Assumption C.15, and further assume:

Assumption C.22. The Lipschitz constant of the V-function with respect to the task parameter space is bounded with respect to the state space. That is,

$$L_{V-\Theta\mathcal{E}\Xi}(\theta,\epsilon,\xi) = \sup_{s\in\mathcal{S}} L^s_{V-\Theta\mathcal{E}\Xi}(\theta,\epsilon,\xi) < \infty,$$

for any $(\theta, \epsilon, \xi) \in (\Theta \times \mathcal{E} \times \Xi)$. Further more, $L_{V-\Theta\mathcal{E}\Xi}(\cdot,\cdot,\cdot)$ is also bounded.

Lemma C.14 (Lipschitz continuity of the Q-function)**.** *Let $\langle \mathcal{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $(s, a) \in \mathcal{S} \times \mathcal{A}$, the function $Q_{(\cdot)}(s, a)$ is $L^{s,a}_{Q-\Theta\mathcal{E}\Xi}$-Pointwise-Lipschitz-Continuous, where*

$$\begin{aligned} L^{s,a}_{Q-\Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi) =& L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon, \xi) + \gamma \frac{R}{1-\gamma} L^{s,a}_{\mathcal{P}-\mathcal{E}}(\epsilon) \\ &+ \gamma L_{V-\Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi). \end{aligned}$$

Proof. Let $(s, a) \in \mathcal{S} \times \mathcal{A}$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$\begin{aligned} & |Q_1(s, a) - Q_2(s, a)| \\ = \;& \Big| \mathcal{R}_1(s, a) + \gamma \int_{\mathcal{S}} V_1(s') \mathcal{P}_1(s'|s, a)\, ds' \\ & - \mathcal{R}_2(s, a) - \gamma \int_{\mathcal{S}} V_2(s') \mathcal{P}_2(s'|s, a)\, ds' \Big| \\ \leqslant \;& |\mathcal{R}_1(s, a) - \mathcal{R}_2(s, a)| \\ & + \gamma \int_{\mathcal{S}} |V_1(s') \mathcal{P}_1(s'|s, a) - V_2(s') \mathcal{P}_2(s'|s, a)|\, ds' \\ \leqslant \;& L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon_1, \xi_1) d_{\mathcal{E}\Xi}((\epsilon_1, \xi_1), (\epsilon_2, \xi_2)) \\ & + \gamma \int_{\mathcal{S}} |V_1(s')|\, |\mathcal{P}_1(s'|s, a) - \mathcal{P}_2(s'|s, a)|\, ds' \\ & + \gamma \int_{\mathcal{S}} \mathcal{P}_2(s'|s, a) |V_1(s') - V_2(s')|\, ds' \\ \leqslant \;& L^{s,a}_{\mathcal{R}-\mathcal{E}\Xi}(\epsilon_1, \xi_1) d_{\mathcal{E}\Xi}((\epsilon_1, \xi_1), (\epsilon_2, \xi_2)) \\ & + \gamma \frac{R}{1-\gamma} L^{s,a}_{\mathcal{P}-\mathcal{E}}(\epsilon_1) d_{\mathcal{E}}(\epsilon_1, \epsilon_2) \lambda(\mathcal{S}) \\ & + \gamma L_{V-\Theta\mathcal{E}\Xi}(\theta_1, \epsilon_1, \xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1, \epsilon_1, \xi_1), (\theta_2, \epsilon_2, \xi_2)) \\ \leqslant \;& L^{s,a}_{Q-\Theta\mathcal{E}\Xi}(\theta_1, \epsilon_1, \xi_1) d_{\Theta\mathcal{E}\Xi}((\theta_1, \epsilon_1, \xi_1), (\theta_2, \epsilon_2, \xi_2)). \end{aligned}$$

■

Assumption C.23. The log-policy gradient is uniformly bounded with respect to the parameter space. That is, for every $k = 1, ..., d_a$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$, the constant $M^{k,s,a}_{\nabla \log \pi}$ satisfies

$$\nabla_{\theta_k} \log \pi_\theta(a|s) \leqslant M_{\nabla \log \pi} k, s, a,$$

for any $\theta \in \Theta$. It also holds that $M_{\nabla \log \pi} k = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} M_{\nabla \log \pi} k, s, a$ is finite.

Assumption C.24. The log-policy gradient is PLC. This means that, whenever $k = 1, ..., n_a$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$ are fixed, the function $\nabla_{\theta_k} \log \pi_{(\cdot)}(a|s)$ is $L^{k,s,a}_{\nabla \log \pi}$-PLC.

Lemma C.15 (Lipschitz continuity of $Q_{(\cdot)}\nabla \log \pi_{(\cdot)}$)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k = 1, ..., d_a$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$, the function $Q_{(\cdot)}(s, a)\nabla_{\theta_k} \log \pi_{(\cdot)}(a|s)$ is $L^{k,s,a}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}$-Pointwise-Lipschitz-Continuous, where*

$$L^{k,s,a}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi) = \frac{1}{1-\gamma} R\, L^{k,s,a}_{\nabla \pi - \Theta}(\theta) + M^{k,s,a}_{\nabla \log \pi} L^{s,a}_{Q - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi).$$

Proof. It follows directly from Lemma C.3, Assumption C.23, Assumption C.24, and Lemma C.14. ∎

Assumption C.25. The Lipschitz constant of the state-action distribution with respect to the parameter space is bounded with respect to the state-action space. That is,

$$L_{\zeta - \Theta\mathcal{E}}(\theta, \epsilon) = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} L^{s,a}_{\zeta - \Theta\mathcal{E}}(\theta, \epsilon) < \infty,$$

for any $(\theta, \epsilon) \in (\Theta \times \mathcal{E})$. Furthermore, $L_{\zeta - \Theta\mathcal{E}}(\cdot, \cdot)$ is bounded.

Assumption C.26. The Lipschitz constant of the Q-function times log-policy gradient is bounded with respect to the state-action space. That is,

$$L^{k}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi) = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} L^{k,s,a}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi) < \infty,$$

for any $(\theta, \epsilon, \xi) \in (\Theta \times \mathcal{E} \times \Xi)$. Furthermore, $L^{k}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}(\cdot, \cdot, \cdot)$ is bounded.

Lemma C.16 (Lipschitz continuity of the gradient $\nabla_\theta J_{(\cdot)}$)**.** *Let $\langle \mathfrak{T}, \Omega \rangle$ be a Lipschitz task environment (Definition 4.2), and Π_Θ be a parametric space of Lipschitz policies (Definition 2.32). For any $k = 1, ..., d_a$, the function $\nabla_\theta J_{(\cdot)}$ is $L^{k}_{\nabla J - \Theta\mathcal{E}\Xi}$-Pointwise-Lipschitz-Continuous, with*

$$L_{\nabla J - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi) = \frac{R}{(1-\gamma)^2} M^{k}_{\nabla \log \pi} L_{\zeta - \Theta\mathcal{E}}(\theta, \epsilon)\lambda(\mathcal{S})\lambda(\mathcal{A}) + \frac{1}{1-\gamma} L^{k}_{Q\nabla \log \pi - \Theta\mathcal{E}\Xi}(\theta, \epsilon, \xi),$$

where $\lambda(\mathcal{S}), \lambda(\mathcal{A})$ are the volumes of $\mathcal{S}, \mathcal{A}$, respectively.

Proof. Let $k = 1, ..., d_a$ be fixed, T_1, T_2 be tasks with parameters $(\epsilon_1, \xi_1), (\epsilon_2, \xi_2) \in \mathcal{E} \times \Xi$, respectively, and π_1, π_2 be policies with parameters $\theta_1, \theta_2 \in \Theta$, respectively, for such tasks. Thus,

$$\begin{aligned} &\left|\nabla_{\theta_k} J_1 - \nabla_{\theta_k} J_2\right| \\ &= \left|\frac{1}{1-\gamma} \int_{\mathcal{S}\mathcal{A}} (Q_1(s, a)\nabla_{\theta_k} \log \pi_1(a|s)\zeta_1(s, a) - \right. \\ &\qquad \left. Q_2(s, a)\nabla_{\theta_k} \log \pi_2(a|s)\zeta_2(s, a))\right| \end{aligned}$$

$$
\begin{aligned}
&\leqslant \frac{1}{1-\gamma}\int_{\mathcal{SA}} |Q_1(s,a)\nabla_{\theta_k}\log\pi_1(a|s)|\,|\zeta_1(s,a)-\zeta_2(s,a)|\,\mathrm{d}sa\\
&+\frac{1}{1-\gamma}\int_{\mathcal{SA}} \zeta_2(s,a)|Q_1(s,a)\nabla_{\theta_k}\log\pi_1(a|s)-\\
&\qquad\qquad Q_2(s,a)\nabla_{\theta_k}\log\pi_2(a|s)|\,\mathrm{d}sa\\
&\leqslant \frac{1}{1-\gamma}\frac{R}{1-\gamma}M^k_{\nabla\log\pi}L_{\zeta-\Theta\mathcal{E}}(\theta_1,\epsilon_1)\mathrm{d}_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_2,\epsilon_2))\lambda(\mathcal{S})\lambda(\mathcal{A})\\
&+\frac{1}{1-\gamma}L^k_{Q\nabla\log\pi-\Theta\mathcal{E}\Xi}(\theta_1,\epsilon_1,\xi_1)\mathrm{d}_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2))\\
&\leqslant L^k_{\nabla J-\Theta\mathcal{E}\Xi}(\theta_1,\epsilon_1,\xi_1)\mathrm{d}_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_2,\epsilon_2,\xi_2)).
\end{aligned}
$$

■

Observation C.4. *The above defined functions representing the Lipschitz constants are bounded with respect to the parameter spaces. This detail actually completes the proofs.*

D

OTHER DERIVATIONS

This chapter presents the remaining important derivations omitted along the document.

D.1 LOCAL LIPSCHITZ CONTINUITY AND KANTOROVICH LIPSCHITZ CONTINUITY

This section shows that the stronger notion of Lipschitz continuity for MDPs presented here (Definition 2.31) implies the one presented in Pirotta, Restelli, and Bascetta (2015). It suffices to proof the following lemma:

Lemma D.1. *Let $X \subset \mathbb{R}^n$ be a bounded metric space with the Euclidean distance and a measurable space with its Borel σ-algebra.*
Let $(P(\cdot|\alpha,\beta))_{\alpha\in\mathcal{A}\ \beta\in\mathcal{B}}$ be a parametric family of probability distributions on X, where $\mathcal{A}$ and $\mathcal{B}$ are the parameter spaces. Assume that, whenever $\beta \in \mathcal{B}$ is fixed, for every $x \in X$, the function $P(x|\cdot,\beta)$ is $L_{\mathcal{A}}^{\beta,x}$-Pointwise-Lipschitz-Continuous. Furthermore, assume that $L_{\mathcal{A}}^{\beta}(\alpha) = \sup_{x\in X} L_{\mathcal{A}}^{\beta,x}(\alpha)$ if finite for every α.
Then, for $p \sim P(\cdot|\alpha_1,\beta)$ and $q \sim P(\cdot|\alpha_2,\beta)$, it holds that

$$\mathcal{K}(p,q) \leqslant \mathcal{L}_{\mathcal{A}}^{\beta}(\alpha_1)\, d_{\mathcal{A}}(\alpha_1,\alpha_2),$$

where $\mathcal{L}_{\mathcal{A}}^{\beta}(\alpha_1) = \frac{1}{2}\mathrm{diam}(X) L_{\mathcal{A}}^{\beta} \lambda(X)$ and $\lambda(X)$ is the volume of X.

Proof. Let TV be the total variation distance. Thus,

$$\begin{aligned}
\mathcal{K}(p,q) &\leqslant \mathrm{diam}(X)\mathrm{TV}(p,q) \\
&= \mathrm{diam}(X) \cdot \frac{1}{2} \max_{|h|\leqslant 1} \left| \int_S h(x)(p(x|\alpha_1,\beta) - q(x|\alpha_2,\beta))\, dx \right| \\
&\leqslant \frac{1}{2}\mathrm{diam}(X) \max_{|h|\leqslant 1} \int_S |h(x)| \cdot |p(x|\alpha_1,\beta) - q(x|\alpha_2,\beta)|\, dx \\
&\leqslant \frac{1}{2}\mathrm{diam}(X) \max_{|h|\leqslant 1} \int_S |h(x)| L_{\mathcal{A}}^{\beta,x}(\alpha_1)\, d_{\mathcal{A}}(\alpha_1,\alpha_2)\, dx \\
&\leqslant \frac{1}{2}\mathrm{diam}(X) d_{\mathcal{A}}(\alpha_1,\alpha_2) \max_{|h|\leqslant 1} \int_S |h(x)| L_{\mathcal{A}}^{\beta}(\alpha_1)\, dx \\
&\leqslant \frac{1}{2}\mathrm{diam}(X) d_{\mathcal{A}}(\alpha_1,\alpha_2) L_{\mathcal{A}}^{\beta}(\alpha_1) \max_{|h|\leqslant 1} \int_S |h(x)|\, dx \\
&\leqslant \frac{1}{2}\mathrm{diam}(X) d_{\mathcal{A}}(\alpha_1,\alpha_2) L_{\mathcal{A}}^{\beta}(\alpha_1) \lambda(X) \\
&= \mathcal{L}_{\mathcal{A}}^{\beta}(\alpha_1)\, d_{\mathcal{A}}(\alpha_1,\alpha_2).
\end{aligned}$$

A proof for the first inequality can be found in (Gibbs and Su, 2002). ∎

This result can be applied, for example, to the transition model, by fixing the parameter model ϵ and taking $\mathcal{A}$ as $\mathcal{SA}$, resulting in

$$\mathcal{K}(\mathcal{P}(\cdot|s_1,a_1),\mathcal{P}(\cdot|s_2,a_2)) \leqslant \mathcal{L}_{\mathcal{SA}}(s_1,a_1)\mathrm{d}_{\mathcal{SA}}((s_1,a_1),(s_2,a_2)).$$

D.2 ON THE OBJECTIVE FUNCTIONS

This section presents the derivation of the alternative forms of the objective function considered by the optimistic (Section 4.5) and the min-max (Section 4.6) approaches. Without loss of generality, only the cases of the actor are proved; the proofs for the critic can be easily derived based on these.

Lemma D.2. *For the optimistic implementation of the actor, the objective function can be rewritten as*

$$\begin{aligned}
&g(\omega) = \\
&\sum_{k=1}^{d_a}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j(\nabla_{\theta_k} J_1 \\
&\qquad - \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1^{(k)}}(s,a)]\Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1}^{(k)}(s,a)] \\
&+ \Big(\frac{1}{n^2}-\frac{1}{n_1^2}\Big)\frac{n_1}{(1-\gamma)^2}\mathop{\mathrm{Var}}_{(s,a)\sim\zeta_1}[\widehat{\eta_1}^{(k)}(s,a)]\Bigg).
\end{aligned}$$

Proof.

$$\begin{aligned}
&g(\omega) \\
&= \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\|\nabla_\theta J_1 - \widehat{\nabla_\theta J_\omega}(\mathcal{D})\|_2^2\Big] - \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\|\nabla_\theta J_1 - \widehat{\nabla_\theta J_\omega}(\mathcal{D}_1)\|_2^2\Big] \\
&= \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\sum_{k=1}^{d_c}\big(\nabla_{\theta_k} J_1 - \widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D})\big)^2\Big] \\
&\quad - \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\sum_{k=1}^{d_c}\big(\nabla_{\theta_k} J_1 - \widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1)\big)^2\Big] \\
&= \sum_{k=1}^{d_c}\Bigg(\mathop{\mathbb{E}}_{\mathcal{D}}\Big[(\nabla_{\theta_k} J_1)^2 - 2\nabla_{\theta_k} J_1 \widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}) + (\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}))^2\Big] \\
&\quad - \mathop{\mathbb{E}}_{\mathcal{D}}\Big[(\nabla_{\theta_k} J_1)^2 - 2\nabla_{\theta_k} J_1 \widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1) + (\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1))^2\Big]\Bigg) \\
&= \sum_{k=1}^{d_c}\Bigg((\nabla_{\theta_k} J_1)^2 + \mathop{\mathbb{E}}_{\mathcal{D}}\Big[(\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}))^2\Big] \pm \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D})\Big]^2 \\
&\quad - 2\nabla_{\theta_k} J_1 \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D})\Big]
\end{aligned}$$

$$
\begin{aligned}
&- (\nabla_{\theta_k} J_1)^2 - \mathop{\mathbb{E}}_{\mathcal{D}}\left[(\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1))^2\right] \mp \mathop{\mathbb{E}}_{\mathcal{D}}\left[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1)\right]^2 \\
&+ 2\nabla_{\theta_k} J_1 \mathop{\mathbb{E}}_{\mathcal{D}}\left[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1)\right]\Big) \\
= \quad & \sum_{k=1}^{d_c}\Big(\Big(\nabla_{\theta_k} J_1 - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D})\right]\Big)^2 \\
&+ \mathop{\mathrm{Var}}_{\mathcal{D}}\left[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D})\right] - \mathop{\mathrm{Var}}_{\mathcal{D}}\left[\widehat{\nabla_\theta^{(k)} J_\omega}(\mathcal{D}_1)\right]\Big) \\
= \quad & \sum_{k=1}^{d_c}\Big(\Big(\nabla_{\theta_k} J_1 - \frac{1}{n(1-\gamma)}\sum_{j=1}^{m} n_j \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1^{(k)}}(s,a)]\Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2}\sum_{j=1}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1}^{(k)}(s,a)] \\
&- \frac{1}{n_1(1-\gamma)^2}\mathop{\mathrm{Var}}_{(s,a)\sim\zeta_1}[\widehat{\eta_1}^{(k)}(s,a)] \\
= \quad & \sum_{k=1}^{d_c}\Big(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j(\nabla_{\theta_k} J_1 - \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1^{(k)}}(s,a)])\Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta_1}^{(k)}(s,a)] \\
&+ \Big(\frac{1}{n^2} - \frac{1}{n_1^2}\Big)\frac{n_1}{(1-\gamma)^2}\mathop{\mathrm{Var}}_{(s,a)\sim\zeta_1}[\widehat{\eta_1}^{(k)}(s,a)]\Big).
\end{aligned}
$$

■

Lemma D.3. *For the optimistic implementation of the V-function critic, the objective function can be rewritten as*

$$
\begin{aligned}
&g(\omega) = \\
&\sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Big(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j(A_{1,\Phi}^{(V)-(k,l)} \\
&\qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\Delta_\Phi^{(k,l)}(s,a,s')])\Big)^2 \\
&+ \frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\Delta_\Phi^{(k,l)}(s,a,s')] \\
&+ \Big(\frac{1}{n^2} - \frac{1}{n_1^2}\Big)n_1 \mathop{\mathrm{Var}}_{(s,a,s')\sim d_1}[\Delta_\Phi^{(k,l)}(s,a,s')]\Big) \\
&+ \sum_{k=1}^{d_c}\Big(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j(b_{1,\Phi}^{(V)-(k)} \\
&\qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\rho_{1,\Phi}^{(k)}(s,a,s')])\Big)^2 \\
&+ \frac{1}{n^2}\sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\rho_{1,\Phi}^{(k)}(s,a,s')]
\end{aligned}
$$

$$+\Big(\frac{1}{n^2}-\frac{1}{n_1^2}\Big)n_1 \operatorname*{Var}_{(s,a,s')\sim d_1}[\rho^{(k)}_{1,\boldsymbol{\phi}}(s,a,s')]\Bigg).$$

Lemma D.4. *For the optimistic implementation of the Q-function critic, the objective function can be rewritten as*

$$\begin{aligned}
&g(\omega) = \\
&\sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j\big(A^{(Q)-(k,l)}_{1,\boldsymbol{\varphi}} \\
&\qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta^{(k,l)}_{\boldsymbol{\varphi}}(s,a,s',a')]\big)\Big)^2 \\
&+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta^{(k,l)}_{\boldsymbol{\varphi}}(s,a,s',a')] \\
&+\Big(\frac{1}{n^2}-\frac{1}{n_1^2}\Big)n_1 \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_1}[\Delta^{(k,l)}_{\boldsymbol{\varphi}}(s,a,s',a')]\Bigg) \\
&+\sum_{k=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j\big(b^{(Q)-(k)}_{1,\boldsymbol{\varphi}} \\
&\qquad\qquad - \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho^{(k)}_{1,\boldsymbol{\phi}}(s,a,s',a')]\big)\Big)^2 \\
&+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho^{(k)}_{1,\boldsymbol{\phi}}(s,a,s',a')] \\
&+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho^{(k)}_{1,\boldsymbol{\phi}}(s,a,s',a')] \\
&+\Big(\frac{1}{n^2}-\frac{1}{n_1^2}\Big)n_1 \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_1}[\rho^{(k)}_{1,\boldsymbol{\phi}}(s,a,s',a')]\Bigg).
\end{aligned}$$

Lemma D.5. *For the min-max implementation of the actor, the objective function can be overstimated by*

$$\begin{aligned}
&g_1(\omega) = \\
&\sum_{k=1}^{d_a}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j L^k_{\nabla J-\Theta\mathcal{E}\Xi}(\theta_j,\epsilon_j,\xi_j)d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j)) \\
&\qquad + \Big|\frac{1}{n(1-\gamma)}\sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j}[\widehat{\eta^{(k)}_j}(s,a)-\omega_j(s,a)\widehat{\eta^{(k)}_1}(s,a)]\Big|\Big)^2 \\
&+\frac{1}{n^2(1-\gamma)^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a)\sim\zeta_j}[\omega_j(s,a)\widehat{\eta^{(k)}_1}(s,a)]\Bigg).
\end{aligned}$$

Proof. By following a process similar to the one above and recalling that the expectations are taken assuming that the target has ζ as state-action distribution, it is possible to derive

$$g_\omega(\zeta)$$

$$
\begin{aligned}
&= \sum_{k=1}^{d_c} \Bigg(\Big(\frac{1}{n} \sum_{j=1}^{m} n_j (\nabla_{\theta_k} J_1(\zeta) - \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} [\omega_j(s,a) \widehat{\eta_1^{(k)}}(s,a)]) \Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2} \sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j} [\omega_j(s,a) \widehat{\eta_1}^{(k)}(s,a)] \\
&+ \Big(\frac{1}{n^2} - \frac{1}{n_1^2} \Big) \frac{n_1}{(1-\gamma)^2} \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_1} [\widehat{\eta_1}^{(k)}(s,a)] \Bigg) \\
&\leqslant \sum_{k=1}^{d_c} \Bigg(\Big(\frac{1}{n} \sum_{j=1}^{m} n_j (\nabla_{\theta_k} J_1(\zeta) - \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} [\omega_j(s,a) \widehat{\eta_1^{(k)}}(s,a)]) \\
&\pm \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} [\widehat{\eta_j^{(k)}}(s,a)] \Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2} \sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j} [\omega_j(s,a) \widehat{\eta_1}^{(k)}(s,a)] \Bigg) \\
&\leqslant \sum_{k=1}^{d_c} \Bigg(\Big(\frac{1}{n} \sum_{j=2}^{m} n_j L^k_{\nabla J - \Theta\mathcal{E}}(\theta_j, \epsilon_j) d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)) \\
&+ \Big| \frac{1}{n(1-\gamma)} \sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a)\sim\zeta_j} [\widehat{\eta_j^{(k)}}(s,a) - \omega_j(s,a) \widehat{\eta_1^{(k)}}(s,a)] \Big| \Big)^2 \\
&+ \frac{1}{n^2(1-\gamma)^2} \sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a)\sim\zeta_j} [\omega_j(s,a) \widehat{\eta_1}^{(k)}(s,a)] \Bigg) \\
&= g_1(\omega).
\end{aligned}
$$

■

Lemma D.6. *For the min-max implementation of the V-function critic, the objective function can be overstimated by*

$$
\begin{aligned}
&g_1(\omega) = \\
&\sum_{k=1}^{d_c} \sum_{l=1}^{d_c} \Bigg(\Big(\frac{1}{n} \sum_{j=2}^{m} n_j L^{k,l}_{A - \Theta\mathcal{E}}(\theta_j, \epsilon_j) d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)) \\
&\qquad + \Big| \frac{1}{n} \sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s')\sim d_j} [(1-\omega_j(s,a,s')) \Delta_{\Phi}^{(k,l)}(s,a,s')] \Big| \Big)^2 \\
&+ \frac{1}{n^2} \sum_{j=2}^{m} n_j \mathop{\mathrm{Var}}_{(s,a,s')\sim d_j} [\omega_j(s,a,s') \Delta_{\Phi}^{(k,l)}(s,a,s')] \Bigg) \\
&+ \sum_{k=1}^{d_c} \Bigg(\Big(\frac{1}{n} \sum_{j=2}^{m} n_j L^{k}_{b - \Theta\mathcal{E}\Xi}(\theta_j, \epsilon_j, \xi_j) d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j)) \\
&\qquad + \Big| \frac{1}{n} \sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s')\sim d_j} [\rho^{(k)}_{j,\Phi}(s,a,s') - \\
&\qquad\qquad \omega_j(s,a,s') \rho^{(k)}_{1,\Phi}(s,a,s')] \Big| \Big)^2
\end{aligned}
$$

$$+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s')\sim d_j}[\omega_j(s,a,s')\rho^{(k)}_{1,\phi}(s,a,s')]\Bigg).$$

Lemma D.7. *For the min-max implementation of the V-function critic, the objective function can be overstimated by*

$$\begin{aligned}
&g_1(\omega) = \\
&\sum_{k=1}^{d_c}\sum_{l=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j L^{k,l}_{A-\Theta\mathcal{E}}(\theta_j,\epsilon_j)d_{\Theta\mathcal{E}}((\theta_1,\epsilon_1),(\theta_j,\epsilon_j)) \\
&\qquad\qquad +\Big|\frac{1}{n}\sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[(1-\omega_j(s,a,s',a'))\Delta^{(k,l)}_{\varphi}(s,a,s',a')]\Big|\Big)^2 \\
&+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\Delta^{(k,l)}_{\varphi}(s,a,s',a')]\Bigg) \\
&+\sum_{k=1}^{d_c}\Bigg(\Big(\frac{1}{n}\sum_{j=2}^{m} n_j L^{k}_{b-\Theta\mathcal{E}\Xi}(\theta_j,\epsilon_j,\xi_j)d_{\Theta\mathcal{E}\Xi}((\theta_1,\epsilon_1,\xi_1),(\theta_j,\epsilon_j,\xi_j)) \\
&\qquad\qquad +\Big|\frac{1}{n}\sum_{j=2}^{m} n_j \mathop{\mathbb{E}}_{(s,a,s',a')\sim\overline{d}_j}[\rho^{(k)}_{j,\varphi}(s,a,s',a')- \\
&\qquad\qquad\qquad\qquad \omega_j(s,a,s',a')\rho^{(k)}_{1,\varphi}(s,a,s',a')]\Big|\Big)^2 \\
&+\frac{1}{n^2}\sum_{j=2}^{m} n_j \operatorname*{Var}_{(s,a,s',a')\sim\overline{d}_j}[\omega_j(s,a,s',a')\rho^{(k)}_{1,\varphi}(s,a,s',a')]\Bigg).
\end{aligned}$$

D.3 On the proximity of the optimal parameters

This section presents a result that explains why transfer from the optimal source policies gives a better performance than transfer from a randomly selected source policy.

Lemma D.8. *Let $X, Y \subset \mathbb{R}^n$ be metric spaces with metrics d_X, d_Y, respectively. Let $f : X \times Y \to \mathbb{R}$ be a continuously differentiable function such that $f(x,\cdot)$ is L_f-LC, and let $y_0, y_1 \in Y$ be two points. If $x_0 \in X$ is such that $\frac{\partial f}{\partial x}(x_0,y_0) = 0$ then there exists a bounded connected neighborhood U of x_0 and a point $x' \in U$ such that $\frac{\partial f}{\partial x}(x',y_1) = 0$.*

Proof. If $\frac{\partial f}{\partial x}(x_0,y_1) = 0$ then the proof over.
Assume then that $\frac{\partial f}{\partial x}(x_0,y_1) \neq 0$. Let U_0 be a block containing x_0 such that $f(x_0,y_0) \geqslant f(x,y_0)$ for any $x \in U_0$. Define $f_M(x) = f(x,y_0) + L_f d_y(y_0,y_1)$ and $f_m(x) = f(x,y_0) - L_f d_y(y_0,y_1)$; note that $f_m(x) \leqslant f(x,y_1) \leqslant f_M(x)$, for any $x \in X$, i. e., the functions model the maximum increase or decrease of f along the space Y.
Let

$$\begin{aligned}
U_1 &= \{x \in U_0 \mid f_M(x) \geqslant f_m(x_0)\} \\
&= \{x \in U_0 \mid f(x,y_0) - f(x_0,y_0) \geqslant -2L_f d_Y(y_0,y_1)\} \\
&= (f(\cdot,y_0))^{-1}([f(x_0,y_0) - 2L_f d_Y(y_0,y_1), f(x_0,y_0)]),
\end{aligned}$$

which is a compact set due to the continuity of f. Note that $f(x, y_1) < f(x_0, y_1)$ for any x in some neighborhood containing U_1. Let U_2 be the connected component in the space U_1 containing x_0, which will be a compact set too as $U_0 \supset U_1$ is bounded. Thus, $f(\cdot, y_1)$ restricted to U_2 will have a maximum at some point in there, namely x'. Given that $\frac{\partial f}{\partial x}(x_0, y_1) \neq 0$, it holds that $f(x', y_1) > f(x_0, y_1)$. Note, however, that x' is in the interior of U_2:
Assume that x' is not in the interior of U_2. It is then possible to have a sequence $(x_i)_{i=1}^{\infty}$ of points outside U_2 that converge to x'. This will create a sequence of points $(z_i = f(x_i, y_1))_{i=1}^{\infty}$ whose limit is $f(x', y_1)$. But this is a contradiction as $f(x', y_1) > f(x_0, y_1) > z_i$.
Hence, the maximum x' is in the interior of U_2, and this implies that $\frac{\partial f}{\partial x}(x_1, y_1) = 0$. ■

Joining this result with Lemma C.12 ensures that a local maximum in the target task will lie in a neighborhood of the source local maximum.