POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

On the Sample Complexity of Inverse Reinforcement Learning

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Filippo Lazzati**

Student ID: 991267
Advisor: Prof. Alberto Maria Metelli
Co-advisors: Marcello Restelli
Academic Year: 2022-2023

Abstract

Reinforcement Learning is one of the most promising research directions in the field of *Artificial Intelligence*. It provides techniques for solving sequential decision-making problems that satisfy specific properties. In many cases, the presence of an *expert*, an agent that knows how to optimally solve the given problem, allows to teach to our agent how to behave by imitating the expert. *Inverse Reinforcement Learning* defines a powerful family of algorithms for solving the *Imitation Learning* problem. The main idea behind Inverse Reinforcement Learning is that our artificial intelligent agent can learn not merely by copying the actions of the expert, but by firstly studying and analyzing the *reasons* and the *motivations* that make the expert behave in that way. In other words, this discipline allows, in specific settings, to retrieve the goal of the expert, and then, to use it to understand which is the best strategy that our agent can apply in order to achieve that goal. In most cases, the strategy of the expert and the dynamics of the environment are not explicitly provided, but they have to be estimated based on expert demonstrations. In this thesis, we focus on the *number of samples* that must be collected in order to solve the Inverse Reinforcement Learning problem in an acceptable way according to a certain index of performance. The analysis conducted is a *worst-case analysis*, namely it aims to compute the minimum number of samples that any algorithm, even the best one, needs to collect in the worst possible problem instance to provide an acceptable solution. One main *setting* is considered that assumes the presence of an oracle that can provide samples for any possible configuration of states where the agent is located and actions taken. The results obtained allow to characterize the sample complexity of Inverse Reinforcement Learning according to the problem definition adopted in this thesis.

Keywords: Inverse Reinforcement Learning, Sample Complexity, Feasible Set, Statistical Learning

Abstract in lingua italiana

L'*Apprendimento per Rinforzo* è, ad oggi, una delle direzioni di ricerca più promettenti nell'ambito dell'*Intelligenza Artificiale*. Esso infatti fornisce tecniche per risolvere specifici problemi in cui un agente deve prendere sequenze di decisioni nel tempo. Spesso, la presenza di un *esperto*, cioè un agente che sa come risolvere il problema in maniera ottimale, ci permette di insegnare al nostro agente come prendere le decisioni semplicemente imitandolo. L'*Apprendimento per Rinforzo Inverso* definisce una consistente famiglia di algoritmi per risolvere il problema dell'*Apprendimento per Imitazione*. L'idea principale dell'Apprendimento per Rinforzi Inverso è che il nostro agente può imparare non soltanto semplicemente copiando le azioni dell'esperto, ma studiando e analizzando le *motivazioni* che guidano l'esperto nelle sue decisioni. In altre parole, questa disciplina permette, in particolari problemi, di recuperare l'obiettivo dell'esperto e poi di usarlo per capire qual è la miglior strategia che il nostro agente può mettere in pratica per raggiungerlo. In questa tesi, mi concentro sul *numero di campioni* che devono essere raccolti per risolvere il problema dell'Apprendimento per Rinforzo Inverso in maniera accettabile secondo un certo indice per misurare le prestazioni. L'analisi condotta è un'*analisi del caso pessimo*, cioè mira a calcolare il minimo numero di campioni che ogni algoritmo, persino il migliore, deve raccogliere nell'istanza di problema peggiore per fornire una soluzione accettabile. La *configurazione* principale che viene considerata è quella in cui è presente un oracolo che fornisce campioni per qualsivoglia combinazione di stati in cui l'agente può trovarsi e azioni che può prendere. I risultati ottenuti permettono di caratterizzare la complessità campionaria dell'Apprendimento per Rinforzo Inverso secondo la definizione di problema adottata in questa tesi.

Keywords: Apprendimento per Rinforzo Inverso, Complessità Campionaria, Insieme di Rinforzi Accettabili, Apprendimento Statistico

Contents

1 | Introduction

1.1. General Overview

Inverse reinforcement learning (IRL) is the problem of inferring the reward function of an agent, given its policy or observed behavior [2]. Given a certain Markov Decision Process [45] without reward function, along with a certain expert policy, IRL aims to compute a specific reward function which is feasible (namely, the expert policy is optimal in the MDP with such reward function) and satisfies specific properties. IRL, along with Behavioral cloning (BC), is one of the techniques we can use for solving the imitation learning problem, *namely the problem of efficiently learn a desired behavior by imitating an expert's behavior* [43]. Similarly to the forward Reinforcement Learning setting [51], it is crucial to efficiently explore the environment and collect the minimum amount of samples that allows solving the problem [26]. In the context of PAC learning [21], we talk about sample complexity to refer to this concept.
The research topic concerns the complexity of solving a problem when some parameters are not known, but they have to be sampled. Collecting samples can be a difficult task, for two reasons. We might need to devise a sampling strategy and we have to understand the number of samples we need to solve the problem in an acceptable way according to a certain performance index. In particular, the research topic focuses on the computation of lower and upper PAC bounds on the sample complexity of such problem analogously to what can be found in the literature for the forward RL problem (like [5, 12]).
The problem of the thesis concerns the PAC sample complexity of estimating the feasible set [39] of an IRL problem. However, we will not consider, as estimation error, the distance between the inferred (at the subsequent RL step) policy and the true policy like in [36, 39], but we will focus on the straightforward Hausdorff distance between the estimated set and the true feasible set. As a metric between functions, we will use the max norm for the generative case. The reasons behind this choice concern the complexity of the considered problem. If we use a generative sampling model, we can reach any (s, a) pair, thus we can provide max-norm guarantees, while with a forward sampling model we might not be able to do so.

1.2. Motivations

The *research topic* has both a theoretical and practical importance. With regards to the former, it allows to characterize the complexity of the IRL problem from the point of view of the number of samples necessary to solve it in an acceptable way. We might then compare the complexity to the forward RL problem and see which of them is easier. Moreover, having a lower bound might help to assess the performance of existing and new algorithms. Considering the latter, PAC analysis allows us to propose algorithms which are PAC optimal and therefore it would allow us to devise algorithms with worst-case theoretical guarantees. Finally, let us remark that IRL has currently a practical importance, but it has not been understood in-depth from a theoretical point of view.
Up to now, IRL researchers have focused on proposing algorithms for computing only one specific feasible reward function that satisfies certain properties. Once obtained a reward function, the next step is straightforward. To perform forward RL in the MDP with such reward to determine a policy to use on our agent (which has now learned from the expert). However, few works have focused on the estimation of the entire feasible set, and in particular on the complexity of doing so. If we were able to characterize such estimation problem, then new IRL algorithms might be devised atop such basic estimation problem, and the PAC framework is a powerful tool for characterizing it. This is why the problem faced in this thesis is important.

1.3. Goals

The main goals of this work concern the development of lower and upper bounds to the sample complexity of inverse reinforcement learning, where the IRL problem is here defined as the problem of computing a set of reward functions instead of a single function. The objective is to better characterize the complexity of IRL from a statistical viewpoint and to help the RL community to better understand this problem from a theoretical perspective. Therefore, there will not be made references to practical applications nor to implementations of the contents discussed in this thesis.

1.4. Contributions

The main contributions concern the definition of a Lipschitzian framework for IRL along with a thorough description of its limits, the development of a novel lower bound and the analysis of an algorithm to obtain an upper bound to the sample complexity that matches the lower bound up to logarithmic factors.

1.5. Structure of the Thesis

The thesis is structured in the following way:

Chapter 2 We present the notation adopted in the thesis and then foundational notions concerning *Reinforcement Learning*, like Markov Decision Processes, Dynamic Programming and some RL algorithms present in literature, *Inverse Reinforcement Learning*, showing its similarities and differences to Behavioral Cloning in the context of Imitation Learning, and the *Probably Approximately Correct (PAC) Learning* framework.

Chapter 3 Here, we present the state-of-the-art literature concerning the sample complexity of Bandits, Reinforcement Learning and Inverse Reinforcement Learning. A different section is dedicated to each topic.

Chapter 4 In this chapter we define a Lipschitz framework for Inverse Reinforcement Learning, clearly defining the setting and thoroughly analysing its limitations.

Chapter 5 This chapter contains the definition of the PAC framework in which we develop the bounds.

Chapter 6 The main theorem with the lower bound result along with its proof is presented in this chapter.

Chapter 7 We introduce the *Uniform-Sampling Algorithm* and we analyze it to obtain the upper bound result.

Chapter 8 This last chapter concludes the thesis. Here we sum up the results, draw some conclusions and propose future extensions.

Appendix The appendix contains a lower bound in a slightly different setting and some technical lemmas.

2 | Background

2.1. Notation

In this section we introduce the notation that will be adopted for the rest of the thesis.

Let $a, b \in \mathbb{N}$ with $a \leqslant b$, we denote with $[\![a, b]\!] := \{a, \dots, b\}$ and with $[\![a]\!] := [\![1, a]\!]$. Let $\mathcal{X}$ be a set, we denote with $\Delta^{\mathcal{X}}$ the set of probability measures over $\mathcal{X}$. Let $\mathcal{Y}$ be a set, we denote with $\Delta^{\mathcal{X}}_{\mathcal{Y}}$ the set of functions with signature $\mathcal{Y} \to \Delta^{\mathcal{X}}$. Let $(\mathcal{X}, d)$ be a (pre)metric space, where $\mathcal{X}$ is a set and $d : \mathcal{X} \times \mathcal{X} \to [0, +\infty]$ is a (pre)metric.[1] Let $\mathcal{Y}, \mathcal{Y}' \subseteq \mathcal{X}$ be non-empty sets, we define the *Hausdorff (pre)metric* [48] $\mathcal{H}_d : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to [0, +\infty]$ between $\mathcal{Y}$ and $\mathcal{Y}'$ induced by the (pre)metric d as follows:

$$\mathcal{H}_d(\mathcal{Y}, \mathcal{Y}') := \max \left\{ \sup_{y \in \mathcal{Y}} \inf_{y' \in \mathcal{Y}'} d(y, y'), \sup_{y' \in \mathcal{Y}'} \inf_{y \in \mathcal{Y}} d(y, y') \right\}. \tag{2.1}$$

2.2. Reinforcement Learning

Reinforcement Learning (RL) is a concept that Computer Science and, in particular, Artificial Intelligence (AI), have borrowed from Psichology. It refers to the learning capabilities of animals of learning how to behave in specific circumstances without being taught by someone, but simply by interacting with the environment and changing the behavioral policy based on specific signals received after the interaction. Such signals are called reinforces, and can be positive or negative, which means that the action taken has brought to something good or, respectively, bad to the agent. In the context of AI, the notion of RL becomes that of a set of techniques and algorithms for solving Markov Decision Processes (MDP).

This chapter will firstly introduce the notion of MDP, highlighting its main features and properties, defining what it means to *solve an MDP*, and then move to an introduction to the Dynamic Programming techniques adopted for solving an MDP. Finally, in the

[1] A *premetric* d satisfies the axioms: $d(x, x') \geqslant 0$ and $d(x, x) = 0$ for all $x, x' \in \mathcal{X}$. Any *metric* is clearly a premetric.

last section, we will introduce the main RL algorithms for solving an MDP both in the discrete and continuous case.

2.2.1. Markov Decision Processes

Let us introduce the main concepts of RL from a high-level point of view, following the presentation in [51], by defining them intuitively, and then formally. Let us consider an **agent**, that is, any instance, person, computer program, that takes actions, and let us put it into an **environment**, any surrounding with which the agent interacts during time. The situation is really general, and every sequential decision-making problem can be formalized in this way. In other words, every time we observe something or someone that has to take a sequence of decisions during time, we can model it as an agent in an environment. The idea is that the agent is located in a certain **state**, which is a general concept representing *whatever information is available to the agent about the environment* [51]. The agent has a **goal**, and he can achieve it by taking decisions, called **actions**, which influence the environment. So, to sum up, there is an agent, located in a certain environment, that has a goal to achieve. He can reach it by taking actions during time that allow it to move from a state to another. In RL, it is assumed that every time the agent takes an action, the environment responds by sending a reinforcement signal, called **reward**, to the agent. To exemplify, let us think to the situation in which there is a dog, the agent, located inside an house with some people, the environment. The dog might be in the situation in which a person tells it "sit", which represents a state. In such situation, called **decision epoch** in [45], the dog has to take a decision, which might be to actually sit down or not. We might think that in case it sits down, the person (environment) will award the dog with a biscuit, while in the opposite case it will not. In RL, we make the assumption that every goal can be entirely described in terms of rewards, and therefore that the goal of our RL agent, the dog, is to maximize the sum of the rewards it will gather over time, namely the amount of biscuits it will receive. There are two main settings that are usually studied, the γ-discounted infinite-horizon and the finite-horizon ones. We will provide background concepts and results for both the settings during the thesis. Formally, we can define a γ-discounted infinite-horizon Markov Decision Process (MDP) as:

Definition 1 (Discounted infinite-horizon Markov Decision Process)**.** *An MDP can be defined as a tuple* $\langle \mathcal{S}, \mathcal{A}, p, R, \gamma, \mu \rangle$*, where* $\mathcal{S}$ *(*$S := |\mathcal{S}|$*) is the set of states,* $\mathcal{A}$ *(*$A := |\mathcal{A}|$*) is the set of possible actions of the agent,* p *is the transition model, namely a conditional probability distribution over the next state reached by the agent given the previous state and the action taken; in symbols:* $p \in \Delta_{\mathcal{S} \times \mathcal{A}}^{\mathcal{S}}$*. Next, there is* $R : \mathcal{S} \times \mathcal{A} \to [-1, 1]$[2]*,*

[2]For the sake of simplicity and w.l.o.g., we restrict to reward functions bounded by 1 in absolute

the reward function, which represents the expected reward obtained when taking action $a \in \mathcal{A}$ *from state* $s \in \mathcal{S}$, $R(s,a)$. $\mu \in \Delta^{\mathcal{S}}$ *is the initial state distribution, the distribution representing the initial state in which the agent is found. Finally,* $\gamma \in [0,1]$ *is the discount factor, which represents the importance the agent gives to the rewards obtained far from the current time instance.*

With regards to the finite-horizon MDPs, we have:

Definition 2 (Finite-horizon Markov Decision Process)**.** *A finite-horizon* Markov Decision Process *is as the tuple* $\langle \mathcal{S}, \mathcal{A}, p, r, H, \mu \rangle$*, where the meaning of most symbols is the same of those presented in the discounted setting.* H *is the horizon, the number of time steps for which the agent has to take decisions. However, the reward function and the transition model can now vary with time, thus we write that* $p = (p_h)_{h \in [\![H]\!]}$ *is the transition model, where for every stage* $h \in [\![H]\!]$ *we have* $p_h \in \Delta^{\mathcal{S}}_{\mathcal{S} \times \mathcal{A}}$*, and that* $r = (r_h)_{h \in [\![H]\!]}$*, where for every stage* $h \in [\![H]\!]$ *we have* $r_h : \mathcal{S} \times \mathcal{A} \to [-1,1]$*, for the time-inhomogeneous case (when* p *and* r *change with time). Instead, for the time-homogeneous setting, the definitions are the same of the infinite-horizon one.*

Another definition that will be useful later is that of time-inhomogeneous finite-horizon *Markov decision process without reward* (MDP\R), defined as a 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, H, \mu \rangle$ with usual definitions. An MDP\R is time-homogeneous if, for every stage $h \in [\![H-1]\!]$, we have $p_h = p_{h+1}$ a.s.; in such a case, we denote the transition model with the symbol p only. We can define the **return** of the agent as:

Definition 3 (γ-discounted return)**.** *The* γ*-discounted return can be defined as the cumulative discounted sum of reward received during time starting from the current time step:* $v_t := r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{+\infty} \gamma^k r_{t+k+1}$*, where* $r_{t+1} := R(s_t, a_t)\ \forall t$.

In the finite-horizon case:

Definition 4 (Finite-horizon return)**.** *The return is defined in this case as* $v := r_1 + r_2 + \ldots + r_H = \sum_{t=1}^{H} r_t$.

However, we have not defined the concept of *solving an MDP* yet. To do so, we have to introduce the notion of **policy**, which is the behavior of the agent, what makes the agent decide which action to take at every decision epoch (time step). A policy can be of various types depending on whether it satisfies the Markovian property and if it changes with time. Formally:

Definition 5 (policy)**.** *A policy* π *is a probability distribution over actions given the*

value.

current state or the past history. Depending on its properties, the policy can take on various names.

There are various kinds of policy:

Markovian/History-dependent if it depends only on the current state $\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}$ then it is Markovian, otherwise it is history-dependent;

Deterministic/Stochastic if the policy assigns always probability 1 to a certain action and 0 to the others, then it is called deterministic, otherwise it is called stochastic;

Stationary/Non-Stationary if the policy does not change with time, then the policy is stationary, otherwise it is called non-stationary.

In the finite-horizon setting, the agent's behavior can be modeled with a time-inhomogeneous policy $\pi = (\pi_h)_{h \in [\![H]\!]}$ where for every stage $h \in [\![H]\!]$, we have $\pi_h \in \Delta_{\mathcal{S}}^{\mathcal{A}}$. Let us introduce two useful operators. Let $f \in \mathbb{R}^{\mathcal{S}}$ and $g \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, we denote with $p_h f(s,a) = \sum_{s' \in \mathcal{S}} p_h(s'|s,a) f(s')$ and with $\pi_h g(s) = \sum_{a \in \mathcal{A}} \pi_h(a|s) g(s,a)$ the expectation operators w.r.t. the transition model and the policy, respectively. Notice that in the following, unless explicitly stated, we will focus on the case of stationary and Markovian policies. We can finally define what we mean when we tell about solving an MDP:

Definition 6 (Solving an MDP)**.** *Given an MDP (of any of the two kinds defined above), we can solve it if we find the optimal policy π^*, the behavior of the agent that maximizes the expected return over time: $\pi^* \in \operatorname{argmax}_{\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}} \mathbb{E}_\pi[v]$.*

In the previous definition, we have talked about expected return; it can be interpreted as the **utility** of a certain state or of a certain state-action pair. It is necessary to introduce two functions, the **state-value function** V^π of policy π and the **action-value function** Q^π of policy π which represent respectively the notion of utility of a state and of a state-action pair. Notice that from now on we provide the definitions only for the case of a discounted infinite-horizon MDP, neglecting those for the finite-horizon case.

Definition 7 (State-value function)**.** *Given an MDP and a policy π, the state-value function of a state is the expected return of the agent when it starts from the state and it follows the policy π: $V^\pi(s) := \mathbb{E}_\pi[v_t | s_t = s]$.*

Definition 8 (Action-value function)**.** *Given an MDP and a policy π, the action-value function of a state-action pair is the expected return of the agent when it starts from the state and playes the action and then it follows the policy π: $Q^\pi(s,a) := \mathbb{E}_\pi[v_t | s_t = s, a_t = a]$.*

A useful notation for the state-value (V-) function and the action-value (Q-) function is the following. We denote the *Q-function* under reward r and policy π as $Q^\pi(\cdot; r) = (Q_h^\pi(\cdot; r))_{h\in[\![H]\!]}$, which represents the expected sum of rewards collected starting from $(s, a, h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$ and following policy π thereafter:

$$Q_h^\pi(s, a; r) := \mathop{\mathbb{E}}_\pi \left[\sum_{l=h}^{H} r_l(s_l, a_l) | s_h = s, a_h = a \right],$$

where $\mathbb{E}_\pi$ denotes the expectation w.r.t. π, i.e., $a_h \sim \pi_h(\cdot|s_h)$ for every stage $h \in [\![h, H]\!]$. We define analogously the *V-function* $V^\pi(\cdot; r) = (V_h^\pi(\cdot; r))_{h\in[\![H]\!]}$. The *advantage function* $A_h^\pi(s, a; r) = Q_h^\pi(s, a; r) - V_h^\pi(s; r)$ represents the relative gain of playing action $a \in \mathcal{A}$ rather than following policy π in the state-stage pair (s, h). This notation is defined for the finite-horizon setting, but can be easily adapted to the infinite-horizon one, by removing the dependence on h from the functions. Given their sequential nature, these functions are often written decomposed using the Bellman expectation equations:

$$V^\pi(s) = \mathbb{E}_\pi[r_t + \gamma V^\pi(s_{t+1})|s_t = s] = \sum_{a\in\mathcal{A}} \pi(a|s) \Big(R(s, a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s, a) V^\pi(s') \Big)$$

and

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_\pi[r_t + \gamma Q^\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a] \\ &= R(s, a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s, a) \sum_{a'\in\mathcal{A}} \pi(a'|s') Q^\pi(s', a'), \end{aligned}$$

where these definitions hold for the infinite-horizon setting. From now on, we will avoid to specify the setting when clear from the context. A policy π induces a Markov Reward Process, i.e. an MDP without actions, for which the Bellman expectation equation of the state-value function can be written in matrix form:

$$V^\pi = R^\pi + \gamma P^\pi V^\pi \iff V^\pi = (I - \gamma P^\pi)^{-1} R^\pi,$$

where R^π, P^π, V^π represent the corresponding expected matrix operators with respect to π. The Bellman expectation equations allow to introduce some operators, called the Bellman operators, one for the state-value function and the other for the action-value function:

Definition 9 (Bellman operator for V^π)**.** *The Bellman operator T^π for the state-value function V^π is a mapping $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ that maps value functions to other value functions:*

$$(T^\pi V)(s) := \sum_{a\in\mathcal{A}} \pi(a|s) \Big(r(s, a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s, a) V(s') \Big) \ \forall V, \ \forall s \in \mathcal{S}.$$

It should be remarked that the value function V^π is the fixed point of such operator. With regards to the action-value function, we have:

Definition 10 (Bellman operator for Q^π)**.** *The Bellman operator T^π for the action-value function Q^π is a mapping $T^\pi : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ that maps action-value functions to other action-value functions:*

$$(T^\pi Q)(s,a) := r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) \sum_{a'\in\mathcal{A}} \pi(a'|s')Q(s',a') \;\forall Q,\; \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

Again, similarly to the state-value function, we can notice that the action-value function Q^π is the fixed point of the Bellman operator. It is clear that, by definition, both the value functions depend on the policy adopted by the agent. Among all the policies, we have seen that the goal of any RL learner is to find the policy that maximizes the expected return, thus that maximizes the value functions. It can therefore be useful to define the value that both the value functions achieve under the optimal policy.

Definition 11 (Optimal state-value function)**.** *The optimal state-value function is the state-value function of the optimal policy, i.e. the maximum state-value function over all the policies:*

$$V^*(s) := \max_{\pi\in\Delta_{\mathcal{S}}^{\mathcal{A}}} V^\pi(s) \quad \forall s \in \mathcal{S}.$$

Definition 12 (optimal action-value function)**.** *The optimal action-value function is the action-value function of the optimal policy, i.e. the maximum action-value function over all the policies:*

$$Q^*(s,a) := \max_{\pi\in\Delta_{\mathcal{S}}^{\mathcal{A}}} Q^\pi(s,a) \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

It should be noticed that, once that the optimal value function is known, the MDP can be considered solved since it allows to retrieve an optimal policy (under certain conditions that will be presented later). They are of extreme importance the Bellman optimality equations, which are the corresponding of the Bellman expectation equations in the case we consider the optimal policy. Indeed, the unique fixed point of such equations are the optimal value functions:

Definition 13 (Bellman optimality equation for the state-value function)**.**

$$V^*(s) = \max_{a\in\mathcal{A}} Q^*(s,a) = \max_{a\in\mathcal{A}} \Big\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)V^*(s') \Big\}.$$

Definition 14 (Bellman optimality equation for the action-value function)**.**

$$Q^*(s) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^*(s') = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a').$$

Based on these equations, we can define the Bellman optimality operators analogously to the Bellman expectation operators:

Definition 15 (Bellman optimality operator for V^*)**.** *The Bellman operator T^* for the optimal state-value function V^* is a mapping $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ that maps value functions to other value functions:*

$$(T^*V)(s) := \max_{a \in \mathcal{A}} \Big\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s') \Big\} \; \forall V, \; \forall s \in \mathcal{S}.$$

Definition 16 (Bellman optimality operator for Q^*)**.** *The Bellman operator T^* for the optimal action-value function Q^* is a mapping $T^\pi : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ that maps action-value functions to other action-value functions:*

$$(T^*Q)(s,a) := r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} Q(s',a') \; \forall Q, \; \forall (s,a) \in \mathcal{S} \times \mathcal{A}.$$

Both the pairs of Bellman operators defined satisfy some important properties, which is worthy to mention here:

Monotonicity given any pair of vectors f_1, f_2 such that $f_1 \preceq f_2$[3], we have that:

$$T^\pi f_1 \preceq T^\pi f_2 \quad \text{and} \quad T^* f_1 \preceq T^* f_2,$$

in other words, the Bellman operators preserve the monotonicity.

Max-Norm Contraction (Lipschitz continuity) given any pair of vectors f_1, f_2, it holds that:

$$||T^\pi f_1 - T^\pi f_2||_\infty \leqslant \gamma ||f_1 - f_2||_\infty \quad \text{and} \quad ||T^* f_1 - T^* f_2||_\infty \leqslant \gamma ||f_1 - f_2||_\infty.$$

Value function convergence for any vector $f \in \mathbb{R}^{|\mathcal{S}|}$ and any policy π it holds that:

$$\lim_{k \to +\infty} (T^\pi)^k f = V^\pi \quad \text{and} \quad \lim_{k \to +\infty} (T^*)^k f = V^*.$$

[3]Symbol $\preceq$ denotes element-wise ordering.

2.2.2. Dynamic Programming

This and the following sections present some algorithms for the computation of the optimal policy. As explained in [54] and [51], we can identify two kinds of methods for solving the RL problem:

Dynamic Programming techniques include algorithms that compute the optimal policy only in the tabular case, i.e., in case both the state and action spaces are finite, and a perfect model of the environment is provided. Broadly speaking, these are techniques that can be applied only when the transition model p is completely known, hypothesis not always guaranteed, and when $\mathcal{S}$ and $\mathcal{A}$ are finite and small, since they are rather expensive;

Reinforcement Learning algorithms that can solve the problem even in the most general case, with even continuous state-action space, without model of the environment.

The former techniques are presented in this section, while the latter are presented in the next section. It should be remarked that a trivial way for finding the optimal policy might be to enumerate all the possible deterministic Markovian policies, evaluate each of them and finally retrieving the best one (but this has an exponential computational complexity). This can be done only in the tabular case. If we define the ordering between policies with:

Definition 17 (Ordering between policies)**.** *We can define an order relation $\geqslant$ among policies by exploiting the corresponding value functions:*

$$\pi_1 \geqslant \pi_2 \quad \textit{if} \quad V^{\pi_1} \geq V^{\pi_2}$$

Then the guarantee that an optimal policy can be found among the deterministic Markovian policies (for the discounted infinite-horizon case) is provided by the following theorem:

Theorem 1 (MDPs and Optimality)**.** *For any discounted infinite-horizon Markov Decision Process:*

- *there exists an optimal policy π^* that is better than or equal to all the other policies: $\pi^* \geqslant \pi \quad \forall \pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}$;*
- *All optimal policies achieve the optimal value function: $V^{\pi^*} = V^*$;*
- *All optimal policies achieve the optimal action–value function: $Q^{\pi^*} = Q^*$;*

- *There is always a deterministic optimal policy for any MDP, and it can be obtained by maximizing the optimal action-value function:*

$$\pi^*(a|s) = \begin{cases} 1 & \text{if} \quad a \in \underset{a' \in \mathcal{A}}{\operatorname{argmax}}\, Q^*(s, a') \\ 0 & \text{otherwise} \end{cases} .$$

One of the fathers of Dynamic Programming is R. Bellman [7]. Such methodology is widespread and can be applied to all the problems which possess an optimal substructure and overlapping subproblems.
Finally, let us conclude this introduction to Dynamic Programming techniques by stating that there are two kinds of RL problems, one which aims to find the optimal policy, and it is the problem considered thus far, and the other one aims to determine the value function of a given policy:

Definition 18 (Prediction problem)**.** *A prediction problem is a problem that, given an MDP and a policy π, aims to compute the value function of π in the MDP: V^π.*

Definition 19 (Control problem)**.** *A control problem is a problem that, given an MDP, aims to compute the optimal value function V^* and the optimal policy π^*.*

For the finite-horizon case, the optimal policy is non-stationary and can be computed by backward recursion. Starting from the last time step, choosing at each decision epoch the action that maximizes the value function from now on. The cost is $H|\mathcal{S} \times \mathcal{A}|$. This is clearly an application of Dynamic Programming, given the structure of the problem.

Policy Iteration

Policy Iteration refers to one technique of determining the optimal policy of an MDP, and therefore of solving the control problem, when the MDP is completely known (its transition model and reward function are, in particular, known). It consists of iteratively alternating between the two tasks of **policy evaluation** and **policy improvement**. Policy evaluation refers to any technique adopted for solving the prediction problem, i.e., given a policy, computing its value function. There are many ways for doing so if the MDP is known. One method consists in solving the system of linear equations that arises

from the Bellman expectation equation:

$$\begin{cases} V^\pi(s_1) = \sum_{a\in\mathcal{A}} \pi(a|s_1)\Big(r(s_1,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s_1,a)V^\pi(s')\Big) \\ V^\pi(s_2) = \sum_{a\in\mathcal{A}} \pi(a|s_2)\Big(r(s_2,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s_2,a)V^\pi(s')\Big) \\ \dots \\ V^\pi(s_{|\mathcal{S}|}) = \sum_{a\in\mathcal{A}} \pi(a|s_{|\mathcal{S}|})\Big(r(s_{|\mathcal{S}|},a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s_{|\mathcal{S}|},a)V^\pi(s')\Big) \end{cases}.$$

However, this method is quite computational expensive, therefore it is often preferred to evaluating a policy with the iterative application of the Bellman expectation equation. Remembering the properties of the Bellman operators presented in section 2.2.1, we see that every time we apply the Bellman expectation operator T^π to any function $V \in \mathbb{R}^{|\mathcal{S}|}$, we are reducing its distance from the value function V^π of policy π. The application of the operator to the function in a state is called **backup**, and the application to all the states is called **sweep**. Given that we are performing a backup to every state before again applying the operator to certain state, we say that we are performing *synchronous backups*.
What we want is a way for moving from a policy π_1 to a policy π_2 such that $\pi_1 \leqslant \pi_2$. This can be achieved through the policy improvement theorem.

Theorem 2 (Policy Improvement Theorem)**.** *Let π and π' be any pair of deterministic policies such that $Q^\pi(s,\pi'(s)) \geqslant V^\pi(s) \quad \forall s \in \mathcal{S}$. Then the policy π' must be as good as, or better than π: $V^{\pi'} \geq V^\pi$.*

Therefore, given a policy π with value function V^π and action-value function Q^π, we can improve if we act greedily, namely if we take in every state the action that maximizes the action-value function: $\pi'(s) \in \operatorname{argmax}_{a\in\mathcal{A}} Q^\pi(s,a)$.
It should be now clear the functioning of policy iteration. Starting from any policy, we evaluate it, then we improve it by acting greedily, then we evaluate the new policy and so on. As Sutton and Barto highlight in their book [51], *because a finite MDP has only a finite number of policies, this process must converge to an optimal policy and optimal value function in a finite number of iterations.* It should be remarked that, during the policy evaluation step, we do not need to converge to the actual value function, but we can approximate it. In such case the method is called **Generalized Policy Iteration**.

Value Iteration

The idea behind **Value Iteration** is straightforward. Given that the application of the Bellman expectation operator of a policy π makes any vector (function) converge to the value function of π, and since the same property holds also for the Bellman optimality operator, we can apply it directly. Again, we are using synchronous backups and again we converge to the optimal policy for discounted infinite-horizon problems. Notice that, differently from policy iteration, now some value functions (of intermediate steps) might not correspond to any policy. However, convergence is asymptotically as opposed to Policy Iteration.

Linear Programming

Although **Linear Programming** defines a different technique from those of **Dynamic Programming**, we have decided to include it here because, similarly to Dynamic Programming techniques, it requires a complete knowledge of the MDP in order to be applied. Simply, there is a theorem that states that the control problem can be solved by finding the solution of a certain linear program:

Theorem 3 (Infinite-horizon Linear Programming)**.** *Given a γ-discounted infinite horizon MDP $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$, we can formalize the problem of computing the optimal value function V^* as a linear program:*

$$\begin{aligned} &\min_{V \in \mathbb{R}^{|\mathcal{S}|}} \sum_{s \in \mathcal{S}} \mu(s) V(s) \\ &s.t. \quad V(s) \geqslant R(s,a) + \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s') \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \end{aligned}$$

Then, V^* is the solution of this LP (Linear Program).

The original formulation of LP for solving MDPs in the discounted infinite-horizon case was introduced in [13]. The LP above has $|\mathcal{S}|$ variables and $|\mathcal{S} \times \mathcal{A}|$ constraints. Given that the model is known, once that V^* is found, the optimal policy can easily be computed. As all the LPs, it admits a dual formulation provided by the following program:

$$\begin{aligned} &\max_{d \in \mathbb{R}^{|\mathcal{S} \times \mathcal{A}|}} \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} d(s,a) R(s,a) \\ &s.t. \quad \sum_{a' \in \mathcal{A}} d(s',a') = \mu(s') + \gamma \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} d(s,a) p(s'|s,a) \quad \forall s \in \mathcal{S} \\ &\qquad d(s,a) \geqslant 0 \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \end{aligned}$$

Thanks to Theorem 6.9.1 of [45], if we define the policy $\bar{\pi}(a|s) := \frac{d(s,a)}{\sum_{a' \in \mathcal{A}} d(s,a')}$, then variable d can be interpreted as the discounted occupancy measure of $\bar{\pi}$:

$$d(s,a) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a | \mu, \bar{\pi}, P).$$

Clearly, the objective coincides with the dot product $\langle d, R \rangle$, which is the expected discounted sum of rewards, and this is why we aim to maximize it. The constraints simply ensure that d has the meaning described above. The optimal policy can be obtained by deterministically choosing in each state the action with maximal occupancy measure: $\pi^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} d(s,a)$. In the worst case, LP has better computational guarantees than dynamic programming, but in practice becomes impractical with a smaller number of states.

2.2.3. Reinforcement Learning

RL techniques are necessary every time that the true MDP is either unknown or too complex (e.g. state-action space continuous). In such cases, Dynamic Programming techniques, which are expensive and usually provide exact solutions, cannot be applied. Instead, RL algorithms are less computationally expensive and can deal with these problems with various tricks. Reinforcement Learning techniques can be described from different point of views, by using the characteristics listed below:

Model–free vs Model–based An algorithm is Model-based when it keeps an estimate $\hat{p}$ of the transition model p, and, then, uses the model for solving the prediction or control problem. In other words, it builds a model from the experience, and then plans the policy or computes the value function from the model. An algorithm which is not Model-based is called Model-free, and simply tries to plan the policy or compute the value function directly from the experience;

On–policy vs Off–policy According to [51], *On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to generate the data*;

Online vs Offline An Online method performs the learning while exploring the environment, in an online manner; therefore, it collects a datum, adjusts its parameters, then collects another datum, and so on. Instead, an Offline method simply uses a batch of data previously collected for the learning, and does not interact with the environment anymore;

Tabular vs Function Approximation According to [51], a Tabular Method can be applied to MDPs in *which the state and action spaces are small enough for the approximate value functions to be represented as arrays, or tables*. In other words, we can afford to store a different value for every state-action pair. When this is not feasible, and, therefore, we are forced to approximate the table with a function, then, the method is said to be a Function Approximation method;

Value–based vs Policy–based vs Actor–Critic This distinction is made for identifying what is actually being approximated. In the case of Value-based methods, we perform function approximation of the (action-) value function; instead, when the method is Policy-based, we approximate the policy, thus we aim to learn a parametric policy. Usually, we call Critic the component that updates the set of parameters of the value function, and Actor the component that updates the parameters of the policy. Therefore, an Actor-Critic algorithm approximates both the value function and the policy.

This section starts by presenting the most used tabular methods in Subsection 2.2.3, then it moves to present value-based methods, then policy-based methods.

Tabular methods

As aforementioned, these methods can be used in "simple" problems, and in many cases they can find optimal solutions. We will present the two most widespread methods, **Monte-Carlo RL** (MC) and **Temporal Difference RL** (TD).

Monte-Carlo Reinforcement Learning Monte-Carlo methods are powerful methods that can address both the prediction and the control problem. Before looking at the technicalities of the methods, it is better to clearly define all the assumptions and the problem they aim to solve. It should be remarked that the environment is not known here, but can only be explored. Exploration produces experience, which can be seen as a sequence:

$$\langle s_1, a_1, r_1, s_2, a_2, r_2, \dots \rangle,$$

which might be unlimited, and, in such case, requires online learning algorithms. Such sequence is called a **trajectory**. However, we restrict here to cases in which the MC algorithm has some sample finite-length **trajectories** available, which can be collected previously (and thus provided as a batch) or in an online manner. The idea is that MC uses such experience to solve the prediction or control task. MC does not estimates the model, but uses directly the collected experience; in other words, it is a model-free

Input: A policy π to be evaluated
Output: An estimate of V^π
Initialize: $V \in \mathbb{R}^{|\mathcal{S}|}$ arbitrarily, $returns(s) \leftarrow$ empty $\forall s \in \mathcal{S}$; `/* An empty list */`
while not all episodes generated **do**
  Generate an episode using π
  for each state s in the episode **do**
    $R \leftarrow$ return following the first occurrence of s
    $V(s) \leftarrow$ average($Returns(s)$)
  end for
end while

Algorithm 1: Monte-Carlo First Visit

method. Moreover, it learns from complete episodes, i.e., it uses trajectories as a whole, and does not update the estimates before the end of a trajectory. Let us consider the following definition of bootstrapping, taken from [51]: *We call Bootstrapping the general idea of updating estimates on the basis of other estimates.* For instance, we can say that some DP methods (e.g. policy iteration) perform bootstrapping since they update some estimates (of the policy) based on other estimates (of the value function). MC is a method that does not perform bootstrapping, while TD, presented in the next section, does. Let us now define MC for episodic tasks.
It relies on the idea of estimating the value function through its sample mean, i.e. given in general n realizations $x_1, x_2, \dots, x_n \overset{i.i.d.}{\sim} X$ of a certain random variable X, the idea of MC is to estimate the expected value of X through its sample mean:

$$\hat{X} = \frac{1}{n} \sum_{i \in [n]} x_i.$$

Thus, given N sample trajectories of length H (the horizon of the episode) collected under a policy π:

$$\begin{aligned}
\tau_1 &= \langle s_{1,1}, a_{1,1}, r_{1,1}, s_{1,2}, a_{1,2}, r_{1,2}, \dots, s_{1,H}, a_{1,H}, r_{1,H} \rangle \\
\tau_2 &= \langle s_{2,1}, a_{2,1}, r_{2,1}, s_{2,2}, a_{2,2}, r_{2,2}, \dots, s_{2,H}, a_{2,H}, r_{2,H} \rangle \\
&\dots \\
\tau_N &= \langle s_{N,1}, a_{N,1}, r_{N,1}, s_{N,2}, a_{N,2}, r_{N,2}, \dots, s_{N,H}, a_{N,H}, r_{N,H} \rangle
\end{aligned}$$

to solve the prediction problem, we can implement MC in the so-called **First-Visit** version, which estimates V^π, the value function, defined as the expected value of the return, with its sample mean, by averaging the returns only of the first occurrence of the considered state in a sample. Instead, MC **Every-Visit** uses all the occurrences of every sample trajectory. The algorithms' pseudocodes as presented in [51] are given in Alg. 1 and Alg. 2. In this way, we are able to explore the environment using policy π, and

Input: A policy π to be evaluated
Output: An estimate of V^π
Initialize: $V \in \mathbb{R}^{|\mathcal{S}|}$ arbitrarily, $returns(s) \leftarrow$ empty $\forall s \in \mathcal{S}$; `/* An empty list */`
while not all episodes generated **do**
  Generate an episode using π
  for each state s in the episode **do**
    for each occurrence of state s in the episode **do**
      $R \leftarrow$ return following the first occurrence of s
      Append R to $Returns(s)$
      $V(s) \leftarrow$ average($Returns(s)$)
    end for
  end for
end while

Algorithm 2: Monte-Carlo Every Visit.

then to estimate its value function. Notice that this setting can work for both discounted and undiscounted episodic MDPs. It should be noticed that MCFV (Monte-Carlo First-Visit) provides unbiased estimates, while MCEV (Monte-Carlo Every-Visit) is biased but consistent. Notice also that this is an On-Policy method, since we are estimating the policy with which we are exploring the environment. An efficient way for implementing the methods is by computing the means incrementally, i.e. with the formula:

$$\hat{V}_t \leftarrow \hat{V}_{t-1} + \frac{1}{t}(v - \hat{V}_{t-1}),$$

where v is the return, so that we can get rid of the samples once used. With regards to the control problem, we can realize that, in absence of the model (the transition model p), we are not able to recover the optimal policy π^* from the optimal value function V^*. Indeed, we are missing some information:

$$\pi^*(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \Big\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^*(s) \Big\} \quad \forall s \in \mathcal{S}.$$

Even if we have the reward function (easier to estimate if we assume that it is deterministic), the absence of p does not allow to get π^*. Instead, the action-value function is sufficient for retrieving the optimal policy even without the transition model and the reward function:

$$\pi^*(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a) \quad \forall s \in \mathcal{S}.$$

Therefore, once that we estimate the action-value function, we can retrieve its policy. Thus, we can use MCFV (or MCEV) to estimate directly the Q^π, namely to perform policy evaluation of π. In order to guarantee a persistent exploration, we need to use

what [51] call a *soft policy*.

Definition 20 (Soft Policy)**.** *A policy π is said to be Soft if $\pi(a|s) > 0 \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$ and it gradually shifts closer and closer to a deterministic policy.*

Definition 21 (ϵ-Soft Policy)**.** *An ϵ-soft policy is a soft policy for which $\pi(a|s) \geqslant \frac{\epsilon}{|\mathcal{A}|} \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$ for some $\epsilon > 0$.*

Definition 22 (ϵ-Greedy Policy)**.** *An ϵ-greedy policy π is an ϵ-soft policy defined as:*

$$\pi(a|s) := \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} & \text{if} \quad a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

Basically, ϵ-greedy policies are the closest to a deterministic policy among the ϵ-soft policies. It can be proved an analogous of Theorem 2 for ϵ-greedy policies:

Theorem 4 (ϵ-Greedy Policy Improvement Theorem)**.** *For any ϵ–greedy policy π, the ϵ–greedy policy π' with respect to Q^π is an improvement.*

It should now be clear that, thanks to MCFV (MCEV) applied to the Q function, we are able to estimate the action-value function of a certain policy π. Next, thanks to Theorem 4, we can perform policy improvement over a given policy π. We are therefore able to perform **Policy Iteration** similarly to what we did in Dynamic Programming.
Up to now, we have talked about how to solve the prediction and the control problem through On-Policy Monte-Carlo algorithms. Now we present the notion of Off-Policy learning with MC. By definition of **Off-Policy**, we aim to learn a target policy π while exploring the environment through a different behavior policy $\bar{\pi}$. Notice that these methods usually suffer from a big variance and are slow to converge [51]. However, they are more general than On-Policy methods, and allow to re-use the experience previously collected. The main notion behind Off-Policy MC is **Importance Sampling**:

Definition 23 (Importance Sampling)**.** *We call Importance Sampling the notion of estimating the expectation of a distribution different from the function model used for collecting the samples. In other words, given two probability distributions $P, Q \in \Delta$, we can compute the expected value of a certain function f under P by using the samples collected*

from Q:

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim P(\cdot)}[f(x)] :&= \sum_x P(x)f(x)\\ &= \sum_x \frac{Q(x)}{Q(x)}P(x)f(x)\\ &= \mathop{\mathbb{E}}_{x\sim Q(\cdot)}\Big[\frac{P(x)}{Q(x)}f(x)\Big].\end{aligned}$$

We can think at the ratio $\frac{P(x)}{Q(x)}$ as the importance weight that allows us to keep into consideration that the sampling distribution is different. Given that a (action-) value function is defined as the expected value of the return under the probability distribution $\mathbb{P}_{p,\pi}$ induced by the transition model p and the exploration policy π, we can simply correct our MC algorithms for prediction and control by multiplying the returns to be averaged by the importance ratio:

$$Q^\pi(s,a) := \mathop{\mathbb{E}}_{\tau\sim\mathbb{P}_{p,\pi}}[v_t|s_t=s,a_t=a] = \mathop{\mathbb{E}}_{\tau\sim\mathbb{P}_{p,\bar{\pi}}}\Big[\prod_t \frac{\pi(s_t,a_t)}{\bar{\pi}(s_t,a_t)}v_t|s_t=s,a_t=a\Big],$$

where τ denotes the trajectory.

Temporal Difference Reinforcement Learning Temporal Difference (TD) methods can address both the prediction and the control problem. Similarly to MC, TD is model-free; however, it performs bootstrapping and it can learn online, i.e. it can learn without waiting until the end of an episode. Simply, TD estimates the return of an episode with the actual reward obtained plus the current estimate of the value function in the next state. The update rule is thus:

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)),$$

This is basically the same update rule of MC, i.e. the formula of the running average:

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t - V(s_t)),$$

where for MC $\alpha = \frac{1}{n}$, the number of returns collected. Instead, for TD, α has the meaning of learning rate. It is clear that we are replacing the return of the episode v_t with an estimate $r_{t+1} + \gamma V(s_{t+1})$, and this explains why TD performs bootstrapping. The quantity $r_{t+1} + \gamma V(s_{t+1})$ is called **TD target**, while $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is the **TD error**. We report in Algorithm 3 the pseudocode of the algorithm for solving the prediction problem as presented in [51].

Input: A policy π to be evaluated, the learning rate $\alpha \in (0, 1]$
Output: An estimate of V^π
Initialize: $V \in \mathbb{R}^{|\mathcal{S}|}$ arbitrarily, except that $V(terminal) = 0$
while not all episodes generated **do**
  Initialize S
  for each step of the episode **do**
    $A \leftarrow$ action sampled from π in S
    Take action A, observe R and S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  end for
end while

Algorithm 3: TD(0).

Input: The learning rate $\alpha \in (0, 1]$, a small $\epsilon > 0$
Output: An estimate of Q^*
Initialize: $Q \in \mathbb{R}^{|\mathcal{S} \times \mathcal{A}|}$ arbitrarily, except that $Q(terminal, \cdot) = 0$
while not all episodes generated **do**
  Initialize S
  choose A from S using the policy derived from Q (e.g. ϵ-greedy)
  for each step of the episode **do**
    Take action A, observe R and S'
    Choose A' from S' using the policy derived from Q (e.g. ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
    $S \leftarrow S'$
    $A \leftarrow A'$
  end for
end while

Algorithm 4: SARSA.

With regards to control problems, since we are considering model-free algorithms, we still need to estimate the action-value function instead of the value function in policy evaluation, and then apply policy improvement through ϵ-policy improvement, in order to guarantee continual exploration. The most famous algorithm for model-free TD RL control is `SARSA`: its name derives from the quintuple $\langle s, a, r, s', a' \rangle$ of observations needed for performing an update step of the algorithm. It is presented in Algorithm 4 the version presented in [51].

SARSA is guaranteed to converge to the optimal value function under certain conditions.

Theorem 5 (Convergence of SARSA)**.** *SARSA converges to the optimal action-value function if:*

- *we choose a GLIE*[4] *sequence of policies* π_t*;*
- *we use a Monroe-Robinson sequence of step-sizes* α_t*, namely* $\sum_{t=1}^{+\infty} \alpha_t = +\infty$ *and* $\sum_{t=1}^{+\infty} \alpha_t^2 < +\infty$

We can apply importance sampling to the TD update in order to make SARSA an offline algorithm; the policies ratio consists now in a single ratio, thus there is much lower variance than in MC:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(r_{t+1} + \gamma \frac{\pi(a_{t+1}|s_{t+1})}{\bar{\pi}(a_{t+1}|s_{t+1})} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big).$$

However, Q-learning [58] is the most used off-policy TD algorithm for control. In this case, we directly learn the optimal policy while using any ϵ-greedy policy as behavioral policy. The update rule in this case is[5]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(r_{t+1} + \gamma \max_{a_{t+1} \in \mathcal{A}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big).$$

Comparison between MC and TD It is important to compare MC with TD by considering the bias-variance trade-off. It is clear that MCFV, since it uses all the data collected during the whole episode, is unbiased, while TD is biased. However, MC has a lot of variance since it depends on many random events (all the actions of the episode), while TD has much lower variance since it depends on only one random event. Moreover, we can see that TD exploits the Markovianity property, while MC does not, thus MC performs better in environments where such property does not hold. Finally, we recall that TD performs bootstrapping, and that it is an online algorithm, while MC is not. We can manage the bias-variance trade-off between MC and TD by considering an algorithm which is somewhat in the middle between them, which is called TD(λ). It defines the n-step return as $v_t^{(n)} := r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$ and then uses a parameter λ to average all such returns and then use them to perform the update. The λ-return is defined as: $v_t^\lambda := (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} v_t^{(n)}$. The forward-view TD($\lambda$) thus uses the update rule:

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t^\lambda - V(s_t)),$$

while a more efficient backward-view TD(λ) can be implemented through the mechanism of the eligibility traces. When $\lambda = 0$, then the update coincides with the usual TD update,

[4]By GLIE we define a Greedy in the Limit of Infinite Exploration, that is that all the state-action pairs are visited an infinite number of times and the sequence of policies converges to a greedy policy.

[5]It should be remarked that, in case of Q-learning, we do not need GLIE conditions.

while for $\lambda = 1$ the sum of errors becomes the MC error, thus TD(1) is more or less equivalent to MCEV. The bias-variance trade-off is managed thanks to the parameter λ. A similar reasoning can be applied to the control problem, where we have the SARSA(λ) algorithm.

Value-based Function Approximation

In the case the state space $\mathcal{S}$ and/or the action space $\mathcal{A}$ are very large and/or continuous, or the time is continuous, then, we cannot use a tabular representation for the value function, since it would be unfeasible or impossible. The idea is to approximate the value function by introducing a vector of weights $\boldsymbol{w}$ with which parametrize it:

$$v_{\boldsymbol{w}} \approx v_\pi,$$
$$q_{\boldsymbol{w}} \approx q_\pi.$$

There are three types of function approximation we can adopt, namely $v_{\boldsymbol{w}} : \mathcal{S} \to \mathbb{R}$, $q_{\boldsymbol{w}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $q_{\boldsymbol{w}} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$. We want the function approximator to be able to generalize well and we will update the value of $\boldsymbol{w}$ at each step of the algorithm. The task is similar to that of supervised learning, but there is an important difference: the experience collected through time is not independent. Moreover, the target can now be non-stationary, since we improve the policy we learn, and with it also the value function improves and thus changes with time. We want to minimize the following loss function:

$$L(\boldsymbol{w}) = \frac{1}{2} \mathop{\mathbb{E}}_{S \sim d_\pi(\cdot)} \Big[(v_\pi(S) - v_{\boldsymbol{w}}(S))^2 \Big],$$

where d_π is the on-policy distribution over states induced by policy π. In other words, we minimize the mean square error (MSE) between the true value function v_π and its approximation $v_{\boldsymbol{w}}$.

Incremental Methods In the case of incremental methods, the loss can be minimized through Gradient Descent or Stochastic Gradient Descent; the gradient of the loss is:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = - \mathop{\mathbb{E}}_{S \sim d_\pi} \Big[(v_\pi(S) - v_{\boldsymbol{w}}(S)) \nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S) \Big],$$

and the update rule of the GD simply $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} L(\boldsymbol{w})$. The Stochastic Gradient Descent (SGD) uses one of the input samples at each iteration to perform the update. We can consider on-policy/off-policy and for prediction/control incremental methods. The idea of the methods for the prediction problem is to approximate the v_π using the rewards,

while methods for the control problem approximate q_π with the rewards and then perform policy improvement. Clearly, v_π and q_π are not provided, thus we need to estimate them with the rewards. We can consider the following incremental methods:

On-policy for prediction We substitute a target for v_π, and the target depends on the method adopted; for MC we use as target the return G_t and the update rule becomes $\Delta \boldsymbol{w} = \alpha(G_t - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$. For TD, we use the TD target $R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1})$ and as update rule $\Delta \boldsymbol{w} = \alpha(R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ while for TD(λ) we use the λ-return G_t^λ, so we have $\Delta \boldsymbol{w} = \alpha(G_t^\lambda - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$. It should be remarked that the MC incremental on-policy algorithm for prediction is a **gradient** method, because $(G_t - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ corresponds to the gradient of a certain objective function, while TD and TD(λ) are called **semi-gradient** methods since the update rule does not correspond to the gradient of any objective function.

Off-policy for prediction In this case, we explore through the behavioral policy b and we want to learn the policy π, thus we can use importance sampling for MC, TD and TD(λ). The update rule of MC become $\Delta \boldsymbol{w} = \alpha\Big(\prod_{l=t}^{T-1} \rho_l\Big)(G_t - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$, where $\rho_t := \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ is the importance weight. For TD we have $\Delta \boldsymbol{w} = \alpha\rho_t(R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ and for TD(λ) we have $\Delta \boldsymbol{w} = \alpha\Big(\prod_{l=t}^{T-1} \rho_l\Big)(G_t^\lambda - v_{\boldsymbol{w}}(S_t))\nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$.

On-policy for control As aforementioned, in the case of control we estimate q_π and then we perform policy improvement. The update rules are the same as those for on-policy prediction methods except for that we have to replace $v_{\boldsymbol{w}}$ with $q_{\boldsymbol{w}}$. The pseudo-code for the semi-gradient SARSA(0) algorithm is presented in Algorithm 5.

Off-policy for control In this case we can directly learn q_* while executing an exploration policy. We can consider for instance the Q-learning target for q_*, namely $R_{t+1} + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{w}}(S_{t+1}, a')$, to obtain the update rule $\Delta \boldsymbol{w} = \alpha(R_{t+1} + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{w}}(S_{t+1}, a') - q_{\boldsymbol{w}}(S_t, A_t))\nabla_{\boldsymbol{w}} q_{\boldsymbol{w}}(S_t, A_t)$. The semi-gradient Q-learning algorithm for control is shown in Algorithm 6.

It is worthy to introduce what is called the *Deadly Triad*, i.e. bootstrapping, non-linear function approximation and off-policy property. In presence of all these three properties, no incremental method is proved to converge.

Input: The learning rate $\alpha \in (0, 1]$
Output: An estimate $q_{\boldsymbol{w}}$ of q^*
Initialize: $\boldsymbol{w}$ arbitrarily
while not all episodes generated **do**
  Initialize S
  choose A from S using the exploration policy
  for each step of the episode **do**
    Take action A, observe R and S'
    Choose A' from S' using the exploration policy
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(R + \gamma q_{\boldsymbol{w}}(S', A') - q_{\boldsymbol{w}}(S, A))\nabla_{\boldsymbol{w}} q_{\boldsymbol{w}}(S, A)$
    $S \leftarrow S'$
    $A \leftarrow A'$
  end for
end while

Algorithm 5: Semi-gradient SARSA(0).

Input: The learning rate $\alpha \in (0, 1]$
Output: An estimate $q_{\boldsymbol{w}}$ of q^*
Initialize: $\boldsymbol{w}$ arbitrarily
while not all episodes generated **do**
  Initialize S
  choose A from S using the exploration policy
  for each step of the episode **do**
    Take action A, observe R and S'
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(R + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{w}}(S, a') - q_{\boldsymbol{w}}(S, A))\nabla_{\boldsymbol{w}} q_{\boldsymbol{w}}(S, A)$
    Choose A' from S' using the exploration policy
    $S \leftarrow S'$
    $A \leftarrow A'$
  end for
end while

Algorithm 6: Semi-gradient Q-learning.

Policy Gradient

Previously, we defined the difference between value-based, policy-based and actor-critic methods. In the previous section, we saw value-based methods, since we were approximating the value functions. Now we consider policy-based methods, in which there is a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ that we use for parametrizing the policy $\pi_{\boldsymbol{\theta}}$. In order to measure the quality of a policy $\pi_{\boldsymbol{\theta}}$, we use a scalar objective like the expected return:

$$J(\boldsymbol{\theta}) := \mathop{\mathbb{E}}_{S_0 \sim \mu_0(\cdot)}\Big[v_{\pi_{\boldsymbol{\theta}}}(S_0)\Big] = \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t R_t | S_0 \sim \mu_0, \pi_{\boldsymbol{\theta}}\right].$$

The expected return can be considered in two equivalent ways: the **trajectory view** and the **occupancy view**. The trajectory view uses the probability of a trajectory $\tau = (S_0, A_0, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, S_T)$ under policy $\pi_{\boldsymbol{\theta}}$:

$$p_{\boldsymbol{\theta}} = \mu_0(S_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(A_t|S_t) p(S_{t+1}|S_t, A_t),$$

and the trajectory return

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r(S_t, A_t),$$

to rewrite the expected return as the expectation of the trajectory return under its probability

$$J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}}\big[G(\tau)\big].$$

Instead, the occupancy view exploits the definition of γ-discounted occupancy:

$$d_{\pi_{\boldsymbol{\theta}}}(s) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(S_t = s | S_0 \sim \mu_0, \pi_{\boldsymbol{\theta}}),$$

to rewrite the expected return as

$$J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{S \sim d_{\pi_{\boldsymbol{\theta}}}(\cdot) \\ A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)}}\big[r(S, A)\big],$$

where the term $\frac{1}{1-\gamma}$ is needed to make the occupancy a probability measure. We have a theorem that states the equivalence between the two.

Theorem 6 (Equivalence between trajectory and occupancy view)**.** *The two representa-*

Input: The learning rate $\alpha \in (0, 1]$, a shape for $\pi_{\boldsymbol{\theta}}$
Output: An estimate $\pi_{\boldsymbol{\theta}}$
Initialize: $\boldsymbol{\theta}$ arbitrarily
for all the iterations $k = 1, \dots, K$ **do**
  sample m trajectories $\tau^i = (S_0^i, A_0^i, S_1^i, A_1^i, \dots, S_{T-1}^i, A_{T-1}^i, S_T^i)$ following $\pi_{\boldsymbol{\theta}}$
  compute the RF gradient estimate $\hat{\nabla}_{\boldsymbol{\theta}}^{RF} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \hat{g}_i$; `/* Sample mean */`
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \hat{\nabla}_{\boldsymbol{\theta}}^{RF} J(\boldsymbol{\theta})$
end for

Algorithm 7: REINFORCE (RF)

tions are equivalent:

$$\mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \big[G(\tau)\big] = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{S \sim d_{\pi_{\boldsymbol{\theta}}}(\cdot) \\ A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)}} \big[r(S, A)\big].$$

The problem of finding the best $\boldsymbol{\theta}$ is an optimization problem; we aim to maximize $J(\boldsymbol{\theta})$. We can thus solve it with the gradient ascent algorithm $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$ where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the policy gradient. We will now present some famous **white-box** approaches, i.e. approaches that compute the gradient analytically. The first one is **REINFORCE** [59], and it works by proposing a certain shape of the policy function $\pi_{\boldsymbol{\theta}}$ like a softmax policy $\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\boldsymbol{x}(s,a)^\intercal \boldsymbol{\theta}}}{\int_{a'} e^{\boldsymbol{x}(s,a')^\intercal \boldsymbol{\theta}} da'}$ or a gaussian policy $\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{a-\mu_{\boldsymbol{\theta}}(s)}{\sigma}\right)^2\right)$ and then noticing that the policy gradient coincides with:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\Big(\sum_{l=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_l|S_l) \Big) \Big(\sum_{t=0}^{T-1} \gamma^t r(S_t, A_t) \Big) \right].$$

REINFORCE simply replaces the expectation with regards to $p_{\boldsymbol{\theta}}$ with its sample mean. The algorithm is presented in Algorithm 7 where we have used:

$$\hat{g}_i := \Big(\sum_{l=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_l^i|S_l^i) \Big) \Big(\sum_{t=0}^{T-1} \gamma^t r(S_t^i, A_t^i) \Big).$$

REINFORCE suffers from a large variance. In [6], Baxter & Bartlett introduce **G(PO)MDP**, a policy gradient algorithm that suffers from less variance than REINFORCE. It is able to reduce the variance by applying the causality property: Baxter & Bartlett noticed that the rewards collected in the past do not depend on the actions played in the future, thus

they rewrite the policy gradient as:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{T-1} \left(\gamma^t r(S_t, A_t) \sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_l | S_l) \right) \right].$$

The pseudo code of G(PO)MDP is the same of REINFORCE 7, the only difference is line 5 in which $\hat{\nabla}_{\boldsymbol{\theta}}^{RF} J(\boldsymbol{\theta})$ is replaced by $\hat{\nabla}_{\boldsymbol{\theta}}^{G(PO)MDP} J(\boldsymbol{\theta})$; simply, we can use the following definition for $\hat{g}_i$ instead of the previous one:

$$\hat{g}_i := \sum_{t=0}^{T-1} \gamma^t r(S_t^i, A_t^i) \sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_l^i | S_l^i).$$

A nice trick for further reducing the variance is to use a **baseline** $\boldsymbol{b}(\tau) \in \mathbb{R}^d$:

$$J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) \odot (G(\tau) - \boldsymbol{b}(\tau)) \right]$$

where $\odot$ represents the element-wise product. Up to now, we have considered only finite-length trajectories. There is a theorem proved in [52] that allows to compute analytically the policy gradient in the case of infinite trajectories.

Theorem 7 (Policy Gradient Theorem)**.** *For an infinite-horizon MDP, let $\pi_{\boldsymbol{\theta}}$ be a stochastic policy differentiable in $\boldsymbol{\theta}$, then the policy gradient is given by*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{S \sim d_{\pi_{\boldsymbol{\theta}}}(\cdot) \\ A \sim \pi_{\boldsymbol{\theta}}(\cdot | S)}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) q_{\pi_{\boldsymbol{\theta}}}(S, A) \right]$$

Since G(PO)MDP still has a large variance, and since Th. 7 provides a form for $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ which includes the value function, then we can introduce a **critic** component to estimate $q_{\boldsymbol{w}} \approx q_{\pi_{\boldsymbol{\theta}}}$ and thus use an actor-critic algorithm. If we use a linear function approximation $q_{\boldsymbol{w}} = \boldsymbol{x}(s, a)^\intercal \boldsymbol{\theta}$, where $\boldsymbol{x}(s, a)$ are the features, then the pseudo-code of an actor-critic algorithm is provided in Alg. 8.

2.3. Inverse Reinforcement Learning

In this section, we introduce the fundamental notion of *Inverse Reinforcement Learning* (IRL). To do so, we will start by defining Imitation Learning, which is the main discipline that comprises all the techniques of IRL, and Behavioral Cloning, which is the main alternative to IRL when tryng to solve an Imitation Learning problem.

Input: The learning rates $\alpha, \beta \in (0, 1]$, respectively for the actor and the critic
Output: An estimate $\pi_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{w}}$
Initialize: $\boldsymbol{\theta} \in \mathbb{R}^n, \boldsymbol{w} \in \mathbb{R}^d$ arbitrarily
for each episode **do**
  initialize S
  sample $A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$
  take action A, observe R and S'
  for each step of the episode **do**
    sample $A' \sim \pi_{\boldsymbol{\theta}}(\cdot|S')$
    $\delta \leftarrow R + \gamma q_{\boldsymbol{w}}(S', A') - q_{\boldsymbol{w}}(S, A)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta\delta\nabla_{\boldsymbol{w}} q_{\boldsymbol{w}}(S, A)$; /* Update the critic */
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) q_{\boldsymbol{w}}(S, A)$; /* Update the Actor */
    $S \leftarrow S'$
    $A \leftarrow A'$
    take action A, observe R and S'
  end for
end for

Algorithm 8: Action-Value Actor-Critic

2.3.1. Imitation Learning

According to [43], *the purpose of imitation learning is to efficiently learn a desired behavior by imitating an expert's behavior*. In other words, this is the setting where we want to teach to our artificial intelligent agent to solve a certain task, and we have an expert available. An expert, usually a human expert, is defined as any entity that knows how to optimally solve a certain task. We aim to teach to our agent what to do based on what the expert does. As an example, let us consider a robotic arm agent, and the goal is to teach it how to play table tennis. If we know a professional table tennis player, we can use him to teach to the arm how to play. Notice that this setting is rather different from that of Reinforcement Learning, where we teach the agent what to do simply by defining its objective using a reward function.

Understanding the Problem

According to the survey [43], we have to ask different questions when deciding to solve a problem using Imitation Learning (IL) techniques. The general aspects we have to consider are:

1. Why and when should imitation learning be used?
2. Who should demonstrate?
3. How should we record data of the expert demonstrations?

4. What should we imitate?

After answering these questions, we will have a clear understanding of the problem, and if we are still convinced to use an Imitation Learning technique, then we can begin to reason about the algorithmic aspects:

1. How should we represent the policy?
2. How should we learn the policy?

Difference between Imitation Learning and Supervised Learning

It should be remarked that Imitation Learning is rather different from Supervised Learning. Indeed, in IL there might be some structural constraints in the solution and gathering new samples is usually expensive. Thus, we have to minimize as much as we can this task.

Problem Definition

Formally, in an IL problem there is an expert demonstrator that provides trajectories that we use to teach our agent what to do. If we denote the features of both agent and environment at a certain time instant by $\boldsymbol{\phi}_t$, then we can write the expert demonstration (trajectory) as $\boldsymbol{\tau} = [\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_T]$. Notice that the trajectory is T time steps long ($\boldsymbol{\phi}_0$ is the initial conditions). The demonstrations often take place in different conditions, so each trajectory has to be considered in a certain context. We introduce a vector $\boldsymbol{s}$ to denote the context in which the demonstration has taken place. Therefore, in an IL setting we have a dataset of demonstrations $\mathcal{D} = \{(\boldsymbol{\tau}_1, \boldsymbol{s}_1), (\boldsymbol{\tau}_2, \boldsymbol{s}_2), \dots, (\boldsymbol{\tau}_N, \boldsymbol{s}_N)\}$, where N is the number of demonstrations provided by the expert. If we denote by $q(\boldsymbol{\phi})$ the distribution over features induced by the expert's policy and $p(\boldsymbol{\phi})$ the distribution over features induced by the learner's policy, then the goal can be defined as that of minimizing the distance between $q(\boldsymbol{\phi})$ and $p(\boldsymbol{\phi})$ according to a certain metric D between probability distributions. In symbols, we aim to find policy π^* such that:

$$\pi^* \in \operatorname{argmin} D\big(q(\boldsymbol{\phi}) || p(\boldsymbol{\phi})\big)$$

Design Choices

The following discussion follows Chapter 2 of [43]. When we aim to solve a problem using an IL method, there are many design choices we must take:

- Access to the reward function: imitation learning or reinforcement learning;

- Parsimonious description of the desired behavior: behavioral cloning or inverse reinforcement learning;
- Access to system dynamics: model-based or model-free;
- Similarity measure between policies;
- Policy representation;
- Features.

In the following paragraphs, we will discuss each of the presented decision points.

Access to the reward function: imitation learning or reinforcement learning We have to distinguish the case in which the learner has access to the expert demonstrations only from the case in which the learner can access both expert demonstrations and a reward signal that would allow him to perform forward RL directly. If we have a reward signal, then we might solve the learning problem simply through RL, without caring about the expert. But if we have also the expert, we can exploit it to limit the exploration needed to apply RL. Therefore, the main trend when having both of them is to initialize a policy using expert demonstrations and then refine it through reinforcement learning.

Parsimonious description of the desired behavior: behavioral cloning or inverse reinforcement learning The main question here is to identify the most parsimonious description of the policy. The two main and contrasting methods for doing so are Behavioral Cloning and Inverse Reinforcement Learning, which will be presented respectively in Section 2.3.2 and Section 2.3.3. Let us mention here just an high-level difference among the two. Behavioral Cloning aims to directly learn a policy that maps inputs (and contexts) to actions/control inputs, and it can be usually computed through supervised learning methods. Instead, Inverse Reinforcement Learning concerns the recovering of a reward function from expert demonstrations that completely encodes the policy we aim to learn. In other words, here we do not learn directly the policy, but we learn a reward function that implicitly represents it. Implicitly here means that we can recover it through forward RL over the learned reward function.

Access to system dynamics: model-based or model-free The difference between model-based and model-free methods concerns the modelling or not of the system dynamics. Model-based methods learn a forward model of the environment and then use it to learn the policy. Instead, Model-free methods directly learn the policy without mod-

elling the dynamics. The distinction is the same presented when talking about RL. In model-free methods, the system dynamics is encoded implicitly inside the learned policy. Behavioral cloning methods have focused on model-free for years. Instead, in model-based methods the system dynamics is explicitly modelled, and such model can be used by IRL methods. Model-free approaches have the advantage of being more efficient since they do not estimate the model, but they can be less precise. Instead, model-based approaches make the learning process data-efficient, but they are more computationally expensive.

Similarity measure between policies To understand what to learn, we need to consider a similarity measure between policies. However, since the expert's policy is not directly observable, we need to learn it from the demonstrations. Therefore, the problem of observability, which concerns what the learner/expert can observe about the environment, arises. We can have settings in which the trajectories of the expert are fully observable, thus, the learner observes both the states and the actions, but also settings in which only the states (or only the actions) can be observed by the learner. It is clear that partial observability can cause many problems during the learning process. Moreover, partial observability can be different also between expert and learner. The expert might partially observe the system state, the control inputs of the expert or the observations of the learner, while the learner might partially observe the system state, the expert's observations, the control inputs by the expert and the control inputs by the learner. An example in [43] of the problems that might arise is that in which a human expert performs a motion which goes around an obstacle that the learner does not observe. Clearly, the agent learns that doing this kind of circular motions is optimal, even in the absence of obstacles.

Policy representation It is fundamental to choose the most appropriate policy representation for the given task. To do so, we have to choose the best features that allow to solve the problem and, at the same time, to limit the complexity of the problem. There are three main kinds of policy representation in this context: symbolic-level abstraction, trajectory-level abstraction and action-state space abstraction. With regards to symbolic-level abstraction, the agent learns a policy that generates an option $o \in \mathcal{O}$ over time. Since in this context it is hard to model a complex task through a single movement/action, we make the policy a sequence of simple movements:

$$\pi : \boldsymbol{x}_t, \boldsymbol{s} \rightarrow [o_1, \ldots, o_T].$$

Instead, in trajectory-level abstraction, the agent learns a policy that maps a context $\boldsymbol{s}$ to a trajectory:

$$\pi : \boldsymbol{s} \rightarrow \boldsymbol{\tau}.$$

Finally, in action-state space abstraction the agent learns a policy that directly maps states and contexts to actions:

$$\pi : \boldsymbol{x}_t, \boldsymbol{s} \rightarrow a_t.$$

This is a fine-grained representation and is that used in RL. Another distinction is between *Monolithic* policies, when we consider a single abstraction level of policy, and *Hierarchical* policies, when we combine different levels of abstractions, and we learn lower-level policies for the primitive behavior and upper-level policies to plan a sequence of lower-level policies. Another distinction is between *Feedback* policies, that iteratively determine the desired action based on the feedback received by the environment, and *Open-Loop* policies, which learn the behavior based only on the initial input. Finally, we can distinguish policies based on their stationarity and stochasticity, analogously to what we did in RL.

Features We need to choose the best policy representation that allows to properly capture the desired behavior. In other words, we have to understand what should be matched between the expert and the learner. There are three main *Behavior Descriptors* that can be matched between the expert and the learner in IL. The first one is the **state-action distribution** $p(s, a)$. Given a dataset $\mathcal{D} = \{(\boldsymbol{s}_i, \boldsymbol{a}_i)\}$, we might learn the policy by supervised learning in this way. However, since state-action distribution only defines the short-term behavior, it might lead to a mismatch with long-term behavior. Another behavior descriptor is the **Trajectory Feature Expectation**: we aim to match features between trajectories, so that to keep into account long-term behavior. Because of stochasticity, we consider the expected value:

$$\mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\cdot)}[\boldsymbol{\phi}(\boldsymbol{\tau})] = \int p(\boldsymbol{\tau}) \boldsymbol{\phi}(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

A third behavior descriptor is the **Trajectory Feature Distribution**: we might aim to match $p(\boldsymbol{\tau})$ not only at the first moment (expectation) but also at higher moments.

2.3.2. Behavioral Cloning

Behavioral Cloning (BC) methods *learn a direct mapping from states/contexts to trajectories/actions without recovering the reward function* [43]. The presentation of the following contents is based on [43].

Problem Statement

BC aims to learn an optimal policy based on the strategy of the expert. As aforementioned, when talking about IL, a dataset of expert demonstrations can be made of trajectories and contexts $\mathcal{D} = \{(\boldsymbol{\tau}_1, \boldsymbol{s}_1), \ldots, (\boldsymbol{\tau}_N, \boldsymbol{s}_N)\}$ or of states and actions $\mathcal{D} = \{(\boldsymbol{s}_1, \boldsymbol{a}_1), \ldots, (\boldsymbol{s}_N, \boldsymbol{a}_N)\}$. By using such datasets, BC aims to learn a mapping from contexts to trajectories or from states to actions. The problem can easily be recast into a *Supervised Learning* problem in which we can learn the policy by solving a regression problem. In other words, we simply have to choose a shape/parametrization for the policy $\pi_{\boldsymbol{\theta}}$ and a loss function, and then optimize $\pi_{\boldsymbol{\theta}}$ w.r.t. $\boldsymbol{\theta}$ according to the defined loss function.

Design Choices

In addition to the design choices presented in section 2.3.1, when developing BC systems we have to reason also to [43]:

1. What surrogate loss function should be used to represent the difference in demonstrated and produced behavior?
2. What regression method should be used to represent the policy?

In other words, we need to understand which metric to use to evaluate the difference between the learned behavior and that shown in the dataset of demonstrations, and also the regression model to use that is complex enough to represent the desired behavior and simple enough for allowing cheap computations.

Model-Free Methods

We consider model-free methods in three different settings: *action-state space*, *learning trajectories* and *task-level planning*.

Model-Free Methods in Action-State Space Here, we consider the setting in which we model the control problem using states and actions. It is intuitive how to use supervised learning. We can implement a regressor for mapping states to actions, and such a function is our policy. However, there is the risk that this approach can fail since samples, differently from usual supervised learning tasks, are not independent, because taking an action instead of another one influences the subsequent sample. Therefore, if the regression method is not provided with a big and various enough amount of samples, then it will not learn how to properly generalize. Moreover, there is the problem of how

to recover from bad actions too. In literature, we can find recent successes using deep neural networks and recurrent neural networks in imitation learning.
It is important to notice that any learned policy will commit some mistakes. Because of the dependence between samples in the IL setting, even small errors might end up in huge errors. In other words, we might need to learn a policy for recovering from errors. Many approaches are present in literature for addressing this task. One way is to use the so-called **structured prediction** for learning a function that maps inputs $\boldsymbol{x}$ to structured outputs $\boldsymbol{y}$. Another approach is the **confidence-based approach**, which aims to ask the expert new demonstrations based on the confidence of a given state. A third approach is **DAGGER** (data aggregation approach) [49], which attempts to collect new expert demonstrations based on the state distribution induced by the learned policy.

Model-Free Methods for Learning Trajectories Now we move to trajectories, whose planning problem is really important because it provides the high-level logic for controlling lower-lever controllers that work in the state-action space. If we consider datasets made of trajectories and contexts $\mathcal{D} = \{(\boldsymbol{\tau}_1, \boldsymbol{s}_1), \ldots, (\boldsymbol{\tau}_N, \boldsymbol{s}_N)\}$, then we can use supervised learning to map contexts to trajectories. However, it should be remarked that we often need to ensure that the planned trajectory is feasible, thus we need to impose some constraints for doing so.
There are various trajectory representations in literature.

Keyframe/Via-Point Based Approaches We can decide to represent trajectories using keyframes, which are called via-points in the context of robotics;

Hidden Markov Models We can use probabilistic models to represent motion, and then we can use some algorithms for estimating the state transition matrix and the output probability matrix. Then, we can estimate a certain sequence given the initial state;

Dynamic Movement Primitives DMPs are representations motivated by differential equations of *well-defined attractor dynamics* [43] and ensure the smoothness and the continuity of the trajectory;

Probabilistic Movement Primitives Since expert behavior is often stochastic, DMPs are not able to represent it, thus ProMPs have been introduced: they represent movement as a distribution over trajectories;

Time-Invariant Dynamical Systems In literature we can find frameworks for representing task trajectories as time-invariant non-linear dynamical systems [28].

Every representation has its strengths and weaknesses.

Model-Free Methods for Task-Level Planning We have seen action-state space and trajectories. Now we see task-level planning, which is useful when a task requires complex motion, since it allows to plan the motion as a sequence of primitive motions. Given that the demonstrated trajectories present in the dataset might consist of sequences of different types of primitive motions, then it might be necessary to segment the demonstrated trajectories in order to retrieve the primitives. After segmentation, clustering is fundamental for learning multiple types of primitive motions. Therefore, after segmentation and clustering, we can *model the structure of the skill and learn the transition between primitive motions from the demonstrated behavior.*

Model-Based Methods

In IL, a problem that often arises is the *correspondence problem*, which concerns the different "embodiment", the different dynamics between the expert and the learner, which might cause the learner to learn something from the demonstrated trajectories. One way for solving this problem is that of explicitly learning the **forward dynamics model** of the system $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and then use it for learning. Various regression approaches can be applied to this problem like Gaussian Mixture and Gaussian Processes. However, it is also possible to avoid to learn the forward dynamics model of the system by using **iterative learning control** [56].

2.3.3. Inverse Reinforcement Learning

In Inverse Reinforcement Learning (IRL), we assume that the expert agent has an objective that can be modelled as the cumulative maximization of a reward signal over time. In other words, the expert agent is playing the optimal policy in a certain Markov Decision Process. If we are able to retrieve the reward function (the goal) of the expert agent, then we are able to subsequently train our learner with forward RL using the learned reward signal. It should be noticed that the definition of the reward function in the RL setting is provided by the engineer that trains the learner, and such manual choice is usually difficult to do. In this section, we will mostly reuse the notation introduced in Section 2.2.

Problem Definition

An *inverse reinforcement learning* problem [IRL, 42] is defined as a pair $(\mathcal{M}, \pi^E)$, where $\mathcal{M}$ is an MDP\R and π^E is an *expert's policy*. Informally, solving an IRL problem consists in finding a reward function $(r_h)_{h \in [\![H]\!]}$ making π^E optimal for the MDP\R $\mathcal{M}$ paired with

reward function r. Any reward function fulfilling this condition is called *feasible* and the set of all such reward functions is called *feasible reward set* [36, 39], defined as:

$$\mathcal{R}_{(\mathcal{M},\pi^E)} \coloneqq \Big\{(r_h)_{h\in[\![H]\!]} \;\Big|\; \forall h \in [\![H]\!] \,:\, r_h : \mathcal{S}\times\mathcal{A} \to [-1,1] \\ \wedge\, \forall (s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!] \,:\, A_h^{\pi^E}(s,a;r) \leqslant 0\Big\}. \tag{2.2}$$

We will omit the subscript $(\mathcal{M},\pi^E)$ whenever clear from the context.

Definition 24 (Feasible Set)**.** *We define Feasible Set the set of all the reward functions which are compatible with a certain expert policy. By compatibility it is meant that reward R makes expert policy π^E optimal in the setting.*

Formally, we can define an IRL instance as:

Definition 25 (IRL)**.** *A finite-horizon IRL instance can be defined as a tuple $\langle \mathcal{S},\mathcal{A},p,\mu_0,H,\pi^E\rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, p is the transition model, μ_0 is the initial state distribution, H is the time horizon and π^E is the expert policy. The goal is to retrieve a function $R:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ such that policy π^E is optimal in the RL problem $\langle \mathcal{S},\mathcal{A},p,\mu_0,H,R\rangle$.*

Notice that if we replace H with γ, we can move to the γ-discounted infinite horizon setting. As shown in [42], the IRL problem is ill-posed, because there are multiple reward functions that make the expert policy optimal. [2] shows that *if the true reward function R^E was available for purposes of evaluation, then the direct distance $||R^E - \hat{R}^E||$ is useless, since an MDP's optimal policy is invariant under affine transformations of the reward function* [50]. Comparing directly the policies might give problems too. Another possibility might be to measure the *Inverse Learning Error*:

Definition 26 (Inverse Learning Error)**.** *Given the expert policy π^E and the policy $\hat{\pi}^E$ induced by the learned reward function, then we define the Inverse Learning Error (ILE) as the distance between the value functions according to a certain p-norm:*

$$||V^{\pi^E} - V^{\hat{\pi}^E}||_p.$$

Methods for IRL

This section takes inspiration from section 4 of [2]. In Algorithm 9, we can see the general routine on which all the IRL algorithms are based.

The majority of IRL algorithms chooses a certain parametric shape for the reward function to return, iteratively computes the optimal policy in the MDP with the reward and

Input: IRL instance, dataset $\mathcal{D}$ of trajectories, π^E
Output: An estimate $\hat{R}^E$
Model the expert's observed behavior as the solution of an MDP whose reward function is not known
Initialize the parameterized form of the reward function using any given features (linearly weighted sum of feature values, distribution over rewards, or other)
Solve the MDP with current reward function to generate the learned behavior or policy
Update the optimization parameters to minimize the divergence between the observed behavior (or policy) and the learned behavior (policy)
Repeat the previous two steps till the divergence is reduced to a desired level.

Algorithm 9: Template for IRL

updates the reward by minimizing the distance between the observed and the learned policy. It is clear that any IRL algorithm is simply a method for choosing only one reward function among all the possible functions from the feasible set. There are many ways for doing so. According to [2], we can group IRL algorithms in 4: *Margin Optimization*, *Entropy Optimization*, *Bayesian Update*, *Classification and Regression*.

Margin Optimization Margin Optimization refers to methods that aim to learn the reward function with a certain margin. Intuitively, the better the margin the better it is. We aim to find the reward function that is better than others with a certain margin. Depending on the definition of margin, there are various families of methods.

Margin of optimal from other actions or policies The margin is here defined as the sum of the differences between the optimal action-value function and the optimal action-value function when the optimal action is removed over all the states:

$$\sum_{s\in\mathcal{S}}\Big(Q^{\pi}(s,a^*) - \max_{a\in\mathcal{A}\setminus\{a^*\}} Q^{\pi}(s,a)\Big),$$

where a^* is the optimal action for state s. We are choosing the reward function that induces a policy π for which the margin, namely the distance between the Q-function when playing π and the Q-function when optimal actions of π are removed, is maximum. We aim to find the reward for which changing the induced optimal policy provides the biggest decrement in action-value function. If the reward function is feature-based, i.e. it can be seen as, for instance, a linear combination of features $\boldsymbol{\phi}$: $r(s,a) = \boldsymbol{w}^{\intercal}\boldsymbol{\phi}(s,a)$, then we can rewrite the previous expression of the margin

using the definition of states distribution d:

$$\sum_{(s,a)\in\boldsymbol{\tau}_t}\Big(d(s)\boldsymbol{w}^\intercal\boldsymbol{\phi}(s,a)-\max_{\boldsymbol{\tau}\in(\mathcal{S}\times\mathcal{A})^H\setminus\{\boldsymbol{\tau}_t\}}d(s)\boldsymbol{w}^\intercal\boldsymbol{\phi}(s,a)\Big)$$

Where we have done the same with trajectories instead of actions. $(\mathcal{S}\times\mathcal{A})^H$ is the set of all the possible trajectories of length H. One of the first methods of this kind has been introduced in [42], the foundational work for this discipline.

Margin of observed from learned feature expectations Some methods aim to minimize the margin defined as the *feature expectation loss*:

$$\left|\boldsymbol{\mu}^{\phi}(\pi)-\hat{\boldsymbol{\mu}}^{\phi}(\mathcal{D})\right|$$

where $\boldsymbol{\mu}^{\phi}(\pi)$ is the expected feature count of all the features under policy π, defined as:

Definition 27 (Expected Feature Count)**.** *Given a policy π and the k-th feature function $\phi_k:\mathcal{S}\to\mathbb{R}$ of the reward function, we define the Expected Feature Count μ^{ϕ_k} as the value of the feature that we expect to see over time under π and under the visit occupancy d^π:*

$$\mu^{\phi_k}(\pi):=\sum_{t=0}^{+\infty}d^\pi(s_t)\phi_k(s_t,\pi(s_t))$$

$\hat{\boldsymbol{\mu}}^{\phi}(\mathcal{D})$ is the empirical feature count of the expert policy in the trajectories present in dataset $\mathcal{D}$. In other words, the idea behind these methods is to find the reward function that provides an optimal policy whose expected feature count is as close as possible to the empirical one of the expert policy. Features summarize states, and we aim to visit states with π (optimal under the learned reward function) so that the expected features are close to the expert ones. Two foundational methods are MAX-MARGIN and PROJECTION [1].

Observed and learned policy distributions over actions Another approach is to minimize the distance between the empirical expert policy and the learned policy in every state:

$$\hat{\pi}^E(a|s)-\pi(a|s).$$

HYBRID-IRL is an algorithm of this kind [41].

Entropy Optimization The methods in this subsection exploit the *maximum entropy principle*. They aim to find the reward function that induces a distribution with maximum entropy. Depending on the distribution, there are different families of methods:

Entropy of the distribution over trajectories or policies These methods aim to learn a reward function that induces a distribution over all the trajectories with maximum entropy:

$$\max_{P \in \Delta^{(\mathcal{S}\times\mathcal{A})^H}} - \sum_{\boldsymbol{\tau} \in (\mathcal{S}\times\mathcal{A})^H} P(\boldsymbol{\tau}) \ln P(\boldsymbol{\tau}).$$

Therefore, the idea is that the reward, that induces a policy under which we visit all the trajectories in the most uniform way, is the closest to the true reward function of the expert. However, since the search in the space of the trajectories grows exponentially, often we prefer to optimize:

$$\max_{P} - \sum_{\pi \in \Delta^{\mathcal{S}\times\mathcal{A}}} P(\pi) \ln P(\pi),$$

the entropy of the distribution over policies induced by the learned reward function. A famous method in this context is MAXENT IRL [61].

Relative entropy of the distribution over trajectories Another idea is to optimize the KL divergence between the distribution P induced by the learned reward function and a certain baseline distribution Q. Notice that in case Q is the uniform distribution it reduces to the maximization of the entropy of P:

$$\max_{P \in \Delta^{(\mathcal{S}\times\mathcal{A})^H}} - \sum_{\boldsymbol{\tau} \in (\mathcal{S}\times\mathcal{A})^H} P(\boldsymbol{\tau}) \ln \frac{P(\boldsymbol{\tau})}{Q(\boldsymbol{\tau})}.$$

This is the objective of REIRL [10].

Bayesian Update The idea is to apply Bayes theorem to a certain prior over reward functions using the dataset $\mathcal{D}$. If we denote the prior over reward functions with $\mathbb{P}(\hat{R})$ and the likelihood with $\mathbb{P}(\boldsymbol{\tau}|\hat{R})$, then the update rule becomes:

$$\mathbb{P}(\hat{R}|\boldsymbol{\tau}) \propto \mathbb{P}(\boldsymbol{\tau}|\hat{R})\, \mathbb{P}(\hat{R}),$$

where the likelihood is typically factored as $\mathbb{P}(\boldsymbol{\tau}|\hat{R}) = \prod_{(s,a)\in\boldsymbol{\tau}} \mathbb{P}((s,a)|\hat{R})$. In this way, after having performed the update as the number of demonstrations in $\mathcal{D}$, we aim to end up with a distribution over reward functions from which we aim to draw the best estimate for $\hat{R}$. This idea can be implemented in various ways:

Boltzmann distribution We can select the likelihood function to be the Boltzmann distribution with the action-value function as energy:

$$\mathbb{P}(\boldsymbol{\tau}|\hat{R}) \propto \exp^{\frac{Q^*(s,a;\hat{R})}{\beta}},$$

where β controls the randomness of the distribution. BIRL [46] works in this way.

Gaussian process We might parametrize the reward function as a non-linear function of features and then model it as a Gaussian process. The algorithm GPIRL [35] computes the posterior using such parameters for the likelihood.

Maximum likelihood estimation Instead of updating the posterior, another possibility is to directly maximize the likelihood function.

Classification and Regression There is also the possibility to apply supervised learning methods for classification and regression for solving the IRL problem:

Classification based on action-value scores We can consider the setting in which we aim to learn the action to play in every state. This is a multi-label (in case of more than two actions) classification problem. Simply, we can consider as score of a classification the Q-function, that is if we predict action a in state s, then the score of this prediction is $Q^\pi(s,a) = \boldsymbol{w}^\intercal \boldsymbol{\mu}^\phi(\pi)(s,a)$. The error can be computed w.r.t. $\max_{a \in \mathcal{A}} Q^\pi(s,a)$. The goal is to learn the weights $\boldsymbol{w}$, since these are the same weights of the reward function. This is the setting as present in SCIRL [29].

Regression tree for state space partitions A linear function of the features might require too many features to represent the entire state-action space. FIRL [34] proposes to use a regression tree with individual features in the intermediate nodes and a *conjunction of indicator feature functions* in the leaves.

2.4. PAC Learning

We will first present the PAC Learning Framework, then we will move to the main results of the Minimax Theory, and finally we will show why this theory is useful in the Reinforcement Learning and related settings.

2.4.1. PAC Learning Framework

In [55], Valiant introduced the Probably Approximately Correct (PAC) framework. In his paper, entitled "A theory of the learnable", he states that he gives *a precise methodology*

for studying this phenomenon of knowledge acquisition in the absence of explicit programming *from a computational viewpoint. It consists of choosing an appropriate information gathering mechanism, the learning protocol, and exploring the class of concepts that can be learned using it in a reasonable (polynomial) number of steps.* As it is emphasized by [21], the *intent of the PAC model is that successful learning of an unknown target concept should entail obtaining, with high probability, a hypothesis that is a good approximation of it.* In other words, PAC learning aims to retrieve, with a certain probability, an hypothesis (the estimator) which is correct modulo a certain error. The presentation of concepts in this section is based on [40].

The Problem Setting

It should be noticed that the PAC learning framework was initially introduced for the Machine Learning field and then adapted to the Reinforcement Learning setting [26]. Here is the setting:

- L is the **learner**, our agent, the algorithm that aims to learn something;
- X is defined as the set of all the **instances** over which target functions can be defined. For instance, X might be a set of people, or a set of objects, each of them characterized by certain features;
- C is a set of **concepts** that the learner might aim to learn. A concept $c \in C$ refers to a subset of items of X that satisfy a certain property. For example, c might represent the subset of people who are skiers, or the objects which are yellow. A concept c can be seen as a boolean function $c : X \to \{0, 1\}$ which returns 1 if the considered instance has the considered property and 0 otherwise;
- $\mathcal{P}$ is a stationary[6] **probability distribution** over X that defines how the instances are generated;
- H is the set of all the **hypothesis** among which the learner L can choose; simply, any hypothesis $h \in H$ is a subset of X and can be seen as a boolean function $h : X \to \{0, 1\}$ similarly to the concepts. An example of hypothesis might be the subset of people that like the mountains.

The goal of the *learner* L is to, after having observed some *samples* $x \sim \mathcal{P}(\cdot)$ along with the label $c(x)$ provided by a certain fixed *concept* c, output an *hypothesis* h that aims to be an estimate of c. As an example, let us consider the set of people X visit Cortina. Every person is characterized by two features, the *age* and the *height*. We sample people

[6] It means that $\mathcal{P}$ does not change with time.

from X using a probability distribution $\mathcal{P}$, which might simply provide as samples only the people that visit the Bar PAC. Therefore, we take every person that visits the Bar PAC in Cortina and label them as a skier or not (the concept c to be learned concerns the subset of all the people that visit Cortina X that are skiers). Our learner L, which might be a computer program or even a person, observes this dataset and aims to learn concept c; in other words, the learner L aims to be able to predict, for a new person that accesses Bar PAC, whether she is a skier or not. However, the learner can only observe the *age* and the *height* of people, and it can only output an hypothesis h that belongs to a certain space H. For example, H might be the set of hypothesis h_1, h_2, h_3, h_4, where h_1 is the subset of people with less than 18 years and h_2 with more than 18 years, and h_3 the subset of people smaller than $1.75m$ and h_4 the subset of people taller than $1.75m$. To sum up, the learner L observes N samples (people) that visit Bar PAC in Cortina ($x \sim \mathcal{P}(\cdot)$) and is told which of them are skiers (belong to concept c). Then, the goal of L is to output an hypothesis $h \in H := \{h_1, h_2, h_3, h_4\}$ which can correctly classify new people that visit Bar PAC based on their features (age and height).
It should be remarked that the hypothesis h outputted by the learner L is not evaluated over new samples collected at random (for example people in a certain square), but samples collected according to the same distribution $\mathcal{P}$ (so new people that visit the same Bar PAC).

Error of a Hypothesis

There are various types of errors that can be considered. Let us start with the true error of an hypothesis h w.r.t. a target concept c and instance distribution $\mathcal{P}$.

Definition 28 (True Error)**.** *The True Error $error_{\mathcal{P}}(h)$ of an hypothesis $h \in H$ with respect to a target concept $c \in C$ and a distribution $\mathcal{P} \in \Delta^X$ is defined as the probability that h will misclassify an instance drawn at random according to $\mathcal{P}$:*

$$error_{\mathcal{P}}(h) := \mathbb{P}_{x \sim \mathcal{P}(\cdot)}\big[h(x) \neq c(x)\big].$$

It should be remarked that the true error depends on the probability distribution $\mathcal{P}$ from which we collect samples. Notice also that the learner L cannot observe the true error of h w.r.t. c, but only the error over the training examples, which are just a part of all the samples that can be sampled from $\mathcal{P}$. We define this error as the **Training Error**:

$$\text{error}_{\mathcal{D}} := \frac{1}{n}\sum_{i=1}^{n}\big\{h(x_i) \neq c(x_i)\big\},$$

where $\mathcal{D}$ is the training set.

PAC learnability

It is clear that, unless the learner L is shown all the possible pairs $\langle x, c(x)\rangle$, then it is impossible to nullify the true error. Therefore, the PAC framework aims to analyze algorithms that output hypothesis with an error bounded by ϵ with a certain probability δ, i.e. that the error is bounded only for a certain percentage of randomly drawn sequences of samples. As [40] remarks, *we require only that the learner probably learn a hypothesis that is approximately correct.* This brings us to the definition of PAC-learnability.

Definition 29 (PAC-Learnable)**.** *Consider a concept class C defined over a set of instances X of length N and a learner L using hypothesis space H. C is PAC-learnable by L using H if for all $c \in C$, distributions $\mathcal{P}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1-\delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{P}}(h) \leqslant \epsilon$, in time that is polynomial in $1/\epsilon, 1/\delta, N$ and $size(c)$.*

Therefore, we are asking two things to our learner L: (ϵ, δ)-correctness (error bounded by ϵ w.h.p.) and that it does so efficiently. In practice, we care about the number of samples required to achieve learning. In other words, we care about the **Sample Complexity**.

Sample Complexity for Finite Hypothesis Spaces

PAC-Learnability is mainly concerned with the number of training samples required by the learner. Mitchell [40] defines the sample complexity as:

Definition 30 (Sample Complexity)**.** *The growth in the number of the required training examples with problem size is called the Sample Complexity of the learning problem.*

Let me introduce other useful definitions.

Definition 31 (Consistent Learner)**.** *A learner is consistent if it outputs hypotheses that perfectly fit the training data, whenever possible.*

Definition 32 (Version Space)**.** *The Version Space $VS_{H,\mathcal{D}}$ is the set of all hypotheses $h \in H$ that correctly classify the training examples $\mathcal{D}$:*

$$VS_{H,\mathcal{D}} := \{h \in H : \forall \langle x, c(x)\rangle \in \mathcal{D} : \ h(x) = c(x)\}.$$

It is clear that a learner is consistent if it outputs only hypotheses that lie in the version space.

Definition 33 (ϵ-exhausted)**.** *Consider a hypothesis space H, target concept c, instance distribution $\mathcal{P}$, and set of training examples $\mathcal{D}$ of c. The version space $VS_{H,\mathcal{D}}$, is said to be ϵ-exhausted with respect to c and $\mathcal{P}$, if every hypothesis $h \in VS_{H,\mathcal{D}}$ has error less than ϵ with respect to c and $\mathcal{P}$:*

$$(\forall h \in VS_{H,\mathcal{D}}) error_{\mathcal{P}}(h) < \epsilon.$$

Basically, an hypothesis space is ϵ-exhausted if all the hypotheses that perfectly classify the training examples have a true error upper bounded by ϵ. [20] provides a result that allows to bound the probability that the version space will be ϵ-exhausted after a certain number of samples:

Theorem 8 (ϵ-exhausting the version space)**.** *If the hypothesis space H is finite, and $\mathcal{D}$ is a sequence of $N \geqslant 1$ independent randomly drawn examples of some target concept c, then for any $0 \leqslant \epsilon \leqslant 1$, the probability that the version space $VS_{H,\mathcal{D}}$ is not ϵ-exhausted (with respect to c) is less than or equal to $|H|e^{-\epsilon N}$:*

$$\mathbb{P}(\exists h \in H | h \in VS_{H,\mathcal{D}} \wedge error_{\mathcal{P}}(h) \geqslant \epsilon) \leqslant |H|e^{-\epsilon N}.$$

To compute the number of samples required to make this failure probability smaller than a certain δ, we can set $|H|e^{-\epsilon N} \leqslant \delta$ and solve with respect to N, obtaining:

$$N \geqslant \frac{1}{\epsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right).$$

Notice that this bound is really general, and as such is weak: the linear dependency on $|H|$ is too much.
In case the hypothesis space H does not contain the target concept c, then it is clear that a zero-error hypothesis is impossible to achieve. What we might ask in this setting is to retrieve the hypothesis $h \in H$ with minimum training error.

Definition 34 (Agnostic Learner)**.** *A learner that makes no assumption that the target concept is representable by H and that simply finds the hypothesis with minimum training error, is called an Agnostic Learner.*

Mitchell highlights the fact that such kind of learner is called agnostic because *it makes no prior commitment about whether or not $C \subseteq H$.* In this context, we usually bound the probability that the true error is greater than the training error (bias) by a certain ϵ. If we denote by $error_{\mathcal{P}}(h)$ the true error of an hypothesis h, $error_{\mathcal{D}}(h)$ the training error of hypothesis h over dataset $\mathcal{D}$ and $h_{best} \in H$ the hypothesis of the hypothesis space with

minimum training error, then our problem becomes that of bounding the probability that

$$error_{\mathcal{P}}(h_{best}) \leqslant error_{\mathcal{D}}(h_{best}) + \epsilon.$$

To do so, we can exploit concentration inequalities [9] and in particular we might use the Hoeffding bound [22] (in case the samples are *i.i.d.* coin flips) to obtain that:

$$\mathbb{P}(\exists h \in H | error_{\mathcal{P}}(h_{best}) - error_{\mathcal{D}}(h_{best}) > \epsilon) \leqslant |H| e^{-2N\epsilon^2}.$$

Again, if we want to bound the failure probability by δ, we can re-arrange the terms and obtain a sample complexity of:

$$N \geqslant \frac{1}{2\epsilon^2}\left(\ln |H| + \ln \frac{1}{\delta}\right).$$

Sample Complexity for Infinite Hypothesis Spaces

Up to now we have considered finite hypothesis spaces. Now we move to infinite hypothesis spaces and, potentially, continuous hypothesis spaces. It is clear that the bounds developed in the previous section are infeasible, because they depend on $|H|$. We therefore replace $|H|$ with a new measure of the complexity of an hypothesis space, the Vapnik-Chervonenkis dimension of H or $VC(H)$. The VC dimension does not measure the complexity of an hypothesis space H with the number of distinct hypotheses in it, but instead it measures *the number of distinct instances from X that can be completely discriminated using H* [40]. To define it formally, we have to firstly introduce some notions.

Definition 35 (Dichotomy)**.** *A dichotomy of a set S is a partition of S into two disjoint subsets.*

Definition 36 (Shattering)**.** *A set of instances S is said to be shattered by hypothesis space H if for every dichotomy of S there exist some hypothesis in H consistent with this dichotomy.*

In other words, the capacity of an hypothesis space to shatter set of instances is a measure of its goodness to represent target concepts defined over these instances. It is clear that an hypothesis space H is unbiased if it shatters the entire space X. We can now define the VC dimension:

Definition 37 (VC Dimension)**.** *The Vapnik-Chervonenkis dimension, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then*

$VC(H) = \infty$.

It is important to notice that every hypothesis space H with finite cardinality has a bounded VC dimension: $VC(H) \leqslant \log_2(|H|)$.
Using the VC dimension, [8] proved a minimum number of training samples N needed to achieve (ϵ, δ)-correctness for any target concept in C:

$$N \geqslant \frac{1}{\epsilon}\Big(4\log_2\frac{2}{\delta} + 8VC(H)\log_2\frac{13}{\epsilon}\Big).$$

2.4.2. Minimax Theory

Minimax Theory *provides a rigorous framework for establishing the best possible performance of a procedure under given assumptions* [24]. We will now present its main notions and concepts based on the lecture notes [24].

The Framework

When we solve a statistical learning problem, i.e., a classification or a regression problem, we can use many different kinds of algorithms/estimators. The minimax theory provides a set of techniques for determining the worst-case complexity of a problem and of a procedure (algorithm). To do so, let us introduce some notation and concepts.
We denote by $\mathcal{P}$ a certain class of probability distributions, and by $X_1, X_2, \ldots, X_n$ a sample from a distribution $P \in \mathcal{P}$. $\theta(P)$ is some function of the true distribution P generating the sample and $\hat{\theta}(X_1, X_2, \ldots, X_n)$ is an estimator of $\theta(P)$, thus a function of the sample. Now that we have introduced the setting, we can define some important notions in the context of minimax theory.

Definition 38 (Minimax Risk)**.** *Given a certain metric d, the minimax risk R_n is defined as:*

$$R_n \equiv R_n(\mathcal{P}) := \inf_{\hat{\theta}} \sup_{P\in\mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))].$$

The minimax risk is the expected distance between the quantity to estimate and its estimator when considering the algorithm/estimator which provides the minimum risk in the worst case. In other words, it provides the minimum error that any algorithm that tries to solve this problem, even the best one, faces in a certain instance of problem. What is interesting is the value of n, the size of the sample:

Definition 39 (Sample Complexity)**.** *Given the minimax risk R_n, we define the sample*

complexity $n(\epsilon, \mathcal{P})$ *as:*

$$n(\epsilon, \mathcal{P}) := \min\Big\{n : R_n \leqslant \epsilon\Big\}.$$

The sample complexity is the minimum number of samples we must gather in order to be sure that the minimax risk is upper bounded by a certain ϵ. The more the samples, the better, because they allow to reduce the error of the estimation problem. However, since the minimax risk is difficult to be computed, what is usually done is to bound it from below and from above until the two bounds match. Such lower and upper bounds to the minimax risk are computed in different ways. The upper bound U_n is "simpler" to compute: indeed, we can fix a certain estimator/algorithm $\hat{\theta}$ and estimate its goodness:

$$R_n(\mathcal{P}) := \inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \leqslant \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \equiv U_n.$$

To reduce such bound, we have to find an estimator which is optimal w.r.t. the worst case. Instead, the computation of the lower bound L_n is much more tricky and it involves the proposal of a certain problem $P \in \mathcal{P}$ and then the computation of the expected error than any possible algorithm/estimation suffers when trying to solve such problem:

$$R_n(\mathcal{P}) := \inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \geqslant \inf_{\hat{\theta}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \equiv L_n.$$

Once that we get L_n and U_n that asymptotically match, then we have succeeded in characterizing the complexity of a statistical learning problem. There are two main methods in minimax theory for computing the lower bound: *Le Cam's inequality* and *Fano's inequality*.

Le Cam's Inequality

Before explaining the method, let us provide the important theorem of Le Cam as presented in the lecture notes of [24].

Theorem 9 (Le Cam's Inequality)**.** *Let* $\mathcal{P}$ *be a set of distributions. For any pair* $P_0, P_1 \in \mathcal{P}$*:*

$$\inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \geqslant \frac{\Delta}{4} \int \min\{p_0^n(x), p_1^n(x)\} dx \geqslant \frac{\Delta}{8} e^{-nKL(P_0||P_1)}, \tag{2.3}$$

where $\Delta := d(\theta(P_0), \theta(P_1))$ *and* p_0, p_1 *are, respectively, the densities of* P_0, P_1*.*

Corollary 1. *Suppose there exist* $P_0, P_1 \in \mathcal{P}$ *such that* $KL(P_0||P_1) \leqslant \log 2/n$*. Then:*

$$\inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \geqslant \frac{\Delta}{16}, \tag{2.4}$$

where $\Delta := d(\theta(P_0), \theta(P_1))$.

Le Cam's inequality allows to lower bound the minimax risk simply by considering just two instances of problem from class $\mathcal{P}$. The distance (according to the previously chosen metric d) between the quantity to be estimated in P_0 and P_1 allows to lower bound the risk. Thus, if we aim to solve a certain statistical learning problem in which we aim to estimate a certain characteristic of an unknown distribution generating data $P \in \mathcal{P}$, we simply have to choose two distributions $P_0, P_1 \in \mathcal{P}$, compute their characteristics, and then depending on the setting, apply Eq. 2.3 or Eq. 2.4 to bound the risk.

Fano's Inequality

However, [24] shows that for metrics like $d(f, g) := \int (f-g)^2$ Le Cam's method usually does not provide tight bounds. Therefore, it is better to apply Fano's method, in which instead of choosing a pair $P_0, P_1 \in \mathcal{P}$ of distributions, we choose a finite set of N distributions $P_1, \ldots, P_N \in \mathcal{P}$:

Theorem 10 (Fano's Inequality)**.** *Let* $F := \{P_1, \ldots, P_N\} \subset \mathcal{P}$. *Let* $\theta(P)$ *be a parameter taking values in a metric space with metric* d. *Then:*

$$\inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \geqslant \frac{\alpha}{2}\Big(1 - \frac{n\beta + \log 2}{\log N}\Big), \tag{2.5}$$

where

$$\alpha := \min_{j \neq k} d(\theta(P_j), \theta(P_k)),$$

and

$$\beta := \max_{j \neq k} KL(P_j || P_k).$$

Corollary 2. *Suppose there exist* $F := \{P_1, \ldots, P_N\} \subset \mathcal{P}$ *such that* $N \geqslant 16$ *and*

$$\beta := \max_{j \neq k} KL(P_j || P_k) \leqslant \frac{\log N}{4n},$$

Then:

$$\inf_{\hat{\theta}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[d(\hat{\theta}, \theta(P))] \geqslant \frac{\alpha}{4}. \tag{2.6}$$

Fano keeps into account more than two distributions of class $\mathcal{P}$ and uses their distances and the distances between their characteristics (θ) to provide the bound. But how should

we build the finite class of distributions F? A possibility is to use the form:

$$F = \{P_\omega : \omega \in \mathcal{X}\},$$

where

$$\mathcal{X} := \{\omega = (\omega_1, \dots, \omega_m) : \omega_i \in \{0,1\}\, \forall i \in [m]\}.$$

is called the hypercube. Thus, there are 2^m distributions in F. Let us denote the Hamming distance between two vectors $\omega, \nu \in \mathcal{X}$ with $H(\omega, \nu) := \sum_{j \in [m]} \mathbb{1}\{\omega_j \neq \nu_j\}$. However, some distributions in F might be really close and this might bring to a poor lower bound. One possible solution is to prune the hypercube, namely to find a subset $\mathcal{X}' \subset \mathcal{X}$ with more or less the same number of elements of $\mathcal{X}$ but such that each pair $P, Q \in F' := \{P_\omega : \omega \in \mathcal{X}'\}$ is far apart. $\mathcal{X}'$ is called *pruned hypercube* and we can use the Varshamov-Gilbert lemma to build it.

Lemma 1 (Varshamov-Gilbert)**.** *Let* $\mathcal{X} := \{\omega = (\omega_1, \dots, \omega_N) : \omega_i \in \{0,1\}\}$. *Suppose that* $N \geqslant 8$. *There exist* $\omega^0, \omega^1, \dots, \omega^M \in \mathcal{X}$ *such that (i)* $\omega^0 = (0, \dots, 0)$, *(ii)* $M \geqslant 2^{N/8}$ *and (iii)* $H(\omega^{(j)}, \omega^{(k)}) \geqslant N/8$ *for* $0 \leqslant j < k \leqslant M$. *We call* $\mathcal{X}' := \{\omega^0, \omega^1, \dots, \omega^M\}$ *a pruned hypercube.*

To apply Fano's inequality to obtain tight bounds, we can firstly use this lemma for finding the pruned hypercube and then apply the inequality.

2.4.3. Sample Complexity in Reinforcement Learning

The concept of *sample complexity*, as presented in the previous sections, is related to the supervised learning setting. In this section, we adapt the notion to the RL setting.

Sampling Models

As Kakade highlights in his doctoral thesis [26], in the RL setting there are two quantities that might be learned through samples: the transition model p and the reward function R. However, the interesting quantity to sample is p. To collect a sample for p, the agent simply has to ask the environment in which state s' it will end up after having taken action a from state s. In this way the agent can increase, for instance, visit counters $n(s,a) \leftarrow n(s,a) + 1$ and $n(s,a,s') \leftarrow n(s,a,s') + 1$ and estimate the transition model with the sample mean $\hat{p}(s'|s,a) = \frac{n(s,a,s')}{n(s,a)}$. Kakade identifies three different sampling strategies in the RL setting:

Forward Model The agent starts to interact with the environment and continues in-

definitely until a certain number of time steps of exploration has been reached. This is the most general and complex setting, in which the agent has to devise an exploration strategy for exploring the environment and visiting most important state-action pairs in order to collect samples for them. Notice that since the agent never resets to a certain state/distribution of states, it has to be careful in visiting certain state-action pairs, because they might get it stuck in sub-areas of the problem.

Generative Model This is the easiest setting. The agent can simply ask the environment to get a sample from any state-action pair (s, a) he wants without having to reach it. Since the exploration phase is missing, it is clear that all the pairs are equally reachable and thus strongest guarantees can be achieved on the estimated $\hat{p}$ and therefore on the policy learned afterwards.

μ-Reset Model This is an intermediate sampling strategy between the other two in terms of difficulty of the task, because we still need an exploration strategy to reach all the state-action pairs we want, but we are helped in doing so by the possibility of performing resets to a certain state sampled from distribution μ. In this case, the complexity will be a function of μ.

In terms of these sampling models, we can provide a more accurate definition of sample complexity for the RL setting:

Definition 40 (Sample Complexity)**.** *We consider the sample complexity to be the number of calls to the sampling model required to satisfy a specified performance criterion.*

The sample complexity, i.e., the number of calls we require to a certain sampling model, depends on a performance criterion.
It should be noticed that the same notion of sample complexity can be adapted to the IRL setting, where we still aim to estimate p to solve the task. Indeed, similarly to RL, in IRL also the expert policy π^E might be unknown and might need to be estimated based on samples, but it is less important than p, analogously to r.

Likelihood Ratio Method

In this section we present the *Likelihood Ratio Method*, which is one of the most powerful methods for proving lower bounds to the sample complexity in the bandits[7] and RL settings. This mathematical tool was introduced in [37] to prove a lower bound to the sample complexity of exploration in the multi-armed bandits problem and then adopted

[7]The bandits setting can be seen as an RL problem with only one state. The goal is to find the action (called arm) that provides largest reward.

by other researchers to prove other bounds in RL (like in the generative model setting [5]).
The proof idea is the following. We assume (by contradiction) that there exists an (ϵ, δ)-correct algorithm that collects, in a certain instance of the problem (say I_1), a number of samples lower than the bound on the sample complexity (say t^*) we want to prove. This brings to an absurd and thus the algorithm is not (ϵ, δ)-correct. However, the brilliant idea lies in the construction of the absurd. To prove that less than t^* samples are not enough for instance I_1, we introduce another instance, I_0. We then consider the likelihood ratio between the two problem instances. Say W the random variable representing a trajectory in a problem, we can consider the ratio of its probability between the two settings and bound such ratio by something $\propto\delta$. The final step is to consider the probability of the event that the algorithm outputs a solution which is not ϵ-correct for I_1, call it $\mathbb{P}_1(B)$ (where subscript 1 means that the probability concerns problem I_1); next, notice that it coincides with the expectation of the indicator function: $\mathbb{P}_1(B) = \mathbb{E}_1[\mathbb{1}\{B\}]$ and finally insert the likelihood ratio to generate the absurd:

$$\mathbb{P}_1(B) = \mathbb{E}_1[\mathbb{1}\{B\}] = \mathbb{E}_0\Big[\frac{L_1(W)}{L_0(W)}\mathbb{1}\{B\}\Big] > K\propto\delta.$$

In other words, the two hypotheses that an algorithm is (ϵ, δ)-correct and that it collects less than t^* samples in instance I_1 are incompatible, because considering a new instance I_0, whose ϵ-correct solutions are not ϵ-correct for I_1, we are able to show that the algorithm outputs an ϵ-correct solution for I_0 when facing instance I_1 and collecting less than t^* samples w.p. greater than δ. This violates the hypothesis of (ϵ, δ)-correctness, thus any (ϵ, δ)-correct algorithm must collect at least t^* samples.

Bretagnolle-Huber Inequality

The Bretagnolle-Huber inequality is a powerful tool that is widely adopted when constructing minimax lower bound proofs in the context of bandits. The Bretagnolle-Huber inequality can be stated as presented by Lattimore in Theorem 14.2 [32]:

Theorem 11 (Bretagnolle-Huber Inequality)**.** *Let P and Q be probability measures on the same measurable space $(\Omega, \mathcal{F})$, and let $A \in \mathcal{F}$ be an arbitrary event. Then,*

$$P(A) + Q(A^{\complement}) \geqslant \frac{1}{2}\exp(-D_{KL}(P||Q)),$$

where $A^{\complement} = \Omega\backslash A$ is the complement of A.

In other words, this theorem allows us to lower bound the sum of probabilities of an event

and its complement with regards to two different probability measures (defined on the same measurable space). If we define the event A (and thus also event $A^{\complement}$) properly, then we can lower bound the sum by a function of the KL divergence, which in turn can be lower bounded by a certain quantity, that depends on the number of samples, exploiting other theorems. Eventually, we can solve with respect to the number of samples and obtain the bound on the sample complexity desired.

3 | State of the Art

The topic of the sample complexity is a very theoretical one, and it is still mostly unexplored for IRL. Instead, for the case of RL, (almost) matching bounds have been computed for a large variety of the problems and quantities of interest. Related works can be classified in IRL and RL works, and then for the sampling model they use. According to [26], we can distinguish between the *generative model* and the *forward model*. Generative model means that we have an oracle that allows us to query whatever (s, a) pair we want, while the forward model requires us to explore the environment to sample the pairs. Another dimension is whether a paper provides both a lower and upper bound or only an upper bound. Finally, we can group works based on the *setting* they are considering, namely whether they consider the discounted/average infinite-horizon setting or the (discounted) fixed finite-horizon setting.
We have decided to organize the presentation of the state of the art in three sections, grouping the main works based on whether they concern *Bandits*, *RL* or *IRL*.

3.1. Sample Complexity in Bandits

As aforementioned, bandits can be seen as RL problems with only one state. The sample complexity will thus depend only on the number of actions (arms) $n := |\mathcal{A}|$, the accuracy ϵ and the failure tolerance δ. In [16] an algorithm was proposed for solving the multi-armed bandit problem with a PAC analysis for the number of time steps to identify a near-optimal arm; (ϵ, δ)-correctness was guaranteed after $\mathcal{O}(\frac{n}{\epsilon^2} \log \frac{1}{\delta})$ time steps (n is the number of arms). Paper [37] provides a matching lower bound. Notice that [37] is the fundamental paper that introduces the *Likelihood Ratio Method* presented in Subsection 2.4.3 for computing lower bounds through a proof by absurd. In [32], some lower bounds are developed using an alternative demonstration method, the one based on the Bretagnolle-Huber inequality.
It should be remarked that, for bandits, we have matching lower and upper bounds.

3.2. Sample Complexity in Reinforcement Learning

With regards to RL, we can have more than one state, therefore, the bounds will depend also on $|\mathcal{S}|$. Historically, one of the first important works on the sample complexity of RL is [26]. Moving to the RL discounted infinite-horizon setting under generative model, the paper that found matching bounds for the estimation of the action-value function in max norm is [5], where a matching bound of $\Theta(\frac{N}{\epsilon^2(1-\gamma)^3}\log\frac{N}{\delta})$, where γ is the discount factor ad $N := |\mathcal{S}\times\mathcal{A}|$ is the size of the state-action space, is proved by exploiting the likelihood ratio method. Such result improves on the previous best lower bound of $\tilde{\Omega}(\frac{N}{\epsilon^2(1-\gamma)^2})$ of [3, 17] and the previous best upper bound of $\tilde{\mathcal{O}}(\frac{N}{\epsilon^2(1-\gamma)^4})$ proved for some algorithms like [4] for instance. If we consider the episodic fixed-horizon setting for RL, then paper [12] is the most interesting result, since the algorithm it proposes, UCFH, produces an upper bound of $\mathcal{O}(\frac{|\mathcal{S}|^2|\mathcal{A}|H^2}{\epsilon^2}\log\frac{1}{\delta})$ to the number of episodes required (H is the length of each episode) to provide an ϵ-optimal estimate of the value function; moreover, [12] proves a lower bound of $\Omega(\frac{|\mathcal{S}||\mathcal{A}|H^2}{\epsilon^2}\log\frac{1}{\delta+c})$ (where c is a constant), so it is clear that the bounds are almost matching. Previous works in this setting are not worthy to be mentioned. For the general RL problem, [33] proposes an algorithm, MERL, with a sample complexity of $\mathcal{O}(\frac{N}{\epsilon^2(1-\gamma)^3}\log^2\frac{N}{\delta\epsilon(1-\gamma)})$, and proves a matching lower bound except for logarithmic factors.

3.3. Sample Complexity in Inverse Reinforcement Learning

The works can be grouped based on the setting they are considering and on the quantity to be estimated.

3.3.1. Sample Complexity for Estimating the Feasible Reward Set

The notion of feasible reward set $\mathcal{R}$ was introduced in [42] in an *implicit* form in the infinite-horizon discounted case as a *linear feasibility* problem and, subsequently, adapted to the finite-horizon case in [36]. Furthermore, in [36, 39] an *explicit* form of the reward functions belonging to the feasible region $\mathcal{R}$ was provided. In these works, the problem of estimating the feasible reward set is studied for the first time considering a "reference" pair of rewards $(\overline{r},\breve{r}) \in \mathcal{R}\times\widehat{\mathcal{R}}$ against which to compare the rewards inside the recovered

sets, leading to the (pre)metric:

$$\widetilde{\mathcal{H}}_d(\mathcal{R}, \mathcal{R}, \overline{r}, \breve{r}) \coloneqq \max \left\{ \inf_{\widehat{r} \in \widehat{\mathcal{R}}} d(\overline{r}, \widehat{r}), \inf_{r \in \mathcal{R}} d(r, \breve{r}) \right\}. \tag{3.1}$$

Compared to the Hausdorff (pre)metric, in Equation (3.1) there is no maximization over the choice of $(\overline{r}, \breve{r})$, leading to a simpler problem.[1] In [39], a uniform sampling approach (similar to Algorithm 10) is proved to achieve a sample complexity of order $\widetilde{O}\left(\frac{\gamma^2|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\right)$ for the index of Equation (3.1) with $d = d^{\mathrm{G}}_{Q*}$[2] in the discounted setting with generative model. For the forward model case, the `AceIRL` algorithm [36] suffers a sample complexity of order $\widetilde{O}\left(\frac{H^5|\mathcal{S}||\mathcal{A}|}{\epsilon^2}\right)$ for the index of Equation (3.1) with $d = d^{\mathrm{F}}_{V*}$, in the finite-horizon case. Unfortunately, the reward recovered by `AceIRL` reward function is not guaranteed to be bounded by a predetermined constant (e.g., $[-1, 1]$). Modified versions of these algorithms allow embedding problem-dependent features under a specific choice of a reward within the set.

3.3.2. Sample Complexity Lower Bounds in IRL

The only work found that proposes a sample complexity lower bound for IRL is [31]. The authors consider a finite state and action MDP\R and the IRL algorithm of [42] for β-strict separable IRL problems (i.e., with suboptimality gap at least β) with state-only rewards in the discounted setting. When only two actions are available ($|\mathcal{A}| = 2$) and the samples are collected starting in each state with equal probability, by means of a geometric construction and Fano's inequality, the authors derive an $\Omega(|\mathcal{S}| \log |\mathcal{S}|)$ lower bound on the number of trajectories needed to identify a reward function. Note that this analysis limits to the *identification* of a reward function within a finite set, rather than evaluating the accuracy of recovering the feasible reward set.

3.3.3. Sample Complexity of IRL Algorithms

Differently from forward RL, the theoretical understanding of the IRL problem is largely less established and the sample complexity analysis proposed in the literature often limit to specific algorithms. In the class of *feature expectation* approaches, the seminal work [1] propose IRL algorithms guaranteed to output an ϵ-optimal policy (made of a mixture of Markov policies) after $\widetilde{O}\left(\frac{k}{\epsilon^2(1-\gamma)^2} \log\left(\frac{1}{\delta}\right)\right)$ trajectories (ideally of infinite length). The result holds in a discounted setting (being γ the discount factor) under the assumption that

[1] In this sense, a PAC guarantee according to Definition 41, implies a PAC guarantee defined w.r.t. (pre)metric of Equation (3.1).

[2] where $d^{\mathrm{G}}_{Q*}(r, \widehat{r}) \coloneqq \max_{(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]} |Q^*_h(s, a; r) - Q^*_h(s, a; \widehat{r})|$

the true reward function $r(s) = w^T\phi(s)$ is state-only and linear in some *known* features ϕ of dimensionality k. In [53], a game-theoretic approach to IRL, named `MWAL`, is proposed improving [1] in terms of computational complexity and allowing the absence of an expert, preserving similar theoretical guarantees in the same setting. `Modular IRL` [57], that integrates supervised learning capabilities in the IRL algorithm, is guaranteed to produce an ϵ-optimal policy after $\widetilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^2\epsilon^2}\log\left(\frac{1}{\delta}\right)\right)$ trajectories. This class of algorithms, however, requires, as an inner step, to compute the optimal policy $\widehat{\pi}$ for every candidate reward function $\widehat{r}$. This step (and the corresponding sample complexity) is somehow hidden in the analysis since they either assume the knowledge of the transition model and apply dynamic programming [e.g., 57] or the access to a black-box RL algorithm [e.g., 1]. In the class of *maximum entropy* approaches [61], the `Maximum Likelihood IRL` [60] converges to a stationary solution with $\widetilde{O}(\epsilon^{-2})$ trajectories for *non-linear* reward parametrization (with bounded gradient and Lipschitz smooth), when the underlying Markov chain is ergodic. Furthermore, the authors prove that, when the reward is linear in some features, the recovered solution corresponds to `Maximum Entropy IRL` [61]. Concerning the *gradient-based* approaches, [44] and [47] prove finite-sample convergence guarantee to the expert's weight under linear parametrization as a function of the accuracy of the gradient estimation. Surprisingly, a theoretical analysis of the IRL progenitor algorithm of [42] has been proposed only recently in [30]. A β-strict separability setting is enforced in which the rewards are assumed to lead to a suboptimality gap of at least $\beta > 0$ when playing any non-optimal action. For finite MDPs, known expert's policy, under the demanding assumption that each state is reachable in one step with a minimum probability $\alpha > 0$, and focusing on state-only reward, the authors prove that the algorithm outputs a β-strict separable feasible reward in at most $\widetilde{O}\left(\frac{1+\gamma^2\Xi^2}{\alpha\beta^2(1-\gamma)^4}\log\left(\frac{1}{\delta}\right)\right)$ trajectories, where $\Xi \leqslant S$ is the number of possible successor states. Recently, an approach with theoretical guarantees has been proposed for continuous states [14].

3.3.4. Reward-Free Exploration

Reward-free exploration [RFE, 23, 27, 38] is a setting for pure exploration in MDPs composed of two phases: exploration and planning. In the exploration phase, the agent learns an estimated transition model $\widehat{p}$ without any reward feedback. In the planning phase, the agent is faced with a reward function r and has to output an estimated optimal policy $\widehat{\pi}^*$, using $\widehat{p}$ since no further interaction with the environment is admitted. In this sense, RFE shares this two-phase procedure with our IRL problem, but, instead

of the *planning* phase, we face the *computation* of the feasible reward set.[3] In RFE exploration, the sample complexity is computed against the performance of the learned policy $\widehat{\pi}^*$ under the reward r, i.e., $V^*(\cdot; r) - V^{\widehat{\pi}^*}(\cdot; r)$, whose lower bound of the sample complexity has order $\Omega\left(\frac{H^2|\mathcal{S}||\mathcal{A}|}{\epsilon^2}\left(H\log\left(\frac{1}{\delta}\right) + |\mathcal{S}|\right)\right)$ [23, 27]. The best known algorithm, `RF-Express`, proposed in [38] archives an almost-matching sample complexity of order $\Omega\left(\frac{H^3|\mathcal{S}||\mathcal{A}|}{\epsilon^2}\left(\log\left(\frac{1}{\delta}\right) + |\mathcal{S}|\right)\right)$. The relevant connection with this thesis is the fact that the derivation of the lower bounds shares similarity especially in the construction of the instances. Nevertheless, in the time-inhomogeneous case, the achieved lower bound is higher of order $\Omega\left(\frac{H^3|\mathcal{S}||\mathcal{A}|}{\epsilon^2}\left(\log\left(\frac{1}{\delta}\right) + |\mathcal{S}|\right)\right)$. The connection between IRL and RFE should be investigated in future works, as also mentioned in [36].

[3] As shown in previous works, the computation of the feasible reward set can be formulated with a *linear feasibility problem* [42].

4 | Lipschitz Framework for IRL

In this section, the regularity properties of the feasible reward set in terms of the Lipschitz continuity w.r.t. the IRL problem are analyzed. To make the idea more concrete, suppose that $\mathcal{R}$ is the feasible reward set obtained from the IRL problem $(\mathcal{M}, \pi^E)$ and that $\widehat{\mathcal{R}}$ is obtained with a different IRL problem $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$, which we can think to as an empirical version of $(\mathcal{M}, \pi^E)$, with an estimated transition model $\widehat{p}$ replacing the true model p. Intuitively, to have any learning guarantee, similar IRL problems ($p \approx \widehat{p}$ and $\pi^E \approx \widehat{\pi}^E$) should lead to similar feasible reward sets ($\mathcal{R} \approx \widehat{\mathcal{R}}$).[1]

To formally define a Lipschitz framework, we need to select a (pre)metric for evaluating dissimilarities between feasible reward sets and IRL problems. While we defer the presentation of the (pre)metric for the IRL problems to Section 4.1, where it will emerge naturally, for the feasible reward sets, we employ the *Hausdorff (pre)metric* $\mathcal{H}_d(\mathcal{R}, \widehat{\mathcal{R}})$ (Equation 2.1), induced by a (pre)metric $d(r, \widehat{r})$ used to evaluate the dissimilarity between individual reward functions $r \in \mathcal{R}$ and $\widehat{r} \in \widehat{\mathcal{R}}$. With this choice, two feasible reward sets are "similar" if every reward $r \in \mathcal{R}$ is "similar" to some reward $\widehat{r} \in \widehat{\mathcal{R}}$ in terms of the (pre)metric d. In the next sections, the metric induced by the L_∞-norm between the reward functions $r \in \mathcal{R}$ and $\widehat{r} \in \widehat{\mathcal{R}}$ is employed as d :[2]

$$d^{\mathrm{G}}(r, \widehat{r}) \coloneqq \max_{(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)|\,, \tag{4.1}$$

where G stands for "generative". In Subsection 4.1, the proof that the Lipschitz continuity is fulfilled when no restrictions on the reward function are enforced (besides boundedness in $[-1, 1]$) is provided. Then, in Subsection 4.2, it is shown that, when further restrictions on the viable rewards are required (e.g., state-only reward), such a regularity property no longer holds.

[1] If not, any arbitrary accurate estimate $(\widehat{p}, \widehat{\pi}^E)$ of (p, π^E), may induce feasible sets $\widehat{\mathcal{R}}$ and $\mathcal{R}$ with finite non-zero dissimilarity.

[2] Other choices of d are discussed in Section 5.

4.1. Lipschitz Continuous Feasible Reward Sets

In order to prove the Lipschitz continuity property, let us use the *explicit* form of the feasible reward sets introduced in [39] and extended by [36] for the finite-horizon case, that is reported below.

Lemma 2 (Lemma 4 of [36]). *A reward function $r = (r_h)_{h\in[\![H]\!]}$ is feasible for the IRL problem $(\mathcal{M}, \pi^E)$ if and only if there exist two functions $(A_h, V_h)_{h\in[\![H]\!]}$ where for every $h \in [\![H]\!]$ we have $A_h : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_{\geqslant 0}$, $V_h : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, and $V_{H+1} = 0$, such that for every $(s, a, h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]$ it holds that:*

$$r_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\pi_h^E(a|s)=0\}} + V_h(s) - p_h V_{h+1}(s,a).$$

Furthermore, if $|r_h(s,a)| \leqslant 1$, if follows that $|V_h(s)| \leqslant H-h+1$ and $A_h(s,a) \leqslant H-h+1$.

A form of regularity of the feasible reward set was already studied in Theorem of 3.1 of [39] and in Theorem 5 of [36], providing an *error propagation* analysis. These results are based on showing the existence of a *particular* reward $\widetilde{r}$ feasible for the IRL problem $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$, whose distance from the original reward function $r \in \mathcal{R}$ is bounded by a dissimilarity term between $(\mathcal{M}, \pi^E)$ and $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$. Unfortunately, such a reward $\widetilde{r}$ is not guaranteed to be bounded in $[-1, 1]$ even when the original reward r is (and, thus, it might be $\widetilde{r} \notin \widehat{R}$).[3] In Lemma 3, with a modified construction, it is shown the existence of another *particular* feasible reward $\widehat{r}$ bounded in $[-1, 1]$ (and, thus, $\widehat{r} \in \widehat{\mathcal{R}}$). From this, the Lipschitz continuity of the feasible reward sets follows.

Theorem 12 (Lipschitz Continuity). *Let $\mathcal{R}$ and $\widehat{\mathcal{R}}$ be the feasible reward sets of the IRL problems $(\mathcal{M}, \pi^E)$ and $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$. Then, it holds that:*[4]

$$\mathcal{H}_{d^G}(\mathcal{R}, \widehat{\mathcal{R}}) \leqslant \frac{2\rho^G((\mathcal{M}, \pi^E), (\widehat{\mathcal{M}}, \widehat{\pi}^E))}{1 + \rho^G((\mathcal{M}, \pi^E), (\widehat{\mathcal{M}}, \widehat{\pi}^E))}, \tag{4.2}$$

where $\rho^G(\cdot,\cdot)$ is a (pre)metric between IRL problems, defined as:

$$\rho^G((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E)) := \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} (H-h+1) \times \left(\left| \mathbb{1}_{\{\pi_h^E(a|s)=0\}} - \mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}} \right| + \|p_h(\cdot|s,a) - \widehat{p}_h(\cdot|s,a)\|_1 \right).$$

Some observations are in order. First, the function ρ^{G} is indeed a (pre)metric since it is

[3] An example of this phenomenon is illustrated in Fact 1.

[4] This implies the standard Lipschitz continuity, by simply bounding $\frac{2\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E))}{1+\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E))} \leqslant 2\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E))$.

non-negative and takes value 0 when the IRL problems coincide. Second, as supported by intuition, ρ^{G} is composed of two terms related to the estimation of the expert's policy and of the transition model. While for the transition model, the dissimilarity is formalized by the L_1-norm distance $\|p_h(\cdot|s,a) - \widehat{p}_h(\cdot|s,a)\|_1$, for the policy, the resulting term deserves some comments. Indeed, the dissimilarity $|\mathbb{1}_{\{\pi_h^E(a|s)=0\}} - \mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}}|$ highlights that what matters is *whether an action $a \in \mathcal{A}$ is played by the expert* and not the corresponding probability $\pi_h^E(a|s)$. Indeed, the expert's policy plays an action (with any non-zero probability) only if it is an optimal action.

To prove Theorem 12, we need the following lemma:

Lemma 3. *Let r be feasible for the IRL problem $(\mathcal{M}, \pi^E)$ bounded in $[-1,1]$ (i.e., $\widehat{r} \in \mathcal{R}$) and defined according to Lemma 2 as $r_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\pi_h^E(a|s)=0\}} + V_h(s) - p_h V_{h+1}(s,a)$. Let $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$ be an IRL problem and define for every $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$:*

$$\begin{aligned}\epsilon_h(s,a) \coloneqq -A_h(s,a)\left(\mathbb{1}_{\{\pi_h^E(a|s)=0\}} - \mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}}\right) \\ + \left(\left(p_h - \widehat{p}_h\right) V_{h+1}\right)(s,a).\end{aligned}$$

Then, the reward function $\widehat{r}$ defined according to Lemma 2 as $\widehat{r}_h(s,a) = -\widehat{A}_h(s,a)\mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}} + \widehat{V}_h(s) - p_h\widehat{V}_{h+1}(s,a)$ for every $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$ with:

$$\widehat{A}_h(s,a) = \frac{A_h(s,a)}{1+\epsilon}, \quad \widehat{V}_h(s) = \frac{V_h(s)}{1+\epsilon}, \quad \widehat{V}_{H+1}(s) = 0.$$

where $\epsilon \coloneqq \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} |\epsilon_h(s,a)|$, is feasible for the IRL problem $(\widehat{\mathcal{M}}, \widehat{\pi}^E)$ and bounded in $[-1,1]$ (i.e., $\widehat{r} \in \widehat{\mathcal{R}}$).

Proof. Given the reward function $r_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\pi_h^E(a|s)=0\}} + V_h(s) - p_h V_{h+1}(s,a)$, we define the reward function:

$$\widetilde{r}_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}} + V_h(s) - \widehat{p}_h V_{h+1}(s,a),$$

that, thanks to Lemma 2, makes policy $\widehat{\pi}^E$ optimal. However, it is not guaranteed that $\widetilde{r} \in \widehat{\mathcal{R}}$ since it can take values larger than 1. Thus, we define the reward:

$$\widehat{r}_h(s,a) = \frac{\widetilde{r}_h(s,a)}{1+\epsilon} = -\frac{A_h(s,a)}{1+\epsilon}\mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}} + \frac{V_h}{1+\epsilon}(s) - \widehat{p}_h\frac{V_{h+1}}{1+\epsilon}(s,a),$$

which simply scales $\widetilde{r}_h$ and preserves the optimality of $\widehat{\pi}^E$. We now prove that $\widehat{r}_h(s,a)$ is

bounded in $[-1,1]$. To do so, we prove that $\widetilde{r}_h(s,a)$ is bounded in $[-(1+\epsilon),(1+\epsilon)]$:

$$\begin{aligned}|\widetilde{r}_h(s,a)| &\leqslant |r_h(s,a)| + |\widetilde{r}_h(s,a) - r_h(s,a)| \\ &= 1 + \left|-A_h(s,a)\mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}} + \widehat{p}_h V_{h+1}(s) - \left(-A_h(s,a)\mathbb{1}_{\{\pi_h^E(a|s)=0\}} + p_h V_{h+1}(s)\right)\right| \\ &= 1 + |\epsilon_h(s,a)| \leqslant 1+\epsilon.\end{aligned}$$

□

We are now able to prove Theorem 12.

Proof. Let $\widetilde{r}$ as defined in the proof of Lemma 3. Then, we have:

$$\begin{aligned}|r_h(s,a) - \widehat{r}_h(s,a)| &= \left|r_h(s,a) - \frac{\widetilde{r}_h(s,a)}{1+\epsilon}\right| \\ &\leqslant \frac{1}{1+\epsilon}\left(|r_h(s,a) - \widetilde{r}_h(s,a)| + \epsilon\,|r_h(s,a)|\right) \\ &\leqslant \frac{2\epsilon}{1+\epsilon}.\end{aligned}$$

By recalling that $\frac{2\epsilon}{1+\epsilon}$ is a non-decreasing function of ϵ, we bound it by replacing ϵ with an upper bound:

$$\begin{aligned}\epsilon &= \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} |\epsilon_h(s,a)| \\ &\leqslant \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} (H-h+1)\left[\left|\mathbb{1}_{\{\pi_h^E(a|s)=0\}} - \mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}}\right| + \|p_h(\cdot|s,a) - \widehat{p}_h(\cdot|s,a)\|_1\right] \\ &=: \rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E)),\end{aligned}$$

where we used Hölder's inequality recalling that $|V_{h+1}(s,a)| \leqslant H-h$ and $|A_h(s,a)| \leqslant H-h+1$. Clearly, $\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E))$ is a (pre)metric. □

Fact 1. *There exist two MDP$\setminus$R $\mathcal{M}$ and $\widehat{\mathcal{M}}$ with transition models p and $\widehat{p}$ respectively, an expert's policy π^E and a reward function $r_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\pi^E(a|s)=0\}} + V_h(s) - p_h V_{h+1}(s)$ feasible for the IRL problem $(\mathcal{M},\pi^E)$ bounded in $[-1,1]$ (i.e., $r\in\mathcal{R}$) such that the reward function $\widehat{r}_h(s,a) = -A_h(s,a)\mathbb{1}_{\{\pi^E(a|s)=0\}} + V_h(s) - \widehat{p}_h V_{h+1}(s)$ is feasible for the IRL problem $(\widehat{\mathcal{M}},\pi^E)$ not bounded in $[-1,1]$.*

Proof. We consider the MDP\R in Figure 4.1 with optimal policy and reward function defined for every $h\in[\![H]\!]$ and $H=10$ as:

$$\begin{aligned}&\pi_h^E(s_1) = a_1,\ \pi_h^E(s_2) = a_2, \\ &r_h(s_1,a_1) = r_h(s_2,a_1) = 0,\ r_h(s_1,a_2) = -1,\ r_h(s_2,a_2) = 1.\end{aligned}$$

Simple calculations lead to the V-function and advantage function values:

$$V_h^{\pi^E}(s_1) = 0,\ V_h^{\pi^E}(s_2) = H - h + 1,$$
$$A_h^{\pi^E}(s_1, a_1) = 0,\ A_h^{\pi^E}(s_1, a_2) = -1 + (H - h)/10$$
$$A_h^{\pi^E}(s_2, a_1) = -1 - (H - h)/10,\ A_h^{\pi^E}(s_2, a_2) = 0.$$

We consider as alternative transition model $\widehat{p} = 1 - p$. After tedious calculations we obtain the alternative reward function:

$$\widehat{r}_h(s_1, a_1) = -(H - h),$$
$$\widehat{r}_h(s_1, a_2) = -1 + 8(H - h)/10,$$
$$\widehat{r}_h(s_2, a_1) = 8(H - h)/10,$$
$$\widehat{r}_h(s_2, a_2) = H - h.$$

It is simple to observe that for some (s, a, h) we have $|\widehat{r}_h(s, a)| > 1$.

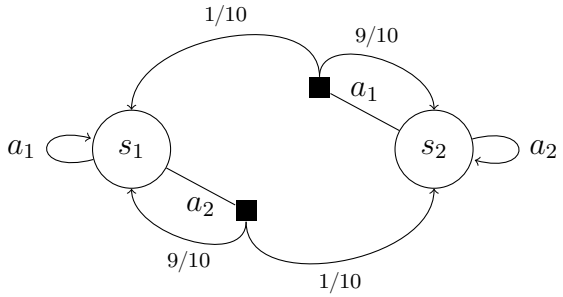


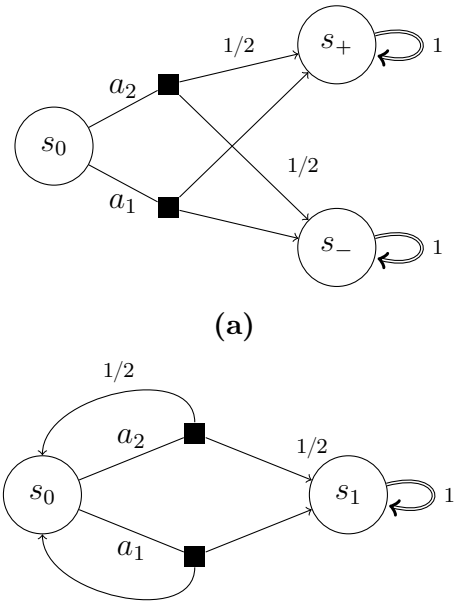
Figure 4.1: The MDP\R employed in Fact 1.

□

4.2. Non-Lipschitz Continuous Feasible Reward Sets

In this section, three cases of feasible reward sets restrictions that turn out not to fulfill the condition of Theorem 12 are illustrated. These examples consider three conditions commonly enforced in the literature: state-only reward function $r_h(s)$ (Example 1), time-homogeneous reward function $r(s, a)$ (Example 2), and β-margin reward function (Example 3). Counter-examples are presented in which in front of ϵ-close transition models, the induced feasible sets are far apart by a constant independent of ϵ.

Example 1 (State-only reward $r_h(s)$)**.** *State-only reward functions have been widely considered in many IRL approaches [e.g., 1, 30, 42, 53]. Let us formalize the state-only*



(a)

(b)

Figure 4.2: The MDP\R employed in the examples of Section 4.2. The arrow with double border denotes a transition executed for multiple actions.

feasible reward set as follows:

$$\mathcal{R}_{state} = \mathcal{R} \cap \{\forall(s, a, a', h) \,:\, r_h(s, a) = r_h(s, a')\}.$$

Consider the MDP\R of Figure 4.2a with $H = 2$, $\pi_h^E(s_0) = \widehat{\pi}_h^E(s_0) = a_1$ *with* $h \in \{1, 2\}$. *Set* $p_1(s_+|s_0, a_1) = 1/2 + \epsilon/4$ *and* $\widehat{p}_1(s_+|s_0, a_1) = 1/2 - \epsilon/4$ *and, thus,* $\|p_1(\cdot|s_0, a_1) - \widehat{p}_1(\cdot|s_0, a_1)\|_1 = \epsilon$. *Let us set* $r_2(s_+) = 1$ *and* $r_2(s_-) = -1$, *which makes* π^E *optimal under* p. *We observe that* $\widehat{\mathcal{R}}$ *is defined by* $\widehat{r}_2(s_-) \leqslant \widehat{r}_2(s_+)$. *Recalling that the rewards are bounded in* $[-1, 1]$, *we have* $\mathcal{H}_{d^G}(\mathcal{R}_{state}, \widehat{\mathcal{R}}_{state}) \geqslant 1$.

Proof. For the MDP\R $\mathcal{M}$, in order to make $\pi_1^E(s_0) = a_1$ optimal, we have to enforce:

$$\begin{aligned} & r_1(s_0) + \frac{2+\epsilon}{4} r_2(s_+) + \frac{2-\epsilon}{4} r_2(s_-) \geqslant r_1(s_0) + \frac{1}{2} r_2(s_+) + \frac{1}{2} r_2(s_-) \\ & \implies r_2(s_+) \geqslant r_2(s_-). \end{aligned}$$

Similarly, to make $\widehat{\pi}_1^E(s_0) = a_1$, we have for $\widehat{\mathcal{M}}$:

$$\begin{aligned} & \widehat{r}_1(s_0) + \frac{2-\epsilon}{4} \widehat{r}_2(s_+) + \frac{2+\epsilon}{4} \widehat{r}_2(s_-) \geqslant \widehat{r}_1(s_0) + \frac{1}{2} \widehat{r}_2(s_+) + \frac{1}{2} \widehat{r}_2(s_-) \\ & \implies \widehat{r}_2(s_+) \leqslant \widehat{r}_2(s_-). \end{aligned}$$

Thus, suppose, we set $r_2(s_-) = 1$ and $r_2(s_+) = -1$, we have:

$$\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\text{state}}, \widehat{\mathcal{R}}_{\text{state}}) \geqslant \min_{\substack{\widehat{r}_2(s_-),\widehat{r}_2(s_+)\in[-1,1] \\ \widehat{r}_2(s_+)\leqslant\widehat{r}_2(s_-)}} \max\{|1-\widehat{r}_2(s_-)|, |-1-\widehat{r}_2(s_+)|\} = 1.$$

□

Example 2 (Time-homogeneous reward $r(s,a)$)**.** *Time-homogeneous reward functions have been employed in several RL [e.g., 12] and IRL settings [e.g., 36]. Let us formalize the time-homogeneous feasible reward set as follows:*

$$\mathcal{R}_{hom} = \mathcal{R} \cap \{\forall(s,a,h,h') \,:\, r_h(s,a) = r_{h'}(s,a)\}.$$

Consider the MDP$\setminus$R of Figure 4.2b with $H=2$, $\pi_1^E(s_0)=\widehat{\pi}_1^E(s_0)=a_1$ and $\pi_2^E(s_0)=\widehat{\pi}_2^E(s_0)=a_2$. For $h\in\{1,2\}$, set $p_h(s_0|s_0,a_1)=1/2+\epsilon/4$ and $\widehat{p}_h(s_0|s_0,a_1)=1/2-\epsilon/4$, thus, $\|p_h(\cdot|s_0,a_1)-\widehat{p}_h(\cdot|s_0,a_1)\|_1=\epsilon$. Set also $r(s_0,a_1)=1$, $r(s_0,a_2)=1-\epsilon/6$, and $r(s_1,a_1)=r(s_1,a_2)=1/2$ making π^E optimal. It can be proved that $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{hom},\widehat{\mathcal{R}}_{hom})\geqslant 1/4$.

Proof. Consider the MDP$\setminus$R $\mathcal{M}$ and we set $r(s_0,a_1) = 1$, $r(s_0,a_2) = 1-\epsilon/12$, and $r(s_1,a) = 1/2$ for $a \in \{a_1,a_2\}$. We immediately observe that π^E is optimal since for $h=2$, $r(s_0,a_1) \geqslant r(s_0,a_2)$ and for $h=1$:

$$\begin{aligned}
&r(s_0,a_2) + \frac{2+\epsilon}{4}r(s_0,a_1) + \frac{2-\epsilon}{4}r(s_1,a) \geqslant r(s_0,a_1) + \frac{1}{2}r(s_0,a_1) + \frac{1}{2}r(s_1,a) \\
&\iff r(s_0,a_2) + \left(\frac{\epsilon}{4}-1\right)r(s_0,a_1) - \frac{\epsilon}{4}r(s_1,a) \geqslant 0 \\
&\iff 1 - \frac{\epsilon}{12} + \frac{\epsilon}{4} - 1 - \frac{\epsilon}{8} \geqslant 0.
\end{aligned}$$

Consider now the alternative MDP$\setminus$R $\widehat{\mathcal{M}}$, we have to enforce the following two conditions:

$$\widehat{r}(s_0,a_1) \geqslant \widehat{r}(s_0,a_2), \tag{4.3}$$

$$\begin{aligned}
&\widehat{r}(s_0,a_2) + \frac{2-\epsilon}{4}\widehat{r}(s_0,a_1) + \frac{2+\epsilon}{4}\widehat{r}(s_1,a) \geqslant \widehat{r}(s_0,a_1) + \frac{1}{2}\widehat{r}(s_0,a_1) + \frac{1}{2}\widehat{r}(s_1,a) \\
&\iff \widehat{r}(s_0,a_2) - \left(\frac{\epsilon}{4}+1\right)\widehat{r}(s_0,a_1) + \frac{\epsilon}{4}\widehat{r}(s_1,a) \geqslant 0.
\end{aligned} \tag{4.4}$$

The way of enforcing Equation (4.3) that is less constraining for Equation (4.4) is setting $\widehat{r}(s_0,a_1) = \widehat{r}(s_0,a_2)$, to get:

$$-\frac{\epsilon}{4}\widehat{r}(s_0,a_1) + \frac{\epsilon}{4}\widehat{r}(s_1,a) \geqslant 0 \iff \widehat{r}(s_1,a) \geqslant \widehat{r}(s_0,a_1).$$

This implies:

$$\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathrm{hom}}, \widehat{\mathcal{R}}_{\mathrm{hom}}) \geqslant \min_{\substack{\widehat{r}(s_1,a),\widehat{r}(s_0,a_1)\in[-1,1] \\ \widehat{r}(s_1,a)\geqslant\widehat{r}(s_0,a_1)}} \max\left\{ \left|1-\widehat{r}(s_0,a_1)\right|, \left|\frac{1}{2}-\widehat{r}(s_1,a)\right| \right\} \geqslant \frac{1}{4},$$

by setting $\widehat{r}(s_0,a_1)=\widehat{r}(s_1,a)=1/4$. □

Example 3 (β-margin reward)**.** *A β-margin reward enforces a suboptimality gap of at least $\beta > 0$ [30, 42]. We formalize it in the finite-horizon case with a sequence $\beta = (\beta_h)_{h\in[\![H]\!]}$, possibly different for every stage:*

$$\mathcal{R}_{\beta\text{-}mar} = \mathcal{R} \cap \{\forall (s,a,h) : A_h^{\pi^E}(s,a;r) \in \{0\} \cup (-\infty, -\beta_h]\}.$$

Consider the MDP$\setminus R$ in Figure 4.2a with $\pi_h^E(s_0) = \widehat{\pi}_h^E(s_0) = a_1$ for $h \in \{1,2\}$. We set $p_1(s_+|s_0,a_1) = 1/2+\epsilon$ and $\widehat{p}_1(s_+|s_0,a_1) = 1/2-\epsilon$. We set for MDP$\setminus R$ $\mathcal{M}$ the reward function as $r_1(s_0,a)=0$ and $r_h(s_+,a) = -r_h(s_-,a) = 1$ for $a \in \{a_1,a_2\}$ and $h \in [\![2,H]\!]$. In $(s_0,1)$ the suboptimality gap is $\beta_1 = 2+2\epsilon(H-1)$. By selecting $H \geqslant 1+1/\epsilon$, the feasible set $\widehat{\mathcal{R}}_{\beta\text{-}mar}$ is empty.

Proof. Concerning the MDP\R $\mathcal{M}$, we observe that by setting $r_1(s_0,a_1)=1$, $r_1(s_0,a_2)=-1$, and $r_h(s_+,a) = -r_h(s_-,a) = 1$ for $a \in \{a_1,a_2\}$ and $h \in [\![2,H]\!]$, the policy π^E is optimal. In particular, in state-stage pair $(s_0,1)$ the suboptimality gap is given by $\beta_1 = 2+2\epsilon(H-1)$. To enforce the optimality of $\widehat{\pi}^E = \pi^E$ in the MDP\R $\widehat{\mathcal{M}}$, we have:

$$\widehat{r}_1(s_0,a_1) + \sum_{h=2}^{H} \frac{1}{2}\widehat{r}_h(s_+,a_1) + \frac{1}{2}\widehat{r}_h(s_-,a_1) \geqslant \widehat{r}_1(s_0,a_2) + \sum_{h=2}^{H} \frac{1}{2}\widehat{r}_h(s_+,a_1) + \frac{1}{2}\widehat{r}_h(s_-,a_1) + \beta_1$$
$$\iff \widehat{r}_1(s_0,a_1) - \widehat{r}_1(s_0,a_2) \geqslant \beta_1.$$

Thus, if $\beta_1 \geqslant 2$, we have that the feasible set $\widehat{\mathcal{R}}_{\beta\text{-sep}}$ is empty. Thus, we select $H \geqslant 1+1/\epsilon$ to have $\beta_1 \geqslant 4$. □

These examples show that, under certain classes of restrictions, the feasible reward set is not Lipschitz continuous w.r.t. the transition model and, more in general, w.r.t. the IRL problem.

5 | PAC Framework for IRL with a generative Model

In this section, we discuss the PAC (Probably Approximately Correct) requirements for estimating the feasible reward set with access to a *generative model* of the environment. We first provide the notion of a learning algorithm estimating the feasible reward set with a generative model (Section 5.1). Then, we formally present the PAC requirement for the Hausdorff (pre)metric $\mathcal{H}_d$ (Section 5.2). Finally, we discuss the relationships between the PAC requirements with different choices of (pre)metric d (Section 5.2.1).

5.1. Learning Algorithms with a Generative Model

A learning algorithm for estimating the feasible reward set is a pair $\mathfrak{A} = (\mu, \tau)$, where $\mu = (\mu_t)_{t\in\mathbb{N}}$ is a *sampling strategy* defined for every time step $t \in \mathbb{N}$ as $\mu_t \in \Delta_{\mathcal{D}_{t-1}}^{\mathcal{S}\times\mathcal{A}\times[\![H]\!]}$ with $\mathcal{D}_t = (\mathcal{S}\times\mathcal{A}\times[\![H]\!]\times\mathcal{S}\times\mathcal{A})^t$ and τ is a stopping time w.r.t. a suitably defined filtration. At every step $t \in \mathbb{N}$, the learning algorithm query the environment in a triple (s_t, a_t, h_t), selected based on the sampling strategy $\mu_t(\cdot|D_{t-1})$, where $D_{t-1} = ((s_l, a_l, h_l, s'_l, a_l^E))_{l=1}^{t-1} \in \mathcal{D}_{t-1}$ is the dataset of past samples. Then, the algorithm observes the next state $s'_t \sim p_{h_t}(\cdot|s_t, a_t)$ and expert's action $a_t^E \sim \pi_{h_t}^E(\cdot|s_t)$ and updates the dataset $D_t = D_{t-1} \oplus (s_t, a_t, h_t, s'_t, a_t^E)$. Based on the collected data D_τ, the algorithm computes the empirical IRL problem $(\widehat{M}^\tau, \widehat{\pi}^{E,\tau})$, based on Equation (7.1) and the empirical feasible reward set $\widehat{\mathcal{R}}^\tau$.

5.2. PAC Requirement

We now introduce a general notion of a PAC requirement for estimating the feasible reward set of an IRL problem. To this end, we consider the Hausdorff (pre)metric introduced in Section 4 defined in terms of the reward (pre)metric $d(r, \widehat{r})$. We denote with d-IRL the problem of estimating the feasible reward set under the Hausdorff (pre)metric $\mathcal{H}_d$.

Definition 41 (PAC Algorithm for d-**IRL**)**.** *Let $\epsilon \in (0, 2)$ and $\delta \in (0, 1)$. An algorithm*

$\mathfrak{A} = (\mu, \tau)$ *is* (ϵ, δ)*-PAC for d-IRL if:*

$$\mathop{\mathbb{P}}_{(\mathcal{M},\pi^E),\mathfrak{A}}\left(\mathcal{H}_d(\mathcal{R}, \widehat{\mathcal{R}}^\tau) \leqslant \epsilon\right) \geqslant 1-\delta,$$

where $\mathbb{P}_{(\mathcal{M},\pi^E),\mathfrak{A}}$ *denotes the probability measure induced by executing the algorithm* $\mathfrak{A}$ *in the IRL problem* $(\mathcal{M}, \pi^E)$ *and* $\widehat{\mathcal{R}}^\tau$ *is the feasible reward set induced by the empirical IRL problem* $(\widehat{\mathcal{M}}^\tau, \widehat{\pi}^{E,\tau})$ *estimated with the dataset* D_τ*. The* sample complexity *is defined as* $\tau := |D_\tau|$.

In the next section, we show the relationship between PAC requirements defined for notable choices of d.

5.2.1. Different Choices of d

So far, we have evaluated the dissimilarity between the feasible reward sets by means of the Hausdorff induced by d^{G}, i.e., the L_∞-norm between individual reward functions. In the literature, other (pre)metrics d have been proposed [e.g., 36, 39].

$d^{\mathrm{G}}_{Q^*}$-IRL

Since the recovered reward functions are often used for performing forward RL, an index of interest is the dissimilarity between optimal Q-functions obtained with the reward $r \in \mathcal{R}$ and $\widehat{r} \in \widehat{\mathcal{R}}$ in the original MDP\R:

$$d^{\mathrm{G}}_{Q^*}(r, \widehat{r}) := \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \left|Q^*_h(s,a;r) - Q^*_h(s,a;\widehat{r})\right|.$$

$d^{\mathrm{G}}_{V^*}$-IRL

We are often interested in not just being accurate in estimating the optimal Q-function, but rather in the performance of an optimal policy $\widehat{\pi}^*$, learned with the recovered reward $\widehat{r} \in \widehat{\mathcal{R}}$, evaluated under the true reward $r \in \mathcal{R}$:

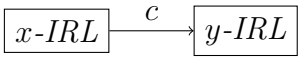
$$d^{\mathrm{G}}_{V^*}(r, \widehat{r}) := \sup_{\widehat{\pi}^*\in\Pi^*(\widehat{r})} \max_{(s,h)\in\mathcal{S}\times[\![H]\!]} \left|V^*_h(s;r) - V^{\widehat{\pi}^*}_h(s;r)\right|$$

where $\Pi^*(\widehat{r}) := \{\pi : \forall (s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!] : A^\pi_h(s,a;\widehat{r}) \leqslant 0\}$ is the set of optimal policies under the recovered reward $\widehat{r}$.

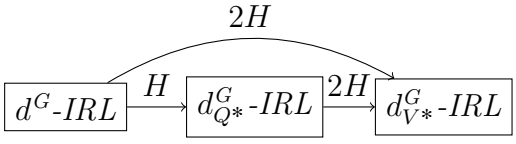
The following result formalizes the relationships between the presented d-IRL problems.

Theorem 13 (Relationships between d-IRL problems)**.** *Let us introduce the graphical*

convention for $c > 0$:



meaning that any (ϵ, δ)-PAC x-IRL algorithm is $(c\epsilon, \delta)$-PAC y-IRL. Then, the following statements hold:



Theorem 13 shows that any (ϵ, δ)-PAC guarantee on d^{G}, implies (ϵ', δ)-PAC guarantees on both d^{G}_{Q*} and d^{G}_{V*}, where $\epsilon' = \Theta(H\epsilon)$ is linear in the horizon H. This justifies why focusing on d^{G}-IRL, as in the following section where sample complexity lower bounds are derived. The lower bound analysis for d^{G}_{Q*}-IRL and d^{G}_{V*}-IRL is left to future works.

Proof. Let $\mathfrak{A}$ be an (ϵ, δ)-PAC d^{G}-IRL algorithm. This means that with probability at least $1-\delta$, we have that for any IRL problem $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^{\tau}) \leqslant \epsilon$. We introduce the following visitation distributions, defined for every $s, s' \in \mathcal{S}$, $h, l \in [\![H]\!]$ with $l \geqslant h$, and $a, a' \in \mathcal{A}$:

$$\eta^{\pi}_{s,a,h,l}(s', a') = \mathop{\mathbb{P}}_{\mathcal{M},\pi}\left(s_l = s', a_l = a' | s_h = s, a_h = a\right),$$
$$\eta^{\pi}_{s,h,l}(s', a') = \sum_{a \in \mathcal{A}} \pi_h(a|s) \eta^{\pi}_{s,a,h,l}(s', a').$$

d^{G}-IRL $\rightarrow$ d^{G}_{Q*}-IRL Let us consider the optimal Q-function difference and let π^* an optimal policy under the reward function r, we have:

$$\begin{aligned}
Q^*_h(s,a;r) - Q^*_h(s,a;\widehat{r}) &\leqslant Q^{\pi^*}_h(s,a;r) - Q^{\pi^*}_h(s,a;\widehat{r}) \\
&= \sum_{l=h}^{H} \sum_{(s',a') \in \mathcal{S}\times\mathcal{A}} \eta^{\pi^*}_{s,a,h,l}(s',a')(r_l(s',a') - \widehat{r}_l(s',a')) \\
&\leqslant \max_{(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)| \sum_{l=h}^{H} \underbrace{\sum_{(s',a') \in \mathcal{S}\times\mathcal{A}} \eta^{\pi^*}_{s,a,h,l}(s',a')}_{=1} \\
&= (H-h+1) \max_{(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)| \\
&\leqslant H \max_{(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)|.
\end{aligned}$$

As a consequence, we have:

$$\mathcal{H}_{d^{\mathrm{G}}_{Q^*}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau) \leqslant H\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau).$$

$d^{\mathbf{G}}$-IRL $\rightarrow$ $d^{\mathbf{G}}_{V^*}$-IRL Let us consider the value functions and let π^* (resp. $\widehat{\pi}^*$) be an optimal policy under reward function r (resp. $\widehat{r}$), we have:

$$\begin{aligned}
V_h^*(s;r) - V_h^{\widehat{\pi}^*}(s;r) &= V_h^{\pi^*}(s;r) - V_h^{\widehat{\pi}^*}(s;r) \pm V_h^{\widehat{\pi}^*}(s;\widehat{r}) \\
&\leqslant V_h^{\pi^*}(s;r) - V_h^{\pi^*}(s;\widehat{r}) + V_h^{\widehat{\pi}^*}(s;\widehat{r}) - V_h^{\widehat{\pi}^*}(s;r) \\
&= \sum_{l=h}^{H} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \eta^{\pi^*}_{s,h,l}(s',a')(r_l(s',a') - \widehat{r}_l(s',a')) + \\
&+ \sum_{l=h}^{H} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \eta^{\widehat{\pi}^*}_{s,h,l}(s',a')(r_l(s',a') - \widehat{r}_l(s',a')) \\
&\leqslant \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)| \cdot \\
&\cdot \left(\sum_{l=h}^{H} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \eta^{\pi^*}_{s,h,l}(s',a') + \sum_{l=h}^{H} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \eta^{\widehat{\pi}^*}_{s,h,l}(s',a') \right) \\
&= 2(H-h+1) \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)| \\
&\leqslant 2H \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} |r_h(s,a) - \widehat{r}_h(s,a)|.
\end{aligned}$$

Thus, it follows that:

$$\mathcal{H}_{d^{\mathrm{G}}_{V^*}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau) \leqslant 2H\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau).$$

$d^{\mathbf{G}}_{Q^*}$-IRL $\rightarrow$ $d^{\mathbf{G}}_{V^*}$-IRL To prove this result, we need to introduce further tools. Specifically, we introduce the Bellman expectation operator and the Bellman optimal operator, defined for a reward function r, policy π, $(s,h) \in \mathcal{S} \times [\![H]\!]$ and function $f : \mathcal{S} \rightarrow \mathbb{R}$:

$$T^*_{r,h} f(s) = \max_{a\in\mathcal{A}} \{r_h(s,a) + p_h f(s,a)\}, \qquad T^\pi_{r,h} f(s) = \pi_h \left(r_h(s,a) + p_h f(s,a)\right).$$

We recall the fixed-point properties: $T^*_{r,h} V^*_h = V^*_h$ and $T^\pi_{r,h} V^\pi_h = V^\pi_h$. Let π^* (resp. $\widehat{\pi}^*$) be

an optimal policy under reward r (resp. $\widehat{r}$). Let us consider the following derivation:

$$\begin{aligned}
V_h^*(s;r) - V_h^{\widehat{\pi}^*}(s;r) &= T_{r,h}^* V_h^*(s;r) - T_{r,h}^{\widehat{\pi}^*} V_h^{\widehat{\pi}^*}(s;r) \pm \\
&\pm T_{r,h}^{\pi^*} V_h^*(s;\widehat{r}) \pm T_{\widehat{r},h}^{\pi^*} V_h^*(s;\widehat{r}) \pm T_{\widehat{r},h}^* V_h^*(s;\widehat{r}) \pm T_{r,h}^{\widehat{\pi}^*} V_h^{\widehat{\pi}^*}(s;\widehat{r}) \\
&= T_{r,h}^{\pi^*} V_h^*(s;r) - T_{r,h}^{\pi^*} V_h^*(s;\widehat{r}) + T_{r,h}^{\pi^*} V_h^*(s,\widehat{r}) - \\
&- T_{\widehat{r},h}^{\pi^*} V_h^*(s;\widehat{r}) + \underbrace{T_{\widehat{r},h}^{\pi^*} V_h^*(s;\widehat{r}) - T_{\widehat{r},h}^* V_h^*(s;\widehat{r})}_{\leqslant 0} \\
&\quad + T_{\widehat{r},h}^{\widehat{\pi}^*} V_h^*(s;\widehat{r}) - T_{r,h}^{\widehat{\pi}^*} V_h^*(s;\widehat{r}) + T_{r,h}^{\widehat{\pi}^*} V_h^*(s;\widehat{r}) - T_{r,h}^{\widehat{\pi}^*} V_h^*(s;r) \\
&= \pi_h^* p_h(V_{h+1}^*(\cdot;r) - V_{h+1}^*(\cdot;\widehat{r}))(s) + \pi_h^*(r_h - \widehat{r}_h)(s) \\
&\quad + \widehat{\pi}_h^*(\widehat{r}_h - r_h)(s) + \widehat{\pi}^* p_h(V_{h+1}^*(\cdot;\widehat{r}) - V_{h+1}^{\widehat{\pi}^*}(\cdot;r))(s) \\
&= (\pi_h^* - \widehat{\pi}_h^*)(Q_h^*(\cdot;r) - Q_h^*(\cdot;\widehat{r}))(s) + \widehat{\pi}^* p_h(V_{h+1}^*(\cdot;r) - V_{h+1}^{\widehat{\pi}^*}(\cdot;r))(s).
\end{aligned}$$

Let us apply the L_∞-norm over the state space and the triangular inequality, we have:

$$\begin{aligned}
\left\| V_h^*(\cdot;r) - V_h^{\widehat{\pi}^*}(\cdot;r) \right\|_\infty &\leqslant \left\| (\pi_h^* - \widehat{\pi}_h^*)(Q_h^*(\cdot;r) - Q_h^*(\cdot;\widehat{r}))(\cdot) \right\|_\infty + \\
&\quad + \left\| \widehat{\pi}^* p_h(V_{h+1}^*(\cdot;r) - V_{h+1}^{\widehat{\pi}^*}(\cdot;r))(\cdot) \right\|_\infty \\
&\leqslant 2 \left\| Q_h^*(\cdot;r) - Q_h^*(\cdot;\widehat{r}))(\cdot) \right\|_\infty + \left\| V_{h+1}^*(\cdot;r) - V_{h+1}^{\widehat{\pi}^*}(\cdot;r) \right\|_\infty .
\end{aligned}$$

By unfolding the recursion over h, we obtain:

$$\left\| V_h^*(\cdot;r) - V_h^{\widehat{\pi}^*}(\cdot;r) \right\|_\infty \leqslant 2 \sum_{l=h+1}^{H} \left\| Q_l^*(\cdot;r) - Q_l^*(\cdot;\widehat{r}))(\cdot) \right\|_\infty .$$

Thus, we have:

$$\max_{(s,h)\in\mathcal{S}\times[\![H]\!]} \left| V_h^*(s;r) - V_h^{\widehat{\pi}^*}(s;r) \right| \leqslant 2H \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \left| Q_h^*(s,a;r) - Q_h^*(s,a;\widehat{r}) \right| .$$

Since the derivation is carried out for arbitrary $\widehat{\pi}^*$, it follows that:

$$\mathcal{H}_{d_{V^*}^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau) \leqslant 2H \mathcal{H}_{d_{Q^*}^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau).$$

□

6 | Lower Bounds

In this section, we establish sample complexity lower bounds for the d^{G}-IRL problem based on the PAC requirement of Definition 41 in the generative model setting. We start presenting the general result (Section 6.1) and, then, we comment on its form and, subsequently, provide a sketch of the construction of the hard instances for obtaining the lower bound (Section 6.2). For the sake of presentation, we assume that the expert's policy π^E is known; the extension to the case of unknown π^E is reported in Appendix A.

6.1. Main Result

In this section, we report the main result of the lower bound of the sample complexity of learning the feasible reward set.

Theorem 14 (Lower Bound for d^{G}-IRL)**.** *Let $\mathfrak{A} = (\mu, \tau)$ be an (ϵ, δ)-PAC algorithm for d^G-IRL. Then, there exists an IRL problem $(\mathcal{M}, \pi^E)$ such that, if $\delta \leqslant 1/32$, $S \geqslant 9$, $A \geqslant 2$, and $H \geqslant 12$, the expected sample complexity is lower bounded by:*

- *if the transition model p is time-inhomogeneous:*

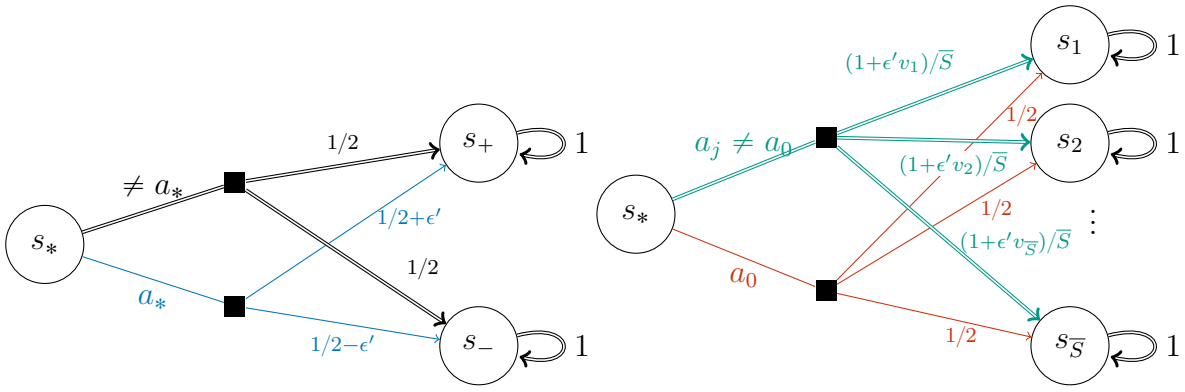
$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{1024}\frac{H^3SA}{\epsilon^2}\left(\frac{1}{2}\log\left(\frac{1}{\delta}\right) + \frac{1}{5}S\right);$$

- *if the transition model p is time-homogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{512}\frac{H^2SA}{\epsilon^2}\left(\frac{1}{2}\log\left(\frac{1}{\delta}\right) + \frac{1}{5}S\right),$$

where $\mathbb{E}_{(\mathcal{M},\pi^E),\mathfrak{A}}$ denotes the expectation w.r.t. the probability measure $\mathbb{P}_{(\mathcal{M},\pi^E),\mathfrak{A}}$.

Some observations are in order. First, the derived lower bound displays a linear dependence on the number of actions A and dependence on the horizon H raised to a power 2 or 3, which depends on whether the underlying transition model is time-homogeneous, as common even for forward RL [e.g., 12, 15]. Second, we identify two different regimes visible inside the parenthesis related to the dependence on the number of states S and



(a) MDP\R used for the small-δ regime. **(b)** MDP\R used for the large-δ regime.

Figure 6.1: The MDP\R employed in the constructions of the lower bounds of Section 6. The expert's policy is $\pi^E(s) = a_0$. The arrow with double border denotes a transition executed for multiple actions.

the confidence δ. Specifically, for small values of δ (i.e., $\delta \approx 0$), the dominating part is $\log\left(\frac{1}{\delta}\right)$, leading to a sample complexity of order $\Omega\left(\frac{H^3SA}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right)$. Instead, for large δ (i.e., $\delta \approx 1/32$), the most relevant part is the one corresponding to S, leading to sample complexity of order $\Omega\left(\frac{H^3S^2A}{\epsilon^2}\right)$ (both for the time-inhomogeneous case). An analogous two-regime behavior has been previously observed in the reward-free exploration setting [23, 27, 38].

6.2. Proof

Proof. We put together the results of Theorem 15 and Theorem 16, by recalling that $\max\{a, b\} \geqslant \frac{a+b}{2}$, or, equivalently, assuming to observe instances like the ones of Theorem 15 w.p. 1/2 as well as those of Theorem 16. □

Theorem 15. *Let $\mathfrak{A} = (\mu, \tau)$ be an (ϵ, δ)-PAC algorithm for d^G-IRL. Then, there exists an IRL problem $(\mathcal{M}, \pi^E)$ such that, if $\epsilon \leqslant 1$, $\delta < 1/16$, $S \geqslant 9$, $A \geqslant 2$, and $H \geqslant 12$, the expected sample complexity is lower bounded by:*

- *if the transition model p is time-inhomogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{2048}\frac{H^3SA}{\epsilon^2}\log\left(\frac{1}{\delta}\right);$$

- *if the transition model p is time-homogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{1024}\frac{H^2SA}{\epsilon^2}\log\left(\frac{1}{\delta}\right).$$

Proof. **Step 1: Instances Construction** The construction of the hard MDP\R instances follows similar steps as the ones presented in the constructions of lower bounds for policy learning [15] and the hard instances are reported in Figure 6.2 in a semi-formal way. The state space is given by $\mathcal{S} = \{s_{\text{start}}, s_{\text{root}}, s_-, s_+, s_1, \dots, s_{\overline{S}}\}$ and the action space is given by $\mathcal{A} = \{a_0, a_1, \dots, a_{\overline{A}}\}$. The transition model is described below and the horizon is $H \geqslant 3$. We introduce the constant $\overline{H} \in [\![H]\!]$, whose value will be chosen later. Let us observe, for now, that if $\overline{H} = 1$, the transition model is time-homogeneous.

The agent begins in state s_{start}, where every action has the same effect. Specifically, if the stage $h < \overline{H}$, then there is probability $1/2$ to remain in s_{start} and a probability $1/2$ to transition to s_{root}. Instead, if $h \geqslant \overline{H}$, the state transitions to s_{root} deterministically. From state s_{root}, every action has the same effect and the state transitions with equal probability $1/\overline{S}$ to a state s_i with $i \in [\![\overline{S}]\!]$. In all states s_i, apart from a specific one, i.e., state s_*, all actions have the same effect, i.e., transitioning to states s_- and s_+ with equal probability $1/2$. State s_* behaves as the other ones if the stage $h \neq h_*$, where $h_* \in [\![H]\!]$ is a predefined stage. If, instead, $h = h_*$, all actions $a_j \neq a_*$ behave like in the other states, while for action a_*, we have a $1/2 + \epsilon'$ probability of reaching s_+ (and consequently probability $1/2 - \epsilon'$ of reaching s_-), with $\epsilon' \in [0, 1/4]$. Notice that, having fixed $\overline{H}$, the possible values of h^* are $\{3, \dots, 2 + \overline{H}\}$. States s_+ and s_- are absorbing states. The expert's policy always plays action a_0.

Let us consider the base instance $\mathcal{M}_0$ in which there is no state behaving like s_*. Additionally, by varying the triple $\ell := (s_*, a_*, h_*) \in \{s_1, \dots, s_{\overline{S}}\} \times \{a_1, \dots, a_{\overline{A}}\} \times [\![3, \overline{H} + 2]\!] =: \mathcal{I}$, we can construct the class of instances denoted by $\mathbb{M} = \{\mathcal{M}_\ell : \ell \in \{0\} \cup \mathcal{I}\}$.

Step 2: Feasible Set Computation Let us consider an instance $\mathcal{M}_\ell \in \mathbb{M}$, we now seek to provide a lower bound to the Hausdorff distance $\mathcal{H}_{d^{\text{G}}}(\mathcal{R}_{\mathcal{M}_0}, \mathcal{R}_{\mathcal{M}_\ell})$. To this end, we focus on the triple $\ell = (s_*, a_*, h_*)$ and we enforce the convenience of action a_0 over action a_*. For the base MDP\R $\mathcal{M}_0$, let $r^0 \in \mathcal{R}_{\mathcal{M}_0}$, we have:

$$r^0_{h_*}(s_*, a_0) + \frac{1}{2}\sum_{l=h_*+1}^{H}\left(r^0_l(s_-) + r^0_l(s_+)\right) \geqslant r^0_{h_*}(s_*, a_*) + \frac{1}{2}\sum_{l=h+1}^{H}\left(r^0_l(s_-) + r^0_l(s_+)\right)$$
$$\implies r^0_{h_*}(s_*, a_0) \geqslant r^0_{h_*}(s_*, a_*),$$

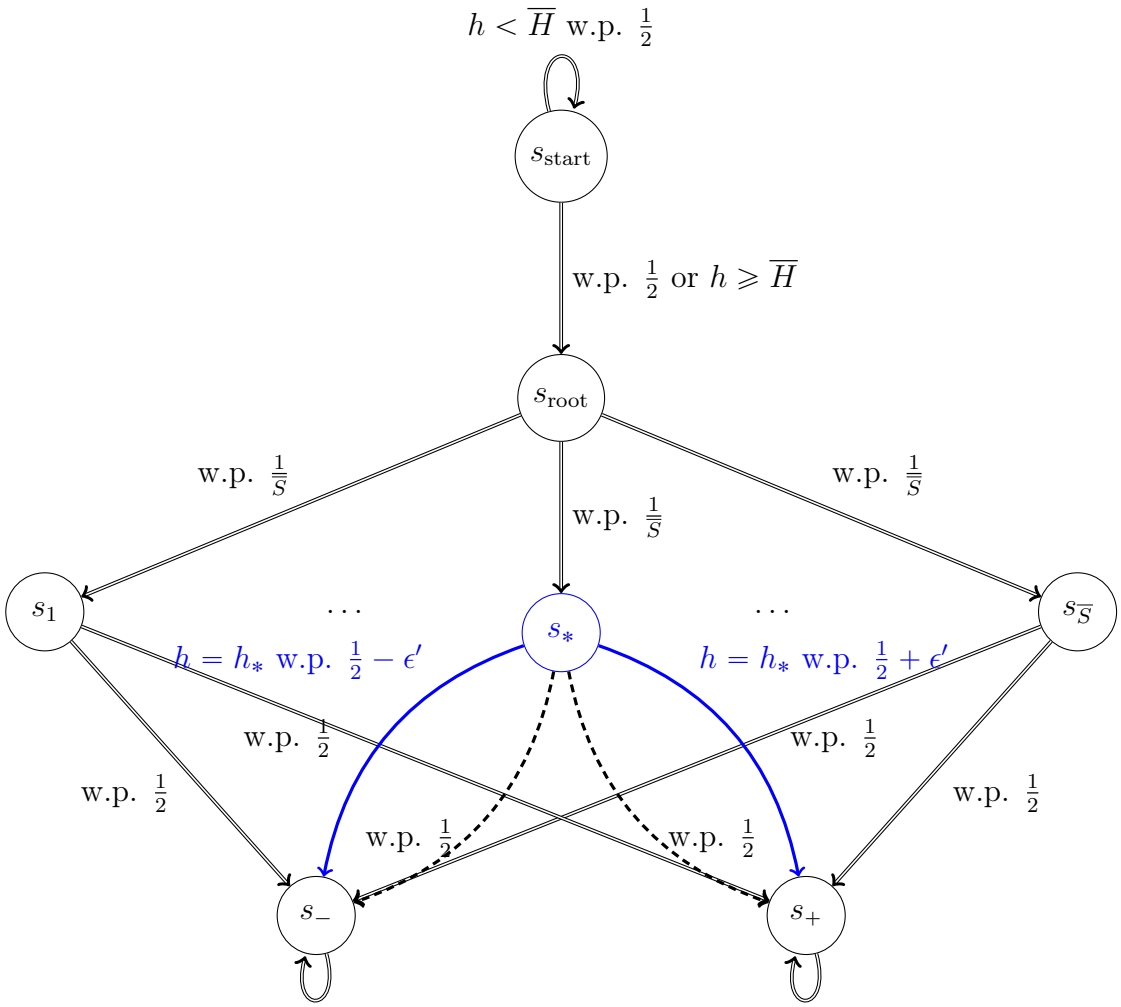


Figure 6.2: Semi-formal representation of the the hard instances MDP\R used in the proof of Theorem 15.

For the alternative MDP\R $\mathcal{M}_\ell$, let $r^\ell \in \mathcal{R}_{\mathcal{M}_\ell}$, we have:

$$
\begin{aligned}
& r^\ell_{h_*}(s_*, a_0) + \frac{1}{2} \sum_{l=h_*+1}^{H} \left(r^\ell_l(s_-) + r^\ell_l(s_+) \right) \geqslant \\
& \geqslant r^\ell_{h_*}(s_*, a_*) + \sum_{l=h_*+1}^{H} \left(\left(\frac{1}{2} - \epsilon' \right) r^\ell_l(s_-) + \left(\frac{1}{2} + \epsilon' \right) r^\ell_l(s_+) \right) \\
& \implies r^\ell_{h_*}(s_*, a_0) \geqslant r^\ell_{h_*}(s_*, a_*) - \epsilon' \sum_{l=h_*+1}^{H} \left(r^\ell_l(s_-) - r^\ell_l(s_+) \right).
\end{aligned}
$$

In order to lower bound the Hausdorff distance $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathcal{M}_0}, \mathcal{R}_{\mathcal{M}_\ell})$, we set for $\mathcal{M}_\ell$:

$$r_l^\ell(s_-) = -r_l^\ell(s_+) = 1,\ r_{h_*}^\ell(s_*, a_*) = 1,\ r_{h_*}^\ell(s_*, a_0) = 1 - 2\epsilon'(H - h_*).$$

Then, for notational convenience, for the MDP\R $\mathcal{M}_0$, we set $x \coloneqq r_{h_*}^0(s_*, a_0)$ and $y \coloneqq r_{h_*}^0(s_*, a_*)$:

$$\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_0}, \mathcal{R}_{\mathcal{M}_\ell}\right) \geqslant \min_{\substack{x,y\in[-1,1]\\ y\geqslant x}} \max\left\{|x-1|, |y-1+2\epsilon'(H-h_*)|\right\} = \epsilon'(H-h_*).$$

We enforce the following constraint on this quantity:

$$\forall h^* \in [\![3, \overline{H}+2]\!] \ :\ (H-h^*)\epsilon' \geqslant 2\epsilon \implies \epsilon' \geqslant \max_{h^*\in[\![3,\overline{H}+2]\!]} \frac{2\epsilon}{(H-h^*)} = \frac{2\epsilon}{(H-\overline{H}-2)}. \tag{6.1}$$

Notice that $\epsilon' \leqslant 1/4$ whenever $H \geqslant \overline{H} + 10$.

Step 3: Lower bounding Probability Let us consider an (ϵ, δ)-correct algorithm $\mathfrak{A}$ that outputs the estimated feasible set $\widehat{\mathcal{R}}$. Thus, for every $\imath \in \mathcal{I}$, we can lower bound the error probability:

$$\begin{aligned}
\delta &\geqslant \sup_{\text{all } \mathcal{M} \text{ MDP\R and expert policies } \pi} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}}, \widehat{\mathcal{R}}\right) \geqslant \epsilon\right) \\
&\geqslant \sup_{\mathcal{M}\in\mathbb{M}} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}}, \widehat{\mathcal{R}}\right) \geqslant \epsilon\right) \\
&\geqslant \max_{\ell\in\{0,\imath\}} \mathbb{P}_{(\mathcal{M}_\ell,\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_\ell}, \widehat{\mathcal{R}}\right) \geqslant \epsilon\right).
\end{aligned}$$

For every $\imath \in \mathcal{I}$, let us define the *identification function*:

$$\Psi_\imath \coloneqq \operatorname*{argmin}_{\ell\in\{0,\imath\}} \mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_\ell}, \widehat{\mathcal{R}}\right).$$

Let $\jmath \in \{0, \imath\}$. If $\Psi_\imath = \jmath$, then, $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathcal{M}_{\Psi_\imath}}, \mathcal{R}_{\mathcal{M}_\jmath}) = 0$. Otherwise, if $\Psi_\imath \neq \jmath$, we have:

$$\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_{\Psi_\imath}}, \mathcal{R}_{\mathcal{M}_\jmath}\right) \leqslant \mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_{\Psi_\imath}}, \widehat{\mathcal{R}}\right) + \mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_\jmath}\right) \leqslant 2\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_\jmath}\right),$$

where the first inequality follows from triangular inequality and the second one from the definition of identification function $\Psi_\imath$. From Equation (6.1), we have that $\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_{\Psi_\imath}}, \mathcal{R}_{\mathcal{M}_\jmath}\right) \geqslant 2\epsilon$. Thus, it follows that $\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_\jmath}\right) \geqslant \epsilon$. This implies the following inclusion of events for $\jmath \in \{0, \imath\}$:

$$\left\{\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_\jmath}\right) \geqslant \epsilon\right\} \supseteq \{\Psi_\imath \neq \jmath\}.$$

Thus, we can proceed by lower bounding the probability:

$$\begin{aligned}\max_{\ell\in\{0,\imath\}} \mathop{\mathbb{P}}_{(\mathcal{M}_\ell,\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\mathcal{M}_\ell},\widehat{\mathcal{R}}\right)\geqslant\epsilon\right) &\geqslant \max_{\ell\in\{0,\imath\}} \mathop{\mathbb{P}}_{(\mathcal{M}_\ell,\pi),\mathfrak{A}}(\Psi_\imath\neq\ell)\\ &\geqslant \frac{1}{2}\left[\mathop{\mathbb{P}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath\neq 0)+\mathop{\mathbb{P}}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath\neq\imath)\right]\\ &= \frac{1}{2}\left[\mathop{\mathbb{P}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath\neq 0)+\mathop{\mathbb{P}}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath= 0)\right],\end{aligned}$$

where the second inequality follows from the observation that $\max\{a,b\}\geqslant\frac{1}{2}(a+b)$ and the equality from observing that $\Psi_\imath\in\{0,\imath\}$. The intuition behind this derivation is that we lower bound the probability of making a mistake $\geqslant\epsilon$ with the probability of failing in identifying the true underlying problem. We can now apply the Bretagnolle-Huber inequality [32, Theorem 14.2] (also reported in Theorem 20 for completeness) with $\mathbb{P}=\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}$, $\mathbb{Q}=\mathbb{P}_{(\mathcal{M}_0,\pi)\mathfrak{A}}$, and $\mathcal{A}=\{\Psi_\imath\neq 0\}$:

$$\mathop{\mathbb{P}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath\neq 0)+\mathop{\mathbb{P}}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath= 0)\geqslant\frac{1}{2}\exp\left(-D_{\mathrm{KL}}\left(\mathop{\mathbb{P}}_{(\mathcal{M}_0,\pi),\mathfrak{A}},\mathop{\mathbb{P}}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}\right)\right).$$

Step 4: KL-divergence Computation Let $\mathcal{M}\in\mathbb{M}$, we denote with $\mathbb{P}_{\mathfrak{A},\mathcal{M},\pi}$ the joint probability distribution of all events realized by the execution of the algorithm in the MDP\R (the presence of π is irrelevant as we assume it known):

$$\mathop{\mathbb{P}}_{(\mathcal{M},\pi),\mathfrak{A}}=\prod_{t=1}^{\tau}\rho_t(s_t,a_t,h_t|H_{t-1})p_{h_t}(s'_t|s_t,a_t).$$

where $H_{t-1}=(s_1,a_1,h_1,s'_1,\ldots,s_{t-1},a_{t-1},h_{t-1},s'_{t-1})$ is the history. Let $\imath\in\mathcal{I}$ and denote with p^0 and $p^\imath$ the transition models associated with $\mathcal{M}_0$ and $\mathcal{M}_\imath$. Let us now move to the KL-divergence:

$$\begin{aligned}D_{\mathrm{KL}}\left(\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}},\mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}\right) &= \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[\sum_{t=1}^{\tau}D_{\mathrm{KL}}\left(p^0_{h_t}(\cdot|s_t,a_t),p^\imath_{h_t}(\cdot|s_t,a_t)\right)\right]\\ &\leqslant \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*,a_*)\right]D_{\mathrm{KL}}\left(p^0_{h_*}(\cdot|s_*,a_*),p^\imath_{h_*}(\cdot|s_*,a_*)\right)\\ &\leqslant 8(\epsilon')^2\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*,a_*)\right].\end{aligned}$$

having observed that the transition models differ in $\imath=(s_*,a_*,h_*)$ and defined $N^\tau_{h_*}(s_*,a_*)=\sum_{t=1}^{\tau}\mathbb{1}\{(s_t,a_t,h_t)=(s_*,a_*,h_*)\}$ and the last passage is obtained by Lemma 7 with $D=2$

(and $\epsilon = 2\epsilon'$). Putting all together, we have:

$$\delta \geqslant \frac{1}{4}\exp\left(-8 \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right](\epsilon')^2\right) \implies$$
$$\implies \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right] \geqslant \frac{\log\frac{1}{4\delta}}{8(\epsilon')^2} = \frac{(H-\overline{H}-2)^2\log\frac{1}{4\delta}}{32\epsilon^2}.$$

Thus, summing over $(s_*, a_*, h_*) \in \mathcal{I}$, we have:

$$\begin{aligned}\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] &\geqslant \sum_{(s_*,a_*,h_*)\in\mathcal{I}} \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right] \\ &= \sum_{(s_*,a_*,h_*)\in\mathcal{I}} \frac{(H-\overline{H}-2)^2\log\frac{1}{4\delta}}{32\epsilon^2} \\ &= \frac{\overline{SAH}(H-\overline{H}-2)^2}{32\epsilon^2}\log\frac{1}{4\delta}.\end{aligned}$$

The number of states is given by $S = |\mathcal{S}| = \overline{S} + 4$, the number of actions is given by $A = |\mathcal{A}| = \overline{A} + 1$. Let us first consider the time-homogeneous case, i.e., $\overline{H} = 1$:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{(S-4)(A-1)(H-3)^2}{32\epsilon^2}\log\frac{1}{4\delta}.$$

For $\delta < 1/16$, $S \geqslant 9$, $A \geqslant 2$, $H \geqslant 10$, we obtain:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{SAH^2}{1024\epsilon^2}\log\frac{1}{\delta}.$$

For the time-inhomogeneous case, instead, we select $\overline{H} = H/2$, to get:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{(S-4)(A-1)(H/2)(H-H/2-2)^2}{\epsilon^2}\log\frac{1}{4\delta}.$$

For $\delta < 1/16$, $S \geqslant 9$, $A \geqslant 2$, $H \geqslant 12$, we obtain:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{SAH^3}{2048\epsilon^2}\log\frac{1}{\delta}.$$

□

Theorem 16. *Let $\mathfrak{A} = (\mu, \tau)$ be an (ϵ, δ)-PAC algorithm for d^G-IRL. Then, there exists an IRL problem $(\mathcal{M}, \pi^E)$ such that, if $\epsilon \leqslant 1$, $\delta \leqslant 1/2$, $S \geqslant 16$, $A \geqslant 2$, $H \geqslant 131$, the expected sample complexity is lower bounded by:*

- *if the transition model p is time-inhomogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{5120}\frac{S^2AH^3}{\epsilon^2};$$

- *if the transition model p is time-homogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{1}{2560}\frac{S^2AH^2}{\epsilon^2}.$$

Proof. **Step 1: Instances Construction** The construction of the hard MDP\R instances for this second bound follows steps similar to those of reward free exploration [23] and the instances are reported in Figure 6.3 in a semi-formal way. The state space is given by $\mathcal{S} = \{s_{\text{start}}, s_{\text{root}}, s_1, \dots, s_{\overline{S}}, s'_1, \dots, s'_{\overline{S}}\}$ and the action space is given by $\mathcal{A} = \{a_0, a_1, \dots, a_{\overline{A}}\}$. We assume $\overline{S}$ to be divisible by 16. The transition model is described below and the horizon is $H \geqslant 3$.

The agent begins in state s_{start}, where every action has the same effect. Specifically, if the stage $h < \overline{H}$ ($\overline{H} \in [\![H]\!]$, whose value will be chosen later), then there is probability $1/2$ to remain in s_{start} and a probability $1/2$ to transition to s_{root}. Instead, if $h \geqslant \overline{H}$, the state transitions to s_{root} deterministically. From state s_{root}, every action has the same effect and the state transitions with equal probability $1/\overline{S}$ to a state s_i with $i \in [\![\overline{S}]\!]$. In every state s_i and every stage h, action a_0 allows reaching states $s'_1, \dots, s'_{\overline{S}}$ with equal probability $1/\overline{S}$. Instead, by playing the other actions a_j with $j \geqslant 1$ at stage h, the probability distribution of the next state is given by $p_h(s'_k|s_i, a_j) = (1 + \epsilon' v_k^{(s_i,a_j,h)})/\overline{S}$ where the vector $v^{(s_i,a_j,h)} = (v_1^{(s_i,a_j,h)}, \dots, v_{\overline{S}}^{(s_i,a_j,h)}) \in \mathcal{V}$, where $\mathcal{V} := \{\{-1,1\}^{\overline{S}} : \sum_{j=1}^{\overline{S}} v_j = 0\}$ and $\epsilon' \in [0, 1/2]$. Notice that, having fixed $\overline{H}$, the possible values of h are $\{3, \dots, 2 + \overline{H}\}$. States $s'_1, \dots, s'_{\overline{S}}$ are absorbing states. The expert's policy always plays action a_0.

Let us introduce the set $\mathcal{I} := \{s_1, \dots, s_{\overline{S}}\} \times \{a_1, \dots, a_{\overline{A}}\} \times [\![3, \overline{H} + 2]\!]$. Let $\boldsymbol{v} = (v^\imath)_{\imath \in \mathcal{I}} \in \mathcal{V}^{\mathcal{I}}$ which is the set of vectors having as components the elements $v^\imath$ determining the probability distribution of the next state starting from the triple $\imath \in \mathcal{I}$. We denote with $\mathcal{M}_{\boldsymbol{v}}$ the MDP\R induced by $\boldsymbol{v}$. We can construct the class of instances denoted by $\mathbb{M} = \{\mathcal{M}_{\boldsymbol{v}} : \boldsymbol{v} \in \mathcal{V}^{\mathcal{I}}\}$. Moreover, we denoted with $\mathcal{M}_{\boldsymbol{v}^{\imath \leftarrow w}}$ the instance in which we replace the $\imath$ component of $\boldsymbol{v}$, i.e., $v^\imath$, with $w \in \mathcal{V}$ and $\mathcal{M}_{\boldsymbol{v}^{\imath \leftarrow 0}}$ the instance in which we replace the $\imath$ component of $\boldsymbol{v}$, i.e., $v^\imath$, with the zero vector.

Step 2: Feasible Set Computation Thanks to Lemma 9, we know that there exists a subset $\overline{\mathcal{V}} \subset \mathcal{V}$ of cardinality at least $|\overline{\mathcal{V}}| \geqslant 2^{\overline{S}/5}$ such that for every $v, w \in \overline{\mathcal{V}}$ with $v \neq w$ we have $\sum_{j=1}^{\overline{S}} |v_j - w_j| \geqslant \overline{S}/16$. Thus, we consider the set $\overline{\mathcal{V}}^{\mathcal{I}} \subset \mathcal{V}^{\mathcal{I}}$ and to build the instances $\boldsymbol{v} \in \overline{\mathcal{V}}^{\mathcal{I}}$ and $v, w \in \overline{\mathcal{V}}$ with $v \neq w$. Let $\imath \in \mathcal{I}$, the induced instances are denoted

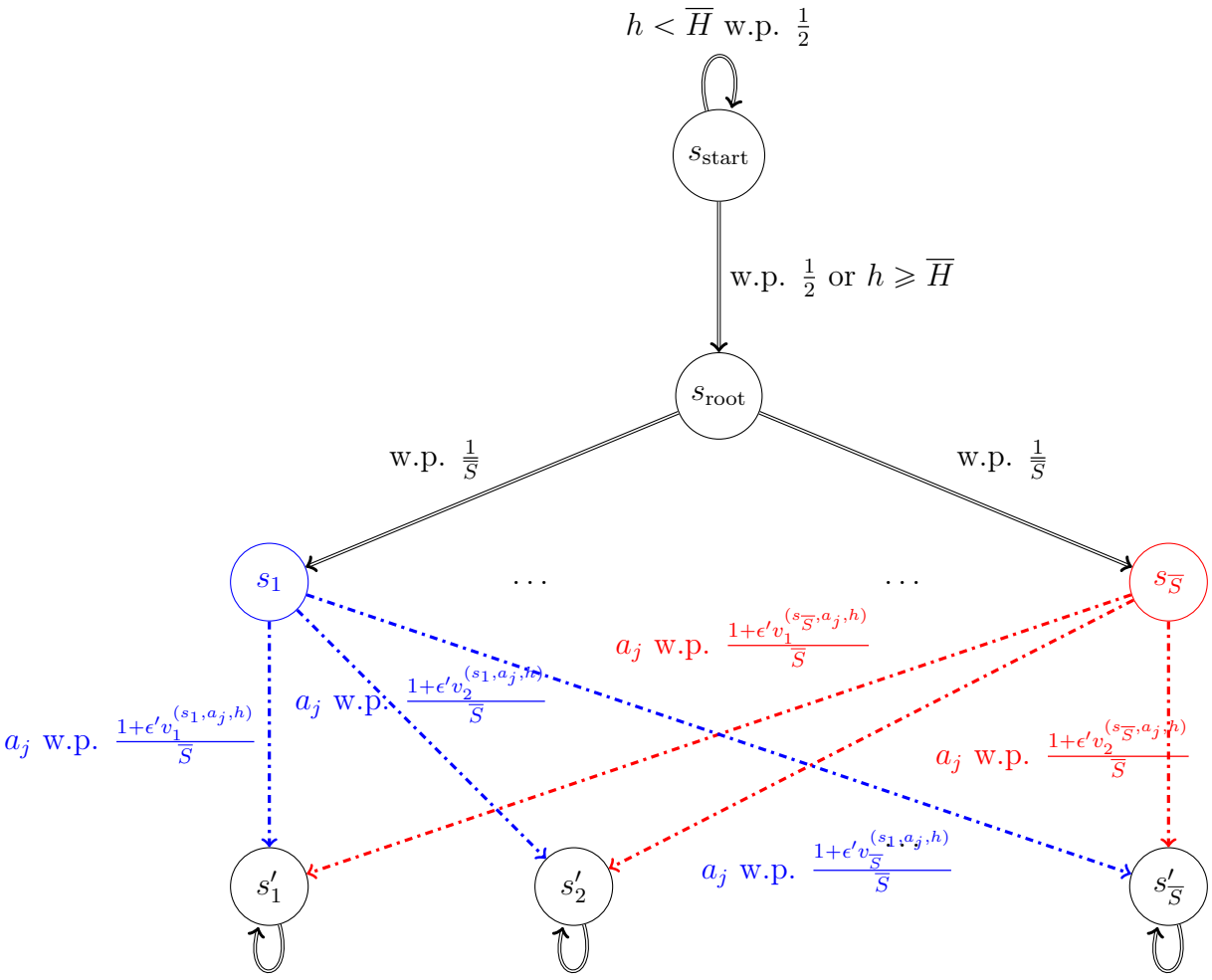


Figure 6.3: Semi-formal representation of the the hard instances MDP\R used in the proof of Theorem 16.

by $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}, \mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w} \in \mathbb{M}$.

To lower bound the Hausdorff distance, we focus on the triple $\imath = (s_*, a_*, h_*)$ and we enforce the convenience of action a_0 over action a_*. For both MDP\R $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}$ and $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}$, let $r^v \in \mathcal{R}_{\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}}$ and $r^w \in \mathcal{R}_{\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}}$, we have:

$$r^v_{h_*}(s_*, a_0) + \frac{1}{\overline{S}} \sum_{l=h_*+1}^{H} \sum_{j=1}^{\overline{S}} r^v_l(s'_j) \geqslant r^v_{h_*}(s_*, a_*) + \sum_{l=h_*+1}^{H} \sum_{j=1}^{\overline{S}} \frac{1+\epsilon' v_j}{\overline{S}} r^v_l(s'_j)$$

$$\implies r^v_{h_*}(s_*, a_0) \geqslant r^v_{h_*}(s_*, a_*) + \frac{\epsilon'}{\overline{S}} \sum_{j=1}^{\overline{S}} v_j \sum_{l=h_*+1}^{H} r^v_l(s'_j).$$

$$r^w_{h_*}(s_*, a_0) + \frac{1}{\overline{S}} \sum_{l=h_*+1}^{H} \sum_{j=1}^{\overline{S}} r^w_l(s'_j) \geqslant r^w_{h_*}(s_*, a_*) + \sum_{l=h_*+1}^{H} \sum_{j=1}^{\overline{S}} \frac{1 + \epsilon' w'_j}{\overline{S}} r^w_l(s'_j)$$

$$\implies r^w_{h_*}(s_*, a_0) \geqslant r^w_{h_*}(s_*, a_*) + \frac{\epsilon'}{\overline{S}} \sum_{j=1}^{\overline{S}} w_j \sum_{l=h_*+1}^{H} r^w_l(s'_j). \tag{6.2}$$

In order to lower bound the Hausdorff distance $\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}, \mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}\right)$, we set for $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}$:

$$r^v_l(s'_j) = -v_j,\ r^v_{h_*}(s_*, a_*) = 1,\ r^v_{h_*}(s_*, a_0) = 1 - \epsilon'(H - h_*).$$

We now want to find the closest reward function r^w for the instance $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}$, recalling that there are at least $\overline{S}/16$ components of the vectors v and w that are different. Clearly, we can set $r^w_l(s'_j) = r^v_l(s'_j) = -v_j$ for all $j \in [\![\overline{S}]\!]$ in which $v_j = w_j$ since this will not increase the Hausdorff distance and make the constraint in Equation (6.2) less restrictive. For symmetry reasons, we can limit our reasoning to the case in which $v_j = -1$ and $w_j = 1$ for the j terms in which they are different. This way, the constraint becomes:

$$\underbrace{r^w_{h_*}(s_*, a_0)}_{=:x} \geqslant \underbrace{r^w_{h_*}(s_*, a_*)}_{=:y} - \frac{N_{v,w}}{\overline{S}} \epsilon'(H - h^*) + \left(1 - \frac{N_{v,w}}{\overline{S}}\right) \epsilon'(H - h^*) \underbrace{\frac{1}{(H - h^*)\left(1 - \frac{N_{v,w}}{\overline{S}}\right)} \sum_{j:v_j \neq w_j}^{\overline{S}} \sum_{l=h_*+1}^{H} r^w_l(s'_j)}_{=:z},$$

where $N_{v,w} = \sum_{j=1}^{\overline{S}} \mathbb{1}\{v_j = w_j\}$. Notice that $z \in [-1, 1]$. Let $\alpha = \frac{N_{v,w}}{\overline{S}}$, the Hausdorff distance can be lower bounded by:

$$\begin{aligned}
\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}v}, \mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}\right) &= \min_{\substack{x,y,z \in [-1,1] \\ y \geqslant x - \alpha\epsilon'(H-h^*) + (1-\alpha)\epsilon'(H-h^*)z}} \max\left\{|x - 1|, |y - (1 - \epsilon'(H - h^*))|, |z + 1|\right\} \\
&\geqslant \min_{\substack{x,y \in [-1,1] \\ y \geqslant x - \alpha\epsilon'(H-h^*)}} \max\left\{|x - 1|, |y - (1 - \epsilon'(H - h^*))|\right\} \\
&= \frac{1}{2}(1 - \alpha)\epsilon'(H - h^*) \geqslant \frac{\epsilon'}{32}(H - h^*),
\end{aligned}$$

where the first inequality derives from the fact that to have a Hausdorff distance smaller than 1, we must take $z < 0$ at least and the second inequality is obtained by recalling that $1 - \alpha \geqslant \frac{1}{16}$ for the packing argument.

We enforce the following constraint on this quantity:

$$\forall h^* \in [\![3, \overline{H}+2]\!] \ : \ \frac{\epsilon'}{32}(H-h^*) \geqslant 2\epsilon \implies \epsilon' \geqslant \max_{h^* \in [\![3,\overline{H}+2]\!]} \frac{\epsilon}{64(H-h^*)} = \frac{64\epsilon}{(H-\overline{H}-2)}. \tag{6.3}$$

Notice that $\epsilon' \leqslant 1/2$ whenever $H \geqslant \overline{H} + 130$.

Step 3: Lower bounding Probability Let us consider an (ϵ, δ)-correct algorithm $\mathfrak{A}$ that outputs the estimated feasible set $\widehat{\mathcal{R}}$. Thus, consider $\imath \in \mathcal{I}$ and $\boldsymbol{v} \in \overline{\mathcal{V}}^{\mathcal{I}}$, we can lower bound the error probability:

$$\begin{aligned}
\delta &\geqslant \sup_{\text{all } \mathcal{M} \text{ MDP}\backslash\text{R and expert policies } \pi} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}} \left(\mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}}, \widehat{\mathcal{R}} \right) \geqslant \epsilon \right) \\
&\geqslant \sup_{\mathcal{M} \in \mathbb{M}} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}} \left(\mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}}, \widehat{\mathcal{R}} \right) \geqslant \epsilon \right) \\
&\geqslant \max_{w \in \overline{\mathcal{V}}} \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w},\pi),\mathfrak{A}} \left(\mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}, \widehat{\mathcal{R}} \right) \geqslant \epsilon \right).
\end{aligned}$$

For every $\imath \in \mathcal{I}$ and $\boldsymbol{v} \in \overline{\mathcal{V}}^{\mathcal{I}}$, let us define the *identification function*:

$$\Psi_{\imath,\boldsymbol{v}} := \operatorname*{argmin}_{w \in \overline{\mathcal{V}}} \mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}, \widehat{\mathcal{R}} \right).$$

Let $w \in \overline{\mathcal{V}}$. If $\Psi_{\imath,\boldsymbol{v}} = w$, then, $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} \Psi_{\imath,\boldsymbol{v}}}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}) = 0$. Otherwise, if $\Psi_{\imath,\boldsymbol{v}} \neq w$, we have:

$$\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} \Psi_{\imath,\boldsymbol{v}}}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}) \leqslant \mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} \Psi_{\imath,\boldsymbol{v}}}}, \widehat{\mathcal{R}}) + \mathcal{H}_{d^{\mathrm{G}}}(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}) \leqslant 2\mathcal{H}_{d^{\mathrm{G}}}(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}),$$

where the first inequality follows from triangular inequality and the second one from the definition of identification function $\Psi_{\imath,\boldsymbol{v}}$. From Equation (6.3), we have that $\mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}_i^{\Psi_i}}, \mathcal{R}_{\mathcal{M}_i^v} \right) \geqslant 2\epsilon$. Thus, it follows that $\mathcal{H}_{d^{\mathrm{G}}}(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}) \geqslant \epsilon$. This implies the following inclusion of events for $w \in \overline{\mathcal{V}}$:

$$\left\{ \mathcal{H}_{d^{\mathrm{G}}}(\widehat{\mathcal{R}}, \mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}) \geqslant \epsilon \right\} \supseteq \{ \Psi_{\imath,\boldsymbol{v}} \neq w \}.$$

Thus, we can proceed by lower bounding the probability:

$$\begin{aligned}
\max_{w \in \overline{\mathcal{V}}} \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w},\pi),\mathfrak{A}} \left(\mathcal{H}_{d^{\mathrm{G}}} \left(\mathcal{R}_{\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w}}, \widehat{\mathcal{R}} \right) \geqslant \epsilon \right) &\geqslant \max_{w \in \overline{\mathcal{V}}} \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w},\pi),\mathfrak{A}} \left(\Psi_{\imath,\boldsymbol{v}} \neq w \right) \\
&\geqslant \frac{1}{|\overline{\mathcal{V}}|} \sum_{w \in \overline{\mathcal{V}}} \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v} \overset{\imath}{\leftarrow} w},\pi),\mathfrak{A}} \left(\Psi_{\imath,\boldsymbol{v}} \neq w \right),
\end{aligned}$$

where the second inequality follows from bounding the maximum of probability with the average. We can now apply the Fano's inequality (Theorem 21) with reference probability $\mathbb{P}_0 = \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}0},\pi),\mathfrak{A}}$, $\mathbb{P}_w = \mathbb{P}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}$, and $\mathcal{A}_w = \{\Psi_{\imath,\boldsymbol{v}} \neq w\}$:

$$\frac{1}{|\overline{\mathcal{V}}|}\sum_{w\in\overline{\mathcal{V}}} \mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}(\Psi_{\imath,\boldsymbol{v}} \neq w) \geqslant 1 - \frac{1}{\log|\overline{\mathcal{V}}|}\left(\frac{1}{|\overline{\mathcal{V}}|}\sum_{w\in\overline{\mathcal{V}}} D_{\mathrm{KL}}\left(\mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}, \mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}0},\pi),\mathfrak{A}}\right) - \log 2\right). \tag{6.4}$$

Step 4: KL-divergence Computation Let $\mathcal{M}$ be an instance, we denote with $\mathbb{P}_{\mathfrak{A},\mathcal{M},\pi}$ the joint probability distribution of all events realized by the execution of the algorithm in the MDP\R (the presence of π is irrelevant as we assume it known):

$$\mathop{\mathbb{P}}_{(\mathcal{M},\pi),\mathfrak{A}} = \prod_{t=1}^{\tau} \rho_t(s_t, a_t, h_t|H_{t-1})p_{h_t}(s'_t|s_t,a_t).$$

where $H_{t-1} = (s_1, a_1, h_1, s'_1, \ldots, s_{t-1}, a_{t-1}, h_{t-1}, s'_{t-1})$ is the history up to time $t-1$. Let $\imath \in \mathcal{I}$ and $v \in \overline{\mathcal{V}}$ and denote with $p^{v\overset{\imath}{\leftarrow}0}$ and $p^{v\overset{\imath}{\leftarrow}w}$ the transition models associated with $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}0}$ and $\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w}$. Let us now move to the KL-divergence and denoting $\imath = (s_*, a_*, h_*)$: Thus, we have:

$$\begin{aligned}
D_{\mathrm{KL}}\left(\mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}, \mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}0},\pi),\mathfrak{A}}\right) &= \mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}\left[\sum_{t=1}^{\tau} D_{\mathrm{KL}}\left(p_{h_t}^{\boldsymbol{v}\overset{\imath}{\leftarrow}w}(\cdot|s_t,a_t), p_{h_t}^{\boldsymbol{v}\overset{\imath}{\leftarrow}0}(\cdot|s_t,a_t)\right)\right] \\
&\leqslant \mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}\left[N_{h_*}^{\tau}(s_*,a_*)\right] D_{\mathrm{KL}}\left(p_{h_*}^{\boldsymbol{v}\overset{\imath}{\leftarrow}w}(\cdot|s_*,a_*), p_{h_*}^{\boldsymbol{v}\overset{\imath}{\leftarrow}0}(\cdot|s_*,a_*)\right) \\
&\leqslant 2(\epsilon')^2 \mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}\left[N_{h_*}^{\tau}(s_*,a_*)\right],
\end{aligned}$$

having observed that the transition models differ in $\imath = (s_*, a_*, h_*)$ and defined $N_{h_*}^{\tau}(s_*,a_*) = \sum_{t=1}^{\tau} \mathbb{1}\{(s_t,a_t,h_t) = (s_*,a_*,h_*)\}$ and the last passage is obtained by Lemma 7 with $D = \overline{S}$. Plugging into Equation (6.4), we obtain:

$$\begin{aligned}
&\delta \geqslant \frac{1}{|\overline{\mathcal{V}}|}\sum_{w\in\overline{\mathcal{V}}} \mathop{\mathbb{P}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}(\Psi_{\imath,\boldsymbol{v}} \neq w) \implies \\
&\implies \frac{1}{|\overline{\mathcal{V}}|}\sum_{w\in\overline{\mathcal{V}}} \mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}\overset{\imath}{\leftarrow}w},\pi),\mathfrak{A}}\left[N_{h_*}^{\tau}(s_*,a_*)\right] \geqslant \frac{(1-\delta)\log|\overline{\mathcal{V}}| - \log 2}{2(\epsilon')^2}.
\end{aligned}$$

Since the derivation is carried out for every $\imath \in \mathcal{I}$ and $\boldsymbol{v} \in \overline{\mathcal{V}}^{\mathcal{I}}$, we can perform the

summation over $\imath$ and the average over $\boldsymbol{v}$:

$$\sum_{\imath\in\mathcal{I}}\frac{1}{|\overline{\mathcal{V}}|^{|\mathcal{I}|}}\sum_{\boldsymbol{v}\in\overline{\mathcal{V}}^{\mathcal{I}}}\frac{1}{|\overline{\mathcal{V}}|}\sum_{w\in\overline{\mathcal{V}}}\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\imath}\leftarrow w},\pi),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right]=\frac{1}{|\overline{\mathcal{V}}|^{|\mathcal{I}|}}\sum_{\boldsymbol{v}\in\overline{\mathcal{V}}^{\mathcal{I}}}\sum_{\imath\in\mathcal{I}}\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}},\pi),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right]$$
$$\geqslant\overline{SAH}\frac{(1-\delta)\log|\overline{\mathcal{V}}|-\log 2}{2(\epsilon')^2}.$$

Notice that we get a guarantee on a mean under the uniform distribution of the instances of the sample complexity. Thus, there must exist one $\boldsymbol{v}^{\text{hard}}\in\overline{\mathcal{V}}$ such that:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\text{hard}},\pi}),\mathfrak{A}}[\tau]\geqslant\sum_{\imath\in\mathcal{I}}\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\text{hard}},\pi}),\mathfrak{A}}\left[N^{\tau}_{h_*}(s_*,a_*)\right]\geqslant\overline{SAH}\frac{(1-\delta)\log|\overline{\mathcal{V}}|-\log 2}{2(\epsilon')^2}.$$

Then, we select $\delta\leqslant 1/2$, recall that $|\overline{\mathcal{V}}|\geqslant 2^{\overline{S}/5}$, we get:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\text{hard}},\pi}),\mathfrak{A}}[\tau]\geqslant\overline{SAH}\frac{\overline{S}/10-\log 2}{2(\epsilon')^2}=\overline{SAH}\frac{(H-\overline{H}-2)^2(\overline{S}/10-\log 2)}{8912\epsilon^2}$$

The number of states is given by $S=|\mathcal{S}|=2\overline{S}+2$, the number of actions is given by $A=|\mathcal{A}|=\overline{A}+1$. Let us first consider the time-homogeneous case, i.e., $\overline{H}=1$, for $S\geqslant 16$, $A\geqslant 2$, $H\geqslant 130$, we have:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\text{hard}},\pi}),\mathfrak{A}}[\tau]\geqslant\frac{S^2AH^2}{2560\epsilon^2}.$$

For the time inhomogeneous case, we select $\overline{H}=H/2$, to get, under the same conditions:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_{\boldsymbol{v}^{\text{hard}},\pi}),\mathfrak{A}}[\tau]\geqslant\frac{S^2AH^3}{5120\epsilon^2}.$$

□

7 | Algorithm

In this chapter, we analyze the sample complexity of a uniform sampling strategy (Uniform Sampling-IRL, `US-IRL`) for the d^{G}-IRL problem (Algorithm 10). We start presenting the sample complexity analysis (Section 7.1) and, then, we provide a sketch of the proof (Section 7.2).

7.1. Main Result

The `US-IRL` algorithm was presented in [36, 39] but analyzed for different IRL formulations. We revise it since it matches our sample complexity lower bounds, provided that more sophisticated concentration tools w.r.t. those employed in [36, 39]. For the sake of presentation, we assume that the expert's policy π^E is known; the extension to unknown π^E is reported in Appendix A. Let $D=\{(s_l,a_l,h_l,s'_l,a^E_l)\}_{l\in[\![t]\!]}$ be a dataset of $t\in\mathbb{N}$ tuples, where for every $l\in[\![t]\!]$, we have $s'_l\sim p_{h_l}(\cdot|s_l,a_l)$ and $a^E_l\sim\pi^E_{h_l}(\cdot|s_l)$. We introduce the counts for every $(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]$: $n^t_h(s,a,s'):=\sum_{l=1}^t\mathbb{1}\{(s_l,a_l,h_l,s'_l)=(s,a,h,s')\}$, $n^t_h(s,a):=\sum_{s'\in\mathcal{S}}n^t_h(s,a,s')$, $n^t_h(s):=\sum_{a\in\mathcal{A}}n^t_h(s,a)$, and $n^{t,E}_h(s,a):=\sum_{l=1}^t\mathbb{1}\{(s_l,a^E_l)=(s,a)\}$. These quantities allow defining the *empirical transition model* $\widehat{p}^t=(\widehat{p}^t_h)_{h\in[\![H]\!]}$ and *empirical expert's policy* $\widehat{\pi}^{t,E}=(\pi^{t,E}_h)_{h\in[\![H]\!]}$ as follows:

$$\begin{aligned}\widehat{p}^t_h(s'|s,a)&:=\begin{cases}\frac{n^t_h(s,a,s')}{n^t_h(s,a)} & \text{if } n^t_h(s,a)>0\\ \frac{1}{S} & \text{otherwise}\end{cases},\\ \widehat{\pi}^{E,t}_h(a|s)&:=\begin{cases}\frac{n^{E,t}_h(s,a)}{n^t_h(s)} & \text{if } n^t_h(s)>0\\ \frac{1}{A} & \text{otherwise}\end{cases}.\end{aligned}\tag{7.1}$$

In the time-homogeneous case, we simply merge the samples collected at different stages $h\in[\![H]\!]$. We denote with $(\widehat{\mathcal{M}}^t,\widehat{\pi}^{E,t})$ the *empirical IRL* problem, where $\widehat{\mathcal{M}}^t=(\mathcal{S},\mathcal{A},\widehat{p}^t,H)$ the empirical MDP\R induced by $\widehat{p}^t$. Finally, we denote with $\widehat{\mathcal{R}}^t:=\mathcal{R}_{(\widehat{\mathcal{M}}^t,\widehat{\pi}^{E,t})}$ the feasible reward set induced $(\widehat{\mathcal{M}}^t,\widehat{\pi}^{E,t})$. We will omit the superscript t, whenever clear from the context and write $\widehat{\mathcal{R}}$. Because of the assumption that expert's policy π^E is known, the algorithm as presented in this chapter uses only the first set of equations. At each

iteration, the algorithm collects a sample from every $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$ and, for time-inhomogeneous models, computes the confidence function:

$$\mathcal{C}_h^t(s,a) := 2\sqrt{2}(H-h+1)\sqrt{\frac{2\beta\big(n_h^t(s,a),\delta\big)}{n_h^t(s,a)}}, \tag{7.2}$$

where $\beta\big(n,\delta\big) := \log(SAH/\delta) + (S-1)\log\big(e(1+n/(S-1))\big)$.[1] The algorithm stops as soon as all confidence functions fall below the threshold ϵ. The following theorem provides the sample complexity of `US-IRL`.

Theorem 17 (Sample Complexity of `US-IRL`). *Let $\epsilon > 0$ and $\delta \in (0,1)$, `US-IRL` is (ϵ,δ)-PAC for d^G-IRL and with probability at least $1-\delta$ it stops after τ samples with:*

- *if the transition model p is time-inhomogeneous:*

$$\tau \leqslant \frac{8H^3SA}{\epsilon^2}\left(\log\left(\frac{SAH}{\delta}\right) + (S-1)C\right),$$

  where $C = \log(e/(S-1) + (8eH^2)/((S-1)\epsilon^2)(\log(SAH/\delta) + 4e))$;
- *if the transition model p is time-homogeneous and :*

$$\tau \leqslant \frac{8H^2SA}{\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right) + (S-1)C_2\right),$$

  where $\widetilde{C} = \log(e/(S-1) + (8eH^2)/((S-1)\epsilon^2)(\log(SA/\delta) + 4e))$.

Thus, time-inhomogeneous (resp. time-homogeneous) transition models, `US-IRL` suffers a sample complexity bound of order $\widetilde{O}\left(\frac{H^3SA}{\epsilon^2}\left(\log\left(\frac{1}{\delta}\right)+S\right)\right)$ (resp. $\widetilde{O}\left(\frac{H^2SA}{\epsilon^2}\left(\log\left(\frac{1}{\delta}\right)+S\right)\right)$) matching the lower bounds of Theorem 14 up to logarithmic factors for both regimes of δ.

[1] In the time-homogeneous case, the algorithm merges the samples collected at different $h \in [\![H]\!]$ for the estimation of the transition model and replaces the confidence function with:

$$\widetilde{\mathcal{C}}_h^t(s,a) := 2\sqrt{2}(H-h+1)\sqrt{\frac{2\widetilde{\beta}\big(n^t(s,a),\delta\big)}{n^t(s,a)}}, \tag{7.3}$$

where $\widetilde{\beta}\big(n,\delta\big) := \log(SA/\delta) + (S-1)\log\big(e(1+n/(S-1))\big)$ and $n^t(s,a) = \sum_{h=1}^H n_h^t(s,a)$.

Input: significance $\delta \in (0,1)$, ϵ target accuracy
$t \leftarrow 0$, $\epsilon_0 \leftarrow +\infty$
while $\epsilon_t > \epsilon$ **do**
  $t \leftarrow t + SAH$
  Collect one sample from each $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$
  Update $\widehat{p}^t$ according with (7.1)
  Update $\epsilon_t = \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \mathcal{C}_h^t(s,a)$ (resp. $\widetilde{\mathcal{C}}_h^t(s,a)$)
end while

Algorithm 10: UniformSampling-IRL (`US-IRL`) for time-inhomogeneous (resp. time-homogeneous) transition models.

7.2. Proof

Proof. We start with the case in which the transition model is time-inhomogeneous. In this case, we introduce the following good event:

$$\mathcal{E} := \left\{ \forall t \in \mathbb{N},\ \forall (s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!] :\ D_{\mathrm{KL}}\Big(\widehat{p}_h^t(\cdot|s,a), p_h(\cdot|s,a)\Big) \leqslant \frac{\beta\big(n_h^t(s,a),\delta\big)}{n_h^t(s,a)} \right\},$$

where p_h is the true transition model and $\widehat{p}_h^t$ is its estimate via Equation (7.1) at time t. Thanks to Lemma 4, we have that $\mathbb{P}_{(\mathcal{M},\pi^E),\mathfrak{A}}(\mathcal{E}) \geqslant 1-\delta$. Thus, under the good event $\mathcal{E}$, we apply Theorem 12:

$$\begin{aligned}
\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau) &\leqslant \frac{2\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}}^t,\widehat{\pi}^{E,t}))}{1+\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}}^t,\widehat{\pi}^{E,t}))} \\
&\leqslant 2\rho^{\mathrm{G}}((\mathcal{M},\pi^E),(\widehat{\mathcal{M}},\widehat{\pi}^E)) \\
&\leqslant 2 \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} (H-h+1)\Bigg(\Big|\mathbb{1}_{\{\pi_h^E(a|s)=0\}} - \mathbb{1}_{\{\widehat{\pi}_h^E(a|s)=0\}}\Big| + \\
&\quad + \|p_h(\cdot|s,a) - \widehat{p}_h(\cdot|s,a)\|_1\Bigg) \\
&\leqslant 2 \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} (H-h+1)\,\|p_h(\cdot|s,a) - \widehat{p}_h(\cdot|s,a)\|_1 \\
&\leqslant 2\sqrt{2} \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} (H-h+1)\sqrt{D_{\mathrm{KL}}\Big(\widehat{p}_h^t(\cdot|s,a), p_h(\cdot|s,a)\Big)} \\
&= \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \mathcal{C}_h^t(s,a),
\end{aligned}$$

where we exploited the fact that the expert's policy is known in the last but one passage and used Pinsker's inequality in the last passage. When the `US-IRL` stops we have that

$\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \mathcal{C}_h^t(s,a) \leqslant \epsilon$ and, consequently, for all $(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]$ we have:

$$\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \mathcal{C}_h^t(s,a) = \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} 2\sqrt{2}(H-h+1)\sqrt{\frac{\beta\big(n_h^t(s,a),\delta\big)}{n_h^t(s,a)}} \leqslant \epsilon.$$

Thus, the algorithm stops at the smallest t such that:

$$\begin{aligned}\implies n_h^t(s,a) &\geqslant \frac{8(H-h+1)^2\beta\big(n_h^t(s,a),\delta\big)}{\epsilon^2} = \\ &= \frac{8(H-h+1)^2}{\epsilon^2}\left(\log(SAH/\delta) + (S-1)\log(e(1+n_h^t(s,a)/(S-1)))\right).\end{aligned}$$

Thus, by applying Lemma 15 of [27], we obtain:

$$\begin{aligned}n_h^\tau(s,a) \leqslant \frac{8(H-h+1)^2}{\epsilon^2}\Bigg(&\log\left(\frac{SAH}{\delta}\right) + \\ &+ (S-1)\log\left(\frac{8e(H-h+1)^2}{(S-1)\epsilon^2}\left(\log\left(\frac{SAH}{\delta}\right)+4e\right)\right)\Bigg).\end{aligned}$$

By recalling that $\tau = SAHn_h^\tau(s,a)$, and bounding $H-h+1 \leqslant H$, we obtain:

$$\tau \leqslant \frac{8H^3SA}{\epsilon^2}\left(\log\left(\frac{SAH}{\delta}\right) + (S-1)\log\left(\frac{e}{S-1} + \frac{8eH^2}{(S-1)\epsilon^2}\left(\log\left(\frac{SAH}{\delta}\right)+4e\right)\right)\right).$$

If the transition model is time-homogeneous, we suppress the subscript h and the algorithm `US-IRL`, will merge together all the samples collected at different stages h. Let us define $n^t(s,a) = \sum_{h=1}^H n_h^t(s,a)$ and $n^t(s,a,s') = \sum_{h=1}^H n_h^t(s,a,s')$. Now the transition model will be estimated straightforwardly as follows:

$$\widehat{p}^t(s'|s,a) := \begin{cases} \frac{n^t(s,a,s')}{n^t(s,a)} & \text{if } n^t(s,a) > 0 \\ \frac{1}{S} & \text{otherwise} \end{cases}.$$

Let us consider now the following good event:

$$\widetilde{\mathcal{E}} := \left\{\forall t\in\mathbb{N},\ \forall(s,a)\in\mathcal{S}\times\mathcal{A}:\ D_{\mathrm{KL}}\Big(\widehat{p}^t(\cdot|s,a), p(\cdot|s,a)\Big) \leqslant \frac{\widetilde{\beta}\big(n^t(s,a),\delta\big)}{n^t(s,a)}\right\}.$$

Thanks to Lemma 4, we have that $\mathbb{P}_{(\mathcal{M},\pi^E),\mathfrak{A}}(\widetilde{\mathcal{E}}) \geqslant 1-\delta$. Thus, in such a case, thanks to

Theorem 12, we have:

$$\begin{aligned}\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R},\widehat{\mathcal{R}}^{\tau}) &\leqslant 2\sqrt{2}\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]}(H-h+1)\sqrt{D_{\mathrm{KL}}\Big(\widehat{p}_h^t(\cdot|s,a),p_h(\cdot|s,a)\Big)}\\ &= \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]}\widetilde{\mathcal{C}}_h^t(s,a).\end{aligned}$$

The algorithm, therefore, stops as soon as:

$$\begin{aligned}\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]}\widetilde{\mathcal{C}}_h^t(s,a) &= \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]}2\sqrt{2}(H-h+1)\sqrt{\frac{\widetilde{\beta}\big(n^t(s,a),\delta\big)}{n^t(s,a)}}\\ &= \max_{(s,a)\in\mathcal{S}\times\mathcal{A}}2\sqrt{2}H\sqrt{\frac{\widetilde{\beta}\big(n^t(s,a),\delta\big)}{n^t(s,a)}}\leqslant\epsilon.\end{aligned}$$

This allows us to compute the maximum value of $n^{\tau}(s,a)$:

$$n^{\tau}(s,a)\leqslant\frac{8H^2}{\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right)+(S-1)\log\left(\frac{e}{S-1}+\frac{8eH^2}{(S-1)\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right)+4e\right)\right)\right).$$

Recalling that $\tau=SAn^{\tau}(s,a)$, we obtain:

$$\tau\leqslant\frac{8H^2SA}{\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right)+(S-1)\log\left(\frac{8eH^2}{(S-1)\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right)+4e\right)\right)\right).$$

□

Lemma 4. *The following statements hold:*

- *for* $\beta\big(n,\delta\big)=\log(SAH/\delta)+(S-1)\log\big(e(1+n/(S-1)\big)$*, we have that* $\mathbb{P}(\mathcal{E})\geqslant 1-\delta$*;*
- *for* $\widetilde{\beta}\big(n,\delta\big)=\log(SA/\delta)+(S-1)\log\big(e(1+n/(S-1)\big)$*, we have that* $\mathbb{P}(\widetilde{\mathcal{E}})\geqslant 1-\delta$*.*

Proof. Let us start with the first statement. Similarly to Lemma 10 of [27], we apply first a union bound and then technical Proposition 1 of [25] (also reported as Lemma 6 for completeness) to concentrate the KL-divergence:

$$\begin{aligned}\mathbb{P}(\mathcal{E}^c) =&\mathbb{P}\left(\exists t\in\mathbb{N},\,\exists(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]:\,D_{\mathrm{KL}}\Big(\widehat{p}_h^t(\cdot|s,a),p_h(\cdot|s,a)\Big)\geqslant\frac{\beta\big(n_h^t(s,a),\delta\big)}{n_h^t(s,a)}\right)\\ &\leqslant\sum_{h\in[\![H]\!]}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\mathbb{P}\left(\exists t\in\mathbb{N}:\,D_{\mathrm{KL}}\Big(\widehat{p}_h^t(\cdot|s,a),p_h(\cdot|s,a)\Big)\geqslant\frac{\beta\big(n_h^t(s,a),\delta\big)}{n_h^t(s,a)}\right)\\ &\leqslant\sum_{h\in[\![H]\!]}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\frac{\delta}{SAH}=\delta.\end{aligned}$$

The proof of the second statement is analogous having simply observed that the union bound has to be performed over $\mathcal{S} \times \mathcal{A}$ only. □

8 | Conclusions

In this chapter, we sum up the main contributions of this thesis and then we devise possible directions for future works in this research topic.

8.1. Contributions of the Present Work

In this work, we provided contributions to the understanding of the complexity of the IRL problem. We conceived a lower bound of order $\Omega\left(\frac{H^3SA}{\epsilon^2}\left(\log\left(\frac{1}{\delta}\right)+S\right)\right)$ on the number of samples collected with a generative model in the finite-horizon setting. This result is of relevant interest since it sets, for the first time, the complexity of the IRL problem, defined as the problem of estimating the feasible reward set. Furthermore, we showed that a uniform sampling strategy matches the lower bound up to logarithmic factors. Nevertheless, the IRL problem is far from being closed. In the following, we outline a road map of open questions, hoping to inspire researchers to work in this appealing area.

8.2. Open Questions

Forward Model The most straightforward extension of our findings is moving to the *forward model* setting, in which the agent can interact with the environment through trajectories only. As we already noted, our lower bounds can be comfortably extended to this setting. However, in this case, the PAC requirement has to be relaxed since controlling the L_∞-norm between rewards is no longer a viable option (e.g., for the possible presence of almost unreachable states). Which distance notion should be used for this setting? Will the Lipschitz regularity of Section 4 still hold?

Problem-Dependent Analysis Our analysis is *worst-case* in the class of IRL problems. Would it be possible to obtain a *problem-dependent* complexity results? Previous problem-dependent analyses provided results tightly connected to the properties of the specific reward selection procedure [36, 39]. Clearly, a currently open question, in all settings in which reward is missing, including reward-free exploration [23] and IRL, is how

to define a problem-dependent quantity in replacement of the suboptimality gaps.

Reward Selection Our PAC guarantees concern with the complete feasible reward set. However, algorithmic solutions to IRL implement a specific criterion for selecting a reward (e.g., maximum entropy, maximum margin). How the PAC guarantee based on the Hausdorff distance relates to guarantees on a single reward selected with a *specific criterion* within $\mathcal{R}$?

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[2] S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artif. Intell.*, 297:103500, 2021.

[3] M. Azar, R. Munos, M. Ghavamzadeh, and H. Kappen. Reinforcement learning with a near optimal rate of convergence. 10 2011.

[4] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. J. Kappen. Speedy q-learning. In *NIPS*, 2011.

[5] M. G. Azar, R. Munos, and B. Kappen. On the sample complexity of reinforcement learning with a generative model, 2012.

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

[7] R. Bellman. *Dynamic Programming*. Dover Publications, 1957.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, oct 1989.

[9] S. Boucheron, G. Lugosi, and P. Massart. Concentration inequalities - a nonasymptotic theory of independence. In *Concentration Inequalities*, 2013.

[10] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. *ICAPS*, 15:20–27, 01 2011.

[11] G. D. Cohen and P. Frankl. Good coverings of hamming spaces with spheres. *Discret. Math.*, 56(2-3):125–131, 1985.

[12] C. Dann and E. Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. *NIPS*, 10 2015.

[13] G. T. de Ghellinck and G. D. Eppen. Linear programming solutions for separable markovian decision problems. *Management Science*, 13(5):371–394, 1967.

[14] G. Dexter, K. Bello, and J. Honorio. Inverse reinforcement learning in a continuous state space with formal guarantees. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 6972–6982, 2021.

[15] O. D. Domingues, P. Ménard, E. Kaufmann, and M. Valko. Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited. In *Algorithmic Learning Theory (ALT)*, volume 132 of *Proceedings of Machine Learning Research*, pages 578–598. PMLR, 2021.

[16] E. Even-Dar, S. Mannor, and Y. Mansour. Pac bounds for multi-armed bandit and markov decision processes. pages 193–209, 07 2002.

[17] E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *J. Mach. Learn. Res.*, 7:1079–1105, 2006.

[18] S. Gerchinovitz, P. Ménard, and G. Stoltz. Fano's inequality for random variables. *CoRR*, abs/1702.05985, 2017.

[19] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer series in statistics. Springer, 2002.

[20] D. Haussler. Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artif. Intell.*, 36:177–221, 1988.

[21] D. Haussler. Probably approximately correct learning. 1990.

[22] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[23] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu. Reward-free exploration for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 4870–4879. PMLR, 2020.

[24] L. W. John Lafferty, Han Liu. Minimax theory.

[25] A. Jonsson, E. Kaufmann, P. Ménard, O. D. Domingues, E. Leurent, and M. Valko. Planning in markov decision processes with gap-dependent sample complexity. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[26] S. Kakade. On the sample complexity of reinforcement learning. 01 2003.

[27] E. Kaufmann, P. Ménard, O. D. Domingues, A. Jonsson, E. Leurent, and M. Valko. Adaptive reward-free exploration. In *Algorithmic Learning Theory (ALT)*, volume 132 of *Proceedings of Machine Learning Research*, pages 865–891. PMLR, 2021.

[28] S. M. Khansari Zadeh and A. Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *IEEE Trans. Robot*, 27, 01 2011.

[29] E. Klein, M. Geist, B. Piot, and O. Pietquin. Inverse reinforcement learning through structured classification. volume 2, 12 2012.

[30] A. Komanduru and J. Honorio. On the correctness and sample complexity of inverse reinforcement learning. pages 7110–7119, 2019.

[31] A. Komanduru and J. Honorio. A lower bound for the sample complexity of inverse reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 5676–5685. PMLR, 2021.

[32] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[33] T. Lattimore, M. Hutter, and P. Sunehag. The sample-complexity of general reinforcement learning. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 28–36, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[34] S. Levine, Z. Popovic, and V. Koltun. Feature construction for inverse reinforcement learning. 12 2010.

[35] S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. 12 2011.

[36] D. Lindner, A. Krause, and G. Ramponi. Active exploration for inverse reinforcement learning. *CoRR*, abs/2207.08645, 2022.

[37] S. Mannor, J. Tsitsiklis, K. Bennett, and N. Cesa-bianchi. The sample complexity of exploration in the multi-armed bandit problem. 07 2004.

[38] P. Ménard, O. D. Domingues, A. Jonsson, E. Kaufmann, E. Leurent, and M. Valko. Fast active learning for pure exploration in reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 7599–7608. PMLR, 2021.

[39] A. M. Metelli, G. Ramponi, A. Concetti, and M. Restelli. Provably efficient learning of transferable rewards. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7665–7676. PMLR, 18–24 Jul 2021.

[40] T. M. Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[41] G. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *CoRR*, abs/1206.5264, 2012.

[42] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 663–670. Morgan Kaufmann, 2000.

[43] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Found. Trends Robotics*, 7(1-2):1–179, 2018.

[44] M. Pirotta and M. Restelli. Inverse reinforcement learning through policy gradient minimization. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 1993–1999. AAAI Press, 2016.

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

[46] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. pages 2586–2591, 01 2007.

[47] G. Ramponi, A. Likmeta, A. M. Metelli, A. Tirinzoni, and M. Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pages 2359–2369. PMLR, 2020.

[48] R. T. Rockafellar and R. J. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1998.

[49] S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.

[50] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[51] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[52] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[53] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. pages 1449–1456, 2007.

[54] C. Szepesvári. *Algorithms for Reinforcement Learning*, volume 4. 01 2010.

[55] L. G. Valiant. A theory of the learnable. In *Symposium on the Theory of Computing*, 1984.

[56] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel. Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. pages 2074–2081, 05 2010.

[57] M. C. Vroman. *Maximum likelihood inverse reinforcement learning*. Rutgers The State University of New Jersey-New Brunswick, 2014.

[58] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[59] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992.

[60] S. Zeng, C. Li, A. Garcia, and M. Hong. Maximum-likelihood inverse reinforcement learning with finite-time guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[61] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI)*, pages 1433–1438. AAAI Press, 2008.

A | Unknown Expert's Policy π^E

In this appendix, we extend the lower bounds and the algorithm for the case in which the expert's policy is unknown. Clearly, if the expert's policy is deterministic, under the generative model setting, its estimation is trivial as it suffices to query every state and stage (resp. state) exactly once for time-inhomogeneous (resp. time-homogeneous) policies, leading to $\mathbb{E}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] = HS$ (resp. $\mathbb{E}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] = S$). Thus, we consider a more general setting in which the expert's policy can be stochastic (still being optimal). Specifically, we consider the following assumption.

Assumption 1. *There exists a known constant $\pi_{\min} \in (0, 1]$ such that every action played by the expert's policy π^E is played with at least probability $\pi_{\min}$:*

$$\forall (s, a, h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!] \, : \, \pi_h^E(a|s) \in \{0\} \cup [\pi_{\min}, 1].$$

Intuitively, Assumption 1 formalizes a form of identifiability for the policy. As already mentioned in Section 4, what matters for learning the feasible reward set is whether an action is played by the agent (not the corresponding probability). Assumption 1 enforces that every optimal action must be played with a minimum (known) non-null probability $\pi_{\min}$. We shall show that if this assumption is violated, the problem becomes non-learnable.

A.1. Lower Bound

The following result provides a lower bound for learning the feasible reward set according to the PAC requirement of Definition (41) when the expert's policy is unknown, but the transition model is known. Clearly, one can combine this result with the ones of Section 6 to address the setting in which both the expert's policy and the transition model are unknown.

Theorem 18. *Let $\mathfrak{A} = (\mu, \tau)$ be an (ϵ, δ)-PAC algorithm for d^G-IRL. Then, there exists an IRL problem $(\mathcal{M}, \pi^E)$ where π^E fulfills Assumption 1 such that, if $\epsilon \leqslant 1/2$, $\delta < 1/16$,*

$S \geqslant 7$, $A \geqslant 2$, and $H \geqslant 3$, the number of samples N is lower bounded in expectation by:

- *if the expert's policy π^E is time-inhomogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{SH}{8 \log \frac{1}{1-\pi_{\min}}} \log\left(\frac{1}{\delta}\right);$$

- *if the expert's policy π^E is time-homogeneous:*

$$\mathop{\mathbb{E}}_{(\mathcal{M},\pi^E),\mathfrak{A}}[\tau] \geqslant \frac{S}{4 \log \frac{1}{1-\pi_{\min}}} \log\left(\frac{1}{\delta}\right).$$

Before presenting the proof, let us comment the result. We observe that when Assumption 1 is violated, i.e., $\pi_{\min} \to 0$, the sample complexity lower bound degenerates to infinity, proving that the problem become non-learnable.

Proof. **Step 1: Instances Construction** The hard MDP\R instances are depicted in Figure A.1 in a semi-formal way. The state space is given by $\mathcal{S} = \{s_{\text{start}}, s_{\text{root}}, s_1, \dots, s_{\overline{S}}, s_{\text{sink}}\}$ and the action space is given by $\mathcal{A} = \{a_0, a_1, \dots, a_{\overline{A}}\}$. The transition model is described below and the horizon is $H \geqslant 3$. We introduce the constant $\overline{H} \in [\![H]\!]$, whose value will be chosen later. Let us observe, for now, that if $\overline{H} = 1$, the transition model is time-homogeneous.

The agent begins in state s_{start}, where every action has the same effect. Specifically, if the stage $h < \overline{H}$, then there is probability $1/2$ to remain in s_{start} and a probability $1/2$ to transition to s_{root}. Instead, if $h \geqslant \overline{H}$, the state transitions to s_{root} deterministically. From state s_{root}, every action has the same effect and the state transitions with equal probability $1/\overline{S}$ to a state s_i with $i \in [\![\overline{S}]\!]$. In all states s_i, apart from a specific one, i.e., state s_*, the expert's policy plays action a_0 deterministically, i.e., $\pi_h^E(a_0|s_i) = 1$ and the state transitions deterministically to s_{sink}. In state s_* the expert's policy plays a_0 as the other ones if the stage $h \neq h_*$, where $h_* \in [\![H]\!]$ is a predefined stage. If, instead, $h = h_*$, the expert's action plays a_0 w.p. $1 - \pi_{\min}$ and a specific action a_* w.p. $\pi_{\min} \in [0, 1/2]$. Then, the transition is deterministic to state s_{sink}. Notice that, having fixed $\overline{H}$, the possible values of h^* are $\{3, \dots, 2 + \overline{H}\}$. State s_{sink} is an absorbing state.

Let us consider the base instance π_0 in which the expert's policy always plays action a_0 deterministically.[1] Additionally, by varying the pair $\ell := (s_*, h_*) \in \{s_1, \dots, s_{\overline{S}}\} \times [\![3, \overline{H} + 2]\!] =: \mathcal{J}$, we can construct the class of instances denoted by $\mathbb{M} = \{\pi_\ell : \ell \in \{0\} \cup \mathcal{J}\}$.

[1] In this construction, the MDP\R does not change across the instances, but what changes is the expert's policy. Thus, we parametrize the instances through the policy rather than the MDP\R.

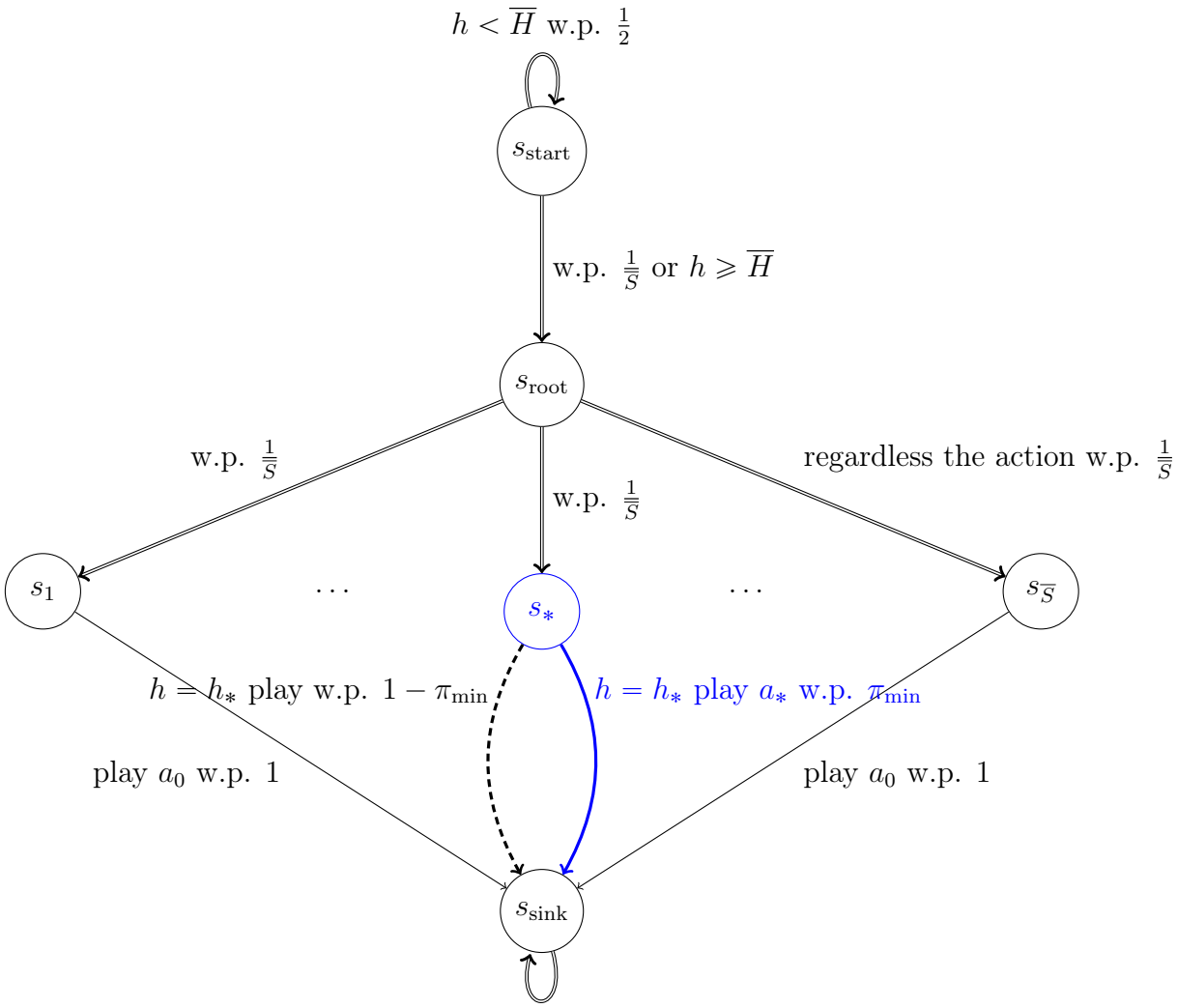


Figure A.1: Semi-formal representation of the the hard instances MDP\R used in the proof of Theorem 18.

Step 2: Feasible Set Computation Let us consider an instance $\pi_\ell \in \mathbb{M}$, we now seek to provide a lower bound to the Hausdorff distance $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\pi_0}, \mathcal{R}_{\pi_\ell})$. To this end, we focus on the pair $\ell = (s_*, h_*)$ and we enforce the convenience of both actions a_0 and a_* over the other actions. Since both actions are played with non-zero probability by the expert's policy, their value function must be the same. Let us denote with $r^\ell \in \mathcal{R}_{\pi_\ell}$, we must have

for all $a_j \notin \{a_0, a_*\}$:

$$
\begin{aligned}
& r^\ell_{h_*}(s_*, a_0) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) \geqslant r^\ell_{h_*}(s_*, a_j) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) \\
& \implies r^\ell_{h_*}(s_*, a_0) \geqslant r^\ell_{h_*}(s_*, a_j), \\
& r^\ell_{h_*}(s_*, a_0) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) = r^\ell_{h_*}(s_*, a_*) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) \\
& \implies r^\ell_{h_*}(s_*, a_0) = r^\ell_{h_*}(s_*, a_*).
\end{aligned}
$$

Consider now the base instance π_0 and denote with $r^0 \in \mathcal{R}_{\pi_0}$. Here we have to enforce the convenience of action a_0 over all the others, including a_*:

$$
\begin{aligned}
& r^0_{h_*}(s_*, a_0) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) \geqslant r^0_{h_*}(s_*, a_j) + \sum_{l=h_*+1}^{H} r^\ell_l(s_{\text{sink}}) \\
& \implies r^0_{h_*}(s_*, a_0) \geqslant r^0_{h_*}(s_*, a_j), \\
& r^0_{h_*}(s_*, a_0) + \sum_{l=h_*+1}^{H} r^0_l(s_{\text{sink}}) \geqslant r^0_{h_*}(s_*, a_*) + \sum_{l=h_*+1}^{H} r^0_l(s_{\text{sink}}) \\
& \implies r^0_{h_*}(s_*, a_0) \geqslant r^0_{h_*}(s_*, a_*).
\end{aligned}
$$

In order to lower bound the Hausdorff distance, we perform a valid assignment of the rewards for the base instance:

$$
r^0_{h_*}(s_*, a_0) = 1,\ r^0_{h_*}(s_*, a_*) = -1,\ r^0_{h_*}(s_*, a_j) = -1.
$$

Thus, the Hausdorff distance can be bounded as follows, having renamed, for convenience $x = r^\ell_{h_*}(s_*, a_0)$ and $y = r^\ell_{h_*}(s_*, a_*)$:

$$
\mathcal{H}_{d^{\text{G}}}(\mathcal{R}_{\pi_0}, \mathcal{R}_{\pi_\ell}) \geqslant \min_{\substack{x,y \in [-1,1] \\ x=y}} \max\{|x-1|, |y+1|\} = 1.
$$

Step 3: Lower bounding Probability Let us consider an (ϵ, δ)-correct algorithm $\mathfrak{A}$ that outputs the estimated feasible set $\widehat{\mathcal{R}}$. Thus, for every $\imath \in \mathcal{J}$, we can lower bound the

error probability:

$$\begin{aligned}\delta &\geqslant \sup_{\text{all } \mathcal{M} \text{ MDP}\setminus\text{R and expert policies } \pi} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi},\widehat{\mathcal{R}}\right) \geqslant \frac{1}{2}\right)\\ &\geqslant \sup_{\pi\in\mathbb{M}} \mathbb{P}_{(\mathcal{M},\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi},\widehat{\mathcal{R}}\right) \geqslant \frac{1}{2}\right)\\ &\geqslant \max_{\ell\in\{0,\imath\}} \mathbb{P}_{(\mathcal{M},\pi_\ell),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_\ell},\widehat{\mathcal{R}}\right) \geqslant \frac{1}{2}\right).\end{aligned}$$

For every $\imath \in \mathcal{J}$, let us define the *identification function*:

$$\Psi_\imath \coloneqq \operatorname*{argmin}_{\ell\in\{0,\imath\}} \mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_\ell},\widehat{\mathcal{R}}\right).$$

Let $\jmath \in \{0,\imath\}$. If $\Psi_\imath = \jmath$, then, $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}_{\pi_{\Psi_\imath}},\mathcal{R}_{\pi_\jmath}) = 0$. Otherwise, if $\Psi_\imath \neq \jmath$, we have:

$$\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_{\Psi_\imath}},\mathcal{R}_{\pi_\jmath}\right) \leqslant \mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_{\Psi_\imath}},\widehat{\mathcal{R}}\right) + \mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}},\mathcal{R}_{\pi_\jmath}\right) \leqslant 2\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}},\mathcal{R}_{\pi_\jmath}\right),$$

where the first inequality follows from triangular inequality and the second one from the definition of identification function $\Psi_\imath$. From Equation (6.3), we have that $\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_{\Psi_\imath}},\mathcal{R}_{\pi_\jmath}\right) \geqslant 1$. Thus, it follows that $\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}},\mathcal{R}_{\pi_\jmath}\right) \geqslant \frac{1}{2}$. This implies the following inclusion of events for $\jmath \in \{0,\imath\}$:

$$\left\{\mathcal{H}_{d^{\mathrm{G}}}\left(\widehat{\mathcal{R}},\mathcal{R}_{\pi_\jmath}\right) \geqslant \frac{1}{2}\right\} \supseteq \{\Psi_\imath \neq \jmath\}.$$

Thus, we can proceed by lower bounding the probability:

$$\begin{aligned}\max_{\ell\in\{0,\imath\}} \mathbb{P}_{(\mathcal{M}_\ell,\pi),\mathfrak{A}}\left(\mathcal{H}_{d^{\mathrm{G}}}\left(\mathcal{R}_{\pi_\ell},\widehat{\mathcal{R}}\right) \geqslant \frac{1}{2}\right) &\geqslant \max_{\ell\in\{0,\imath\}} \mathbb{P}_{(\mathcal{M}_\ell,\pi),\mathfrak{A}}(\Psi_\imath \neq \ell)\\ &\geqslant \frac{1}{2}\left[\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath \neq 0) + \mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath \neq \imath)\right]\\ &= \frac{1}{2}\left[\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath \neq 0) + \mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath = 0)\right],\end{aligned}$$

where the second inequality follows from the observation that $\max\{a,b\} \geqslant \frac{1}{2}(a+b)$ and the equality from observing that $\Psi_\imath \in \{0,\imath\}$. We can now apply the Bretagnolle-Huber inequality [32, Theorem 14.2] (also reported in Theorem 20 for completeness) with $\mathbb{P} = \mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}$, $\mathbb{Q} = \mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}$, and $\mathcal{A} = \{\Psi_\imath \neq 0\}$:

$$\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}(\Psi_\imath \neq 0) + \mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}(\Psi_\imath = 0) \geqslant \frac{1}{2}\exp\left(-D_{\mathrm{KL}}\left(\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}, \mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}\right)\right).$$

Step 4: KL-divergence Computation Let $\mathcal{M} \in \mathbb{M}$, we denote with $\mathbb{P}_{\mathfrak{A},\mathcal{M},\pi}$ the joint probability distribution of all events realized by the execution of the algorithm in the MDP\R (the presence of p is irrelevant as it does not change across the different instances):

$$\mathop{\mathbb{P}}_{(\mathcal{M},\pi),\mathfrak{A}} = \prod_{t=1}^{\tau} \rho_t(s_t, a_t, h_t | H_{t-1}) p_{h_t}(s'_t | s_t, a_t) \pi^E_{h_t}(a^E_t | s_t).$$

where $H_{t-1} = (s_1, a_1, h_1, s'_1, a^E_1, \dots, s_{t-1}, a_{t-1}, h_{t-1}, s'_{t-1}, a^E_{t-1})$ is the history. Let $\imath \in \mathcal{I}$. Let us now move to the KL-divergence between the instances π_0 and $\pi_\imath$ for some $\imath = (s_*, h_*) \in \mathcal{J}$:

$$\begin{aligned} D_{\mathrm{KL}}\left(\mathbb{P}_{(\mathcal{M}_0,\pi),\mathfrak{A}}, \mathbb{P}_{(\mathcal{M}_\imath,\pi),\mathfrak{A}}\right) &= \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[\sum_{t=1}^{\tau} D_{\mathrm{KL}}\left(\pi^0_{h_t}(\cdot|s_t), \pi^\imath_{h_t}(\cdot|s_t)\right)\right] \\ &\leqslant \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*)\right] D_{\mathrm{KL}}\left(\pi^0_{h_*}(\cdot|s_*), \pi^\imath_{h_*}(\cdot|s_*)\right) \\ &\leqslant \log\frac{1}{1-\pi_{\min}} \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*, a_*)\right]. \end{aligned}$$

having observed that the transition models differ in $\imath = (s_*, h_*)$ and defined $N^\tau_{h_*}(s_*) = \sum_{t=1}^{\tau} \mathbb{1}\{(s_t, h_t) = (s_*, h_*)\}$ and the last passage is obtained by explicitly computing the KL-divergence:

$$\begin{aligned} D_{\mathrm{KL}}\left(\pi^0_{h_*}(\cdot|s_*), \pi^\imath_{h_*}(\cdot|s_*)\right) &= \sum_{a\in\mathcal{A}} \pi^0_{h_*}(a|s_*) \log\left(\frac{\pi^0_{h_*}(a|s_*)}{\pi^\imath_{h_*}(a|s_*)}\right) \\ &= \pi^0_{h_*}(a_0|s_*) \log\left(\frac{\pi^0_{h_*}(a_0|s_*)}{\pi^\imath_{h_*}(a_0|s_*)}\right) = \log\frac{1}{1-\pi_{\min}}. \end{aligned}$$

Putting all together, we have:

$$\delta \geqslant \frac{1}{4}\exp\left(-\log\frac{1}{1-\pi_{\min}} \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*)\right]\right) \implies \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*)\right] \geqslant \frac{\log\frac{1}{4\delta}}{\log\frac{1}{1-\pi_{\min}}}.$$

Thus, summing over $(s_*, a_*) \in \mathcal{J}$, we have:

$$\begin{aligned} \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] &\geqslant \sum_{(s_*,a_*)\in\mathcal{J}} \mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}\left[N^\tau_{h_*}(s_*, a_*)\right] \\ &= \sum_{(s_*,a_*,h_*)\in\mathcal{I}} \frac{(H - \overline{H} - 2)^2 \log\frac{1}{4\delta}}{2\epsilon^2} \\ &= \overline{SH}\frac{\log\frac{1}{4\delta}}{\log\frac{1}{1-\pi_{\min}}}. \end{aligned}$$

The number of states is given by $S = |\mathcal{S}| = \overline{S} + 3$. Let us first consider the time-

Input: significance $\delta \in (0,1)$, ϵ target accuracy
$t \leftarrow 0$, $\epsilon_0 \leftarrow +\infty$
while $\epsilon_t > \epsilon$ **do**
  $t \leftarrow t + SAH$
  Collect one sample from each $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$
  Update $\widehat{p}^t$ and $\widehat{\pi}^{E,t}$ according to (7.1)
  Update $\epsilon_t = \max_{(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!]} \overline{\mathcal{C}}_h^t(s,a)$ (resp. $\widetilde{\overline{\mathcal{C}}}_h^t(s,a)$)
end while

Algorithm 11: UniformSampling-IRL (`US-IRL`) for time-inhomogeneous (resp. time-homogeneous) transition models and expert's policies.

homogeneous case, i.e., $\overline{H} = 1$:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant (S-3)\frac{\log \frac{1}{4\delta}}{\log \frac{1}{1-\pi_{\min}}}.$$

For $\delta < 1/16$, $S \geqslant 7$, $A \geqslant 2$, $H \geqslant 2$, we obtain:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{S}{4\log \frac{1}{1-\pi_{\min}}} \log \frac{1}{\delta}.$$

For the time-inhomogeneous case, instead, we select $\overline{H} = H/2$, to get:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{(S-3)(H/2)}{\epsilon^2}\frac{\log \frac{1}{4\delta}}{\log \frac{1}{1-\pi_{\min}}}.$$

For $\delta < 1/16$, $S \geqslant 7$, $A \geqslant 2$, $H \geqslant 2$, we obtain:

$$\mathop{\mathbb{E}}_{(\mathcal{M}_0,\pi),\mathfrak{A}}[\tau] \geqslant \frac{SH}{8\log \frac{1}{1-\pi_{\min}}} \log \frac{1}{\delta}.$$

□

A.2. Algorithm

In this appendix, we extend `US-IRL` to the expert's policy estimation under Assumption 1. The pseudocode is reported in Algorithm 11. The interaction protocol follows the same principles of Algorithm 10, with the only difference that the confidence function, now, must account for the policy estimation, leading to the following function for every

$(s, a, h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!]$:[2]

$$\overline{C}_h^t(s,a) := 2(H-h+1)\left(\mathbb{1}_{\left\{n_h^t(s) \geqslant \max\left\{1, \xi(n_h^t(s),\delta/2)\right\}\right\}} + \sqrt{\frac{2\beta\left(n_h^t(s,a),\delta/2\right)}{n_h^t(s,a)}}\right). \tag{A.2}$$

where:

$$\xi(n,\delta) := \frac{\log(2SAHn^2/\delta)}{\log(1/(1-\pi_{\min}))}.$$

It is worth noting that we have distributed the confidence δ equally between the problem estimating the policy and that of estimating the transition model. The following theorem provides the sample complexity of `US-IRL`.

Theorem 19 (Sample Complexity of `US-IRL`)**.** *Let $\epsilon > 0$ and $\delta \in (0,1)$, under Assumption 1, `US-IRL` is (ϵ,δ)-PAC for d^G-IRL and with probability at least $1-\delta$ it stops after τ samples with:*

- *if the transition model p and the expert's policy π^E are time-inhomogeneous:*

$$\tau \leqslant \frac{8H^3SA}{\epsilon^2}\left(\log\left(\frac{SAH}{\delta}\right) + (S-1)\overline{C}_1\right) + SH + \\ + \frac{SH}{\log(1/(1-\pi_{\min}))}\left(\log\left(\frac{4SAH}{\delta}\right) + \overline{C}_2\right),$$

  where $\overline{C}_1 = \log(e/(S-1)) + (8eH^2)/((S-1)\epsilon^2)(\log(2SAH/\delta) + 4e))$ and $\overline{C}_2 = 2\log\left(\frac{\log(4SAH/\delta)+2}{\log(1/(1-\pi_{\min}))}\right)$.
- *if the transition model p and the expert's policy π^E are time-homogeneous:*

$$\tau \leqslant \frac{8H^2SA}{\epsilon^2}\left(\log\left(\frac{SA}{\delta}\right) + (S-1)\overline{C}_1\right) + SH + \\ + \frac{S}{\log(1/(1-\pi_{\min}))}\left(\log\left(\frac{4SA}{\delta}\right) + \overline{C}_2\right),$$

[2] As for the transition model, one can adapt the confidence function for the case of stationary policy in straightforward way:

$$\widetilde{\overline{C}}_h^t(s,a) := 2(H-h+1)\left(\mathbb{1}_{\left\{n_h^t(s) \geqslant \max\left\{1, \widetilde{\xi}(n^t(s),\delta/2)\right\}\right\}} + \sqrt{\frac{2\widetilde{\beta}\left(n^t(s,a),\delta/2\right)}{n^t(s,a)}}\right), \tag{A.1}$$

where:

$$\widetilde{\xi}(n,\delta) := \frac{\log(2SAn^2/\delta)}{\log(1/(1-\pi_{\min}))}.$$

In principle, one can also consider the case of a time-homogeneous transition model and time-inhomogeneous expert's policy. We omit it because it adds nothing to the characteristics of the problem and of the algorithms.

where $\widetilde{\overline{C}}_1 = \log(e/(S-1)) + (8eH^2)/((S-1)\epsilon^2)(\log(2SA/\delta) + 4e))$ *and* $\widetilde{\overline{C}}_2 = 2\log\left(\frac{\log(4SA/\delta)+2}{\log(1/(1-\pi_{\min}))}\right)$.

Before moving to the proof, let us observe that the result matches the rate of the lower bound of Theorem 18 up to logarithmic terms.

Proof. We make use of the notation of the proof of Theorem 19. We start with the case in which the transition model is time-inhomogeneous. In addition to the good event $\mathcal{E}$ related to the transition model, we introduce the following one:

$$\mathcal{E}_\pi := \Bigg\{ \forall t \in \mathbb{N},\, \forall (s,a,h) \in \mathcal{S} \times \mathcal{A} \times [\![H]\!] : \\ \left| \mathbb{1}_{\pi_h^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}_h^{E,t}(a|s)=0} \right| \leqslant \mathbb{1}_{\left\{ n_h^t(s) \geqslant \max\left\{1, \xi(n_h^t(s),\delta/2)\right\}\right\}} \Bigg\},$$

where π_h^E is the true expert's policy and $\widehat{\pi}^{E,t}$ is its estimate via Equation (7.1) at time t. Thanks to Lemma 4 and Lemma 5, we have that $\mathbb{P}(\mathcal{E} \cap \mathcal{E}_\pi) \geqslant 1-\delta$. Thus, under the good event $\mathcal{E} \cap \mathcal{E}_\pi$, we apply Theorem 12 to obtain $\mathcal{H}_{d^{\mathrm{G}}}(\mathcal{R}, \widehat{\mathcal{R}}^\tau) \leqslant \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} \overline{\mathcal{C}}_h^t(s,a)$. A sufficient condition to make this term $\leqslant \epsilon$ is to request the following ones:

$$\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} 2(H-h+1)\mathbb{1}_{\left\{ n_h^t(s) \geqslant \max\left\{1, \xi(n_h^t(s),\delta/2)\right\}\right\}} = 0,$$

$$\max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[\![H]\!]} 2\sqrt{2}(H-h+1)\sqrt{\frac{\beta\left(n_h^t(s,a),\delta/2\right)}{n_h^t(s,a)}} \leqslant \epsilon.$$

For the first one, we first enforce the condition:

$$n_h^t(s) \geqslant \xi(n_h^t(s),\delta/2) = \frac{\log(4SAH(n_h^t)^2/\delta)}{\log(1/(1-\pi_{\min}))} = \frac{\log(4SAH/\delta)}{\log(1/(1-\pi_{\min}))} + \frac{2\log n_h^t}{\log(1/(1-\pi_{\min}))}.$$

Using Lemma 15 of [27] and enforcing $n_h^t(s) \geqslant 1$, we obtain:

$$n_h^\tau(s) \leqslant 1 + \frac{1}{\log(1/(1-\pi_{\min}))}\left(\log(4SAH/\delta) + 2\log\left(\frac{\log(4SAH/\delta)+2}{\log(1/(1-\pi_{\min}))}\right)\right).$$

Combining this result with that of Theorem 19 for what concerns the transition model, we obtain:

$$\tau \leqslant \frac{8H^3SA}{\epsilon^2}\left(\log\left(\frac{2SAH}{\delta}\right) + (S-1)\log\left(\frac{8eH^2}{(S-1)\epsilon^2}\left(\log\left(\frac{2SAH}{\delta}\right) + 4e\right)\right)\right) \\ + SH + \frac{SH}{\log(1/(1-\pi_{\min}))}\left(\log(4SAH/\delta) + 2\log\left(\frac{\log(4SAH/\delta)+2}{\log(1/(1-\pi_{\min}))}\right)\right).$$

Analogous derivations can be carried out for the case of time-homogenous policy using the good event:

$$\widetilde{\mathcal{E}}_\pi := \left\{\forall t \in \mathbb{N},\, \forall (s,a) \in \mathcal{S}\times\mathcal{A} : \left|\mathbb{1}_{\pi^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}^{E,t}(a|s)=0}\right| \leqslant \mathbb{1}_{\left\{n^t(s)\geqslant \max\left\{1,\widetilde{\xi}(n^t(s),\delta/2)\right\}\right\}}\right\},$$

where $\widetilde{\xi}(n,\delta) := \frac{\log(2SAn^2/\delta)}{\log(1/(1-\pi_{\min}))}$. We omit the tedious but straightforward derivation. □

Lemma 5. *Under Assumption 1, the following statements hold:*

- *for* $\xi(n,\delta) := \frac{\log(2SAHn^2/\delta)}{\log(1/(1-\pi_{\min}))}$, *we have that* $\mathbb{P}(\mathcal{E}_\pi) \geqslant 1-\delta$;
- *for* $\widetilde{\xi}(n,\delta) := \frac{\log(2SAn^2/\delta)}{\log(1/(1-\pi_{\min}))}$, *we have that* $\mathbb{P}(\widetilde{\mathcal{E}}_\pi) \geqslant 1-\delta$.

Proof. Let us start with the first statement. We apply first a union bound and, then, Lemma 8 to perform the concentration:

$$\begin{aligned}
\mathbb{P}(\mathcal{E}_\pi^c) &= \mathbb{P}\Bigg(\exists t \in \mathbb{N},\, \exists (s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!] : \\
&\quad \left|\mathbb{1}_{\pi_h^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}_h^{E,t}(a|s)=0}\right| \leqslant \mathbb{1}_{\left\{n_h^t(s)>\max\left\{1,\xi(n_h^t(s),\delta)\right\}\right\}}\Bigg) \\
&= \mathbb{P}\Bigg(\exists n \in \mathbb{N},\, \exists (s,a,h) \in \mathcal{S}\times\mathcal{A}\times[\![H]\!] : \\
&\quad \left|\mathbb{1}_{\pi_h^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}_h^{E,[n]}(a|s)=0}\right| > \mathbb{1}_{\{n\geqslant\max\{1,\xi(n,\delta)\}\}}\Bigg) \\
&\leqslant \sum_{h\in[\![H]\!]}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\sum_{n\geqslant 0}\mathbb{P}\left(\left|\mathbb{1}_{\pi_h^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}_h^{E,[n]}(a|s)=0}\right| \leqslant \mathbb{1}_{\{n>\max\{1,\xi(n,\delta)\}\}}\right) \\
&\leqslant \sum_{h\in[\![H]\!]}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\sum_{n\geqslant 1}\mathbb{P}\left(\left|\mathbb{1}_{\pi_h^E(a|s)=0} - \mathbb{1}_{\widehat{\pi}_h^{E,[n]}(a|s)=0}\right| \leqslant \mathbb{1}_{\{n>\max\{1,\xi(n,\delta)\}\}}\right) \\
&\leqslant \sum_{h\in[\![H]\!]}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\frac{\delta}{2SAHn^2} = \frac{\pi^2}{6}\frac{\delta}{2} \leqslant \delta.
\end{aligned}$$

where on the first passage we enforced the condition on the time instants in which the policy estimate changes (i.e., when (s,h) is visited) and we denotes such an estimate as $\widehat{\pi}_h^{E,[n]}$. Then, after a union bound, we apply Lemma 8. The proof of the second statement is analogous having simply observed that the union bound has to be performed over $\mathcal{S}\times\mathcal{A}$ only. □

B | Technical Lemmas

Theorem 20. *(*Bretagnolle-Huber inequality *[32, Theorem 14.2]) Let* $\mathbb{P}$ *and* $\mathbb{Q}$ *be probability measures on the same measurable space* $(\Omega, \mathcal{F})$*, and let* $\mathcal{A} \in \mathcal{F}$ *be an arbitrary event. Then,*

$$\mathbb{P}(\mathcal{A}) + \mathbb{Q}(\mathcal{A}^c) \geqslant \frac{1}{2} \exp\left(-D_{KL}(\mathbb{P}, \mathbb{Q})\right),$$

where $\mathcal{A}^c = \Omega \backslash \mathcal{A}$ *is the complement of* $\mathcal{A}$*.*

Theorem 21. *(*Fano inequality *[18, Proposition 4]) Let* $\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_M$ *be probability measures on the same measurable space* $(\Omega, \mathcal{F})$*, and let* $\mathcal{A}_1, \ldots, \mathcal{A}_M \in \mathcal{F}$ *be a partition of* Ω*. Then,*

$$\frac{1}{M} \sum_{i=1}^{M} \mathbb{P}_i(\mathcal{A}_i^c) \geqslant 1 - \frac{\frac{1}{M} \sum_{i=1}^{M} D_{KL}(\mathbb{P}_i, \mathbb{P}_0) - \log 2}{\log M},$$

where $\mathcal{A}^c = \Omega \backslash \mathcal{A}$ *is the complement of* $\mathcal{A}$*.*

Lemma 6. *[25, Proposition 1] Let* $\mathbb{P} = (p_1, \ldots, p_D)$ *be a categorical probability measure on the support* $[\![D]\!]$*. Let*$\mathbb{P}_n = (\widehat{p}_1, \ldots, \widehat{p}_D)$ *be the maximum likelihood estimate of* $\mathbb{P}$ *obtained with* $n \geqslant 1$ *independent samples. Then, for every* $\delta \in (0, 1)$ *it holds that:*

$$\mathbb{P}\left(\exists n \geqslant 1 \,:\, nD_{KL}\left(\mathbb{P}_n, \mathbb{P}\right) > \log(1/\delta) + (D-1)\log\left(e(1 + n/(D-1))\right)\right) \leqslant \delta$$

Lemma 7. *Let* $\epsilon \in [0, 1/2]$ *and* $\mathbf{v} \in \{-\epsilon, \epsilon\}^D$ *such that* $\sum_{i=1}^{d} v_i = 0$*. Consider the two categorical distributions* $\mathbb{P} = \left(\frac{1}{D}, \frac{1}{D}, \ldots, \frac{1}{D}\right)$ *and* $\mathbb{P} = \left(\frac{1+v_1}{D}, \frac{1+v_2}{D}, \ldots, \frac{1+v_D}{D}\right)$*. Then, it holds that:*

$$D_{KL}(\mathbb{P}, \mathbb{Q}) \leqslant 2\epsilon^2 \qquad \textit{and} \qquad D_{KL}(\mathbb{Q}, \mathbb{P}) \leqslant 2\epsilon^2.$$

Proof. First of all we recall that since $\sum_{i=1}^{M} v_i = 0$, we have $|\{i \in [\![D]\!] : v_i = \epsilon\}| = |i \in$

$[\![D]\!] : v_i = -\epsilon| = D/2$. Let us compute the KL-divergence $D_{\mathrm{KL}}(\mathbb{P}, \mathbb{Q})$:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{P}, \mathbb{Q}) &= \sum_{i=1}^{D} \frac{1+v_i}{D} \log \frac{\frac{1+v_i}{D}}{\frac{1}{D}} \\
&= \sum_{i \in [\![D]\!] : v_i = \epsilon} \frac{1+\epsilon}{D} \log(1+\epsilon) + \sum_{i \in [\![D]\!] : v_i = -\epsilon} \frac{1-\epsilon}{D} \log(1-\epsilon) \\
&= \frac{1+\epsilon}{2} \log(1+\epsilon) + \frac{1-\epsilon}{2} \log(1-\epsilon) \\
&= \underbrace{\frac{1}{2} \log(1-\epsilon^2)}_{\leqslant 0} + \frac{\epsilon}{2} \log(1+\epsilon) - \frac{\epsilon}{2} \log(1-\epsilon) \\
&\leqslant \frac{\epsilon^2}{2} + \frac{\epsilon}{2} \left(\frac{1}{1-\epsilon} - 1 \right) = \epsilon^2 \frac{2-\epsilon}{2(1-\epsilon)} \leqslant \frac{3}{2} \epsilon^2 \leqslant 2\epsilon^2.
\end{aligned}$$

where we used the inequality $\log(1+x) \leqslant x$ for $x \geqslant 0$ and $-\log(1-x) \leqslant \frac{1}{1-x} - 1$ for $0 < x < 1$ and exploited that $\epsilon \leqslant \frac{1}{2}$. Let us now move to the second KL-divergence $D_{\mathrm{KL}}(\mathbb{Q}, \mathbb{P})$:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbb{Q}, \mathbb{P}) &= \sum_{i=1}^{D} \frac{1}{D} \log \frac{\frac{1}{D}}{\frac{1+v_i}{D}} \\
&= \sum_{i \in [\![D]\!] : v_i = \epsilon} \frac{1}{D} \log \frac{1}{1+\epsilon} + \sum_{i \in [\![D]\!] : v_i = -\epsilon} \frac{1}{D} \log \frac{1}{1-\epsilon} \\
&= -\frac{1}{2} \log(1-\epsilon^2) \\
&\leqslant \frac{1}{2} \left(\frac{1}{1-\epsilon^2} - 1 \right) = \frac{\epsilon^2}{2(1-\epsilon^2)} \leqslant \frac{2}{3} \epsilon^2 \leqslant 2\epsilon^2,
\end{aligned}$$

where we used the inequality $-\log(1-x) \leqslant \frac{1}{1-x} - 1$ for $0 < x < 1$ and observed that $\epsilon \leqslant \frac{1}{2}$. □

Lemma 8. *Let $\mathbb{P} = (p_1, \ldots, p_D)$ be a categorical probability measure on the support $[\![D]\!]$. Let$\mathbb{P}_n = (\widehat{p}_1, \ldots, \widehat{p}_D)$ be the maximum likelihood estimate of $\mathbb{P}$ obtained with $n \geqslant 1$ independent samples. Then, if $p_i \in \{0\} \cup [p_{\min}, 1]$ for some $p_{\min} \in (0, 1]$. Then, for every $i \in [\![D]\!]$ individually, for every $\delta \in (0, 1)$, it holds that:*

$$\left| \mathbb{1}_{\{p_i = 0\}} - \mathbb{1}_{\{\widehat{p}_i = 0\}} \right| \leqslant \mathbb{1}_{\left\{ n \geqslant \max\left\{ 1, \frac{\log\left(\frac{1}{\delta}\right)}{\log\left(\frac{1}{1-p_{min}}\right)} \right\} \right\}}.$$

Proof. Let $i \in [\![D]\!]$ such that $p_i > 0$ and, thus, $\mathbb{1}_{\{p_i = 0\}} = 0$. By assumption, it must be that $p_i \geqslant p_{\min}$. To make a mistake, we must have that $\mathbb{1}_{\{\widehat{p}_i = 0\}} = 1$, and, thus, $\widehat{p}_i = 0$.

Thus, we compute the probability that no sample i is observed among the n ones:

$$\mathbb{P}\left(\bigcap_{j\in[\![n]\!]} X_j \neq i\right) = \prod_{j\in[\![n]\!]} \mathbb{P}\left(X_j \neq i\right) = \mathbb{P}\left(X_1 \neq i\right)^n = (1-p_i)^n \leqslant (1-p_{\min})^n,$$

where we exploited the fact that the random variables X_j are i.i.d.. If $n = 0$ the latter expression is 1. If, instead, $n \geqslant 1$, by setting the last expression equal to δ, we get:

$$(1-p_{\min})^n \leqslant \delta \implies n \geqslant \frac{\log\left(\frac{1}{\delta}\right)}{\log\left(\frac{1}{1-p_{min}}\right)}.$$

The result follows. □

Lemma 9. *Let $\mathcal{V} = \{v \in \{-1,1\}^D : \sum_{j=1}^D v_j = 0\}$. Then, the $\frac{D}{16}$-packing number of $\mathcal{V}$ w.r.t. the metric $d(v,v') = \sum_{j=1}^D |v_j - v'_j|$ is lower bounded by $2^{\frac{D}{5}}$.*

Proof. Let us denote the packing number with $M(\epsilon;\mathcal{V},d)$ and the covering number with $N(\epsilon;\mathcal{V},d)$. It is well known that $N(\epsilon;\mathcal{V},d) \leqslant M(\epsilon;\mathcal{V},d)$ [19]. Thus, a lower bound to the covering number is a lower bound to the packing number. Let us consider the (pseudo)metric $d'(v,v') = \sum_{j=1}^{D/2} |v_j - v'_j|$ that considers the first half of the components only. Clearly, we have that $d'(v,v') \leqslant d(v,v')$. Therefore, any ϵ-cover w.r.t. $d(v,v')$ is an ϵ-cover w.r.t. $d'(v,v')$ and, consequently, $N(\epsilon;\mathcal{V},d') \leqslant N(\epsilon;\mathcal{V},d)$. Since the (pseudo)metric d' considers only the first half of the components, constructing an ϵ-cover of $\mathcal{V}$ w.r.t. d' is equivalent to constructing an ϵ-cover of $\mathcal{V}'$ w.r.t. d', where $\mathcal{V}' = \{-1,1\}^{D/2}$. $\mathcal{V}'$ considers the first half of the components of vectors of $\mathcal{V}$, that can be freely chosen, disregarding the summation constraint.[1] Thus, $N(\epsilon;\mathcal{V},d') = N(\epsilon;\mathcal{V}',d')$. Notice that d' is now a proper metric on $\mathcal{V}' = \{-1,1\}^{D/2}$. Now, we reduce the problem to constructing cover on the Hamming space $\mathcal{H} = \{0,1\}^{D/2}$. Indeed, we can always map an $(\epsilon/2)$-cover for the Hamming space $\mathcal{H}$ to an ϵ-cover for the space $\mathcal{V}'$. Specifically, let $(h_l)_l$ an $(\epsilon/2)$-cover for the Hamming space, we construct $(v'_l)_l$ by applying the following transformation:

$$v'_j = \begin{cases} -1 & \text{if } h_j = 0 \\ 1 & \text{if } h_j = 1 \end{cases},$$

[1] From an algebraic perspective, $\mathcal{V}'$ can be considered the quotient set obtained from $\mathcal{V}$ by means of the equivalence relation $v \sim v' \iff v_j = v_{j'}$ for all $j \in [\![D/2]\!]$.

or, in more convenient way, $v' = 2h - 1$. Let $v' \in \mathcal{V}'$:

$$\min_l d'(v', v'_{l,j}) = \min_l \sum_{j=1}^{D/2} |v'_j - v'_{l,j}| = 2 \min_l \sum_{j=1}^{D/2} |h'_j - h'_{l,j}| \leqslant \epsilon.$$

The covering number of a Hamming space has been lower bounded in [11] for $\epsilon \in [\![D/2]\!]$ as:

$$\log_2 N(\epsilon; \mathcal{H}, d') \geqslant \frac{D}{2} - \log_2 \sum_{k=0}^{\epsilon} \binom{D/2}{k}.$$

We take $\epsilon = D/16$, and we use the known bound $\sum_{i=0}^{k} \binom{n}{k} \leqslant \left(\frac{en}{k}\right)^k$:

$$\sum_{k=0}^{D/16} \binom{D/2}{k} \leqslant (8e)^{D/16}.$$

From, which, we get:

$$\log_2 N(\epsilon; \mathcal{H}, d') \geqslant \frac{D}{2} - \log_2 \sum_{k=0}^{\epsilon} \binom{D/2}{k} \geqslant \frac{D}{2} - \frac{D}{16} \log_2(8e) \geqslant \frac{D}{5}.$$

□

Acknowledgements

Questa tesi e in particolare questa laurea non sono un traguardo che ho raggiunto da solo, bensì sono il risultato di un lavoro di squadra. Voglio ringraziare i miei genitori, mio papà *Stefano* e mia mamma *Monica*, perché senza di loro dubito che sarei mai riuscito a raggiungere un obiettivo come questo nella mia vita. Mi supportano e sopportano sin da quando ero piccolo, ci sono sempre stati per me e mi hanno insegnato a vivere (e continuano a farlo). Papà, mamma, sono molto fortunato ad avervi come genitori e ci tengo a sottolineare che questa laurea è soprattutto merito vostro. Siete voi che mi avete insegnato sin da quando ero piccolo l'importanza dell'istruzione e mi avete accompagnato giorno dopo giorno in questo percorso. Ve ne sarò sempre grato. Grazie a *Fede* e a *Leo*, perché quando ho bisogno ci siete sempre. Voglio ringraziare mia nonna *Anna*, la nunèta, perché ci vogliamo molto bene e troviamo sempre qualcosa su cui ridere e scherzare. Mi hai aiutato durante tutti questi anni e sei stata dalla mia parte, e per questo ti ringrazio molto. Ringrazio lo zio *Tesa* e la zia *Maurizia*, lo zio *Luca* e lo zio *Mauri*, la *Roby*, la zia *Patty*, i cuginetti, perché con voi sto bene, sono molto felice e orgoglioso di avervi dalla mia parte. Ringrazio la *Caty*, che da quasi tre anni a questa parte ha deciso di sopportarmi. Grazie per sostenermi sempre. Ringrazio tutti i miei amici, chi vedo più spesso e chi meno, per essere stati parte di questo percorso. Ringrazio inoltre *Alberto* per avermi seguito durante la tesi e il prof. *Restelli* per questa opportunità.

Filippo