POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in
Computer Science and Engineering

Beyond Maximum Likelihood Model Estimation in Model-based Policy Search

Supervisor:
PROF. MARCELLO RESTELLI

Assistant Supervisors:
DOTT. ALBERTO MARIA METELLI
DOTT. ANDREA TIRINZONI
DOTT. MATTEO PAPINI

Master Graduation Thesis by:
PIERLUCA D'ORO
Student Id n. 893751

Academic Year 2018-2019

All'Etna.

RINGRAZIAMENTI

Desidero ringraziare innanzitutto il Prof. Marcello Restelli, la cui disponibilità e la cui passione per la ricerca mi hanno profondamente ispirato e donato una nuova prospettiva su come sia possibile condurre attività scientifica di alta qualità. Ringrazio anche il Dott. Tirinzoni e il Dott. Papini per il loro fondamentale contributo a questo lavoro, ed in particolar modo il Dott. Metelli, il cui talento mi ha illuminato numerose volte sul valore di una matematica rigorosa, creativa ed elegante.

Ringrazio Alessia, che mi è stata accanto nei momenti più incerti e difficili di questi due anni.

Vorrei anche ringraziare Ennio, con cui ho condiviso progetti, gioie e dolori accademici, ma anche molti sorrisi, e gli amici della Residenza Galileo.

Infine, vorrei riconoscere quanto importante sia stato l'aiuto datomi dalla mia famiglia, che nonostante la distanza è sempre stata presente e di supporto.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 5.2 Hyperparameters used for algorithm comparison in the different environments. The apexes *model* and *policy* indicate the parameters employed in optimizing the two using the Adam [50] optimizer. Clearly, hyperparameters concerning the estimation of the forward model are ignored in model-free algorithms. The hypeparameters, except for q and γ, were chosen by trial and error from a range of $(0.001, 0.9)$. 47

ACRONYMS

RL	Reinforcement Learning
MDP	Markov Decision Process
MBRL	Model-Based Reinforcement Learning
PGT	Policy Gradient Theorem
ML	Maximum Likelihood
MVG	Model-Value-based Gradient
MFG	Model-Free Gradient
FMG	Fully Model-based Gradient
GAMPS	Gradient-Aware Model-based Policy Search

ABSTRACT

Reinforcement Learning allows an agent to learn behaviors for solving sequential decision making problems. When learning such control policies, an algorithm can take advantage of a learned model of the dynamics of the environment. This is the rationale behind Model-Based Reinforcement Learning (MBRL), in which the agent learns, and then employs, estimated models. MBRL approaches present several advantages, for instance in terms of sample efficiency, compared to the ones, known as model-free, that learn a control policy without explicitly representing the dynamics. However, the dynamics of the environment can be extremely complex and very hard to model using few data, endangering the promise of data efficiency that underlies MBRL. Fortunately, in many interesting application domains, perfectly modeling the dynamics of the whole environment is not necessary for a model to be effectively used by a learning agent. Instead, it is possible to use simpler model classes, whose estimation requires few interactions with the environment, and focus their limited expression capability where it is more needed for control purposes. Nonetheless, most MBRL methods learn the model by maximum likelihood estimation, judging the relative importance of environment dynamics just upon visitation, and completely ignoring the underlying decision problem. This thesis proposes Gradient-Aware Model-based Policy Search (GAMPS), a novel model-based algorithm for policy improvement that, by leveraging a weighting scheme on the loss function, learns a model focused on the aspects of the dynamics that are most relevant for estimating the policy gradient. GAMPS uses the Model-Value-based Gradient, a newly formalized approximation for the policy gradient that employs collected trajectories together with an estimated value function. The empirical evaluation for the method, carried out on simple yet illustrative tasks, both in discrete and continuous domains, shows that it is able to outperform standard model-free policy gradient methods and model-based methods based on maximum likelihood model estimation.

SOMMARIO

L'Apprendimento per Rinforzo consente a un agente di imparare i comportamenti necessari per risolvere problemi che richiedono sequenze di decisioni. Durante l'apprendimento di una politica di controllo, un algoritmo può far uso di un modello stimato delle dinamiche dell'ambiente. Questa caratteristica contraddistingue gli approcci di Apprendimento per Rinforzo basato su modelli, che apprendono ed impiegano modelli stimati a beneficio dell'agente. Questi approcci presentano diversi vantaggi, per esempio in termini di efficienza, in confronto ai metodi non basati su modelli, che apprendono una politica di controllo senza rappresentare esplicitamente le dinamiche. Ciononostante, le dinamiche dell'ambiente possono essere estremamente complesse e molto difficili da apprendere, minando la promessa di efficienza degli approcci basati su modelli. Fortunatamente, in molti domini applicativi di generale interesse, apprendere perfettamente la dinamica nell'intero ambiente non è necessario affinché un modello possa essere usato in maniera proficua da un agente che apprende. Invece, è possibile usare classi di modelli più semplici, che possono dunque essere stimati con meno campioni, e concentrare la loro limitata capacità di rappresentazione dove più è necessaria in termini di controllo. Pur tuttavia, la maggior parte degli approcci esistenti impara la dinamica dell'ambiente per massima verosimiglianza, implicitamente giudicando importante la sola frequenza di visita a determinati stati, ed ignorando completamente il problema decisionale che si desidera risolvere. Questa tesi propone Gradient-Aware Model-based Policy Search (GAMPS), un algoritmo innovativo basato su modelli per migliorare una politica di controllo che, sfruttando una pesatura sulla funzione di costo, apprende un modello accurato sugli aspetti delle dinamiche ambientali che più sono rilevanti per stimare il gradiente della politica. GAMPS fa uso del Model-Value-based Gradient, una nuova formalizzazione per un'approssimazione del gradiente della politica, che usa traiettorie collezionate nell'ambiente insieme ad una funzione di valore stimata. La valutazione empirica del metodo, eseguita su problemi semplici ma illustrativi, sia in domini discreti che continui, mostra come sia capace di sorpassare le prestazioni di algoritmi standard che usano il gradiente della politica senza alcun modello ma anche quelle di algoritmi basati su modelli addestrati per massima verosimiglianza.

1 INTRODUCTION

Being able to solve *sequential decision making problems* is at the core of what human beings define as intelligence. An intelligent agent must be able to tackle such problems in order to accomplish its *goals* or, in other terms, to maximize its *utility*. Reinforcement Learning (RL) offers a very general approach for formalizing and solving sequential decision making problems: the agent is provided with a reward signal and its goal is to maximize the total amount of reward that it collects. Problems in reinforcement learning are conveniently modeled using the concept of Markov Decision Process (MDP), assuming transitions in the world only depend on the previous state of it and the action executed by the agent. The objective is usually to find an optimal policy to be used in the environment.

One of the most important challenges in reinforcement learning is the strive for sample efficiency: the agent must reach satisfying performance using as little experience as possible. Humans, the best example of intelligent being that is known to us, acquired outstanding tools for solving sequential decision making problems using very little data, through years of hardware (i.e., of the body, including the brain) and software (i.e., of the mind) evolution. One of the most important of these tools that are available to us is, roughly speaking, *anticipation*: during most of the interactions of a human with any familiar environment, the mind is able to anticipate how the world will evolve, naturally or in response to actions of the human being herself. This capability can be leveraged in several, powerful, ways: a baseball player can predict what the trajectory of a ball will be; an expert musician can predict which the effect of playing a certain sequence of notes over a chord will be; an expert hiker can imagine which its arriving time will be by looking at a trail from above; an athlete can watch a video of one of its performances and learn from it by counterfactually predicting what the alternative outcome of its movements could have been. Such examples of internal *world models* can be incredibly useful to humans, guiding them towards performing optimal actions or learning how to do so.

Drawing inspiration from this kind of human mental simulation, Model-Based Reinforcement Learning (MBRL) [107, 74] approaches were proposed. Instead of directly letting the agent learn from the experience what the optimal behavior is, an internal *world model* (sometimes called *forward model* or just *model*) is constructed and used for planning or learning in the environment. A world model can be desirable for other properties than the sole sample efficiency. As in the case, for instance, of human intuitive knowledge of physics, an agent that can leverage a good approximation of the dynamics of the environment can transfer it across different tasks that have it

in common; or it can use the uncertainty of its own world model to know where to explore, in a similar manner with respect to human curiosity. Moreover, using a model of the world (or system) dynamics for control is deeply linked with the classical control literature and planning-based artificial intelligence: MBRL offers an interesting gateway for combining the machine learning approach with ideas from these other two, perhaps more mature, research fields.

The MBRL approach looks promising. However, *model-free* RL, that does not use any world model, is still the standard technique to solve many sequential decision making problems. This mainly happens because trying to model the dynamics of the environment in a thorough way can be extremely complex and, thus, require the use of very powerful model classes and considerable amounts of data, betraying the original goal of MBRL and often yielding unfavorable comparisons with model-free techniques. Fortunately, in many interesting application domains (e.g., robotics), perfectly modeling the dynamics across the whole state-action space is not necessary for a model to be effectively used by a learning agent [1, 75, 59].

Stemming from these ideas, an approach that is wiser than a pure brute-force learning of the complete dynamics consists in using simpler model classes, whose estimation requires few interactions with the environment, and focus their limited capacity on the most relevant parts of the environment. These parts could present a local dynamics that is inherently simpler than the global one, or at least easier to model using prior knowledge.

The vast majority of MBRL methods employs a maximum likelihood estimation process for learning the model [21]. In other words, the importance of any aspect of the dynamics of the world is only determined based on how much it can be observed in the available data. Nonetheless, this completely ignores the underlying decision problem, the control approach, and, importantly, the policy played by the agent. For instance, suppose a learning agent acts deterministically in a certain region of the environment, possibly thanks to some prior knowledge, and has no interest in changing its behavior in that area; or that some regions of the state space are extremely unlikely to be reached by an agent following the current policy. There would be no benefit in approximating the corresponding aspects of the dynamics, since that knowledge cannot contribute to the agent's learning process. Therefore, with a limited model expressiveness, an approach for learning the world model that explicitly accounts for the current policy and for how it will be improved can be more powerful than the traditional maximum likelihood estimation.

As an additional example, suppose we wish to update the controller of a robotic arm from an inefficient hand-crafted algorithm to one using MBRL. We can reuse previously collected data to learn the dynamics. However, many of such transitions could be very unlikely, at a given iteration, to be visited by the policy: instead, we would like to be able to automatically discriminate among useless and useful data, improving the learned policy.

The idea of capturing the underlying decision problem into the way a world model is learned is known as *decision-awareness* [29] and can be leveraged in different ways. The most direct approach consists in accounting for the performance of a policy that uses the learned model into the loss function used for learning the model itself [44, 7]. More sophisticated methods [76, 101, 29, 28] consider how the control policy is derived from the model. In particular, existing decision-aware approaches focus on *value-based* RL, in which the policy is obtained by using a value function for assessing how promising being in a given state is for the agent.

CONTRIBUTIONS

In this thesis, we focus our attention on direct *policy search*, proposing the first model-based policy search [21, 109] method that leverages awareness of the current agent's policy in the estimation of a forward model, used to perform policy optimization. Unlike existing approaches, which typically ignore all the knowledge available on the running policy during model estimation, we incorporate it into a weighting scheme for the objective function used for model learning. We focus on a widespread class of policy search methods, the ones based on the *policy gradient*, that uses gradient ascent for improving a control policy belonging to a space of differentiable and stochastic parametric policies. We choose to focus on this class of approaches because their main advantages, clearly stated in the rest of this thesis, make them amenable for a use in solving real world problems, for instance in the field of robotics.

The contributions of this thesis are theoretical, algorithmic and experimental. First, we provide a novel categorization of existing policy gradient approaches through the formalization of the concepts of *model-free gradient*, *fully-model-based gradient* and *model-value-based gradient*. We analyze the latter type of gradient, proving that model learning by maximum likelihood is not the best option when leveraging this type of approximation. Starting from this analysis, we derive a novel model-based policy improvement algorithm, that we name Gradient-Aware Model-based Policy Search (GAMPS), the first that incorporates information about the policy gradient into the objective used for model estimation. We analyze the algorithm on a theoretical level, formally confirming the effectiveness of a decision-aware approach to model-based reinforcement learning. Lastly, we compare against common baselines on benchmark domains both the performance of our approach for model learning and of the overall GAMPS algorithm, showing favorable results.

OVERVIEW

We now outline the general structure of the thesis. In Chapter 2, we formally define the reinforcement learning problem and the notion of Markov Decision Process. We first present a set of general mathematical tools used when dealing with sequential

decision making problems. Then, we outline a taxonomy of approaches, focusing on the classifications that are particularly useful for further analysis in the following chapters. We reserve particular attention to model-free policy gradient methods, presenting some of the details of the most common algorithms.

In Chapter 3, we review the state of the art in model-based reinforcement learning. We first give a high-level view of existing approaches by analyzing the most important decisions in the design of a model-based reinforcement learning algorithm, concerning the choice of the model class to be used, the approach taken for learning the model from data and the technique through which the agent derives a policy from the learned model. Then, we present model-based policy gradient approaches. In doing so, we employ an original categorization for the previous literature, categorizing existing approaches under the umbrella of two possible formulations of a model-based approximation for the policy gradient. We formalize these two approximations and give some insights about their advantages and drawbacks.

Chapter 4 is devolved to the derivation and analysis of our algorithm. We start by analyzing the error of one of the policy gradient approximations described in the previous chapter. Directly inspired by this analysis, that suggests the potential of a decision-aware loss function to be used in a model-based policy gradient approach, we derive and describe GAMPS. We break the presentation of GAMPS following the three main steps that occur at each one of its iterations: first, we derive a gradient-aware loss function to be used for model learning; then, we outline the available approaches for obtaining a value function starting from the learned model, in both discrete and continuous domains; lastly, we show how the computed value function can be used for estimating the policy gradient and improving the policy.

Chapter 5 contains an analysis of the empirical performance of the algorithm, that is compared against a model-based baseline that is not using gradient-awareness as well as against commonly-employed model-free approaches. First, we test the performance of both our gradient-aware loss function and the overall policy optimization algorithm on a small and discrete gridworld domain, that allows a precise analysis of the features of the approach (e.g., an accurate comparison between the real gradient and the one provided by the different algorithms). Then, we show the capabilities of GAMPS also in a continuous environment, namely a simulated minigolf domain.

In Chapter 6, we summarize the most important contributions of the thesis, together with the current limitations of the proposed approach. Then, we suggest some future work that can stem from our results.

The proofs of the results included in the main text of the thesis are reported in Appendix A. Additional details regarding the algorithm, as well as an alternative derivation for our gradient-aware loss function for model learning, are reported in Appendix B.

2

REINFORCEMENT LEARNING

In this chapter, we introduce the basics of reinforcement learning, formalizing the mathematical and algorithmic tools, and their related notation, that will serve as building blocks for the rest of the thesis. In Section 2.1, we introduce the formalism regarding the Markov Decision Process (MDP), a common framework used to model sequential decision making problems. In Section 2.2, we provide a general overview of commonly employed algorithms, while in Section 2.3 we focus on the specific family of model-free policy gradient approaches, whose details will be useful in the following chapters. Refer to [107, 85] for an extensive introduction to the field.

MARKOV DECISION PROCESSES

Markov Decision Processes [85] provide a mathematical framework that can be used to model a vast class of problems in which an agent acts with an environment with the aim of collecting as much reward as possible. At each time instant, the agent must decide, by observing the current state, what is the action that it wants to execute; this action will determine a *transition* in the environment, that, in the general case, is stochastic. A pictorial representation of this cycle of interactions is shown in Figure 2.1. The fundamental property of any MDP, hence contained in its name, is the *Markov property*: the transition from a state of the environment to another does not depend on what the previous states were but only on the current state and executed action. This property can appear to be rather limiting, since in many interesting problems,

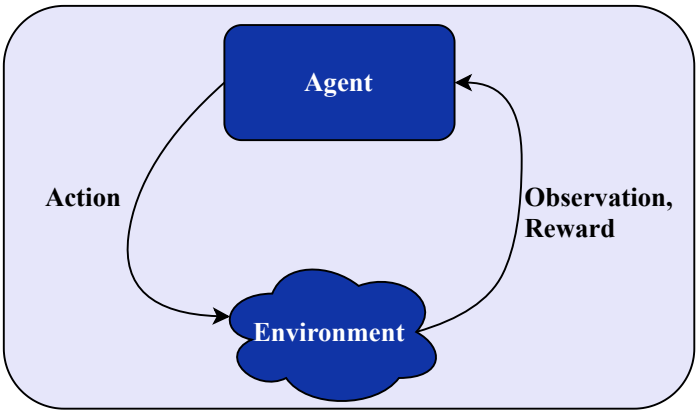

Figure 2.1: Scheme representing the interactions occurring in a MDP.

it is difficult to imagine the environment to be memory-less. However, the Markov property is a formally useful assumption that can always hold, provided a redefinition of the states that considers all the possible histories. For the rest of the thesis, we assume, for simplicity of presentation, that the environment dynamics cannot change over time (i.e., we make a *stationarity assumption*) and that the agent can observe a complete state of the environment during its interaction (i.e., we make a *complete observability assumption*).

Formally, a discrete-time MDP [85] is described by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \mu, \gamma)$, where:

- $\mathcal{S}$ is the space of possible states.
- $\mathcal{A}$ is the space of possible actions.
- $r : \mathcal{S} \times \mathcal{A} \to [-R_{max}, R_{max}]$ is the reward received by executing action a in state s, uniformly bounded by $R_{max} > 0$.
- $p(\cdot|s, a)$ is the transition model, providing the distribution of the next state when performing action a in state s.
- $\mu(s_0)$ is the distribution of the initial state.
- $\gamma \in [0, 1)$ is a discount factor, that penalizes rewards that are later in time.

The behavior of an agent is described by a policy $\pi(\cdot|s)$ that provides the distribution over the action space for every state s. In general, p is assumed to be unknown. Instead, the knowledge of r depends on the domain: while it is sometimes impossible to infer, in other cases (e.g., when learning is applied to robotics [20, 59]) a direct representation of the performance of an agent according to the preferences of who is implementing a control system is available. Think for instance to the case of a robotic arm that must reach a target position: in that case, the reward given to an agent that is controlling the arm can simply be the known distance from the target. An agent interacting with the environment generates a set of *trajectories* of the form $\tau = (s_0, a_0, s_1, a_1, ..., s_T, a_T)$. If we consider an *episodic* setting, in which the agent interacts with the environment during finite-length runs, a trajectory can correspond to a whole episode. When required for simplicity of presentation, we can consider the maximum length of an episode H, the *horizon*. Moreover, it is useful to consider the distribution over the trajectories, characterized by a probability density function $\zeta_\mu^{\pi,p}(\tau)$, defined, thanks to the Markov property, by factorization into single-steps transition probabilities:

$$\zeta_\mu^{\pi,p}(\tau) = \mu(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t). \tag{2.1}$$

In describing an MDP, it is useful to quantify how good is for the agent to be in a given state or execute a certain action. With this aim, the action-value function

$Q^{\pi,p} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides a way to express the cumulative expected reward that follows the execution of action a in state s, when policy π is followed in an MDP with underlying model p. The action-value function, or Q-function [107], can be recursively defining by means of the *Bellman equation*, as:

$$Q^{\pi,p}(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} p(s'|s,a) \int_{\mathcal{A}} \pi(a'|s') Q^{\pi,p}(s',a') ds' da'. \tag{2.2}$$

This definition is rooted in *dynamic programming* [10]. Similarly, the state-value function, or V-function, can be defined starting from it as:

$$V^{\pi,p}(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[Q^{\pi,p}(s,a)]. \tag{2.3}$$

The goal of the agent is to find an optimal policy π^*, i.e., a policy that maximizes the overall *expected return*:

$$J^{\pi,p} = \mathbb{E}_{s_0\sim\mu}\left[V^{\pi,p}(s_0)\right]. \tag{2.4}$$

In the definition of all these quantities, we made explicit the dependency on both the policy π and the model p. This will be useful in following chapters: $V^{\pi,\overline{p}}(s)$ corresponds, for instance, to the expected cumulative reward obtained starting from state s in an MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, r, \overline{p}, \mu, \gamma)$, and following policy π.

We can further define two useful distributions. The first one is the *state distribution* [109] under policy π and model p, defined as:

$$d_\mu^{\pi,p}(s) = (1-\gamma)\sum_{t=0}^{+\infty} \gamma^t \Pr(s_t = s|\mathcal{M},\pi). \tag{2.5}$$

In an equivalent way, we can define the state-action distribution $\delta_\mu^{\pi,p}$ as:

$$\delta_\mu^{\pi,p}(s,a) = (1-\gamma)\sum_{t=0}^{+\infty} \gamma^t \Pr(s_t = s, a_t = a|\mathcal{M},\pi). \tag{2.6}$$

These distributions model the probability of incurring in the different states and actions when policy π is run into the MDP. Note that $\delta_\mu^{\pi,p}(s,a) = \pi(a|s) d_\mu^{\pi,p}(s)$. We can now rewrite in a different, easily manageable, way the performance of the policy:

$$J^{\pi,p} = \frac{1}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) r(s,a) ds da. \tag{2.7}$$

When $\mathcal{S}$ and $\mathcal{A}$ are finite and small sets, a simple tabular representation can be used for learning value functions and policies. When this case, commonly defined as *tabular scenario*, is not applicable, V, Q, π and any other function of interest must be modeled through the use of appropriate function approximators. Common choices are linear models, for which more theoretical guarantees are available [107], or neural networks [94, 58], whose potentially huge number of parameters, together with compositionality assumptions, yield high representation capabilities.

REINFORCEMENT LEARNING ALGORITHMS

In this section, we present a non-exhaustive classification of the different approaches to solve RL problems that have been studied in the literature. The taxonomy we will present is focused on the categories that are relevant in the further discussion contained in the rest of this work. In particular, we will discuss about the distinction between model-free and model-based, value-based and policy-based, on-policy and off-policy approaches. Algorithms combining different choices among the three categories exist. Moreover, the three categorizations are not dichotomies: there exist algorithms that combine elements from both approaches for each of them.

Model-free and Model-based

A first high-level classification of RL methods can be done considering whether an estimated model of the dynamics of the environment is used or not. The first type of approaches, commonly known as *model-free*, learns a policy π without explicitly considering the underlying dynamics of the environment in which the agent is acting. Model-Based Reinforcement Learning (MBRL) approaches, instead, employ an internal *world model* $\widehat{p}$ for choosing how to act [93, 105]. Several strategies exist for obtaining a policy from an estimated model: a discussion and analysis of some of them will be the topic of the following chapters.

These two types of approaches are related to inherent aspects of human intelligence. Intuitively, following a parallel with behavioral science [46], the model-free approach is similar to instinctive *fast-thinking*, while model-based methods are more related to complex reasoning, *slow-thinking*. Recent studies supported this connections, suggesting that both the systems cohabit in the human brain [24] or employing ideas from MBRL as a thorough model for human creativity [95].

Value-based and Policy-based

Another important categorization of existing RL algorithms concerns how the policy is obtained. *Value-based* methods use the interaction with the environment to learn a value-function, from which a policy is derived. This can be done, for instance, in a *greedy* way, by setting

$$\pi(\cdot|s) = \arg\max_{a \in \mathcal{A}} Q(s, a). \tag{2.8}$$

Policy-based approaches search, instead, directly in the space of possible policies. Although value-based approaches are often more effective in some domains (e.g., [70]), learning an explicit policy is more practical, mainly because:

- prior knowledge can be integrated in an easier way;

- for many parameterization choices (e.g., neural networks), small perturbations to the state result in small changes in the probabilities of actions, a fact that is not true for value-based approaches, where instead a small state perturbation can yield the execution by the policy of a completely different action;
- the policy can flexibly have different degrees of stochasticity;
- for some problems, directly modeling the optimal behavior can be substantially easier than modeling the value of each state, for instance, if an agent is driving a car that is going towards a wall, knowing that the optimal action is to steer away is more direct than trying to infer the action from the value of being in that state.

On-policy and Off-policy

In a RL method, the policy that is improved by the algorithm, called *target policy* π, is not necessarily the same as the one used for collecting the data, referred to as *behavior policy* π_b. When the target and behavior policy coincides, the approach is said to be *on-policy*, while an algorithm that learns to improve a policy by using data collected by means of a different behavior is said to be *off-policy*. There exist both value-based [120, 72] and policy-based [98, 62] off-policy methods. Learning off-policy is potentially more data-efficient than a strict on-policy approach, that must actually forget all previous interactions with the environment every time the policy is improved.

Pushing to the limit off-policy learning, we obtain a setting commonly known as *batch reinforcement learning* [56, 27]. In this setting, π_b generates a dataset

$$\mathcal{D} = \{\tau^i\}_{i=1}^N = \left\{\left(s_0^i, a_0^i, s_1^i, a_1^i, ..., s_{T_i-1}^i, a_{T_i-1}^i, s_{T_i}^i\right)\right\}_{i=1}^N \tag{2.9}$$

of N independent trajectories τ^i, each composed of T_i transitions, and further interactions with the environment are not allowed.

In *off-policy* estimation [84], we have to take into account that policy π is different from the policy π_b that generated the data. To correct for the distribution mismatch, a common solution is to employ *importance sampling* [45, 79], re-weighing the transitions based on the probability of being observed under the policy π. Namely, we can define the importance weight relative to a subtrajectory $\tau_{t':t''}$ of τ, occurring from time t' to t'', and to policies π and π_b as:

$$\begin{aligned}\rho_{\pi/\pi_b}(\tau_{t':t''}) &= \frac{\zeta_\mu^{\pi,p}(\tau_{t':t''})}{\zeta_\mu^{\pi_b,p}(\tau_{t':t''})} = \frac{\prod_{t=t'}^{t''-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)}{\prod_{t=t'}^{t''-1} p(s_{t+1}|s_t, a_t)\pi_b(a_t|s_t)} \\ &= \prod_{t=t'}^{t''-1} \frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}.\end{aligned} \tag{2.10}$$

Note that the importance weight can be computed without knowledge of the transition model p, since the ratios of probabilities depending on it can be simplified.

Importance sampling can provide an unbiased estimate for the quantities of interest to be computed for policy π, using the data sampled from π_b.

In the general case, we wish to estimate the expected value of a function f under a probability distribution P by drawing N samples from a probability distribution Q, both defined on a measurable space. Considering a characterization provided by, respectively, two density functions p and q, the importance sampling estimator is defined as:

$$\widehat{\mu}_{P/Q} = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i) = \frac{1}{N}\sum_{i=1}^{N}\rho_{P/Q}(x_i)f(x_i). \tag{2.11}$$

The assumption required for using this estimator is that $P \ll Q$, P is *absolutely continuous* w.r.t. Q. Formally, this means that there exist a function f such that $dP = f dQ$. In other words, we require that, whenever $q(x_i) = 0$, also $p(x_i) = 0$.

The price to pay for using an importance sampling estimator instead of directly sampling from P is an increase in variance. Intuitively, this price becomes larger as the mismatch between P and Q increases. In [66], an upper bound on the variance of the estimator is provided, by quantifying the dissimilarity between P and Q in terms of the exponentiated *Rényi divergence* [87, 116] $d_\alpha(P\|Q) = \exp\left(D_\alpha(P\|Q)\right)$, where:

$$D_\alpha(P\|Q) = \frac{1}{\alpha-1}\log\int_{\mathcal{X}}\left(\frac{dP}{dQ}\right)^{\alpha} dQ = \frac{1}{\alpha-1}\log\int_{\mathcal{X}} q(x)\left(\frac{p(x)}{q(x)}\right)^{\alpha} dx. \tag{2.12}$$

We restate here the result for completeness.

Lemma 2.1 (Variance of the importance sampling estimator [66])**.** *Let P and Q be two probability measures on the measurable space $(\mathcal{X}, \mathcal{F})$ such that $P \ll Q$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ i.i.d. random variables sampled from Q and $f : \mathcal{X} \to \mathbb{R}$ be a bounded function ($\|f\|_\infty < +\infty$). Then, for any $N > 0$, the variance of the importance sampling estimator $\widehat{\mu}_{P/Q}$ can be upper bounded as:*

$$\underset{\mathbf{x}\sim Q}{\mathbb{V}\mathrm{ar}}\left[\widehat{\mu}_{P/Q}\right] \leqslant \frac{1}{N}\|f\|_\infty^2 d_2\left(P\|Q\right). \tag{2.13}$$

This upper bound suggests that the variance is smaller when more samples are collected, the function f whose expected value is being estimated is bounded by a small constant or P is very similar to Q.

In the context of reinforcement learning, this implies that we should trust less the estimation provided by importance sampling when the behavior policy π_b is very different from π (i.e., they induce two trajectory distributions with high $d_2(\cdot\|\cdot)$).

MODEL-FREE POLICY GRADIENT

In this section, we present an overview of fundamental model-free policy gradient techniques. This will build the foundations for the presentation of the model-based policy search approaches, that are treated in the following chapter.

Policy Gradient

An important class of policy-based RL algorithms are the one based on the *policy gradient*. The general idea behind these approaches is to consider a parametric space of policies $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta \subseteq \mathbb{R}^d\}$ and to learn a policy π_θ by following improvement directions in this space. Generally, stochastic policies are considered, in order to naturally induce exploration. The policy is improved by using a gradient ascent approach, assuming that the policy is differentiable w.r.t. its parameters θ. Starting from some estimate of the gradient of the performance of a policy $\widehat{\nabla}_\theta J(\theta)$, an improvement step of the form:

$$\theta \leftarrow \theta + \alpha \widehat{\nabla}_\theta J(\theta) \tag{2.14}$$

is repeatedly applied. α is a non-negative step size, that determines how much the parameters are changed at each update. The pseudocode for a general policy gradient algorithm is illustrated in Algorithm 2.3.1.

Algorithm 2.3.1: Policy Gradient meta-algorithm

Input: Initial policy parameters θ, step size α

while stopping condition not reached **do**
  obtain an exploratory policy π_b from π_θ (e.g., $\pi_b \leftarrow \pi_\theta$)
  collect trajectories with policy π_b
  obtain an estimate $\widehat{\nabla}_\theta J(\theta)$ of the policy gradient
  $\theta \leftarrow \theta + \alpha \widehat{\nabla}_\theta J(\theta)$
end while

In each iteration of this meta-algorithm, a behavior policy π_b is obtained and possibly run in the environment for collecting trajectories. The exploratory policy π_b to be used in a given iteration can be retrieved in several ways. Perhaps the most common choice is to set $\pi_b = \pi_\theta$, thus leaving the whole responsibility of exploration to the stochasticity of policy π_θ. Another alternative, when the action space $\mathcal{A}$ is discrete, is to use a uniformly random policy π_r for collecting the data, or a mixture $\pi_b = \alpha\pi_r + (1-\alpha)\pi_\theta$ between the random and the current policy (using any $\alpha \in [0, 1]$). Approaches [62] that learn a deterministic policy $\pi_\theta(s)$ usually obtain the behavior as $\pi_b = \pi(s) + \mathcal{N}$, by perturbation of the current policy by means of noise, for instance *white* or obtained from an *Ornstein-Uhlenbeck process* [114] to alleviate low-pass filter effects over the control variables in robotics.

In the second step of Algorithm 2.3.1, an arbitrary number of trajectories is obtained by running π_b in the environment. The step as a whole can be skipped in some of the iterations for off-policy policy gradient algorithms. In batch approaches, the

step is actually executed only for the first iteration and all the policy improvements of subsequent iterations are done by starting from that initial batch of collected experience.

After enough data has been collected, an estimate for the policy gradient is computed, for instance with a method described in this section or in Section 3.5, and the policy is updated. The procedure is repeated until a stopping condition is reached, usually determined by the convergence to a local optimum (i.e., $\|\widehat{\nabla}_\theta J(\theta)\| \simeq 0$).

The Policy Gradient Theorem

The Policy Gradient Theorem (PGT) [109] provides an expression connecting the gradient of the performance of a policy with the value function. We present the theorem in a form that is more adapt to the following discussion, obtained starting from the original formulation by using a substitution known as the log-derivative trick [121] (i.e., by trivially observing that $\nabla_\theta \pi(a|s) = \pi(a|s)\nabla_\theta \log \pi(a|s)$). We now state and comment the result.

Theorem 2.2 (Policy Gradient Theorem [109])**.** *In a Markov Decision Process, the gradient of the policy performance w.r.t. to the parameters of a policy π_θ can be written as:*

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) \nabla_\theta \log \pi(a|s) Q^{\pi,p}(s,a) ds da.$$

The factor $\nabla_\theta \log \pi(a|s)$ is called *score function*. It is connected to the possibility for each one of the parameters in θ to be changed by a policy update: if a state-action pair is well-visited and has an high score in a component of a θ, the gradient will point in the parameter space to the direction favoring an increase of the probabilities of the actions that lead to greater values of the Q-function. Both $\delta_\mu^{\pi,p}(s,a)$ and $Q^{\pi,p}$ are, in general, unknown quantities. Model-Free Gradient (MFG) approaches use the direct interaction with the environment for obtaining a Monte Carlo estimate for both of them.

An estimator for the gradient in the form provided by Theorem 2.2 can have large variance. One of the sources of this variance is rather clear: if the Q-function is always positive, the gradient will suggest to increase the probabilities of all the actions, even the ones that are very bad compared to the others. A common way to solve this issue is the use of a control variate called *baseline* $b(s)$, a quantity to be subtracted from the estimate for the Q-function. In this case, the policy gradient can be rewritten as:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) \nabla_\theta \log \pi(a|s) \left(Q^{\pi,p}(s,a) - b(s)\right) ds da. \tag{2.15}$$

For the PGT to still hold, $b(s)$ must not depend on the action. In fact:

$$\begin{aligned}\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}[\nabla_\theta \log \pi(a|s)b(s)] &= \mathop{\mathbb{E}}_{s\sim d_\mu^{\pi,p}}[\pi(a|s)\nabla_\theta \log \pi(a|s)b(s)] \\ &= \mathop{\mathbb{E}}_{s\sim d_\mu^{\pi,p}}[\nabla_\theta \pi(a|s)b(s)] \\ &= \int_{\mathcal{S}} d_\mu^{\pi,p}(s)b(s)\nabla_\theta \int_{\mathcal{A}} \pi(a|s)da\,ds \\ &= \int_{\mathcal{S}} d_\mu^{\pi,p}(s)b(s)\nabla_\theta 1\,ds = 0.\end{aligned} \tag{2.16}$$

Thus, no bias is introduced, and a Monte Carlo estimator that uses a baseline remains unbiased.

The score function

It is useful to consider the expression of the score function for commonly employed policy classes. When the action space $\mathcal{A}$ is discrete, a *Boltzmann* (or softmax) policy is a common choice. Given $\phi(s,a)$, some feature representation for a state-action pair, the probability of an action is computed as:

$$\pi(a|s) = \frac{e^{\Phi(s,a)}}{\sum_{\overline{a}\in\mathcal{A}} e^{\theta^T \Phi(s,\overline{a})}}. \tag{2.17}$$

The corresponding score for this class of parameterized policy is given by:

$$\nabla_\theta \log \pi_\theta(a|s) = \Phi(s,a) - \sum_{\overline{a}\in\mathcal{A}} \pi_\theta(\overline{a}|s)\Phi(s,\overline{a}). \tag{2.18}$$

Boltzmann policies can only be used for discrete action spaces. In the general case of continuous $\mathcal{S}$ and $\mathcal{A}$, a common choice is to use Gaussian policies. In particular, when a small parameterization is enough, policy linear in a representation of the state are usually considered. For instance, considering a parameterization for the mean of the policy:

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{a-\theta^T\Phi(s)}{\sigma}\right)^2\right). \tag{2.19}$$

In this case, the score is given by:

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{\left(a-\theta^T\Phi(s)\right)\Phi(s)}{\sigma^2} \tag{2.20}$$

An interesting result about the score is highlighted in the following proposition.

Proposition 2.3. *The expected value of the score in any state $s \in \mathcal{S}$ is:*

$$\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[\nabla_\theta \log \pi_\theta(a|s)] = 0. \tag{2.21}$$

Of course, the variance of the score is instead in general different from zero.

REINFORCE

REINFORCE [121] is perhaps the first MFG algorithm based on the likelihood ratio that was proposed in the RL literature. The underlying idea is simple: in an episodic setting, the Q-function can be estimated by looking at the return obtained by running the policy in the environment. REINFORCE uses the cumulative return from the start to the end of the episode as an estimate for the Q-function, resulting, considering N trajectories indexed by i, in the following estimator:

$$\widehat{\nabla}_\theta^{MFG} J^{RF}(\theta) = \left\langle \sum_{t=0}^{H-1} \nabla_\theta \log \pi(a_t^i | s_t^i) \left(\sum_{l=0}^{H-1} \gamma^l r(s_l^i, a_l^i) \right) \right\rangle_N , \tag{2.22}$$

where we indicate by $\langle \cdot \rangle_N$ an average over trajectories and we use an apex to highlight that it is an estimator for the MFG.

The name of the algorithm derives from the behavioral psychology inspiration: the agent must increase the probability of, or *reinforce*, the actions that lead to a greater cumulative reward. An optimal baseline w.r.t. the minimization of the variance is available thanks to [82], that can be estimated using the following expression:

$$b^{RF} = \frac{\left\langle \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi(a_t^i | s_t^i) \right)^2 \left(\sum_{l=0}^{H-1} \gamma^l r(s_t^i, a_t^i) \right) \right\rangle_N}{\left\langle \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi(a_t^i | s_t^i) \right)^2 \right\rangle_N}. \tag{2.23}$$

Improvements over REINFORCE

The basic algorithm we presented in Section 2.3.4 considers the cumulative reward collected during a whole episode when reinforcing any action. However, since the agent is acting in a world rooted in causal relationships, this is not sound, and pointlessly increases the variance of the estimation: the fact that an action can lead to potentially good consequences (i.e., high reward) must be inferred only from the rewards that came after it. This can be also easily seen in the use of the Q-function in the expression of the policy gradient provided by Theorem 2.2: the value function only deals with future rewards and rewards collected in the time instants that precede a particular action are irrelevant for the policy gradient, only leading to increased variance if used in an estimator.

Therefore, a more reasonable model-free estimate for $Q(s_t^i, a_t^i)$ is the *reward-to-go* of the state-action pair, consisting in the discounted cumulative reward collected starting from time instant t. This directly derives from Theorem 2.2, and leads to the following

PGT estimator, which has in most cases less variance [122] than the one provided by REINFORCE:

$$\widehat{\nabla}^{MFG}_{\theta} J^{PGT}(\theta) = \left\langle \sum_{t=0}^{H-1} \nabla_\theta \log \pi(a^i_t|s^i_t) \left(\sum_{l=t}^{H-1} \gamma^l r(s^i_l, a^i_l) \right) \right\rangle_N . \tag{2.24}$$

In a way that is equivalent from an estimation perspective but with a different algorithmic formulation [82], we can also define the G(PO)MDP [9] estimator:

$$\widehat{\nabla}^{MFG}_{\theta} J^{G(PO)MDP}(\theta) = \left\langle \sum_{t=0}^{H-1} \left(\sum_{l=0}^{H-1} \nabla_\theta \log \pi(a^i_l|s^i_l) \right) \gamma^l r(s^i_t, a^i_t) \right\rangle_N . \tag{2.25}$$

As in the case of REINFORCE, also the PGT estimator admits an optimal baseline [82], that is this time step-dependent. It can be computed using the following estimator:

$$b^{PGT}_l = \frac{\left\langle \left(\sum_{l=0}^{t} \nabla_\theta \log \pi(a^i_l|s^i_l) \right)^2 \gamma^l r(s^i_t, a^i_t) \right\rangle_N}{\left\langle \left(\sum_{l=0}^{t} \nabla_\theta \log \pi(a^i_l|s^i_l) \right)^2 \right\rangle_N}. \tag{2.26}$$

Actor-Critic Methods

In actor-critic methods [53], a parameterized policy π_θ (the *actor*) is learned together with a parameterized value function $\widehat{V}_\omega$ or $\widehat{Q}_\omega$ (the *critic*). In some sense, these techniques are on the border between the value-based and the policy-based worlds: learning the value function is instrumental for estimating the policy gradient, then used as in actor-only methods to improve the policy by gradient ascent. The difference between actor-only and actor-critic methods is not completely neat, since an estimate of the value function can be used as a valid baseline in methods such as REINFORCE. The rationale of [107] is to refer to actor-critic methods as the ones using some form of *bootstrapping* [106] in learning the value function. When using bootstrapping, the estimate of the value function is updated partially based on a previous estimate. For instance, if we consider only one step, actor and critic can be updated based on *temporal difference* error computed on a transition (s_t, a_t, r_t, s_{t+1}), namely:

$$\delta_t \leftarrow r(s_t, a_t) + \gamma \widehat{V}_\omega(s_{t+1}) - \widehat{V}_\omega(s_t). \tag{2.27}$$

This quantity can then be used for updating the value function:

$$\omega \leftarrow \omega + \beta \gamma^t \delta_t \nabla_\omega \widehat{V}_\omega(s_t), \tag{2.28}$$

with a learning rate $\beta \geqslant 0$ and the policy:

$$\theta \leftarrow \theta + \alpha \gamma^t \delta_t \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{2.29}$$

This basic scheme is at the root of most recent actor-critic algorithms, subject to various extensions. For instance, *deep deterministic policy gradient* [62] modifies it to learn a deterministic policy while acting with a stochastic one; *Asynchronous Advantage Actor-Critic* [69] adapts the basic actor-critic procedure to a heavily-parallel setting; *Soft-Actor-Critic* [40] encourages exploration by incorporating an entropy regularization term into the objective. For a detailed survey of more traditional actor-critic techniques refer to [38].

3

MODEL-BASED REINFORCEMENT LEARNING

In this chapter, we review the state of the art of model-based reinforcement learning. First, in Section 3.1, we present a general overview of MBRL. Then, we discuss three central issues in MBRL, concerning the choice of the model class to be used for approximating the dynamics (Section 3.2), the loss function to be minimized for obtaining the estimated model (Section 3.3) and the different techniques for making use of the learned model for improving or running a control policy (Section 3.4). Lastly, in Section 3.5, we focus our attention on model-based policy gradient approaches, outlining a taxonomy that will be useful in the further discussion of Chapter 4.

OVERVIEW

Motivation

In recent years, the RL research community was able to obtain outstanding results. Superhuman level was reached by novel algorithms in tasks that are generally judged challenging, such as playing a vast range of diverse videogames [70, 117], or even ancestrally complex, like excelling at the game of Go [99]. However, even if the available hardware seems to be ready for it [31], current RL algorithms struggle in performing very basic sensorimotor-intensive tasks. Learning interactions that are trivial even for the smallest human child is incredibly hard for most current algorithms; on the other hand, performing tasks that seem impossible for most humans (e.g., playing Go or Chess at top level) is now possible to be done by an algorithm using reasonable computational resources. This counterintuitive fact is commonly known as the *Moravec's paradox* [71].

The Moravec's paradox is due to evolutionary reasons: the sensorimotor intelligence of human beings had about 3.7 billions years [77] to develop to the final form that is known to us, but the rational intelligence required to perform tasks such as playing complex games is no more than 300.000 years old [91]. Therefore, it is fairly easier to engineer a system that overtakes the performance of a simpler mechanism. However, those very basic, sensorimotor-related tasks are among the most useful in the real world, thinking, for instance, to industrial or daily applications.

Model-based reinforcement learning offers a powerful solution for alleviating the effects that contribute to the existence of the Moravec's paradox. It has been empirically studied by neuroscientists and behavioral psychologists that humans make extensive use of an internal *world model* to run mental simulations or to predict the effects of

their own actions [48, 32, 8, 41]. The attitude to learn and use an internal model of the dynamics of the environment was thus encoded into humans by evolution: artificial agents using MBRL try to apply the same principle, using an estimated model to decide how to act.

The promise of MBRL is to address important issues affecting model-free reinforcement learning. Namely, among the most important advantages of using a model of the dynamics of the environment inside an RL algorithm, we can count:

- *Sample-efficiency*. For many problems, learning a model to be used for policy improvement is easier than directly learning a policy [5]. A reduced amount of experience can be therefore used by an agent to obtain satisfying performance.
- *Easier transfer*. Once it has learned to perform a task, it is easier for an agent to learn to perform a related task. This comes naturally from the fact that, if the environment dynamics remains similar, but the task has changed (e.g., the reward function was modified), an agent can still reuse the same world model.
- *More effective exploration*. Estimating a model of the dynamics can be useful for knowing where to explore. For instance, an agent could try to explore regions of the environment where its own internal world model is more uncertain.
- *Safety*. A world model that is accurate enough can be used for *off-policy policy evaluation* [107], estimating the performance and understanding the behavior of a policy without running it into the actual environment.

A definition for MBRL

A precise definition of MBRL is not easy to give. In the most general case, a MBRL approach is any RL approach that uses a model of the dynamics of the environment in the learning or control algorithm. This definition comprises both the approaches that use either the real model of the environment or an estimate of it. The former class requires that the model of the environment in its exact form is available. For instance, for a board game such as Go or Chess, the model would be just the complete set of its rules; for a generic dynamical system, the model can be an extremely accurate writeup of the physical laws that determine its evolution. Approaches based on this precious, albeit often unaccessible, kind of knowledge are closely related to state-space search [99, 100, 97] or classical control [60].

For the rest of the thesis, we will focus on the second type of approaches, that create an internal world model by observing the dynamics of the environment and, in general, we refer just to them when using the term MBRL. More precisely, an MBRL algorithm wants to find an approximately optimal policy $\pi \approx \pi^*$ through the use of a learned model of the dynamics $\widehat{p} \approx p$.

Three important questions are central in MBRL and help in categorizing existing approaches:

1. Which is an appropriate model class to be used for representing the dynamics?
2. Which loss function should be minimized for learning the model?
3. How can the learned model be used by the agent?

These three questions will be the subject of the discussion in the following sections.

For simplicity of presentation, prior to answering these questions, we provide a general overview on recent generative models, the basic tool used for obtaining a world model, both discussing model classes and loss functions. Next, we will focus the discussion to RL.

Overview on modern generative models

The dynamics of an environment are, in the general case, stochastic. Therefore, the problem of approximating the probability distribution implied by p is a *density estimation* problem [102]. Density estimation is performed with the use of *generative models*, either in an explicit way, for instance by defining a parameterized estimated density $\hat{p}_\phi$, or implicitly, by being able to sample from $\hat{p}$. A common approach consists in maximizing the log-likelihood of the data, or, equivalently, to minimize $D_{KL}(p\|\hat{p})$, the Kullback–Leibler divergence between the real and the estimated distributions. Nonetheless, this objective is usually intractable and simplifying assumptions are needed.

There are three main categories of likelihood-based methods that explicitly learn $\hat{p}(\mathbf{x})$: *fully observed*, *flow* and *variational* methods. *Fully observed* models do not employ any latent variables to explain the hidden factors of the data. Two examples are *Gaussian processes* [86], that model functions as collections of normally distributed random variables, and *autoregressive approaches*, that try to make the problem tractable using a factorization of the kind $\hat{p}(\mathbf{x}) = \prod_{t=0}^{T} p(x_t|x_0, ..., x_{t-1})$, corresponding to the generation, for each sample, of a single feature at a time [37, 78, 115]. *Flow methods* leverage invertible transformations of random variables [22, 23, 51]. *Variational methods* [52, 88] try to learn an *approximate density* by maximizing a computable surrogate objective $\mathcal{L}(\mathbf{x};\theta)$ subject to the *evidence lower bound* $\mathcal{L}(\mathbf{x};\theta) \leqslant \log \hat{p}(\mathbf{x};\theta)$; a common approach is to train a *variational autoencoder* (VAE), using $-\mathcal{L}(\mathbf{x};\theta)$ as a loss function. A VAE is composed of an *encoder* that maps the input data into a latent representation and a *decoder*, able both to reconstruct the encoded data and to generate new samples, decoding points sampled from the prior distribution of the latent variables. *Generative adversarial networks* (GANs) are *implicit density models* that provide a way to sample from $\hat{p}(\mathbf{x})$: in GAN training, we look for a *Nash equilibrium* in the game between a generator G, producing samples resembling the real ones, and

a discriminator D, which learns to distinguish the samples drawn from the dataset from the ones produced by G. Several value functions were devised for this game, each of which leads to an equivalent minimization of a divergence between $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$ (e.g., Jensen-Shannon [36], Wasserstein [3], Pearson χ^2 [64]). Refer to [35] for a more exhaustive taxonomy of recent generative models.

WHICH MODEL CLASS TO USE

The choice of a model class $\mathcal{P}$ is the first, important, step for any model-based approach. We now briefly discuss the critical aspects relevant for the RL problem, mentioning which are the approaches that are commonly employed.

Dealing with uncertainty

A fundamental problem in MBRL is called *model bias*. An implicit assumption of many methods is that the learned world model accurately reflects the real dynamics of the environment. However, with limited data, this assumption can be very problematic, since the bias introduced by the use of an imperfect world model can severely hurt the performance of the resulting policy. A solution for alleviating model bias is to consider *uncertainty* about the prediction of the dynamics, by means of different model classes and techniques. Intuitively, a model-based algorithm should rely less upon its world model when the uncertainty of its prediction is higher.

A quite effective approach has been proposed within the framework named *probabilistic inference for learning control* (PILCO) [20]. The basic idea is to leverage the fact that Gaussian processes are able to output the uncertainty about their predictions. The approach has demonstrated good empirical performance, albeit the intrinsic limitations of Gaussian processes, a non-parametric method that struggles in scaling to high-dimensional state spaces, limit its applicability.

Another alternative for incorporating uncertainty estimation into an approximate forward model is to use *Bayesian neural networks* [33, 63]. Bayesian neural networks are able to augment traditional neural networks with the possibility of obtaining a degree of uncertainty from model predictions, by modeling a distribution over the parameters of the network instead of a simple point estimate. Perhaps one of the most straightforward approaches, proposed in [33], consists in the use of the dropout regularization technique [103], which allows to perform approximate inference by running multiple forward passes of the same model. These techniques were originally used for MBRL by integration into the PILCO framework [34], but have been also employed in different scenarios [49, 17].

A third viable alternative for estimating uncertainty in MBRL consists in the use of ensembles of forward models. For instance, multiple models can be part of an ensemble obtained by *bagging* [13], where each model is fitted to slightly different sets

of environmental transitions, created by sampling with reinsertion from the original dataset of transitions. Intuitively, if all models agree on the prediction of the next state, then the uncertainty is low; if, instead, the members of the ensemble give very different predictions, uncertainty is high. This type of estimation has been used in MBRL both for control [55, 14] and to improve exploration [96].

Single-step and multi-step dynamics modeling

Another core issue in MBRL is the model's *compounding error*. For a model to be particularly useful for an agent, it must be able to predict the long-term future. However, if a model $\widehat{p}(\cdot|s, a)$ is used for estimating the dynamics of the environment, the only option to be used for predicting the effects of the actions of an agent over multiple future time steps is to unroll this prediction several times. This compounds the error that the model commits on average in a single time step in a way that is exponential w.r.t. to the planning horizon. Recently, leveraging recent progress in generative modeling led by deep learning techniques, it has been proposed as a solution to instead model the probability distribution $\zeta_\mu^{\pi,p}$ of whole trajectories. In particular, existing approaches [68, 18] use trajectory-level world models based on variational autoencoders for control. Naturally, the benefits of modeling whole trajectories come with some disadvantages, namely:

- There is a reduced amount of data to be used for training the world model because, obviously, there are way more transitions than trajectories.
- For many problems, modeling $\zeta_\mu^{\pi,p}$ can be significantly harder than modeling $p(\cdot|s, a)$.

For these reasons, single-step forward models are still the most common approach in MBRL.

Locally-accurate models

The dynamics of an arbitrary environment can be extremely complex. Therefore, it is easy to think that very powerful model classes are required for a MBRL approach to work in an effective way. Nonetheless, the estimation of a high-capacity model requires considerable amounts of data, betraying the promise of sample-efficiency of MBRL.

Fortunately, in many interesting application domains (e.g., robotics), perfectly modeling the dynamics across the whole state-action space is not necessary for a model to be effectively used by an agent. This intuition is the foundation of the classical control technique of linearizing a dynamical system [67, 61]: in order to control a complex nonlinear system around an equilibrium point, it can be sufficient to use a local linear approximation of its dynamics.

It is reasonable, in MBRL, to use simple model classes and focus their limited capacity only on some parts of the environment. This reduces the amount of data needed for learning the model, in the hope that the bits of the dynamics that are better approximated are the ones that are more relevant for control purposes.

For instance, the approach proposed in [1] improves a policy using a model that is constrained to be fully consistent with the trajectories observed while acting in the environment. This is done by assuming a deterministic environment and using a parametric model class. Model updates for the estimated deterministic forward model f have the following form:

$$f^{k+1}(s_t, a_t) \leftarrow f^k(s_t, a_t) + s^i_{t+1} - f^k(s^i_t, a^i_t). \tag{3.1}$$

The model f thus perfectly predicts observed trajectories, eliminating a source of error, despite being, of course, only a local approximation.

Other approaches use multiple local models in order to overcome the scalability limits of Gaussian processes [75]. In [59], inspired by work in trajectory optimization [111], time-varying linear dynamics models are used together with a guided policy search approach to learn a control policy: the authors show that such simple models are sufficient to solve contact-rich robotics tasks that have discontinuous dynamics.

HOW TO LEARN THE MODEL

Decision-unaware MBRL

Considering the traditional Maximum Likelihood (ML) estimation procedure, model learning is performed in most MBRL approaches by solving an optimization problem affine to the following one:

$$\widehat{p} = \arg\min_{\overline{p} \in \mathcal{P}} \mathop{\mathbb{E}}_{s,a \sim \delta_\mu^{\pi_b, p}} \left[D_{KL}(p(\cdot|s,a) \| \overline{p}(\cdot|s,a)) \right], \tag{3.2}$$

where we denote by D_{KL} the Kullback-Leibler divergence [54]. For continuous state spaces, a common choice is to minimize the mean squared error between predicted and observed states:

$$\widehat{p} = \arg\min_{\overline{p} \in \mathcal{P}} \mathop{\mathbb{E}}_{\substack{s,a \sim \delta_\mu^{\pi_b, p} \\ \hat{s} \sim \overline{p}(\cdot|s,a), s' \sim p(\cdot|s,a)}} \left[\|\hat{s} - s'\|^2 \right]. \tag{3.3}$$

This approach is *decision-unaware*, in the sense that it is completely agnostic of the decision problem that is being solved by the agent that will use the estimated model. The advantage of this method for model learning is its ease of use and applicability: any of the general density estimation techniques discussed in Section 3.2 can be used for obtaining the world model, without any further modification. When the actual

model of the dynamics p belongs to the model class $\mathcal{P}$ that has been selected for its estimation, the maximum likelihood guess in $\mathcal{P}$ corresponds to p. However, as we discussed in Section 3.2.3, it is often convenient to employ a simpler albeit *misspecified* model class in place of a very powerful one. Although theoretical guarantees exist for this to be addressed by ML-based approaches [90], it is likely that the best model in $\mathcal{P}$ according to the ML objective is not the most effective to be used by a policy to solve the control problem.

Decision-aware MBRL

The observation that, under misspecified model classes, the dynamics of the environment must be captured foreseeing the final task to be performed led to the development of model-learning approaches that are *decision-aware* [29]. One of the first examples in machine learning was a financial application [11], in which, to train a model for generating financial time series, a financial criterion together with a differentiable control module is used in place of a prediction error (e.g., mean squared error).

In the reinforcement learning setting, decision-awareness consists of incorporating, in addition to the effect of the actual model of the environment p on the trajectory and next-state distributions, one or more of these elements into the loss function used for model learning:

- The reward function r
- The policy π that will use the estimated model
- The policy π_b that collected the data

The idea was introduced into MBRL [44, 7] and the related adaptive optimal control literature [83] by considering evaluations of a control policy in the environment as a performance index for model learning. For instance, in [44], the model is updated through the computation of the gradient of the return obtained by a control policy, in a similar manner w.r.t. policy gradient approaches.

Recently, a theoretical framework called *value-aware* MBRL [29] was proposed. The rationale of this value-based method is that some aspects of the dynamics are irrelevant for estimating the optimal value function $\max_{\pi} \left\{ V^{\pi,p} \right\}$. The loss function is obtained by minimizing the expected error on the Bellman optimality operator, explicitly considering that the estimated forward model is then used in an approximate value iteration [12] algorithm. Thus, the model is obtained by minimizing an empirical version of the following loss function:

$$\mathcal{L}(p, \overline{p}) = \mathop{\mathbb{E}}_{s,a \sim \delta_{\mu}^{\pi_b,p}} \left[\sup_{Q \in \mathcal{F}} \left| \int \left[p(s'|s,a) - \overline{p}(s'|s,a) \right] \max_a Q(s,a) ds' \right|^2 ds da \right] \quad (3.4)$$

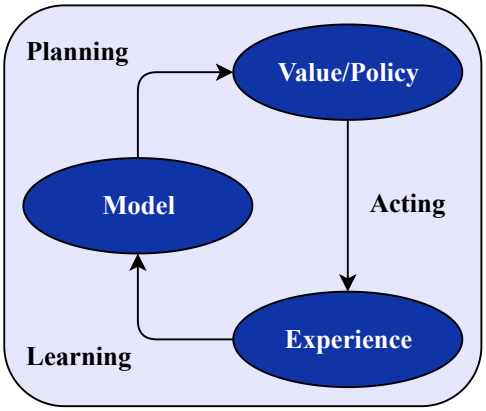

Figure 3.1: Scheme for the standard MBRL approach.

given a class $\mathcal{F}$ of Q-functions. It can be noted that this is a *worst-case* formulation, since the supremum of $\mathcal{F}$ is considered.

Starting from [29], further theoretical considerations and approaches have been proposed. For instance, an iterative version of the original algorithm [28] has been developed, with the aim of alleviating its overly conservative treatment of the value function. In [4], the authors prove that, in Lipschitz MDPs, the use of the value-aware loss from [29] is equivalent to employing the Wasserstein metric as a loss for model learning.

Awareness of the final task to be performed has been also incorporated into stochastic dynamic programming [25] and, albeit implicitly, into neural network-based works [76, 101], in which value functions and models consistent with each other are learned.

HOW TO USE THE LEARNED MODEL

Once a model class and a loss function for model learning are selected, the remaining central question in MBRL concerns how to use the world model. Although one of the possible uses of a learned model is for *exploration* purposes (e.g., [92, 112, 80, 104, 15, 2, 96]), in the rest of this section and of the thesis we will focus on the most common use case: *planning*. There are essentially two possible options, to which we refer to as *online* and *offline* planning. In both cases, a policy is derived from a model that has been learned from experience; the policy can be then used to collect new experience into the environment and restart the cycle. This standard [108] scheme, describing an high-level general view of MBRL, is depicted in Figure 3.1.

Online planning

Using the model for online planning consists in obtaining a control policy on-the-fly, by unrolling the estimated model into the future every time the execution of an action is required, without any explicit parameterization of a policy. This is also known under the name of trajectory optimization [118] in the optimal control literature and, in the case of discrete action spaces, it is closely related to classical [42] and modern [16] search algorithms.

These approaches have been successfully applied in MBRL, for instance using trajectory-level world models [68] or particle methods [17] to perform trajectory optimization, or unrolling a generative adversarial network to adapt the very effective algorithm of Monte Carlo Tree Search to an approximate case [6].

However, we are more often interested in learning an explicit policy π_θ, to increase the generality of the control law that can be derived from the learned model.

Offline planning

Offline planning constitutes perhaps the most used class of MBRL approaches. Pseudocode outlining how it works, from an high-level perspective, is given in Algorithm 3.4.1.

Algorithm 3.4.1: Model-based RL meta-algorithm

Input: Empty dataset $\mathcal{D}$, initial policy π, initial model $\widehat{p}$

while π is not satisfactory **do**
  $\pi_b \leftarrow \texttt{exploratory\_policy}(\pi)$
  ▷ Interact with the environment
  $s \sim \mu$
  while not enough data is collected **do**
    $a \sim \pi_b(\cdot|s)$
    $s' \sim p(\cdot|s, a)$
    $r = r(s, a)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s, a, s', r)$
    $s \leftarrow s'$
  end while
  ▷ Obtain model from collected data
  $\widehat{p} \leftarrow \texttt{obtain\_model}(\widehat{p}, \pi, \mathcal{D})$
  ▷ Obtain a new policy using the model
  $\pi \leftarrow \texttt{obtain\_policy}(\widehat{p}, \pi, \mathcal{D})$
end while

In every iteration of the algorithm, a behavior policy π_b is first obtained from the current policy π. For instance, we can have $\pi_b = \pi$, π_b being a randomly uniform policy or π_b being a mixture between a randomly uniform policy and π [90]. Then, we let π_b interact with the environment and we add the transitions resulting from these interactions to a dataset $\mathcal{D}$. Afterwards, we estimate a forward model $\widehat{p}$. Note that the pseudo-procedure $\texttt{obtain\_model}(\cdot)$ takes three parameters: the first one, the current estimation $\widehat{p}$, is useful for performing some form of online learning; the second one, the policy π, is usually ignored, but can be used by decision-aware approaches to model learning; the third one, the dataset of transitions $\mathcal{D}$, is of course used by all MBRL algorithms. The last step in an iteration of the general algorithm is where planning happens: policy π is updated using $\widehat{p}$ and, potentially, the dataset $\mathcal{D}$.

This meta-algorithm is followed by several methods, ranging from the classical value-based method *Dyna-Q* [105] to the recently proposed *recurrent world models* [39], which use modern generative models and evolutionary techniques to perform policy search. In general, an enormous range of techniques can be used as instantiation of the $\texttt{obtain\_policy}(\cdot)$ step in Algorithm 3.4.1: the policy can be, for instance, obtained by temporal difference learning [108] or by imitation of locally learned policies [59]. In the rest of the thesis, we focus our attention to a particular type of policy search techniques: the ones using the gradient of the performance of policy with respect to its parameters for improving it.

MODEL-BASED POLICY GRADIENT

Overview

Although any policy search approach can be in principle adopted in MBRL, making use of the policy gradient usually yields faster convergence and better scalability.

There are essentially two possible strategies for estimating the policy gradient in a model-based approach:

- *Sampling*. The world model is used for sampling trajectories and then the policy gradient is estimated using any technique affine to the model-free ones presented in Section 2.3.
- *Gradient Propagation*. Given a differentiable reward function, policy and world model, gradients of the performance of the policy are computed by chain-rule-based propagation.

Gradient-propagation approaches clearly have lower variance and are computationally appealing. Nonetheless, they usually require very strong assumptions, especially in the case of stochastic environments. For instance, in PILCO [20], the distribution of states is assumed to be approximately Gaussian and moment matching approximations are used.

Regardless of the use of the sample-based or propagated gradient, a further distinction on existing model-based policy gradient approaches can be made, depending on which element of the policy gradient is derived from the estimated model and which one from the observed transitions.

Fully-Model-based Gradient

A first option for leveraging a world model in approximating the policy gradient is to use it for generating trajectories (or *imagination rollouts*) and then use them for a sampling-based estimate or for propagating the gradient. We can interpret this technique under the light of the PGT and give it a formal definition.

Definition 3.1. *Given a Markov Decision Process $\mathcal{M}$, let Π_Θ be a parametric space of stochastic policies, $\mathcal{P}$ a class of transition models. Given $\pi \in \Pi_\Theta$ and $\widehat{p} \in \mathcal{P}$, the* Fully Model-based Gradient (FMG) *is defined as:*

$$\nabla_\theta^{\text{FMG}} J(\theta) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,\widehat{p}}(s,a) \nabla_\theta \log \pi(a|s) Q^{\pi,\widehat{p}}(s,a) \mathrm{d}s\mathrm{d}a. \tag{3.5}$$

As highlighted in the definition, the estimated model $\widehat{p}$ is used for providing an approximation of both the state-action distribution $\delta_\mu^{\pi,p}$ and the action-value function $Q^{\pi,p}$.

This is by far the most used approach for model-based policy gradient. For instance, PILCO and its extensions [20, 34] alternate the collection of experience in the environment with gradient updates obtained by unrolling the estimated model. Other works try to replicate the model-free policy gradient approaches and integrating them in MBRL, with the use of tabular models [119], least-squares density estimation techniques [110], specifically-designed video predictors [47] or ensembles of models [55].

Model-Value-based Gradient

A different gradient approximation, less common w.r.t. the FMG, is the Model-Value-based Gradient (MVG), defined as follows, again following the same perspective adopted by the PGT.

Definition 3.2. *Let* p *be the transition model of a Markov Decision Process* $\mathcal{M}$, Π_Θ *a parametric space of stochastic policies,* $\mathcal{P}$ *a class of transition models. Given* $\pi \in \Pi_\Theta$ *and* $\hat{p} \in \mathcal{P}$, *the* Model-Value-based Gradient (MVG) *is defined as:*

$$\nabla_\theta^{\text{MVG}} J(\theta) = \frac{1}{1-\gamma} \int_S \int_A \delta_\mu^{\pi,p}(s,a) \nabla_\theta \log \pi(a|s) Q^{\pi,\hat{p}}(s,a) ds da. \tag{3.6}$$

Thus, the MVG employs experience collected in the real environment under the model p, i.e., sampling from $\delta_\mu^{\pi,p}(s,a)$, and uses the generative power of the world model $\hat{p}$ in the computation of an approximate action-value function $Q^{\pi,\hat{p}}$ only.

The MVG finds a compromise between a FMG, in which the experience is directly generated from $\delta_\mu^{\pi,\hat{p}}$ [20, 21], and a Monte Carlo estimator for the MFG (e.g., GPOMDP [9]) in which also the Q-function is computed from experience collected in the real environment.

The idea of grounding the computation of a model-based policy gradient on real trajectories was first introduced for deterministic policies and world models using gradient-propagation [73, 1]. Along these lines, [43] extends the approach to stochastic policies and environments, by differentiating the Bellman equation thanks to the *reparameterization trick* [52].

Other MVG approaches compute the gradient by sampling. For instance, the ones based on model-based value expansion [30, 14] use a fixed-horizon unrolling of an estimated forward model for obtaining a better value function in an actor-critic setting. In order to do so, they assume to have a model whose performance is trusted to be satisfying when unrolled for a limited horizon H. Starting from this consideration, when computing the targets for critic learning, H steps of rewards are simulated and used in the Bellman equation, in place of the immediate next value of the value function or of a Monte Carlo return. This improves the approximation of the value function and, consequently, the performance of the resulting policy.

COMPARISON AMONG GRADIENTS

We have seen three different types of policy gradients: MFG, FMG and MVG. It is worth comparing the advantages and disadvantages of these three approaches.

On the one hand, a model-free estimation of the policy gradient is usually unbiased. However, this has a huge cost in terms of variance, especially in an off-policy setting, thus requiring substantial amounts of data and leading to poor sample efficiency.

On the other hand, the FMG, employing trajectories generated by $\widehat{p}$ for estimating both the distribution of states and the value function, suffers from the full influence of the bias introduced by a world model. This, for complex and stochastic environments, can have a non-negligible impact on the performance of a policy trained by using this gradient approximation.

The MVG, instead, limits the bias effect of $\widehat{p}$ to the Q-function approximation $Q^{\pi,\widehat{p}}$, reducing the compounding of errors due to $\widehat{p}$. At the same time, it enjoys a smaller variance w.r.t. a Monte Carlo estimator, as the Q-function is no longer estimated from samples but just approximated using $\widehat{p}$.

A summary of the advantages of these three general approaches for computing the policy gradient is provided in Table 3.1.

It is worth noting that when the environment dynamics can be approximated *locally* with a simple model, or some prior knowledge on the environment is available, selecting a suitable approximator $\widehat{p}$ for the transition model is easier than choosing an appropriate function approximator for a critic in an actor-critic architecture. Therefore, it is reasonable, in Definition 3.1 and Definition 3.2, to write an approximation for the Q-function in a model-based form (as $Q^{\pi,\widehat{p}}$) in place of a generic $\widehat{Q}$.

Table 3.1: Qualitative comparison of estimations for Model-Free Gradient (MFG), Fully Model-based Gradient (FMG) and Model-Value-based Gradient (MVG) in terms of bias and variance.

	MFG	FMG	MVG
Bias	Zero	Potentially high	Potentially moderate
Variance	Potentially very high	Potentially high	Potentially moderate

4

GRADIENT-AWARE MODEL-BASED POLICY SEARCH

This chapter is devoted to deriving and presenting a novel decision-aware model-based policy search approach. In Section 4.1, we analyze the MVG, providing a relationship between the error made in model estimation and the one made in approximating the policy gradient. Starting from this result, in Section 4.2, we derive the first decision-aware approach that explicitly considers information about the gradient of the current policy in the loss function used for model learning and integrate it into a batch policy search algorithm. Then, in Section 4.3, we provide a theoretical analysis of the proposed algorithm.

ANALYSIS OF THE MODEL-VALUE-BASED GRADIENT

Decision-Aware Bound

A central question concerning Definition 3.2 is how the choice of $\widehat{p}$ affects the quality of the gradient approximation, i.e., how much bias an MVG introduces in the gradient approximation. To this end, we bound the approximation error by the expected KL-divergence between p and $\widehat{p}$.

Theorem 4.1. *Let $q \in [1, +\infty]$ and $\widehat{p} \in \mathcal{P}$. Then, the L^q-norm of the difference between the policy gradient $\nabla_\theta J(\theta)$ and the corresponding MVG $\nabla_\theta^{\text{MVG}} J(\theta)$ can be upper bounded as:*

$$\left\|\nabla_\theta J(\theta) - \nabla_\theta^{\text{MVG}} J(\theta)\right\|_q \leqslant \frac{\gamma\sqrt{2} Z R_{\max}}{(1-\gamma)^2} \sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))]},$$

where

$$\eta_\mu^{\pi,p}(s,a) = \frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi_\theta(a'|s')\right\|_q \delta_{s',a'}^{\pi,p}(s,a) ds' da'$$

is a probability distribution over $\mathcal{S}\times\mathcal{A}$ and

$$Z = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi_\theta(a'|s')\right\|_q ds' da'$$

is a normalization constant, both independent from $\widehat{p}$.[1]

1 We need to assume that $Z > 0$ in order for $\eta_\mu^{\pi,p}$ to be well-defined. This is not a limitation, as if $Z = 0$, then $\nabla_\theta J(\theta) = 0$ and there is no need to define $\eta_\mu^{\pi,p}$ in this case.

Intuitively, Theorem 4.1 suggests that the quality of an MVG approximation is intimately linked to the quality of the estimated model $\widehat{p}$. However, the relative importance of the performance of the world model is not uniform across the whole $\mathcal{S} \times \mathcal{A}$ space, nor it is uniquely determined upon the state-action visitation implied by $\delta_\mu^{\pi,p}$ as assumed in previous MVG-based, decision-unaware, approaches. Instead, the expected value of the distribution mismatch between real and estimated model, quantified by the Kullback-Liebler divergence D_{KL}, is taken under the distribution $\eta_\mu^{\pi,p}$.

In order to understand how the weighting distribution $\eta_\mu^{\pi,p}$ enlarges the relative importance of some transitions with respect to others, we define an auxiliary distribution $\nu_\mu^{\pi,p}$ as:

$$\nu_\mu^{\pi,p}(s', a') = \frac{1}{Z} \left\| \nabla_\theta \log \pi(a'|s') \right\|_q \delta_\mu^{\pi,p}(s', a'), \tag{4.1}$$

where Z, as defined in Theorem 4.1, is a normalization constant required for $\nu_\mu^{\pi,p}$ to be a well-defined probability distribution. Z can be seen as the *expected score magnitude* in the MDP $\mathcal{M}$ under policy π. The distribution $\nu_\mu^{\pi,p}$ is high for states and actions that are both likely to be visited executing π and corresponding to high norm of its score. Intuitively, a low magnitude for the score is related to a smaller possibility for policy π to be improved. However, the connection between the score-magnitude for states and actions and the relative importance of those states and actions for minimizing the approximation error caused by the MVG is not direct. In other words, it is not possible to say that the most important transitions to be learned for a model to be good for an MVG approach are the ones featuring the largest score magnitude and frequently encountered by the policy (i.e., $\nu_\mu^{\pi,p}$ is *not* the correct weighting distribution). Nonetheless, $\nu_\mu^{\pi,p}$ plays an important role in defining the whole weighting distribution $\eta_\mu^{\pi,p}$. In fact, it can be rewritten as:

$$\eta_\mu^{\pi,p}(s, a) = \int_{\mathcal{S}} \int_{\mathcal{A}} \underbrace{\nu_\mu^{\pi,p}(s', a')}_{\substack{\text{gradient magnitude} \\ \text{distribution}}} \underbrace{\delta_{s',a'}^{\pi,p}(s, a)}_{\substack{\text{state-action} \\ \text{reachability}}} \, ds' da' = \mathop{\mathbb{E}}_{s', a' \sim \nu_\mu^{\pi,p}} \left[\delta_{s',a'}^{\pi,p}(s, a) \right]. \tag{4.2}$$

Under the interpretation suggested by Equation 4.2, $\eta_\mu^{\pi,p}$ can be seen as the expected *state-action reachability* under the *gradient magnitude distribution*. $\delta_{s',a'}^{\pi,p}(s, a)$ is the state-action distribution of (s, a) after executing action a' in state s'. It is equivalent to the state-action distribution $\delta_\mu^{\pi,p}$ in an MDP where

$$\mu(s) = \begin{cases} 1 & s = s' \\ 0 & \text{otherwise} \end{cases},$$

and the first action executed by the agent is always a'. Each state-action couple (s', a') with high score magnitude that precedes (s, a) brings a contribution to the final weighting factor for (s, a).

Summarizing, the most relevant (s, a) pairs for propagating the model error to the MVG error are those that are *likely to be reached from the policy starting from high gradient-magnitude state-action pairs*. If, for instance, $(\bar{s}, \bar{a})$ is only encountered after low score-magnitude state-action pairs, corresponding transitions of the type $(\hat{s}, \hat{a}, \hat{s}' \sim p(\cdot|\hat{s}, \hat{a}))$ contribute less in propagating the error of the model $\hat{p}$ to the bias induced by the MVG.

Maximum Likelihood Bound

Theorem 4.1 shows that the state-action distribution is not the only factor to give attention to when learning a model employed in an MVG. However, existing approaches (e.g., [73, 1, 43, 30, 14]) employ a maximum likelihood objective for estimating the forward model: is this a theoretically sound approach? We answer this question by stating the following proposition.

Proposition 4.2. *Let* $q \in [1, +\infty]$ *and* $\hat{p} \in \mathcal{P}$. *If* $\|\nabla_\theta \log \pi(a|s)\|_q \leqslant K$ *for all* $s \in \mathcal{S}$ *and* $s \in \mathcal{A}$, *then, the* L^q*-norm of the difference between the policy gradient* $\nabla_\theta J(\theta)$ *and the corresponding MVG* $\nabla_\theta^{MVG} J(\theta)$ *can be upper bounded as:*

$$\begin{aligned}\|\nabla_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\|_q &\leqslant \frac{\gamma\sqrt{2} Z R_{max}}{(1-\gamma)^2} \sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\hat{p}(\cdot|s,a))]} \\ &\leqslant \frac{\gamma\sqrt{2} K R_{max}}{(1-\gamma)^2} \sqrt{\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\hat{p}(\cdot|s,a))]}.\end{aligned}$$

This proposition gives motivation to both the common maximum likelihood approach and the decision-aware one suggested by Theorem 4.1. We can derive two insights from it: the first is that minimizing $\mathbb{E}_{s,a\sim\delta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\hat{p}(\cdot|s,a))]$, (i.e., learning the model by maximum likelihood) can therefore minimize an upper bound on the error of the approximation provided by the MVG; the second, however, is that finding a minimizer for the expectation of the KL-divergence under $\delta_\mu^{\pi,p}$ generally leads to a worse MVG approximation than the one provided by minimizing the expected KL-divergence under $\eta_\mu^{\pi,p}$.

To understand how much the bound provided by Theorem 4.1 is actually tighter, we can analyze the case in which the magnitude of the gradient is a constant value, such that $\|\nabla_\theta \log \pi(a|s)\|_q = K_0$. In this case, we observe that:

$$Z = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s', a') \left\|\nabla_\theta \log \pi_\theta(a'|s')\right\|_q ds'da' = K_0 \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s', a') ds'da' = K_0 \tag{4.3}$$

and that:

$$\eta_\mu^{\pi,p}(s, a) = \frac{1}{K_0}\int_{\mathcal{S}}\int_{\mathcal{A}} K_0 \delta_\mu^{\pi,p}(s', a')\delta_{s',a'}^{\pi,p}(s, a) ds'da' = \delta_\mu^{\pi,p}(s, a). \tag{4.4}$$

Picking a value of $K = K_0$, we can notice that, in the case of constant gradient-magnitude, the last inequality presented in Proposition 4.2 holds therefore as an equality. Thus, a maximum likelihood approach for model learning would lead to the same result w.r.t. a decision-aware one. Intuitively, as the magnitude of the score starts to vary in a range that is large enough, and $K \gg Z$, the bound provided by Theorem 4.1 can become significantly tighter, especially if this variation occurs in states and actions that are frequently visited by the policy π.

Related considerations were previously stated for other forms of decision-aware MBRL. In particular, it has been shown that, despite being less effective than *value-aware model learning* [29], maximum likelihood estimation can be used for minimizing the error induced by a world model on the Bellman operator.

GRADIENT-AWARE MODEL-BASED POLICY SEARCH ALGORITHM

Inspired by Theorem 4.1, we propose a policy search algorithm that employs an MVG approximation, combining trajectories generated in the real environment together with a model-based approximation of the Q-function obtained with the estimated transition model $\widehat{p}$. The algorithm, Gradient-Aware Model-based Policy Search (GAMPS), consists of three steps: learning the forward model $\widehat{p}$ (Section 4.2.1), computing the action-value function $Q^{\pi,\widehat{p}}$ (Section 4.2.2) and updating the policy using the estimated gradient $\widehat{\nabla}_\theta J(\theta)$ (Section 4.2.3).

Learning the transition model

To learn $\widehat{p}$, we aim at minimizing the bound in Theorem 4.1, over a class of transition models $\mathcal{P}$, using the trajectories $\mathcal{D}$ collected with $\zeta_\mu^{\pi_b,p}$. However, to estimate an expected value computed over $\eta_\mu^{\pi,p}$, as in Theorem 4.1, we face two problems. First, the policy mismatch between the behavioral policy π_b used to collect $\mathcal{D}$ and the current agent's policy π. This can be easily addressed by using importance sampling. Second, given a policy π we need to be able to compute the expectations over $\eta_\mu^{\pi,p}$ using samples from $\zeta_\mu^{\pi,p}$. In other words, we need to reformulate the expectation over $\eta_\mu^{\pi,p}$ in terms of expectation over trajectories. To this end, we provide the following result.

Lemma 4.3. *Let π and π_b be two policies such that $\pi \ll \pi_b$ (π is absolutely continuous w.r.t. to π_b). Let $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$ be an arbitrary function defined over the state-action space. Then, it holds that:*

$$\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}[f(s,a)] = \frac{(1-\gamma)^2}{Z} \mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}} \left[\sum_{t=0}^{+\infty} \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}) \sum_{l=0}^{t} \|\nabla_\theta \log \pi(a_l|s_l)\|_q \, f(s_t,a_t)\right].$$

To specialize Lemma 4.3 for our specific case, it is sufficient to set $f(s,a) = D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))$. Note that Z is independent from $\widehat{p}$ and thus it can be ig-

nored in the minimization procedure. Furthremore, minimizing the KL-divergence is equivalent to maximizing the log-likelihood of the observed transitions. Putting it all together, we get to the objective:

$$\begin{aligned}\widehat{p} &= \arg\max_{\overline{p}\in\mathcal{P}} \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_i-1} \omega_t^i \log \overline{p}\left(s_{t+1}^i | s_t^i, a_t^i\right), \\ \omega_t^i &= \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}^i) \sum_{l=0}^{t} \left\| \nabla_\theta \log \pi(a_l^i | s_l^i) \right\|_q .\end{aligned} \tag{4.5}$$

The factors contained in the weight ω_t^i accomplish three goals in weighting the transitions for the model. The discount factor γ^t encodes that later transitions are exponentially less important in the gradient computation. The importance weight $\rho_{\pi/\pi_b}(\tau_{0:t}^i)$ is larger for the transitions that are more likely to be generated by the current policy π. This incorporates a key consideration into model learning: since the running policy π can be quite different from the policy that generated the data π_b, typically very explorative [21, 90], an accurate approximation of the dynamics for the regions that are rarely reached by the current policy is not useful. Lastly, the factor $\sum_{l=0}^{t} \left\| \nabla_\theta \log \pi(a_l^i | s_l^i) \right\|_q$ prefers the transitions that occur at the end of a subtrajectory $\tau_{0:t}$ with a high cumulative score-magnitude. This score accumulation resembles the expression of some model-free gradient estimators, such as G(PO)MDP [9]. Intuitively, the magnitude of the score of a policy is related to its *opportunity to be improved*, i.e., the possibility to change the probability of actions. Our gradient-aware weighting scheme encourages a better approximation of the dynamics for states and actions found in trajectories that can potentially lead to the most significant improvements to the policy. We provide an alternative derivation for this gradient-aware weighting scheme in Appendix B.1.

Computing the value function

The estimated transition model $\widehat{p}$ can be used to compute the action-value function $Q^{\pi,\widehat{p}}$ for any policy π. At this stage, we need to consider the reward function r of $\mathcal{M}$. For the rest of the presentation of our algorithm, we assume r is known. This is not a strong assumption in many domains, as discussed in Chapter 2. However, when it is not directly available, the actual reward function can be replaced by the use of an estimated reward function $\widehat{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, obtained, for instance, by using an ML approach.

Computing the action-value function amounts to *evaluating* the current policy using $\widehat{p}$ instead of the actual transition probability kernel p. In the case of finite MDPs, the evaluation can be performed by finding the fixed point of the Bellman equation

$$\widehat{Q}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\widehat{p}(\cdot|s,a), a'\sim\pi(\cdot|s')} \left[\widehat{Q}(s',a')\right], \tag{4.6}$$

either in closed form or in an iterative manner via dynamic programming [10, 107].

For continuous MDPs, $Q^{\pi,\hat{p}}$ cannot, in general, be represented exactly. A common approach consists of employing a parameterized function approximator $\widehat{Q}_\omega \in \mathcal{Q}$ and apply approximate dynamic programming [12]. The regression targets to be used for learning these parameters must be derived from the estimated model, in order to inject the learned knowledge into another stage of estimation: it can be done by choosing an arbitrary amount of unrolling of $\hat{p}$ before performing bootstrapping by temporal difference, using the the *model-based value expansion* approach [30]. For instance, with one step of model unrolling, the state-action value function could be found by iteratively solving the following optimization problem:

$$\widehat{Q} = \arg\min_{Q} \sum_{\tau} \sum_{t} \left(Q(s_t, a_t) - \left(r(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{s_{t+1} \sim \hat{p}(\cdot|s_t, a_t) \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} [Q(s_{t+1}, a_{t+1})] \right) \right)^2, \tag{4.7}$$

where we used a lighter notation for indicating summations over trajectories (with index τ) and time steps of these trajectories. The expected value in Equation (4.7) can be approximated by sampling from the estimated model $\hat{p}$ and the policy π. In practice, a further parameterized state-value function $\widehat{V}(s) \approx \mathbb{E}_{a \sim \pi(\cdot|s)} \left[\widehat{Q}(s, a) \right]$ can be learned jointly with the action-value function.

However, this method requires a proper choice of a functional space $\mathcal{Q}$ and the definition of the regression targets, which should be derived using the estimated model $\hat{p}$ [27, 89], possibly introducing further bias.

For our algorithm, we instead encourage the use a different approach, that consists in the use of $\hat{p}$ as a generative model for the sole purpose of approximating $Q^{\pi,\hat{p}}$. Recalling that we will use $\widehat{Q}$ to estimate the policy gradient from the available trajectories, we can just obtain a Monte Carlo approximation of $Q^{\pi,\hat{p}}$ on the fly, in an unbiased way, averaging the return from a (possibly large) number of imaginary trajectories obtained from the estimated model $\hat{p}$:

$$\widehat{Q}(s, a) = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=0}^{T_j - 1} \gamma^t r(s_t^j, a_t^j), \quad \tau^j \sim \zeta_{s,a}^{\pi,\hat{p}}. \tag{4.8}$$

This approach has the advantage of avoiding the harsh choice of an appropriate model complexity $\mathcal{Q}$ and the definition of the regression targets, while providing an unbiased estimate for the quantity of interest.

Estimating the policy gradient

After computing $Q^{\pi,\hat{p}}$ (or some approximation $\widehat{Q}$), all the gathered information can be used to improve policy π. As we are using a *model-value-based gradient*, the trajectories

we will use have been previously collected in the real environment. Furthermore, the data have been generated by a possibly different policy π_b, and, to account for the difference in the distributions, we need importance sampling again. Therefore, by writing the sample version of Equation (3.6) we obtain:

$$\widehat{\nabla}_\theta J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi/\pi_b}(\tau^i_{0:t}) \nabla_\theta \log \pi(a^i_t|s^i_t) Q^{\pi,\widehat{p}}(s^i_t, a^i_t). \tag{4.9}$$

For performing batch policy optimization, we iteratively repeat the three steps presented in this section using the data collected by the behavior policy π_b. At each iteration, we fit the model with the weights relative to the current policy, we employ it in the computation of the state-action value function and we then improve the policy with one or more steps of gradient ascent. The overall procedure is summarized in Algorithm 4.2.1.

Algorithm 4.2.1: Gradient-Aware Model-based Policy Search

Input: Trajectory dataset $\mathcal{D}$, behavior policy π_b,
initial parameters θ_0, step size schedule $(\alpha_k)_{k=0}^{K-1}$

for $k = 0, 1, ..., K-1$ **do**
▷ Learn the model (Section 4.2.1)
$\omega^i_{t,k} \leftarrow \gamma^t \rho_{\pi_{\theta_k}/\pi_b}(\tau^i_{0:t}) \sum_{l=0}^{t} \|\nabla_\theta \log \pi_{\theta_k}(a^i_l|s^i_l)\|_q$
$\widehat{p}_k \leftarrow \arg\max_{\overline{p}\in\mathcal{P}} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \omega^i_{t,k} \log \overline{p}(s^i_{t+1}|s^i_t, a^i_t)$
▷ Estimate the value function (Section 4.2.2)
Generate M trajectories for each (s, a) using $\widehat{p}_k$
$\widehat{Q}_k(s, a) = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=0}^{T_j-1} \gamma^t r(s^j_t, a^j_t)$
▷ Improve the policy (Section 4.2.3)
$\widehat{\nabla}_\theta J(\theta_k) \leftarrow \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi_{\theta_k}/\pi_b}(\tau^i_{0:t})$
$\times \nabla_\theta \log \pi_{\theta_k}(a^i_t|s^i_t) \widehat{Q}_k(s^i_t, a^i_t)$
$\theta_{k+1} \leftarrow \theta_k + \alpha_k \widehat{\nabla}_\theta J(\theta_k)$
end for

THEORETICAL ANALYSIS

In this section, we provide a finite-sample bound for the gradient estimation of Equation 4.9, assuming to have the exact value of $Q^{\pi,\widehat{p}}$. This corresponds to the analysis of a single iteration of GAMPS.

The objective of this analysis, carried out considering the actual action-value function under the estimated model $\widehat{p}$, is to extend the insights provided by Theorem 4.1 to the

case where the MVG is estimated, instead of computed exactly. The result we provide is a PAC learning bound, including a model approximation error, depending on the model class used for the world model, and an estimation error, influenced by the available data.

Note that it should be possible, albeit not straightforward, to extend our single-iteration analysis to multiple iterations of GAMPS, in order to understand how the propagation of the error occurs. Likewise, our analysis could be extended to the case in which $Q^{\pi,\widehat{p}}$ and r are not assumed to be available, incorporating the error made in their estimation into the final result.

Assumptions

We now present the assumptions that are required for our theoretical analysis of GAMPS. First, we define two useful functions. Let τ be a trajectory, $\pi \in \Pi_\Theta$ and $\overline{p} \in \mathcal{P}$. We define

$$l^{\pi,\overline{p}}(\tau) = \sum_{t=0}^{+\infty} \omega_t \log \overline{p}\left(s_{t+1}|s_t, a_t\right), \tag{4.10}$$

$$\mathbf{g}^{\pi,\overline{p}}(\tau) = \sum_{t=0}^{+\infty} \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}) \nabla_\theta \log \pi(a_t|s_t) Q^{\pi,\overline{p}}(s_t, a_t). \tag{4.11}$$

Using these functions, we can state the following assumptions.

Assumption 1. *The second moment of $l^{\pi,\overline{p}}$ and $\mathbf{g}^{\pi,\overline{p}}$ are uniformly bounded over $\mathcal{P}$ and Π_Θ. In this case, given a dataset $\mathcal{D} = \{\tau^i\}_{i=1}^N$, there exist two constants $c_1, c_2 < +\infty$ such that:*

$$\sup_{\overline{p}\in\mathcal{P}} \sup_{\pi\in\Pi_\Theta} \max\left\{ \mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}} \left[l^{\pi,\overline{p}}(\tau)^2\right], \frac{1}{N}\sum_{i=1}^N l^{\pi,\overline{p}}(\tau^i)^2 \right\} \leqslant c_1^2,$$

$$\sup_{\overline{p}\in\mathcal{P}} \sup_{\pi\in\Pi_\Theta} \max\left\{ \left\| \mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}} \left[\mathbf{g}^{\pi,\overline{p}}(\tau)^2\right] \right\|_\infty, \left\| \frac{1}{N}\sum_{i=1}^N \mathbf{g}^{\pi,\overline{p}}(\tau^i)^2 \right\|_\infty \right\} \leqslant R_{\max}^2 c_2^2.$$

Assumption 2. *The pseudo-dimension of the hypothesis spaces $\left\{l^{\pi,\overline{p}} : \overline{p} \in \mathcal{P}, \pi \in \Pi\right\}$ and $\left\{\mathbf{g}^{\pi,\overline{p}} : \overline{p} \in \mathcal{P}, \pi \in \Pi\right\}$ are bounded by $\nu < +\infty$.*

Assumption 1 is requiring that the overall effect of the importance weight ρ_{π/π_b}, the score $\nabla_\theta \log \pi$ and the approximating transition model $\overline{p}$ preserves the finiteness of the second moment. Clearly, a sufficient (albeit often unrealistic) condition is requiring

all these quantities to be uniformly bounded. Assumption 1 is equivalent to require that there exists two finite constants $c_1 < +\infty$ and $c_2 < +\infty$ such that:

$$\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\sum_{t=0}^{+\infty}\gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\log p(s_{t+1}|s_t,a_t)\right)^2\right]\leqslant c_1^2, \tag{4.12}$$

$$\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\sum_{t=0}^{+\infty}\gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\nabla_{\theta_j}\log\pi(a_t|s_t)Q^{\pi,\overline{p}}(s_t,a_t)\right)^2\right]\leqslant R_{\max}^2c_2^2,\ j\in[1..d]. \tag{4.13}$$

Assumption 2 concerns the capacity of the model classes used for learning the estimated model $\overline{p}$ and the policy π. This capacity, quantified by means of the pseudo-dimension of the hypothesis spaces for $l^{\pi,\overline{p}}$ and $\mathbf{g}^{\pi,\overline{p}}$, must be bounded. This assumption is required to state learning theory guarantees.

Arguably, it is not easy to understand which are the conditions that the policies π and π_b, together with the transition models p (the real one) and $\overline{p}$ (the approximating one) should satisfy in order to fulfill Assumption 1. Thus, we decouple the assumption into two separate conditions, more intelligible, through the following result.

Corollary 4.4. *Assumption 1 is satisfied if there exist three constants χ_1, χ_2 and χ_3, with $\chi_1 < \frac{1}{\gamma}$.*

$$\sup_{\pi\in\Pi_\Theta}\sup_{s\in\mathcal{S}}\mathop{\mathbb{E}}_{a\sim\pi_b(\cdot|s)}\left[\left(\frac{\pi(a|s)}{\pi_b(a|s)}\right)^2\right]\leqslant\chi_1,$$

$$\sup_{\pi\in\Pi_\Theta}\sup_{s\in\mathcal{S}}\mathop{\mathbb{E}}_{a\sim\pi_b(\cdot|s)}\left[\left(\frac{\pi(a|s)}{\pi_b(a|s)}\|\nabla_\theta\log\pi(a|s)\|_q^2\right)^2\right]\leqslant\chi_2,$$

$$\sup_{\overline{p}\in\mathcal{P}}\sup_{\substack{s\in\mathcal{S}\\a\in\mathcal{A}}}\mathop{\mathbb{E}}_{s'\sim p(\cdot|s,a)}\left[\left(\log\overline{p}(s'|s,a)\right)^2\right]\leqslant\chi_3.$$

In such case, Equation (A.23) and Equation (A.24) are satisfied with constants:

$$c_1^2=\frac{\chi_3\chi_2(1+\gamma\chi_1)}{(1-\gamma)(1-\gamma\chi_1)^3},\quad c_2^2=\frac{\chi_3\chi_2}{(1-\gamma)^3(1-\gamma\chi_1)}.$$

The first two inequalities impose a condition on the policies π_b and π, while the last concerns the models p and $\overline{p}$.

Finite-sample bound

Under the assumptions stated in the previous section, we are now ready to present the main result. It employs the learning theory tools of [19].

Theorem 4.5. *Let* $q \in [1, +\infty]$, d *be the dimensionality of* Θ *and* $\widehat{p} \in \mathcal{P}$ *be the maximizer of the objective function in Equation* (4.5)*, obtained with* $N > 0$ *independent trajectories* $\{\tau^i\}_{i=1}^N$*. Under Assumption 1 and 2, for any* $\delta \in (0, 1)$*, with probability at least* $1 - 4\delta$ *it holds that:*

$$\left\| \widehat{\nabla}_\theta J(\theta) - \nabla_\theta J(\theta) \right\|_q \leqslant \underbrace{\frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2} \inf_{\overline{p} \in \mathcal{P}} \sqrt{\mathop{\mathbb{E}}_{s,a \sim \eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a) \| \overline{p}(\cdot|s,a))]}}_{\textit{approximation error}}$$
$$+ \underbrace{2R_{\max} \left(d^{\frac{1}{q}} c_2 \epsilon + \frac{\gamma\sqrt{2Z c_1 \epsilon}}{1-\gamma} \right)}_{\textit{estimation error}},$$

given

$$\epsilon = \sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{8(d+1)}{\delta}}{N}} \Gamma\left(\sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{8(d+1)}{\delta}}{N}} \right),$$

$$\Gamma(\xi) \coloneqq \frac{1}{2} + \sqrt{1 + \frac{1}{2} \log \frac{1}{\xi}}.$$

The theorem justifies the intuition behind the gradient estimation based on the MVG. A model $\overline{p}$ is good when it achieves a reasonable trade-off between the errors in approximation and estimation.[2]

With a huge amount of data (i.e., $N \to \infty$), the second term, concerning the estimation error, becomes negligible, since $\epsilon \approx 0$. Thus, it is better to select a powerful model class $\mathcal{P}$, such that the best model contained in it is able to reach a very low approximation error.

However, in the much more frequent case of scarce data (i.e., small N), it is convenient to choose a low-capacity model class $\mathcal{P}$ in order to reduce the error-enlarging effect the pseudo-dimension ν, and, consequently, the second term in the bound. Naturally, this carries the risk of being unable to approximate the original model. Nonetheless, the gradient-aware nature of GAMPS is reflected into the behavior of the first term of the bound: the approximation error depends, in fact, on an expected value under $\eta_\mu^{\pi,p}$. Thus, even a model class that would be highly misspecified w.r.t. an expectation computed under the state-action distribution $\delta_\mu^{\pi,p}$ can, perhaps surprisingly, lead to an accurate gradient estimation using our approach.

This is a direct incarnation of the rationale behind decision-aware MBRL approaches: if not enough data is available (i.e., N is not large enough), the best choice is to pick a fairly simple model class and employ the limited capacity for approximating the bits of the dynamics that are most important for the learning algorithm, in the case of GAMPS a policy gradient method based on the MVG.

2 It is worth noting that the estimation error is $\widetilde{\mathcal{O}}(N^{-\frac{1}{4}})$.

In the bound presented in Theorem 4.5, other factors play a role. For instance, the dimensionality of the parameter vector of the policy space Π_Θ has an effect on the estimation error. This happens because estimating the gradient for a policy that features a small number of parameters is intuitively easier, albeit the policy might not be able to reach satisfying performance on a given problem.

5

EXPERIMENTS

In this chapter, we present an experimental evaluation of GAMPS, whose objective is two-fold: assessing the effect of our weighting scheme for model learning and comparing the performance in batch policy optimization of our algorithm against model-based and model-free policy search baselines.

TWO-AREAS GRIDWORLD

This experiment is meant to show how decision-awareness can be an effective tool to improve the accuracy of policy gradient estimates when using a forward model. The environment, depicted in Figure 5.1, is a 5×5 gridworld, divided into two areas (lower and upper) with different dynamics: the effect of a movement action of the agent is reversed in one area w.r.t. the other. Once the agent gets to the lower area, it is not possible for it to go back in the upper one.

More precisely, the gridworld we use in our experiments features two subspaces of the state space $\mathcal{S}$, to which we refer to as $\mathcal{S}_1$ (*lower*) and $\mathcal{S}_2$ (*upper*). The agent can choose among four different actions: in the lower part, a sticky area, each action corresponds to an attempt to go up, right, down or left, and has a 0.9 probability of success and a 0.1 probability of causing the agent to remain in the same state; in the upper part, the four actions have deterministic movement effects, all different from the ones they have in the other area (rotated of 90 degrees). Representing as $(p_1, p_2, p_3, p_4, p_5)$ the probabilities p_1, p_2, p_3, p_4 and p_5 of, respectively, going up, right, down, left and

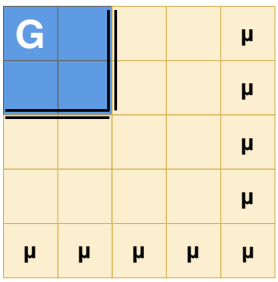

Figure 5.1: Gridworld representation. The goal state is **G** and the possible initial states are μ. The two areas with different dynamics are represented with different colors and the black bars remark that is not possible for the agent to reach the lower area once it is left.

remaining in the same state, the transition model of the environment is defined as follows:

$$s \in S_1 : p(\cdot|s,a) = \begin{cases} (0, \underset{\Uparrow}{0.9}, 0, 0, 0.1), & \text{if } a = 0 \\ (0, 0, \underset{\Rightarrow}{0.9}, 0, 0.1), & \text{if } a = 1 \\ (0, 0, 0, \underset{\Leftarrow}{0.9}, 0.1), & \text{if } a = 2 \\ (\underset{\Uparrow}{0.9}, 0, 0, 0, 0.1), & \text{if } a = 3 \end{cases},$$

$$s \in S_2 : p(\cdot|s,a) = \begin{cases} (\underset{\Uparrow}{1}, 0, 0, 0, 0), & \text{if } a = 0 \\ (0, \underset{\Rightarrow}{1}, 0, 0, 0), & \text{if } a = 1 \\ (0, 0, \underset{\Downarrow}{1}, 0, 0), & \text{if } a = 2 \\ (0, 0, 0, \underset{\Leftarrow}{1}, 0), & \text{if } a = 3 \end{cases}.$$

There is a reward of -1 in all states apart a single absorbing goal state, located on the upper left corner, that yields zero reward. The initial state is uniformly chosen among the ones on the low and right border and the agent cannot go back to the sticky part once it reached the second area, in which it passes through the walls to get to the other side.

As policy class Π_Θ, we use policies linear in the one-hot representation of the current state. The policy outputs a Boltzman probability distribution over the four possible actions. In the lower part of the environment, we initialize the policy as deterministic: the agent tries to go up as long as it can, and goes left when a wall is encountered. Being the policy deterministic for these actions, the corresponding score is zero. We collect experience with policy π_b that is randomly initialized in the upper area. π_b is also used as initial policy for the learning algorithm.

As model class $\mathcal{P}$, we employ the one in which each $\widehat{p} \in \mathcal{P}$ is such that $\widehat{p}(m|s,a) = \text{softmax}(\mathbb{1}_a^\intercal \mathbf{W})$, where $\mathbf{W}$ is a matrix of learnable parameters, $\mathbb{1}_a$ is the one-hot representation of the action and $m \in \{\Uparrow, \Rightarrow, \Downarrow, \Leftarrow, \Leftrightarrow\}$ is a movement effect. This model class has very little expressive power: the forward model is, in practice, executing a probabilistic lookup using the current actions, trying to guess what the next state is without even looking at the current one.

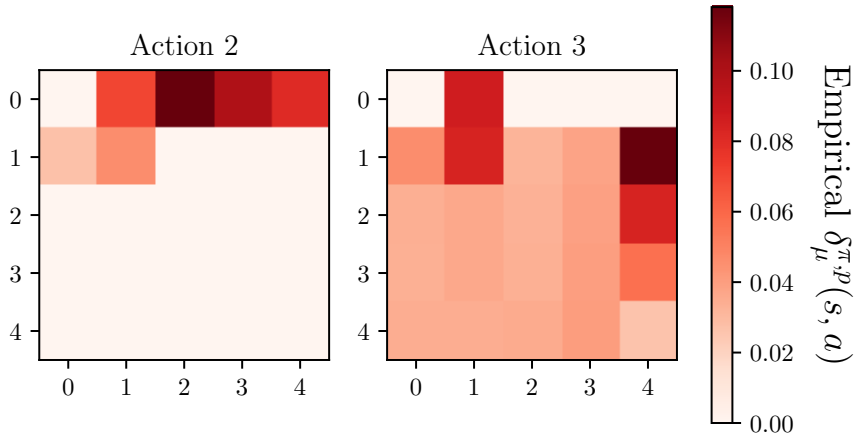

Figure 5.2: Normalized values of the empirical state-action distribution $\delta_\mu^{\pi,p}$. Each grid represents every state of the environment for the two most representative actions.

Properties of Gradient-Aware Model Learning

As stated above, the first goal of this experiment is to show that, with the use of gradient-awareness, even an extremely simple model class can be sufficiently expressive to provide an accurate estimate of the policy gradient. The considered model class cannot perfectly represent the whole environment dynamics at the same time, as it nonlinearly changes between the two areas. However, given the nature of policy π, this is not necessary, since only the modeling of the upper area, which is indeed representable with our model, would be enough to perfectly improve the policy. Nonetheless, this useful information has no way of being captured using the usual maximum likelihood procedure, which, during model learning, weights the transitions just upon visitation, regardless the policy. To experimentally assess how our approach addresses this intuitive point, we generate 1000 trajectories running π_b in the environment, and we first compare the maximum likelihood and the gradient-aware weighting factors, $\delta_\mu^{\pi,p}(s,a)$ and $\eta_\mu^{\pi,p}(s,a)$. The results (Figure 5.2 and Figure 5.3) show that our method is able, in a totally automatic way, not to assign importance to the transitions in which the policy cannot be improved.

In order to further understand the properties of our method for model learning, we compare the maximum likelihood model (ML) and the one obtained with GAMPS, in terms of accuracy in next state prediction and MSE with the real Q-function w.r.t. to the one derived by dynamic programming; lastly, we use the computed action-value functions to provide two approximations to the sample version of Equation 3.6. The intuitive rationale behind decision-aware model learning is that the raw quality of the estimate of the forward model itself or any intermediate quantity is pointless: the accuracy on estimating the quantity of interest for improving the policy, in our case its

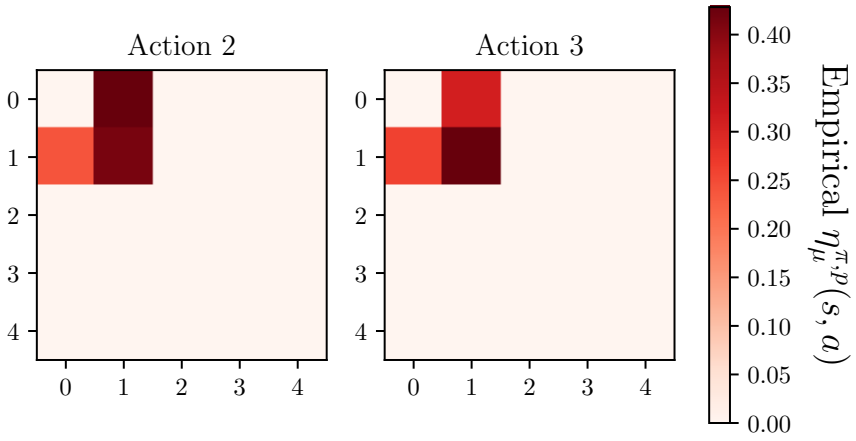

Figure 5.3: Normalized values of the gradient-aware weighting factor $\eta_\mu^{\pi,p}$. Each grid represents every state of the environment for the two most representative actions.

gradient, is the only relevant metric. The results, shown in Table 5.1, illustrate exactly this point, showing that, although our method offers worse performance in model and Q-function estimation, it is able to perfectly estimate the correct direction of the policy gradient.

Table 5.1: Estimation performance on the gridworld environment comparing Maximum Likelihood estimation (ML) and our approach (GAMPS). 1000 training and 1000 validation trajectories per run. Average results on 10 runs with a 95% confidence interval.

Approach	$\widehat{p}$ accuracy	$\widehat{Q}$ MSE	$\widehat{\nabla}_\theta J$ cosine similarity
ML	0.765 ± 0.001	11.803 ± 0.158	0.449 ± 0.041
GAMPS	0.357 ± 0.004	633.835 ± 12.697	1.000 ± 0.000

We now present a clear definition of the metrics used for making the comparison, computed over an hold-out set of 1000 validation trajectories. The model accuracy for an estimated model $\hat{p}$ is defined as:

$$\text{acc}(\widehat{p}) = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s') \in \mathcal{D}} \mathbb{1}(s' = \arg\max_{\bar{s}} \widehat{p}(\bar{s}|s,a)). \tag{5.1}$$

The MSE for measuring the error in estimating the tabular Q-function is computed by averaging the error obtained for every state and action. Lastly, the cosine similarity between the real gradient $\nabla_\theta J(\theta)$ and the estimated gradient $\widehat{\nabla}_\theta J(\theta)$ is defined as:

$$\text{sim}(\nabla_\theta J(\theta), \widehat{\nabla}_\theta J(\theta)) = \frac{\nabla_\theta J(\theta) \cdot \widehat{\nabla}_\theta J(\theta)}{\max(\|\nabla_\theta J(\theta)\|_2\| \cdot \widehat{\nabla}_\theta J(\theta)\|_2, \epsilon)}, \tag{5.2}$$

where ϵ is set to 10^{-8}.

Performance in Policy Improvement

We further investigate the performance of GAMPS compared to batch learning with the maximum likelihood transition model (ML) and two classical model-free learning algorithms REINFORCE [121] and PGT [109]. To adapt the latter two to the batch setting, we employ importance sampling in the same way as described in Equation 4.9, but estimating the Q-function using the same trajectories (and importance sampling as well). We learn both the policy and the models by minimizing the corresponding loss function via gradient descent, using the hyperparameters reported in Table 5.2. Note that in stating the results the results presented in Chapter 4 we made no assumption on the norm to be used on the score for the gradient-aware weights: thus, it is an additional hyperparameter available for GAMPS. In practice, given $\|\nabla_\theta \log \pi(a|s)\|_q$, we employ $q = 2$.

Table 5.2: Hyperparameters used for algorithm comparison in the different environments. The apexes *model* and *policy* indicate the parameters employed in optimizing the two using the Adam [50] optimizer. Clearly, hyperparameters concerning the estimation of the forward model are ignored in model-free algorithms. The hypeparameters, except for q and γ, were chosen by trial and error from a range of $(0.001, 0.9)$.

Environment	α^{model}	β_1^{model}	β_2^{model}	α^{policy}	β_1^{policy}	β_2^{policy}	q	γ
Gridworld	0.01	0.9	0.999	0.2	0.9	0.999	2	0.99
Minigolf	0.02	0.9	0.999	0.08	0	0.999	2	0.99

The results obtained by collecting different numbers of trajectories and evaluating on the environment are shown in Figure 5.5, Figure 5.4 and Figure 5.6. Note that, in batch learning, performance degradation when the current policy π becomes too far from the behavioral policy π_b is natural due to the variance of the importance weights. To avoid this effect, a stopping condition connected to the effective sample size [79] could be employed.

As the evaluation shows, REINFORCE is generally affected by severe variance, and thus struggles in reaching satisfying performance with any amount of trajectories; the

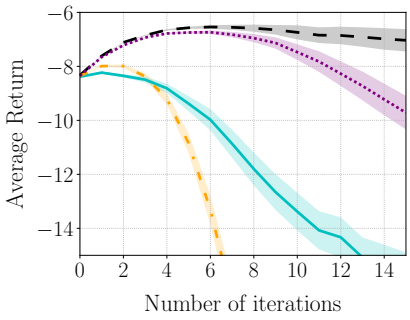

Figure 5.4: GAMPS performance in gridworld using 50 trajectories.

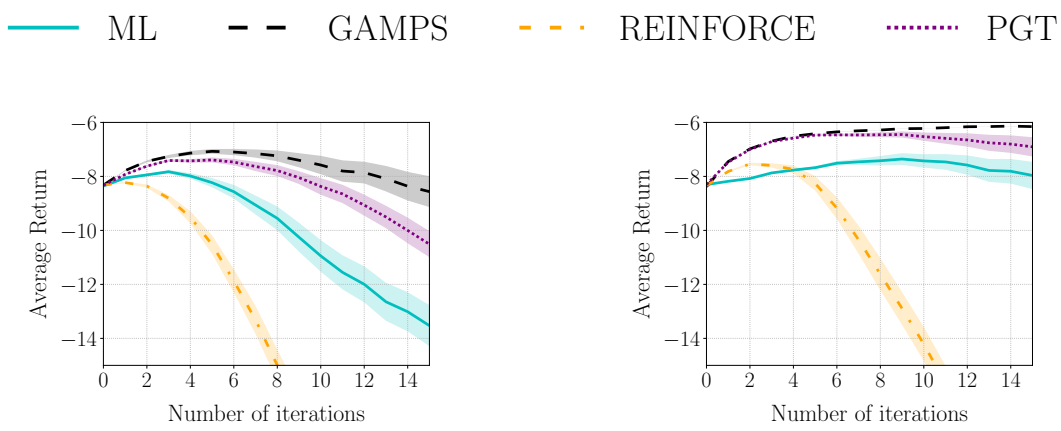

Figure 5.5: GAMPS performance in gridworld using 10 trajectories.

Figure 5.6: GAMPS performance in gridworld using 100 trajectories.

reduction of the variance provided by the PGT algorithm improves its performance, that are even superior than the ones of the version of GAMPS not using the gradient-aware weights. Generally, all the algorithms benefit from the increasing number of trajectories. However, it is worth noting that, with any amount of data we tested, the GAMPS learning curve is consistently above the others, showing superior performance considering the best iteration of any algorithm.

MINIGOLF

In the minigolf game, the agent hits a ball using a putter, with the goal of reaching a hole in the minimum number of trials. This problem was originally proposed for RL in [57], but we employ the dynamics of the more realistic model developed by [81]. More precisely, the goal of the agent is to shoot a ball with radius r inside a hole of diameter D, using a putter of length l. We assume that the ball moves along a level surface with a negative acceleration $d = \frac{5}{7}\rho g$, where ρ is the dynamic friction coefficient between the ball and the ground and g is determined by gravity. Given the distance x_0 of the ball from the hole, that constitutes the state seen by the agent, an angular velocity ω must be determined for the putter. This, in turn, determines the initial velocity $v_0 = \omega l$ to put the ball. For each distance x_0, the ball falls in the hole, yielding success for the agent, if its initial velocity v_0 ranges from $v_{min} = \sqrt{2dx_0}$ to $v_{max} = \sqrt{(2D - r)^2 \frac{g}{2r} + v_{min}^2}$. The ball is placed at random at the beginning of each episode, far from the hole between 0 and $2m$. Given an action a executed by the agent, the angular velocity of the putter is computed as follows: $\omega = al(1 + \epsilon)$, where $\epsilon \sim \mathcal{N}(0, 0.3)$. This implies that the stronger the action chosen the more uncertain its outcome will be. Thus, the agent is discouraged to attempt to score in a single shot when it is away from the hole and, if equipped with a policy near to the optimal one, will prefer to perform a sequence of low-intensity shots. Refer to [113] for a complete description of the dynamics of the environment. We divide the state space into two parts: the first one, bigger twice the other, is the nearest to the hole and features $\rho_1 = 0.131$; the second one is smaller and has an higher friction with $\rho_1 = 0.19$. Hence, the effect of each action significantly changes between these regions.

We use a linear-Gaussian policy that is linear on six equally-spaced radial basis function features. Four of the basis functions are therefore in the first area, while two are in the other one. The parameters of the policy are initialized equal to one for the mean and equal to zero for the standard deviation.

As a model class, we use parameterized linear-Gaussian models that predict the next state by difference with respect to the previous one, as often done in MBRL [20]. We enforce the fact that the next state is always to the left w.r.t. the current state by

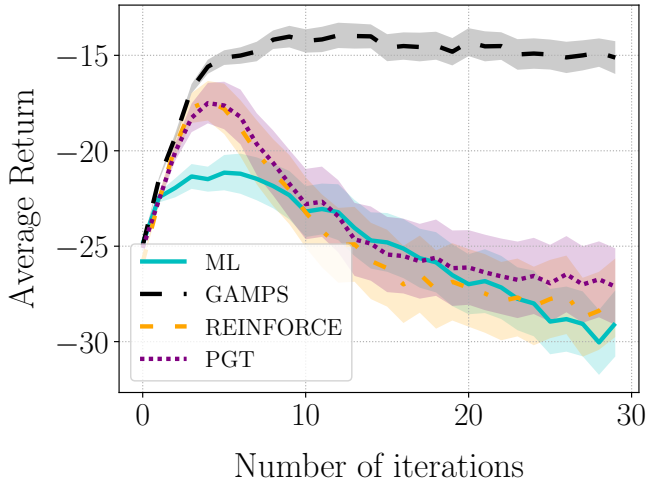

Figure 5.7: Performance of GAMPS in terms of average return using a 50 trajectories dataset on the minigolf environment (10 runs, mean ± std).

using a rectifier function. The overall prediction of the next state by the model is given by

$$\widehat{s}_{t+1} = s_t - \max(0, \epsilon), \epsilon \sim \mathcal{N}(V_\mu[s_t, a_t], V_\sigma[s_t, a_t]), \tag{5.3}$$

where V_μ and V_σ are two learnable parameters and $[\cdot, \cdot]$ is the concatenation operator.

We evaluate GAMPS against the same baselines employed for the previous experiments. We collect a dataset of 50 trajectories using an explorative policy. For training the model used by GAMPS and GAMPS-ML, we minimize the MSE weighted through our weighting scheme. For the estimation of the Q-function, we use the on-the-fly procedure outlined in Section 4.2.2, with an horizon of 20 and averaging over 10 rollouts. More details on the used hyperparameters are reported in Table 5.2.

The results, shown in Figure 5.7, show that GAMPS is able to reach a good performance, corresponding to a policy that allows the ball to reach the hole most of the times. The other algorithms, instead, are prone to overfitting after the optimized policy has become too different from the one that generated the data.

6 CONCLUSIONS

SUMMARY AND ADDITIONAL INSIGHTS

In this thesis, we presented our theoretical, algorithmic and experimental contributions to model-based reinforcement learning. MBRL has the potential to make the generality and flexibility of reinforcement learning scale to real world scenarios. This family of approaches presents advantages in terms of data efficiency, transfer, incorporation of prior knowledge, safety: all these features are extremely appealing when dealing with complex domains in the wild. Moreover, MBRL can be a point of conjunction for more traditional and well-established methods such as classical planning and optimal control, at the same time unleashing them to the next level, by injecting the fundamentally important ability to learn. However, although MBRL methods are in this sense highly promising, there are still several open challenges to be solved before they can show their true capabilities in these hard scenarios. These challenges touch many levels in the formulation of MBRL methods, concerning both how to learn the model of the dynamics and how to use it for obtaining a control policy.

Given the promising results achieved in the last years, a good research direction to look at concerns the use of policy gradient methods. In Chapter 3, we interpreted existing approaches through a novel lens, outlining a taxonomy and defining the FMG, MFG and MVG. The latter family of approaches, that combines the knowledge contained in a learned model with the one contained in the interaction data collected from the environment, can help in mitigating model bias, one of the crucial problems affecting MBRL.

As we discussed, most existing methods in MBRL ignore the question concerning the choice of an appropriate loss function to be used for learning about the dynamics of the environment from interaction data. In this thesis, we argued that this is instead a fundamental problem and that specific solutions are needed for taking advantage of all the information that is available about a task. In particular, Section 4.1 presented an analysis of the bias of the MVG, showing how it is related to the error introduced in the model learning phase. Interestingly, as Theorem 4.1 shows, this bias is not indiscriminately determined by the error in the whole state-action space, nor state-action couples are weighted only depending upon their visitation; instead, information about the policy followed by the agent, for instance provided by the magnitude of its score, greatly contributes in shaping the bias of the MVG. Note that, since different MVG-based algorithms have been recently shown to perform well in practice, Theorem 4.1 is of independent interest w.r.t. the algorithm proposed in Chapter 4.

A main contribution of this thesis is summarized by Proposition 4.2, which shows that, despite policy gradient methods are widespread in the MBRL community, their common choice of using maximum likelihood model estimation is not the optimal one when learning a model of the dynamics.

In Chapter 4, we built upon our theoretical analysis of the MVG to derive GAMPS, a practical model-based policy improvement algorithm designed for the batch setting. Although we focused on one specific instantiation, that unrolls the learned model for estimating the value function used then in the policy gradient estimation, our algorithm has an open ended nature and could be interpreted as a general framework. The finite-sample analysis of GAMPS provides additional insights, showing that, in case very few data is available, a good approach consists in using a tiny model class together with a decision-aware loss function.

Through the experiments presented in Chapter 5, we were mainly interested in proving two points. On the one hand, we reinforced the rationale behind MBRL, showing that the performance in prediction of the next state or any intermediate quantity is not an indicator of how useful a model will be for a learning algorithm. The only meaningful metric to be used for measuring the quality of a learned model in reinforcement learning should assess the resulting accuracy on the actual quantity used for improving the policy (in our case, the policy gradient). On the other hand, we showed that the decision-aware nature of GAMPS allows it to surpass the performance of both standard model-free methods and model-based approaches based on maximum likelihood estimation.

We argue that GAMPS, apart from being a decision-aware MBRL approach, can be interpreted as a form of *meta-learning*. If we consider the combination of model and policy to be part of a same agent, the model-learning step in GAMPS corresponds indeed to an update of an internal component (the model) as a function of the other component (the policy), with the help of the training data. The overall objective of this update is to make one module more helpful for learning another internal module: this is aligned with the meta-learning paradigm, in which an agent is aware of its own learning process and can intervene on it.

CURRENT LIMITATIONS AND FUTURE WORK

The contributions presented in this thesis are open to several extensions and improvements. In this section, we outline the ones we believe being particularly interesting.

Extension to the online setting

Our proposed method is made for a batch reinforcement learning setting. However, there is no theoretical limitation nor practical barrier to an extension to the online case. Therefore, future work could focus on adapting GAMPS to the interactive scenario, for

instance mixing on-policy and off-policy experience, in order to make the algorithm scale on complex high-dimensional environments. We feel that an interesting problem concerns the correct way to mix on-policy and off-policy experience. In particular, since GAMPS makes use of importance sampling for leveraging off-policy data, the information on the variance of the importance weights could be used for understanding when to stop the exploitation of a given set of collected experience and start to collect new trajectories.

Other techniques for estimating Q

As briefly mentioned during the presentation of the algorithm, unrolling model, policy and reward function on the fly is not the only way to obtain an estimate of the action-value function to be used in computing the policy gradient. Although with the disadvantages implied by the choice of both a functional space and reasonable targets, a parametric estimate for the Q-function can be trained, as commonly done in many modern reinforcement learning algorithms.

In some sense, especially if we use an explicit function approximator for approximating the value function, GAMPS can be seen as an actor-critic algorithm, where the critic incorporates the gradient-awareness induced in the model.

Deeper Theoretical Analysis

As clearly shown in the formulation given by the Policy Gradient Theorem (Theorem 2.2), three factors determine the policy gradient. The first two, the state-action distribution induced by the policy and the score, only depend on the policy and the environmental dynamics. The third factor, consisting in the action-value function, also depends on the reward function. In our analysis of the MVG, we bounded the value function using the maximum reward, actually loosing the information about the reward function. This also propagates to our gradient-aware loss function, that is therefore reward-agnostic. While not being a problem in case of smooth reward, incorporating such an information to cope with arbitrary, possibly sparse, rewards can be a promising direction for future work.

Moreover, some refinements to our finite-sample analysis of GAMPS are possible. For instance, instead of assuming the actual action-value function under the estimated model, we could consider its estimation error in terms of unrolling horizon for the model; or, instead of assuming to have a way to evaluate the reward function without interaction with the environment, we could consider also the estimation error induced by the use of a trained function approximator in place of the actual reward.

Other gradient-aware MVGs

Another possible direction for future work consists in the generalization of our gradient-aware loss function to a totally different algorithm based on the MVG. For instance, it would be interesting to design a modified version the model-based Stochastic Value Gradient algorithm [43], also known as SVG(∞), that computes the policy gradient by backpropagating through the value function. The modification can in principle occur only on the objective function used for model learning, without any further modification to other aspects of the algorithm.

BIBLIOGRAPHY

[1] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. "Using inaccurate models in reinforcement learning." In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 1–8 (cit. on pp. 2, 22, 28, 33).

[2] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. "Learning to poke by poking: Experiential learning of intuitive physics." In: *Advances in Neural Information Processing Systems*. 2016, pp. 5074–5082 (cit. on p. 24).

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks." en. In: *International Conference on Machine Learning*. July 2017, pp. 214–223 (cit. on p. 20).

[4] Kavosh Asadi, Evan Cater, Dipendra Misra, and Michael L Littman. "Equivalence between wasserstein and value-aware model-based reinforcement learning." In: *arXiv preprint arXiv:1806.01265* (2018) (cit. on p. 24).

[5] Christopher G Atkeson and Juan Carlos Santamaria. "A comparison of direct and model-based reinforcement learning." In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4. IEEE. 1997, pp. 3557–3564 (cit. on p. 18).

[6] Kamyar Azizzadenesheli, Brandon Yang, Weitang Liu, Emma Brunskill, Zachary C Lipton, and Animashree Anandkumar. "Sample-Efficient Deep RL with Generative Adversarial Tree Search." In: *arXiv preprint arXiv:1806.05780* (2018) (cit. on p. 25).

[7] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J. Tomlin. "Goal-driven dynamics learning via Bayesian optimization." In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (2017), pp. 5168–5173 (cit. on pp. 3, 23).

[8] Christopher Bates, Peter Battaglia, Ilker Yildirim, and Joshua B Tenenbaum. "Humans predict liquid dynamics using probabilistic simulation." In: *CogSci*. 2015 (cit. on p. 18).

[9] Jonathan Baxter and Peter L Bartlett. "Infinite-horizon policy-gradient estimation." In: *Journal of Artificial Intelligence Research* 15 (2001), pp. 319–350 (cit. on pp. 15, 28, 35).

[10] Richard Bellman et al. "The theory of dynamic programming." In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515 (cit. on pp. 7, 36).

[11] Yoshua Bengio. "Using a Financial Training Criterion Rather than a Prediction Criterion." In: *International journal of neural systems* 8 4 (1997), pp. 433–43 (cit. on p. 23).

[12] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*. Vol. 1. 2. Athena scientific Belmont, MA, 1995 (cit. on pp. 23, 36).

[13] Leo Breiman. "Bagging predictors." In: *Machine learning* 24.2 (1996), pp. 123–140 (cit. on p. 20).

[14] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. "Sample-efficient reinforcement learning with stochastic ensemble value expansion." In: *Advances in Neural Information Processing Systems*. 2018, pp. 8224–8234 (cit. on pp. 21, 28, 33).

[15] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. "Large-scale study of curiosity-driven learning." In: *arXiv preprint arXiv:1808.04355* (2018) (cit. on p. 24).

[16] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. "Monte-Carlo Tree Search: A New Framework for Game AI." In: 2008 (cit. on p. 25).

[17] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." In: *Advances in Neural Information Processing Systems*. 2018, pp. 4754–4765 (cit. on pp. 20, 25).

[18] John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. "Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings." In: *arXiv preprint arXiv:1806.02813* (2018) (cit. on p. 21).

[19] Corinna Cortes, Spencer Greenberg, and Mehryar Mohri. "Relative deviation learning bounds and generalization with unbounded loss functions." In: *arXiv preprint arXiv:1310.5796* (2013) (cit. on pp. 39, 73).

[20] Marc Deisenroth and Carl E Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472 (cit. on pp. 6, 20, 27, 28, 49).

[21] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. "A survey on policy search for robotics." In: *Foundations and Trends® in Robotics* 2.1–2 (2013), pp. 1–142 (cit. on pp. 2, 3, 28, 35).

[22] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear independent components estimation." In: *arXiv preprint arXiv:1410.8516* (2014) (cit. on p. 19).

[23] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP." In: *arXiv:1605.08803 [cs, stat]* (May 2016) (cit. on p. 19).

[24] Bradley B Doll, Dylan A Simon, and Nathaniel D Daw. "The ubiquity of model-based reinforcement learning." In: *Current opinion in neurobiology* 22.6 (2012), pp. 1075–1081 (cit. on p. 8).

[25] Priya Donti, Brandon Amos, and J Zico Kolter. "Task-based end-to-end model learning in stochastic optimization." In: *Advances in Neural Information Processing Systems*. 2017, pp. 5484–5494 (cit. on p. 24).

[26] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. "Go-Explore: a New Approach for Hard-Exploration Problems." In: *arXiv preprint arXiv:1901.10995* (2019) (cit. on p. 81).

[27] Damien Ernst, Pierre Geurts, and Louis Wehenkel. "Tree-based batch mode reinforcement learning." In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 503–556 (cit. on pp. 9, 36).

[28] Amir-massoud Farahmand. "Iterative value-aware model learning." In: *Advances in Neural Information Processing Systems*. 2018, pp. 9072–9083 (cit. on pp. 3, 24).

[29] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. "Value-aware loss function for model-based reinforcement learning." In: *Artificial Intelligence and Statistics*. 2017, pp. 1486–1494 (cit. on pp. 3, 23, 24, 34).

[30] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. "Model-based value estimation for efficient model-free reinforcement learning." In: *arXiv preprint arXiv:1803.00101* (2018) (cit. on pp. 28, 33, 36).

[31] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. "Optimization based full body control for the atlas robot." In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2014, pp. 120–127 (cit. on p. 17).

[32] Jason Fischer, John G Mikhael, Joshua B Tenenbaum, and Nancy Kanwisher. "Functional neuroanatomy of intuitive physical inference." In: *Proceedings of the national academy of sciences* 113.34 (2016), E5072–E5081 (cit. on p. 18).

[33] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In: *international conference on machine learning*. 2016, pp. 1050–1059 (cit. on p. 20).

[34] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. "Improving PILCO with Bayesian neural network dynamics models." In: *Data-Efficient Machine Learning workshop, ICML*. Vol. 4. 2016 (cit. on pp. 20, 28).

[35] Ian Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks." In: *arXiv:1701.00160 [cs]* (Dec. 2016) (cit. on p. 20).

[36] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on p. 20).

[37] Alex Graves. "Generating sequences with recurrent neural networks." In: *arXiv preprint arXiv:1308.0850* (2013) (cit. on p. 19).

[38] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1291–1307 (cit. on p. 16).

[39] David Ha and Jürgen Schmidhuber. "Recurrent world models facilitate policy evolution." In: *Advances in Neural Information Processing Systems*. 2018, pp. 2450–2462 (cit. on p. 26).

[40] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 1861–1870 (cit. on p. 16).

[41] Jessica B Hamrick, Peter W Battaglia, Thomas L Griffiths, and Joshua B Tenenbaum. "Inferring mass in complex scenes by mental simulation." In: *Cognition* 157 (2016), pp. 61–76 (cit. on p. 18).

[42] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 25).

[43] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. "Learning continuous control policies by stochastic value gradients." In: *Advances in Neural Information Processing Systems*. 2015, pp. 2944–2952 (cit. on pp. 28, 33, 54).

[44] Joshua Mason Joseph, Alborz Geramifard, John W. Roberts, Jonathan P. How, and Nicholas Roy. "Reinforcement learning with misspecified model classes." In: *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 939–946 (cit. on pp. 3, 23).

[45] Herman Kahn and Andy W Marshall. "Methods of reducing sample size in Monte Carlo computations." In: *Journal of the Operations Research Society of America* 1.5 (1953), pp. 263–278 (cit. on p. 9).

[46] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011 (cit. on p. 8).

[47] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. "Model-Based Reinforcement Learning for Atari." In: *arXiv preprint arXiv:1903.00374* (2019) (cit. on p. 28).

[48] Mitsuo Kawato. "Internal models for motor control and trajectory planning." In: *Current opinion in neurobiology* 9.6 (1999), pp. 718–727 (cit. on p. 18).

[49] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. "Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 6250–6261 (cit. on p. 20).

[50] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 (cit. on p. 47).

[51] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions." In: *arXiv:1807.03039 [cs, stat]* (July 2018) (cit. on p. 19).

[52] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *arXiv:1312.6114 [cs, stat]* (Dec. 2013) (cit. on pp. 19, 28).

[53] Vijay R Konda and John N Tsitsiklis. "Actor-critic algorithms." In: *Advances in neural information processing systems*. 2000, pp. 1008–1014 (cit. on p. 15).

[54] Solomon Kullback and Richard A Leibler. "On information and sufficiency." In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86 (cit. on p. 22).

[55] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. "Model-Ensemble Trust-Region Policy Optimization." In: *International Conference on Learning Representations*. 2018 (cit. on pp. 21, 28).

[56] Sascha Lange, Thomas Gabel, and Martin Riedmiller. "Batch reinforcement learning." In: *Reinforcement learning*. Springer, 2012, pp. 45–73 (cit. on p. 9).

[57] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. "Reinforcement learning in continuous action spaces through sequential monte carlo methods." In: *Advances in neural information processing systems*. 2008, pp. 833–840 (cit. on p. 49).

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), p. 436 (cit. on p. 7).

[59] Sergey Levine and Pieter Abbeel. "Learning neural network policies with guided policy search under unknown dynamics." In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079 (cit. on pp. 2, 6, 22, 26).

[60] Sergey Levine and Vladlen Koltun. "Guided policy search." In: *International Conference on Machine Learning*. 2013, pp. 1–9 (cit. on p. 18).

[61] Weiwei Li and Emanuel Todorov. "Iterative linear quadratic regulator design for nonlinear biological movement systems." In: (cit. on p. 21).

[62] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971* (2015) (cit. on pp. 9, 11, 16).

[63] David JC MacKay. "A practical Bayesian framework for backpropagation networks." In: *Neural computation* 4.3 (1992), pp. 448–472 (cit. on p. 20).

[64] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. "Least squares generative adversarial networks." In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2813–2821 (cit. on p. 20).

[65] Alberto Maria Metelli, Mirco Mutti, and Marcello Restelli. "Configurable Markov Decision Processes." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 3488–3497 (cit. on p. 67).

[66] Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. "Policy optimization via importance sampling." In: *Advances in Neural Information Processing Systems*. 2018, pp. 5442–5454 (cit. on p. 10).

[67] John Milnor. *Dynamics in One Complex Variable.(AM-160):(AM-160)-*. Vol. 197. Princeton University Press, 2011 (cit. on p. 21).

[68] Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. "Prediction and Control with Temporal Segment Models." en. In: *International Conference on Machine Learning*. July 2017, pp. 2459–2468 (cit. on pp. 21, 25).

[69] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In: *International conference on machine learning*. 2016, pp. 1928–1937 (cit. on p. 16).

[70] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (2015), p. 529 (cit. on pp. 8, 17).

[71] Hans Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988 (cit. on p. 17).

[72] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. "Safe and efficient off-policy reinforcement learning." In: *Advances in Neural Information Processing Systems*. 2016, pp. 1054–1062 (cit. on p. 9).

[73] Kumpati S Narendra and Kannan Parthasarathy. "Identification and control of dynamical systems using neural networks." In: *IEEE Transactions on neural networks* 1.1 (1990), pp. 4–27 (cit. on pp. 28, 33).

[74] Duy Nguyen-Tuong and Jan Peters. "Model learning for robot control: a survey." In: *Cognitive processing* 12.4 (2011), pp. 319–340 (cit. on p. 1).

[75] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. "Model learning with local gaussian process regression." In: *Advanced Robotics* 23.15 (2009), pp. 2015–2034 (cit. on pp. 2, 22).

[76] Junhyuk Oh, Satinder Singh, and Honglak Lee. "Value prediction network." In: *Advances in Neural Information Processing Systems*. 2017, pp. 6118–6128 (cit. on pp. 3, 24).

[77] Yoko Ohtomo, Takeshi Kakegawa, Akizumi Ishida, Toshiro Nagase, and Minik T Rosing. "Evidence for biogenic graphite in early Archaean Isua metasedimentary rocks." In: *Nature Geoscience* 7.1 (2014), p. 25 (cit. on p. 17).

[78] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks." In: *ICML*. 2016 (cit. on p. 19).

[79] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013 (cit. on pp. 9, 47, 72).

[80] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. "Curiosity-driven exploration by self-supervised prediction." In: *International Conference on Machine Learning (ICML)*. Vol. 2017. 2017 (cit. on p. 24).

[81] AR Penner. "The physics of putting." In: *Canadian Journal of Physics* 80.2 (2002), pp. 83–96 (cit. on p. 49).

[82] Jan Peters and Stefan Schaal. "Reinforcement learning by reward-weighted regression for operational space control." In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 745–750 (cit. on pp. 14, 15, 83).

[83] L. Piroddi and W. Spinelli. "An identification algorithm for polynomial NARX models based on simulation error minimization." In: *International Journal of Control* 76.17 (2003), pp. 1767–1781. eprint: https://doi.org/10.1080/00207170310001635419 (cit. on p. 23).

[84] Doina Precup, Richard S. Sutton, and Satinder P. Singh. "Eligibility Traces for Off-Policy Policy Evaluation." In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, 2000, pp. 759–766 (cit. on p. 9).

[85] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014 (cit. on pp. 5, 6).

[86] Carl Edward Rasmussen. "Gaussian processes in machine learning." In: *Summer School on Machine Learning*. Springer. 2003, pp. 63–71 (cit. on p. 19).

[87] Alfréd Rényi et al. "On measures of entropy and information." In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California. 1961 (cit. on p. 10).

[88] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 2014, pp. 1278–1286 (cit. on p. 19).

[89] Martin Riedmiller. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method." In: *European Conference on Machine Learning*. Springer. 2005, pp. 317–328 (cit. on p. 36).

[90] Stéphane Ross and J Andrew Bagnell. "Agnostic system identification for model-based reinforcement learning." In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Omnipress. 2012, pp. 1905–1912 (cit. on pp. 23, 26, 35).

[91] Carina M Schlebusch, Helena Malmström, Torsten Günther, Per Sjödin, Alexandra Coutinho, Hanna Edlund, Arielle R Munters, Mário Vicente, Maryna Steyn, Himla Soodyall, et al. "Southern African ancient genomes estimate modern human divergence to 350,000 to 260,000 years ago." In: *Science* 358.6363 (2017), pp. 652–655 (cit. on p. 17).

[92] Jürgen Schmidhuber. "A possibility for implementing curiosity and boredom in model-building neural controllers." In: *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*. 1991, pp. 222–227 (cit. on p. 24).

[93] Jürgen Schmidhuber. "An on-line algorithm for dynamic reinforcement learning and planning in reactive environments." In: *1990 IJCNN international joint conference on neural networks*. IEEE. 1990, pp. 253–258 (cit. on p. 8).

[94] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural networks* 61 (2015), pp. 85–117 (cit. on p. 7).

[95] Jürgen Schmidhuber. "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)." In: *IEEE Transactions on Autonomous Mental Development* 2.3 (2010), pp. 230–247 (cit. on p. 8).

[96] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. "Model-Based Active Exploration." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, pp. 5779–5788 (cit. on pp. 21, 24).

[97] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." In: *Science* 362.6419 (2018), pp. 1140–1144 (cit. on p. 18).

[98] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic policy gradient algorithms." In: *ICML*. 2014 (cit. on p. 9).

[99] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." In: *nature* 529.7587 (2016), p. 484 (cit. on pp. 17, 18).

[100] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of go without human knowledge." In: *Nature* 550.7676 (2017), p. 354 (cit. on p. 18).

[101] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. "The predictron: End-to-end learning and planning." In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3191–3199 (cit. on pp. 3, 24).

[102] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018 (cit. on p. 19).

[103] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958 (cit. on p. 20).

[104] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. "Incentivizing exploration in reinforcement learning with deep predictive models." In: *arXiv preprint arXiv:1507.00814* (2015) (cit. on p. 24).

[105] Richard S. Sutton. "Dyna, an Integrated Architecture for Learning, Planning, and Reacting." In: *SIGART Bull.* 2.4 (July 1991), pp. 160–163. ISSN: 0163-5719 (cit. on pp. 8, 26).

[106] Richard S Sutton. "Learning to predict by the methods of temporal differences." In: *Machine learning* 3.1 (1988), pp. 9–44 (cit. on p. 15).

[107] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018 (cit. on pp. 1, 5, 7, 15, 18, 36).

[108] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. "Dyna-style planning with linear function approximation and prioritized sweeping." In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2008, pp. 528–536 (cit. on pp. 24, 26).

[109] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation." In: *Advances in Neural Information Processing Systems*. 2000, pp. 1057–1063 (cit. on pp. 3, 7, 12, 47, 69).

[110] Voot Tangkaratt, Syogo Mori, Tingting Zhao, Jun Morimoto, and Masashi Sugiyama. "Model-based policy gradients with parameter-based exploration by least-squares conditional density estimation." In: *Neural networks* 57 (2014), pp. 128–140 (cit. on p. 28).

[111] Yuval Tassa, Tom Erez, and Emanuel Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4906–4913 (cit. on p. 22).

[112] Sebastian B Thrun and Knut Möller. "Active exploration in dynamic environments." In: *Advances in neural information processing systems*. 1992, pp. 531–538 (cit. on p. 24).

[113] Andrea Tirinzoni, Mattia Salvini, and Marcello Restelli. "Transfer of Samples in Policy Search via Multiple Importance Sampling." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, pp. 6264–6274 (cit. on p. 49).

[114] George E Uhlenbeck and Leonard S Ornstein. "On the theory of the Brownian motion." In: *Physical review* 36.5 (1930), p. 823 (cit. on p. 11).

[115] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. "WaveNet: A generative model for raw audio." In: *SSW*. 2016, p. 125 (cit. on p. 19).

[116] Tim Van Erven and Peter Harremos. "Rényi divergence and Kullback-Leibler divergence." In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 3797–3820 (cit. on p. 10).

[117] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. "Starcraft ii: A new challenge for reinforcement learning." In: *arXiv preprint arXiv:1708.04782* (2017) (cit. on p. 17).

[118] Oskar Von Stryk and Roland Bulirsch. "Direct and indirect methods for trajectory optimization." In: *Annals of operations research* 37.1 (1992), pp. 357–373 (cit. on p. 25).

[119] Xin Wang and Thomas G Dietterich. "Model-based policy gradient reinforcement learning." In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 776–783 (cit. on p. 28).

[120] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8.3-4 (1992), pp. 279–292 (cit. on p. 9).

[121] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256 (cit. on pp. 12, 14, 47).

[122] Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. "Analysis and improvement of policy gradient estimation." In: *Advances in Neural Information Processing Systems*. 2011, pp. 262–270 (cit. on p. 15).

A

PROOFS AND DERIVATIONS

In this appendix, we report the proofs of the results presented in the thesis, together with some additional results and extended discussion.

VARIOUS PROOFS

In this section, we see the proofs of various results proven throughout the thesis, divided by sections.

Proofs of Section 4.1

The following lemma is used in proving Theorem 4.1.

Lemma A.1. *Considering the state-action distributions $\delta_\mu^{\pi,p}$ and $\delta_\mu^{\pi,\widehat{p}}$ under policy π and models p and $\widehat{p}$, the following upper bound holds:*

$$\left\|\delta_\mu^{\pi,p} - \delta_\mu^{\pi,\widehat{p}}\right\|_1 \leqslant \frac{\gamma}{1-\gamma} \mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}} \left[\|p(\cdot|s,a) - \widehat{p}(\cdot|s,a)\|_1\right].$$

Proof. Recalling that $\delta_\mu^{\pi,p}(s,a) = \pi(a|s) d_\mu^{\pi,p}(s)$ we can write:

$$\begin{aligned}
\left\|\delta_\mu^{\pi,p} - \delta_\mu^{\pi,\widehat{p}}\right\|_1 &= \int_{\mathcal{S}}\int_{\mathcal{A}} \left|\delta_\mu^{\pi,\widehat{p}}(s,a) - \delta_\mu^{\pi,\widehat{p}}(s,a)\right| ds da \\
&= \int_{\mathcal{S}}\int_{\mathcal{A}} \pi(a|s) \left|d_\mu^{\pi,p}(s) - d_\mu^{\pi,\widehat{p}}(s)\right| ds da \\
&= \int_{\mathcal{S}} \left|d_\mu^{\pi,p}(s) - d_\mu^{\pi,\widehat{p}}(s)\right| \int_{\mathcal{A}} \pi(a|s) da ds \\
&= \int_{\mathcal{S}} \left|d_\mu^{\pi,p}(s) - d_\mu^{\pi,\widehat{p}}(s)\right| ds = \left\|d_\mu^{\pi,p} - d_\mu^{\pi,\widehat{p}}\right\|_1,
\end{aligned}$$

where $d_\mu^{\pi,p}(s) = (1-\gamma)\sum_{t=0}^{+\infty} \gamma^t \Pr(s_t = s|\mathcal{M},\pi)$. In order to bound $\left\|d_\mu^{\pi,p} - d_\mu^{\pi,\widehat{p}}\right\|_1$, we can use Corollary 3.1 from [65]:

$$\left\|d_\mu^{\pi,p} - d_\mu^{\pi,\widehat{p}}\right\|_1 \leqslant \frac{\gamma}{1-\gamma} \mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}} \left[\|p(\cdot|s,a) - \widehat{p}(\cdot|s,a)\|_1\right].$$

■

Now, we can prove Theorem 4.1.

Theorem 4.1. *Let* $q \in [1, +\infty]$ *and* $\widehat{p} \in \mathcal{P}$*. Then, the* L^q*-norm of the difference between the policy gradient* $\nabla_\theta J(\theta)$ *and the corresponding MVG* $\nabla_\theta^{MVG} J(\theta)$ *can be upper bounded as:*

$$\left\|\nabla_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q \leqslant \frac{\gamma\sqrt{2} Z R_{max}}{(1-\gamma)^2} \sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))]},$$

where

$$\eta_\mu^{\pi,p}(s,a) = \frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi_\theta(a'|s')\right\|_q \delta_{s',a'}^{\pi,p}(s,a) ds' da'$$

is a probability distribution over $\mathcal{S}\times\mathcal{A}$ *and*

$$Z = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi_\theta(a'|s')\right\|_q ds' da'$$

is a normalization constant, both independent from $\widehat{p}$*.*

Proof.

$$\begin{aligned}
\left\|\nabla_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q &= \left\|\frac{1}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a)(Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a)) \right.\\
&\qquad \left. \times \nabla_\theta \log \pi(a|s) ds da\right\|_q \\
&\leqslant \frac{1}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a)\left|Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a)\right| \|\nabla_\theta \log \pi(a|s)\|_q ds da \qquad \text{(A.1)}\\
&= \frac{Z}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\left|Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a)\right| ds da \qquad \text{(A.2)}\\
&= \frac{Z}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\left|\int_{\mathcal{S}}\int_{\mathcal{A}} r(s',a')(\delta_{s,a}^{\pi,p}(s',a') - \delta_{s,a}^{\pi,\widehat{p}}(s',a'))ds' da'\right| ds da \qquad \text{(A.3)}\\
&\leqslant \frac{Z R_{max}}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\left|\int_{\mathcal{S}}\int_{\mathcal{A}} (\delta_{s,a}^{\pi,p}(s',a') - \delta_{s,a}^{\pi,\widehat{p}}(s',a'))ds' da'\right| ds da \qquad \text{(A.4)}\\
&\leqslant \frac{Z R_{max}}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\left\|\delta_{s,a}^{\pi,p} - \delta_{s,a}^{\pi,\widehat{p}}\right\|_1 ds da\\
&\leqslant \frac{Z R_{max}\gamma}{(1-\gamma)^2}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{s,a}^{\pi,p}(s',a') \qquad \text{(A.5)}\\
&\qquad \times \left\|p(\cdot|s',a') - \widehat{p}(\cdot|s',a')\right\| ds' da' ds da\\
&= \frac{Z R_{max}\gamma}{(1-\gamma)^2}\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_\mu^{\pi,p}(s',a')\int_{\mathcal{S}}\left|p(s''|s',a') - \widehat{p}(s''|s',a')\right| ds'' ds' da'\\
&\leqslant \frac{Z R_{max}\gamma}{(1-\gamma)^2}\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_\mu^{\pi,p}(s,a)\sqrt{2 D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))} ds da \qquad \text{(A.6)}\\
&\leqslant \frac{Z R_{max}\gamma}{(1-\gamma)^2}\sqrt{2\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_\mu^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a)) ds da}, \qquad \text{(A.7)}
\end{aligned}$$

where in Equation (A.2), we define a new probability distribution $\nu_\mu^{\pi,p}(s,a) = \frac{1}{Z}\delta_\mu^{\pi,p}(s,a)\|\nabla_\theta \log \pi(a|s)\|_q$ by means of an appropriate normalization constant Z, assumed $Z > 0$. In Equation (A.3), we use the definition of Q-function as $Q^{\pi,p}(s,a) = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{s,a}^{\pi,p}(s',a')r(s',a')ds'da'$. After bounding the reward in Equation (A.4), in Equation (A.5) we apply Lemma A.1. Then we obtain Equation (A.6) by employing Pinsker's inequality, defining the overall weighting term

$$\eta_\mu^{\pi,p}(s',a') = \int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a)\delta_{s,a}^{\pi,p}(s',a')dsda,$$

and renaming variables for clarity. Last passage follows from Jensen inequality. ■

Proofs of Section 4.2

We start introducing the following lemma that states that taking expectations w.r.t. $\delta_\mu^{\pi,p}$ is equivalent to taking proper expectations w.r.t. $\zeta_\mu^{\pi,p}$.

Lemma A.2. *Let $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^k$ an arbitrary function defined over the state-action space. Then, it holds that:*

$$\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}[f(s,a)] = (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t \mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi,p}}[f(s_t,a_t)] = (1-\gamma)\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi,p}}\left[\sum_{t=0}^{+\infty}\gamma^t f(s_t,a_t)\right]. \tag{A.8}$$

Proof. We denote with $\mathcal{T}$ the set of all possible trajectories. We just apply the definition of $\delta_\mu^{\pi,p}$ [109]:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}[f(s,a)] = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a)f(s,a)dsda \\
&= (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t\int_{\mathcal{S}}\int_{\mathcal{A}} \Pr(s_t = s,\, a_t = a|\mathcal{M},\,\pi)\, f(s,a)dsda \\
&= (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t\int_{\mathcal{S}}\int_{\mathcal{A}} \left(\int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t})\mathbb{1}(s_t = s,\, a_t = a)\, d\tau_{0:t}\right) f(s,a)dsda \\
&= (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t\int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t})\left(\int_{\mathcal{S}}\int_{\mathcal{A}} \mathbb{1}(s_t = s,\, a_t = a)\, f(s,a)dsda\right) d\tau_{0:t} \\
&= (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t\int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t})f(s_t,a_t)d\tau_{0:t} \\
&= (1-\gamma)\int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau)\sum_{t=0}^{+\infty}\gamma^t f(s_t,a_t)d\tau,
\end{aligned}$$

where we exploited the fact that the probability $\Pr(s_t = s,\, a_t = a|\mathcal{M}, \pi)$ is equal to the probability that a prefix of trajectory $\tau_{0:t}$ terminates in (s_t, a_t), i.e.,

$$\int_{\mathcal{T}} \zeta_{\mu}^{\pi,p}(\tau_{0:t}) \mathbb{1}(s_t = s,\, a_t = a)\, d\tau_{0:t}.$$

The last passage follows from the fact that $f(s_t, a_t)$ depends on random variables realized at time t we can take the expectation over the whole trajectory. ■

We can apply this result to rephrase the expectation w.r.t. $\eta_{\mu}^{\pi,p}$ as an expectation w.r.t. $\zeta_{\mu}^{\pi,p}$.

Lemma A.3. *Let $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$ an arbitrary function defined over the state-action space. Then, it holds that:*

$$\mathop{\mathbb{E}}_{s,a\sim\eta_{\mu}^{\pi,p}}[f(s,a)] = \frac{(1-\gamma)^2}{Z} \mathop{\mathbb{E}}_{\tau\sim\zeta_{\mu}^{\pi,p}}\left[\sum_{t=0}^{+\infty}\gamma^t \sum_{l=0}^{t} \|\nabla_\theta \log \pi(a_l|s_l)\|_q\, f(s_t,a_t)\right]. \quad \text{(A.9)}$$

Proof. We just need to apply Lemma A.2 twice and exploit the definition of $\eta_{\mu}^{\pi,p}$:

$$\begin{aligned}\mathop{\mathbb{E}}_{s,a\sim\eta_{\mu}^{\pi,p}}[f(s,a)] &= \int_{\mathcal{S}}\int_{\mathcal{A}} \eta_{\mu}^{\pi,p}(s,a) f(s,a) ds da \\ &= \frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{\mu}^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi(a'|s')\right\|_q \\ &\qquad \times \delta_{s',a'}^{\pi,p}(s,a) ds' da' f(s,a) ds da.\end{aligned}$$

Let us first focus on the expectation taken w.r.t. $\delta_{\mu}^{\pi,p}(s',a')$. By applying Lemma A.2 with $f(s',a') = \|\nabla_\theta \log \pi(a'|s')\|_q\, \delta_{s',a'}^{\pi,p}(s,a)$, we have:

$$\begin{aligned}&\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{\mu}^{\pi,p}(s',a') \left\|\nabla_\theta \log \pi(a'|s')\right\|_q \delta_{s',a'}^{\pi,p}(s,a) ds' da' \\ &= (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t \int_{\mathcal{T}} \zeta_{\mu}^{\pi,p}(\tau_{0:t}) \left\|\nabla_\theta \log \pi(a_t|s_t)\right\|_q \delta_{s_t,a_t}^{\pi,p}(s,a) d\tau_{0:t}.\end{aligned}$$

Now, let us consider $\delta^{\pi,p}_{s_t,a_t}(s,a)$. We instantiate again Lemma A.2:

$$\begin{aligned}
\mathop{\mathbb{E}}_{s,a\sim\eta^{\pi,p}_{\mu}}[f(s,a)] &= \frac{(1-\gamma)}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}}\sum_{t=0}^{+\infty}\gamma^t \int_{\mathcal{T}} \zeta^{\pi,p}_{\mu}(\tau_{0:t})\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q \\
&\qquad\qquad \times \delta^{\pi,p}_{s_t,a_t}(s,a) f(s,a) d\tau_{0:t} ds da \\
&= \frac{(1-\gamma)}{Z}\sum_{t=0}^{+\infty}\gamma^t \int_{\mathcal{T}} \zeta^{\pi,p}_{\mu}(\tau_{0:t})\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q \\
&\qquad\qquad \times \int_{\mathcal{S}}\int_{\mathcal{A}} \delta^{\pi,p}_{s_t,a_t}(s,a) f(s,a) ds da d\tau_{0:t} \\
&= \frac{(1-\gamma)^2}{Z}\sum_{t=0}^{+\infty}\gamma^t \int_{\mathcal{T}} \zeta^{\pi,p}_{\mu}(\tau_{0:t})\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q \\
&\qquad\qquad \times \sum_{l=0}^{+\infty}\gamma^l \int_{\mathcal{T}} \zeta^{\pi,p}_{s_t,a_t}(\tau_{0:l}) f(s_l,a_l) d\tau_{0:l} d\tau_{0:t} \\
&= \frac{(1-\gamma)^2}{Z}\int_{\mathcal{T}} \zeta^{\pi,p}_{\mu}(\tau)\sum_{t=0}^{+\infty}\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q \sum_{h=t}^{+\infty}\gamma^h f(s_h,a_h) d\tau,
\end{aligned}$$

where the last passage derives from observing that, for each t and l we are computing an integral over the trajectory prefixes of length $h := t+l$ and observing that (s_l, a_l) can be seen as the h-th state-action pair of a trajectory $\tau \sim \zeta^{\pi,p}_{\mu}$. We now rearrange the summations:

$$\sum_{t=0}^{+\infty}\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q \sum_{h=t}^{+\infty}\gamma^h f(s_h,a_h) = \sum_{h=0}^{+\infty}\gamma^h f(s_h,a_h)\sum_{t=0}^{h}\left\|\nabla_\theta \log\pi(a_t|s_t)\right\|_q .$$

By changing the names of the indexes of the summations, we get the result. ■

We are now ready to prove Lemma 4.3.

Lemma A.4. *Let π and π_b be two policies such that $\pi \ll \pi_b$ (π is absolutely continuous w.r.t. to π_b). Let $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^k$ be an arbitrary function defined over the state-action space. Then, it holds that:*

$$\mathop{\mathbb{E}}_{s,a\sim\eta^{\pi,p}_{\mu}}[f(s,a)] = \frac{(1-\gamma)^2}{Z}\mathop{\mathbb{E}}_{\tau\sim\zeta^{\pi_b,p}_{\mu}}\left[\sum_{t=0}^{+\infty}\gamma^t \rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\left\|\nabla_\theta \log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right].$$

Proof. What changes w.r.t. Lemma A.3 is that we are now interested in computing the expectation w.r.t. to a target policy π while trajectories are collected with a behavioral

policy π_b, fulfilling the hypothesis stated in the lemma. We start from Lemma 4.3 and we just need to apply importance weighting [79]:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi,p}}\left[\sum_{t=0}^{+\infty}\gamma^t\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right]\\
&=\sum_{t=0}^{+\infty}\gamma^t\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi,p}}\left[\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right]\\
&=\sum_{t=0}^{+\infty}\gamma^t\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi_b,p}}\left[\frac{\zeta_\mu^{\pi,p}(\tau_{0:t})}{\zeta_\mu^{\pi_b,p}(\tau_{0:t})}\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right]\\
&=\sum_{t=0}^{+\infty}\gamma^t\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi_b,p}}\left[\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right]\\
&=\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right].
\end{aligned}
$$

■

Proofs of Section 4.3

Under Assumption 1, we prove the following intermediate result about the objective function in Equation (4.5).

Lemma A.5. *Let $\widehat{p}\in\mathcal{P}$ be the maximizer of the objective function in Equation (4.5), obtained with $N>0$ independent trajectories $\{\tau^i\}_{i=1}^N$. Under Assumption 1 and 2, for any $\delta\in(0,1)$, with probability at least $1-2\delta$ it holds that:*

$$
\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widehat{p}}(\tau)\right]\geqslant\sup_{\overline{p}\in\mathcal{P}}\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\overline{p}}(\tau)\right]-4c_1\epsilon, \tag{A.10}
$$

where $\epsilon=\sqrt{\frac{v\log\frac{2eN}{v}+\log\frac{4}{\delta}}{N}}\Gamma\left(\sqrt{\frac{v\log\frac{2eN}{v}+\log\frac{4}{\delta}}{N}}\right)$ *and* $\Gamma(\xi):=\frac{1}{2}+\sqrt{1+\frac{1}{2}\log\frac{1}{\xi}}=\widetilde{\mathcal{O}}(1)$.

Proof. We use a very common argument of empirical risk minimization. Let us denote with $\widetilde{p}\in\arg\max_{\overline{p}\in\mathcal{P}}\mathbb{E}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\overline{p}}(\tau)\right]$ and $\widehat{\mathcal{L}}^{\pi,\overline{p}}=\frac{1}{N}\sum_{i=1}^N l^{\pi,\overline{p}}(\tau^i)$:

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widehat{p}}(\tau)\right]-\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widetilde{p}}(\tau)\right]&=\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widehat{p}}(\tau)\right]-\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widetilde{p}}(\tau)\right]\pm\widehat{\mathcal{L}}^{\pi,\widehat{p}}\\
&\geqslant\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widehat{p}}(\tau)\right]-\widehat{\mathcal{L}}^{\pi,\widehat{p}}\\
&\quad-\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widetilde{p}}(\tau)\right]+\widehat{\mathcal{L}}^{\pi,\widetilde{p}}\\
&\geqslant-2\sup_{\overline{p}\in\mathcal{P}}\left|\widehat{\mathcal{L}}^{\pi,\overline{p}}-\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\overline{p}}(\tau)\right]\right|,
\end{aligned}
$$

where we exploited the fact that $\widehat{\mathcal{L}}^{\pi,\widetilde{p}} \leqslant \widehat{\mathcal{L}}^{\pi,\widehat{p}}$, as $\widehat{p}$ is the maximizer of $\widehat{\mathcal{L}}^{\pi,\cdot}$. The result follows from the application of Corollary 14 in [19], having bounded the growth function with the pseudodimension, as in Corollary 18 of [19]. ■

We can derive a concentration result for the gradient estimation (Equation (4.9)), recalling the fact that $\mathbf{g}^{\pi,\overline{p}}$ is a vectorial function.

Lemma A.6. *Let $q \in [1,+\infty]$, d be the dimensionality of Θ and $\widehat{p} \in \mathcal{P}$ be the maximizer of the objective function in Equation (4.5), obtained with $N > 0$ independent trajectories $\{\tau^i\}_{i=1}^N$. Under Assumption 1 and 2, for any $\delta \in (0,1)$, with probability at least $1-2d\delta$, simultaneously for all $\overline{p} \in \mathcal{P}$, it holds that:*

$$\left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q \leqslant 2d^{\frac{1}{q}} R_{max} c_2 \epsilon, \tag{A.11}$$

where $\epsilon = \sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{4}{\delta}}{N}} \Gamma\left(\sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{4}{\delta}}{N}}\right)$ *and* $\Gamma(\xi) := \frac{1}{2} + \sqrt{1 + \frac{1}{2}\log\frac{1}{\xi}} = \widetilde{\mathcal{O}}(1)$.

Proof. We observe that $\widehat{\nabla}_\theta J(\theta)$ is the sample version of $\nabla_\theta^{MVG} J(\theta)$. Under Assumption 1 and 2, and using Corollary 14 in [19] as in Lemma A.5, we can write for any $j = 1, \dots d$ the following bound that holds with probability at least $1-2\delta$, simultaneously for all $\widehat{p} \in \mathcal{P}$:

$$\left|\widehat{\nabla}_{\theta_j} J(\theta) - \nabla_{\theta_j}^{MVG} J(\theta)\right| \leqslant 2 R_{max} c_2 \epsilon. \tag{A.12}$$

Considering the L^q-norm, and plugging the previous equation, we have that with probability at least $1-2d\delta$ it holds that, simultaneously for all $\widehat{p} \in \mathcal{P}$:

$$\left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q = \left(\sum_{j=1}^{d} \left|\nabla_{\theta_j}^{MVG} J(\theta) - \nabla_{\theta_j} J(\theta)\right|^q\right)^{\frac{1}{q}} \leqslant 2 d^{\frac{1}{q}} R_{max} c_2 \epsilon,$$

having exploited a union bound over the dimensions d. ■

We are now ready to prove the main result.

Theorem 4.5. *Let $q \in [1,+\infty]$, d be the dimensionality of Θ and $\widehat{p} \in \mathcal{P}$ be the maximizer of the objective function in Equation (4.5), obtained with $N > 0$ independent trajectories $\{\tau^i\}_{i=1}^N$. Under Assumption 1 and 2, for any $\delta \in (0,1)$, with probability at least $1-4\delta$ it holds that:*

$$\left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta J(\theta)\right\|_q \leqslant \underbrace{\frac{\gamma\sqrt{2}ZR_{max}}{(1-\gamma)^2} \inf_{\overline{p}\in\mathcal{P}} \sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a)\|\overline{p}(\cdot|s,a))]}}_{\textit{approximation error}}$$
$$+ \underbrace{2R_{max}\left(d^{\frac{1}{q}} c_2 \epsilon + \frac{\gamma\sqrt{2Zc_1\epsilon}}{1-\gamma}\right)}_{\textit{estimation error}},$$

given

$$\epsilon = \sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{8(d+1)}{\delta}}{N}} \Gamma\left(\sqrt{\frac{\nu \log \frac{2eN}{\nu} + \log \frac{8(d+1)}{\delta}}{N}}\right),$$

$$\Gamma(\xi) \coloneqq \frac{1}{2} + \sqrt{1 + \frac{1}{2}\log \frac{1}{\xi}}.$$

Proof. Let us first consider the decomposition, that follows from triangular inequality:

$$\begin{aligned}\left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta J(\theta)\right\|_q &= \left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta J(\theta) \pm \nabla_\theta^{MVG} J(\theta)\right\|_q \\ &\leqslant \underbrace{\left\|\widehat{\nabla}_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q}_{\text{(i)}} + \underbrace{\left\|\nabla_\theta^{MVG} J(\theta) - \nabla_\theta J(\theta)\right\|_q}_{\text{(ii)}}.\end{aligned}$$

We now bound each term of the right hand side. (i) is bounded in Lemma A.6. Let us now consider (ii). We just need to apply Theorem 4.3 and Lemma A.5, recalling the properties of the KL-divergence. From Theorem 4.1:

$$\left\|\nabla_\theta^{\mathrm{MVG}} J(\theta) - \nabla_\theta J(\theta)\right\|_q \leqslant \frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2}\sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}\left[D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\right]}$$

$$= \frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2}\Bigg(\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}\Bigg[\int_S p(s'|s,a)\log p(s'|s,a)ds' - \int_S p(s'|s,a)\log\widehat{p}(s'|s,a)ds'\Bigg]\Bigg)^{\frac{1}{2}} \tag{A.13}$$

$$= \frac{\gamma\sqrt{2Z}R_{\max}}{(1-\gamma)}\sqrt{\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\omega_t\left(\log p(s_{t+1}|s_t,a_t) - \log\widehat{p}(s_{t+1}|s_t,a_t)\right)\right]} \tag{A.14}$$

$$= \frac{\gamma\sqrt{2Z}R_{\max}}{(1-\gamma)}\sqrt{\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\omega_t\log p(s_{t+1}|s_t,a_t)\right] - \mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\widehat{p}}(\tau)\right]}$$

$$\leqslant \frac{\gamma\sqrt{2Z}R_{\max}}{(1-\gamma)}\sqrt{\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\omega_t\log p(s_{t+1}|s_t,a_t)\right] - \sup_{\overline{p}\in\mathcal{P}}\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[l^{\pi,\overline{p}}(\tau)\right] + 4c_1\epsilon} \tag{A.15}$$

$$= \frac{\gamma\sqrt{2Z}R_{\max}}{(1-\gamma)}\Bigg(\inf_{\overline{p}\in\mathcal{P}}\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\omega_t\left(\log p(s_{t+1}|s_t,a_t) - \log\overline{p}(s_{t+1}|s_t,a_t)\right)\right] + 4c_1\epsilon\Bigg)^{\frac{1}{2}} \tag{A.16}$$

$$\leqslant \frac{\gamma\sqrt{2Z}R_{\max}}{(1-\gamma)}\sqrt{\inf_{\overline{p}\in\mathcal{P}}\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\sum_{t=0}^{+\infty}\omega_t\left(\log p(s_{t+1}|s_t,a_t) - \log\overline{p}(s_{t+1}|s_t,a_t)\right)\right]} + \frac{2\gamma R_{\max}\sqrt{2Zc_1\epsilon}}{1-\gamma} \tag{A.17}$$

$$= \frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2}\sqrt{\inf_{\overline{p}\in\mathcal{P}}\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}\left[D_{KL}(p(\cdot|s,a)\|\overline{p}(\cdot|s,a))\right]} + \frac{2\gamma R_{\max}\sqrt{2Zc_1\epsilon}}{1-\gamma}, \tag{A.18}$$

where Equation (A.13) and Equation (A.18) follow from the definition of KL-divergence and Lemma 4.3. Equation (A.14) is derived from Lemma 4.3 where

$$\omega_t = \gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q.$$

Equation (A.15) is obtained by applying Lemma A.5. Equation (A.17) follows from the subadditivity of the square root. Putting together (i) and (ii) we get the result that holds with probability at least $1-2(d+1)\delta$ as bound (i) holds w.p. $1-2\delta$ and bound (ii) w.p. $1-2d\delta$. By rescaling δ we get the result. ■

GRADIENT-UNAWARE MODEL LEARNING

We now show that maximum-likelihood model estimation is a sound way of estimating the policy gradient when using the MVG, although it is optimizing a looser bound with respect to the one provided by Theorem 4.1. For proving the following result, we assume the score is bounded by $\|\nabla_\theta \log \pi(a|s)\|_q \leqslant K$.

Proposition A.7. *Let* $q \in [1, +\infty]$ *and* $\widehat{p} \in \mathcal{P}$. *If* $\|\nabla_\theta \log \pi(a|s)\|_q \leqslant K$ *for all* $s \in \mathcal{S}$ *and* $s \in \mathcal{A}$, *then, the* L^q*-norm of the difference between the policy gradient* $\nabla_\theta J(\theta)$ *and the corresponding MVG* $\nabla_\theta^{MVG} J(\theta)$ *can be upper bounded as:*

$$\begin{aligned}\|\nabla_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\|_q &\leqslant \frac{\gamma\sqrt{2}ZR_{max}}{(1-\gamma)^2}\sqrt{\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}[D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))]} \\ &\leqslant \frac{\gamma\sqrt{2}KR_{max}}{(1-\gamma)^2}\sqrt{\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}[D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))]}.\end{aligned}$$

Proof.

$$\begin{aligned}\left\|\nabla_\theta J(\theta) - \nabla_\theta^{MVG} J(\theta)\right\|_q &\leqslant \frac{\gamma\sqrt{2}ZR_{max}}{(1-\gamma)^2}\left(\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_\mu^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))dsda\right)^{\frac{1}{2}} \\ &= \frac{\gamma\sqrt{2}ZR_{max}}{(1-\gamma)^2}\Bigg(\int_{\mathcal{S}}\int_{\mathcal{A}}\frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}} \|\nabla_\theta \log\pi(a'|s')\|_q \delta_\mu^{\pi,p}(s',a')\delta_{s',a'}^{\pi,p}(s,a)ds'da' \\ &\qquad \times D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))dsda\Bigg)^{\frac{1}{2}} \qquad (A.19) \\ &\leqslant \frac{\gamma\sqrt{2KZ}R_{max}}{(1-\gamma)^2}\Bigg(\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a')\delta_{s',a'}^{\pi,p}(s,a)ds'da' \\ &\qquad \times D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))dsda\Bigg)^{\frac{1}{2}} \qquad (A.20) \\ &= \frac{\gamma\sqrt{2KZ}R_{max}}{(1-\gamma)^2}\left(\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))dsda\right)^{\frac{1}{2}} \qquad (A.21) \\ &\leqslant \frac{\gamma\sqrt{2}KR_{max}}{(1-\gamma)^2}\left(\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))dsda\right)^{\frac{1}{2}}, \qquad (A.22)\end{aligned}$$

where we started from Theorem 4.1. Equation (A.21) follows from the fact that $\int \delta_\mu^{\pi,p}(s',a')\delta_{s',a'}^{\pi,p}(s,a)ds'da' = \delta_\mu^{\pi,p}(s,a)$, as we are actually recomposing the state-

action distribution that was split at (s', a') and Equation (A.22) is obtained by observing that $Z \leqslant K$. ■

We can observe that maximum likelihood provides a looser bound w.r.t. to the one provided by Theorem 4.1. This reflects the fact that the standard approach for model learning in MBRL does not make use of all the available information, in this case related to the gradient of the current agent policy.

WEIGHTED KL DIVERGENCE

Minimizing D_{KL} between real and estimated model is equivalent to to maximizing the weighted log-likelihood of the data collected under the real model. This is formally stated in the following proposition.

Proposition A.8. *Given an arbitrary weighting function $\xi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, this equality holds:*

$$\begin{aligned}
&\arg\min_{\overline{p}} \int \xi(s,a) D_{KL}(p(s'|s,a) \| \overline{p}(s'|s,a)) ds da \\
&= \arg\max_{\overline{p}} \lim_{N\to\infty} \frac{1}{N} \sum_{\substack{i=1 \\ s'_i \sim p(\cdot|s_i,a_i)}}^{N} \xi(s,a) \log \overline{p}(s'_i|s_i,a_i)
\end{aligned}$$

Proof.

$$\begin{aligned}
&\arg\min_{\overline{p}} \int \xi(s,a) D_{KL}(p(s'|s,a) \| \overline{p}(s'|s,a)) ds' ds da \\
&= \arg\min_{\overline{p}} \int \xi(s,a) \int p(s'|s,a) \log \frac{p(s'|s,a)}{\overline{p}(s'|s,a)} ds' ds da \\
&= \arg\min_{\overline{p}} \int \xi(s,a) \Big(\int p(s'|s,a) \log p(s'|s,a) ds' \\
&\qquad\qquad - \int p(s'|s,a) \log \overline{p}(s'|s,a) ds' \Big) ds da \\
&= \arg\min_{\overline{p}} - \int \xi(s,a) \int p(s'|s,a) \log \overline{p}(s'|s,a) ds' ds da \\
&= \arg\max_{\overline{p}} \int \xi(s,a) \int p(s'|s,a) \log \overline{p}(s'|s,a) ds' ds da \\
&= \arg\max_{\overline{p}} \int p(s'|s,a) \xi(s,a) \log \overline{p}(s'|s,a) ds' ds da \\
&= \arg\max_{\overline{p}} \lim_{N\to\infty} \frac{1}{N} \sum_{\substack{i=1 \\ s'_i \sim p(\cdot|s_i,a_i)}}^{N} \xi(s_i,a_i) \log \overline{p}(s'_i|s_i,a_i)
\end{aligned}$$

■

In the case of GAMPS, the weighting function is given by $\xi(s,a) = \eta_\mu^{\pi,p}(s,a)$.

DETAILS ABOUT THE ASSUMPTION

Assumption 1 is equivalent to require that there exists two finite constants $c_1 < +\infty$ and $c_2 < +\infty$ such that:

$$\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\sum_{t=0}^{+\infty}\gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\log p(s_{t+1}|s_t,a_t)\right)^2\right]\leqslant c_1^2, \tag{A.23}$$

$$\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\sum_{t=0}^{+\infty}\gamma^t\rho_{\pi/\pi_b}(\tau_{0:t})\nabla_{\theta_j}\log\pi(a_t|s_t)Q^{\pi,\overline{p}}(s_t,a_t)\right)^2\right]\leqslant R_{\max}^2c_2^2, j\in[1..d]. \tag{A.24}$$

We now state the following result that allows decoupling Assumption 1 into two separate conditions for the policies π and π_b and the transition models p (the real one) and $\overline{p}$ (the approximating one).

Corollary A.9. *Assumption 1 is satisfied if there exist three constants χ_1, χ_2 and χ_3, with $\chi_1 < \frac{1}{\gamma}$.*

$$\sup_{\pi\in\Pi_\Theta}\sup_{s\in\mathcal{S}}\mathop{\mathbb{E}}_{a\sim\pi_b(\cdot|s)}\left[\left(\frac{\pi(a|s)}{\pi_b(a|s)}\right)^2\right]\leqslant\chi_1,$$

$$\sup_{\pi\in\Pi_\Theta}\sup_{s\in\mathcal{S}}\mathop{\mathbb{E}}_{a\sim\pi_b(\cdot|s)}\left[\left(\frac{\pi(a|s)}{\pi_b(a|s)}\|\nabla_\theta\log\pi(a|s)\|_q^2\right)^2\right]\leqslant\chi_2,$$

$$\sup_{\overline{p}\in\mathcal{P}}\sup_{\substack{s\in\mathcal{S}\\a\in\mathcal{A}}}\mathop{\mathbb{E}}_{s'\sim p(\cdot|s,a)}\left[\left(\log\overline{p}(s'|s,a)\right)^2\right]\leqslant\chi_3.$$

In such case, Equation (A.23) and Equation (A.24) are satisfied with constants:

$$c_1^2 = \frac{\chi_3\chi_2(1+\gamma\chi_1)}{(1-\gamma)(1-\gamma\chi_1)^3},\quad c_2^2 = \frac{\chi_3\chi_2}{(1-\gamma)^3(1-\gamma\chi_1)}.$$

Proof. Let us start with Equation (A.23). We first apply Cauchy Swartz inequality to bring the expectation inside the summation:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\left[\left(\sum_{t=0}^{+\infty}\gamma^{\frac{t}{2}}\cdot\gamma^{\frac{t}{2}}\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\log p(s_{t+1}|s_t,a_t)\right)^2\right]\\
&\quad\leqslant\sum_{t=1}^{+\infty}\gamma^t\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\Big[\sum_{t=0}^{+\infty}\gamma^t\Big(\rho_{\pi/\pi_b}(\tau_{0:t})\\
&\qquad\qquad\times\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\log p(s_{t+1}|s_t,a_t)\Big)^2\Big]\\
&\quad\leqslant\frac{1}{1-\gamma}\sum_{t=0}^{+\infty}\gamma^t\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\Big[\Big(\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\\
&\qquad\qquad\times\log p(s_{t+1}|s_t,a_t)\Big)^2\Big].
\end{aligned}$$

Let us fix a timestep t. We derive the following bound:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\left[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\sum_{l=0}^{t}\|\nabla_\theta\log\pi(a_l|s_l)\|_q\log p(s_{t+1}|s_t,a_t)\right)^2\right]\\
&=\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\left[\left(\sum_{l=0}^{t}\rho_{\pi/\pi_b}(\tau_{0:t})\|\nabla_\theta\log\pi(a_l|s_l)\|_q\right)^2(\log p(s_{t+1}|s_t,a_t))^2\right]\\
&\leqslant\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\left[(t+1)\sum_{l=0}^{t}\left(\rho_{\pi/\pi_b}(\tau_{0:t})\|\nabla_\theta\log\pi(a_l|s_l)\|_q\right)^2(\log p(s_{t+1}|s_t,a_t))^2\right],
\end{aligned}$$

where we applied Cauchy-Swartz inequality to bound the square of the summation. We now rewrite the expectation in a convenient form to highlight the different components.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b},p}\left[(t+1)\sum_{l=0}^{t}\left(\rho_{\pi/\pi_b}(\tau_{0:t})\|\nabla_\theta\log\pi(a_l|s_l)\|_q\right)^2(\log p(s_{t+1}|s_t,a_t))^2\right]\\
&\quad=(t+1)\sum_{l=0}^{t}\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi_b},p}\Big[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\|\nabla_\theta\log\pi(a_l|s_l)\|_q\right)^2\\
&\qquad\qquad\times\mathop{\mathbb{E}}_{s_{t+1}\sim p(\cdot|s_t,a_t)}\left[(\log p(s_{t+1}|s_t,a_t))^2\right]\Big]\\
&\quad\leqslant(t+1)\chi_3\sum_{l=0}^{t}\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi_b},p}\left[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\|\nabla_\theta\log\pi(a_l|s_l)\|_q\right)^2\right].
\end{aligned}$$

Let us fix l and bound the expectation inside the summation, by unrolling the trajectory and recalling the definition of $\rho_{\pi/\pi_b}(\tau_{0:t})$:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi_b,p}}\left[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\left\|\nabla_\theta \log\pi(a_l|s_l)\right\|_q\right)^2\right] \\
&\quad= \mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_0\sim\pi(\cdot|s_0)}}\left[\left(\frac{\pi(a_0|s_0)}{\pi_b(a_0|s_0)}\right)^2 \mathop{\mathbb{E}}_{\substack{s_1\sim p(\cdot|s_0,a_0)\\ a_1\sim\pi(\cdot|s_1)}}\left[\left(\frac{\pi(a_1|s_1)}{\pi_b(a_1|s_1)}\right)^2\cdots\right.\right. \\
&\qquad\times \mathop{\mathbb{E}}_{\substack{s_l\sim p(\cdot|s_{l-1},a_{l-1})\\ a_l\sim\pi(\cdot|s_l)}}\left[\left(\frac{\pi(a_l|s_l)}{\pi_b(a_l|s_l)}\left\|\nabla_\theta\log\pi(a_l|s_l)\right\|_q\right)^2\right. \\
&\qquad\left.\left.\cdots \mathop{\mathbb{E}}_{\substack{s_t\sim p(\cdot|s_{t-1},a_{t-1})\\ a_t\sim\pi(\cdot|s_t)}}\left[\left(\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}\right)^2\right]\cdots\right]\cdots\right] \\
&\quad\leqslant \chi_2\chi_1^t.
\end{aligned}$$

Plugging this result in the summation we get the result, recalling that $\gamma\chi_1 < 1$ and using the properties of the geometric series, we obtain:

$$\frac{1}{1-\gamma}\sum_{t=0}^{+\infty}(t+1)^2\gamma^t\chi_1^t\chi_2\chi_3 = \frac{\chi_3\chi_2(1+\gamma\chi_1)}{(1-\gamma)(1-\gamma\chi_1)^3}.$$

We now consider Equation (A.24) and we apply Cauchy Swartz as well:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\sum_{t=0}^{+\infty}\gamma^{\frac{t}{2}}\cdot\gamma^{\frac{t}{2}}\rho_{\pi/\pi_b}(\tau_{0:t})\nabla_{\theta_j}\log\pi(a_t|s_t)Q^{\pi,\overline{p}}(s_t,a_t)\right)^2\right] \\
&\qquad\leqslant \frac{1}{1-\gamma}\sum_{t=0}^{+\infty}\gamma^t \mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\nabla_{\theta_j}\log\pi(a_t|s_t)Q^{\pi,\overline{p}}(s_t,a_t)\right)^2\right].
\end{aligned}$$

By observing that $\left|Q^{\pi,\overline{p}}(s_t,a_t)\right| \leqslant \frac{R_{max}}{1-\gamma}$ and $\left|\nabla_{\theta_j}\log\pi(a_t|s_t)\right| \leqslant \left\|\nabla_\theta\log\pi(a_t|s_t)\right\|_\infty \leqslant \left\|\nabla_\theta\log\pi(a_t|s_t)\right\|_q$ we can use an argument similar to the one used to bound Equation (A.23) to get:

$$\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi_b,p}}\left[\left(\rho_{\pi/\pi_b}(\tau_{0:t})\nabla_{\theta_j}\log\pi(a_t|s_t)Q^{\pi,\overline{p}}(s_t,a_t)\right)^2\right] \leqslant \frac{R_{max}^2}{(1-\gamma)^2}\chi_2\chi_1^t.$$

Plugging this result into the summation, we have:

$$\frac{1}{1-\gamma}\sum_{t=0}^{+\infty}\gamma^t\chi_1^t\chi_2\chi_3\frac{R_{max}^2}{(1-\gamma)^2} = \frac{\chi_3\chi_2R_{max}^2}{(1-\gamma)^3(1-\gamma\chi_1)}.$$

■

B

DETAILS ON THE ALGORITHM

ALTERNATIVE DERIVATION

In this section, we provide an alternative derivation for our gradient-aware weighting scheme for model learning, starting from Theorem 4.1.

To make the weighting for transitions suggested by Theorem 4.1 practical, we have to choose a sampling strategy for two distributions, namely $\nu_\mu^{\pi,p}(s,a)$, as defined in Equation 4.1, and $\delta_{s,a}^{\pi,p}(s',a')$.

Concerning $\nu_\mu^{\pi,p}(s,a)$, we can recall its definition and use importance sampling, adopting $\delta_\mu^{\pi,p}(s,a)$ as the proposal distribution. For a generic function $f(s,a)$, we have:

$$\begin{aligned}\mathop{\mathbb{E}}_{s,a\sim\nu_\mu^{\pi,p}}[f(s,a)] &= \mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}\left[\frac{\nu_\mu^{\pi,p}(s,a)}{\delta_\mu^{\pi,p}(s,a)}f(s,a)\right] \\ &= \mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}\left[\frac{\|\nabla_\theta \log\pi(a|s)\|}{Z}f(s,a)\right].\end{aligned} \tag{B.1}$$

The importance weight is therefore the normalized norm of the score. In practice, we will be interested in finding the model parameters that maximize the likelihood objective implied by the KL-divergence: we can therefore ignore the computation of the normalizing constant Z, that does not depend on them.

Sampling from $\delta_{s,a}^{\pi,p}(s',a')$ is harder. In principle, to obtain independent samples, we should restart the environment in state s, execute action a and then observe the subsequent (s,a) tuples; however, even if sometimes used in reinforcement learning (e.g., for exploration [26]), restarting the environment to an arbitrary state is often impossible or at least very expensive. Therefore, we propose an approximate sampling procedure, based on the reuse of transitions belonging to the same trajectory. For instance, given a trajectory $\tau = (s_0, a_0, s_1, a_1, ..., s_t, a_t, ...) \in \mathcal{D}$, we consider (s_t, a_t) a sample from $\delta_{s_0,a_0}^{\pi,p}$ as well as a sample from $\delta_{s_1,a_1}^{\pi,p}$, both (s_1,a_1) and (s_t,a_t) as samples from $\delta_{s_0,a_0}^{\pi,p}$.

Let us assume, for the moment, that τ has been generated by policy π. For computing the overall weighting factor for transition $(s_t, a_t, a_{t+1}) \in \tau$, we should consider the contribution of every preceding transition in τ. Let us fix our attention, for instance, on transition $(s_\ell, a_\ell, s_{\ell+1}) \in \tau$, with $\ell < t$: we weight the sample collected according to $\delta_\mu^{\pi,p}(s_\ell,a_\ell)$ with a partial factor of $\gamma^\ell\|\nabla_\theta \log\pi(a_\ell|s_\ell)\|$, given by discounting and the above discussed importance sampling on $\nu_\mu^{\pi,p}(s_\ell,a_\ell)$. Then, we must consider (s_t,a_t) as a sample from $\delta_{s_\ell,a_\ell}^{\pi,p}(s,a)$: to do it, we simply apply the discount by multiplying by

another factor $\gamma^{t-\ell}$, given that transition (s_t, a_t) comes after $t-\ell$ steps. The overall weight is obtained by summing the contribution of all transitions in τ_i that come before (s_t, a_t, a_{t+1}):

$$\tilde{\omega}_t = \sum_{\ell=0}^{t} \gamma^{\ell} \|\nabla_\theta \log \pi(a_\ell|s_\ell)\| \gamma^{t-\ell} = \gamma^t \sum_{\ell=0}^{t} \|\nabla_\theta \log \pi(a_\ell|s_\ell)\|. \tag{B.2}$$

In Equation B.2, we assumed the policy with respect to which we compute the weights to be the same as the one that generated trajectory τ. However, the data has been generated by a policy π_b, in general different from π, and the computation of the weights ω_t happens in an off-policy scenario. Therefore, we must consider the importance weights for given trajectories when performing sampling from $\delta_\mu^{\pi,p}(s,a)$ and from $\delta_{s,a}^{\pi,p}(s',a')$. Hence, the weights for learning the model must be modified accordingly:

$$\begin{aligned} \omega_t &= \gamma^t \sum_{\ell=0}^{t} \prod_{m=0}^{\ell} \frac{\pi(a_m|s_m)}{\pi_b(a_m|s_m)} \|\nabla_\theta \log \pi(a_\ell|s_\ell)\| \prod_{r=\ell+1}^{t} \frac{\pi(a_r|s_r)}{\pi_b(a_r|s_r)} = \\ &= \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}) \sum_{\ell=0}^{t} \|\nabla_\theta \log \pi(a_\ell|s_\ell)\|. \end{aligned} \tag{B.3}$$

Once computed the weights ω_t^i for all the transitions in our dataset, we find the forward model by minimizing the weighted Kullback-Liebler divergence between it and the actual environment model. In practice, since the problem is equivalent to maximizing the weighted log-likelihood of observed transitions (as shown in Appendix A.3), we solve the following optimization problem:

$$\widehat{p} = \arg\max_{\overline{p} \in \mathcal{P}} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T^{(i)}-1} \omega_t^i \log \overline{p}\left(s_{t+1}^i | s_t^i, a_t^i\right) \tag{B.4}$$

where T^i denotes the length of trajectory τ_i and N is the total number of collected transitions.

TIME COMPLEXITY

Let us consider that the algorithm is run for K iterations on a dataset of N trajectories. Suppose a parametric model class for which at most E epochs are necessary for estimation. We define H as the maximum length of a trajectory (or *horizon*) and use an estimate of the Q-function derived by sampling M trajectories from the estimated model, as described in Section 4.2.2. For every iteration, we first compute the weights for every transition in every trajectory $\mathcal{O}(NH)$ and then estimate the corresponding forward model (order of NHE). Then, we estimate the gradient given all the transitions,

using the trajectories imagined by the model for obtaining the value function (order of NMH^2). The overall time complexity of the algorithm is therefore $\mathcal{O}(KNHE + KNMH^2)$.

A connection with reward-weighted regression

Interestingly, our gradient-aware procedure for model learning has some connections with the reward-weighted regression (RWR) [82] techniques, that solve reinforcement learning problems by optimizing a supervised loss. To see this, we shall totally revert our perspective on a non-Markovian decision process. First, we interpret a model $\widehat{p}_\phi$ parameterized by ϕ as a *policy*, whose action is to pick a new state after observing a previous state-action combination. Then, we see the policy π as the *model*, that samples the transition to the next state given the output of $\widehat{p}_\phi$. Finally, the cumulative absolute score at time t is the (non-markovian) reward. To strengthen the parallel, let us consider an appropriate transformation u_c on the weights ω_t.

We can now give an expectation-maximization formulation for our model learning problem as reward-weighted regression in this newly defined decision process:
E-step:

$$q_{k+1}(t) = \frac{p_{\phi_k}(s_{t+1}|s_t, a_t)u_{c_k}(\omega_t)}{\sum_{t'} p_{\phi_k}(s_{t'+1}|s_{t'}, a_{t'})u_{c_k}(\omega_{t'})} \tag{B.5}$$

M-step for model parameters:

$$\phi_{k+1} = \arg\max \sum_t q_{k+1}(t) \log p_\phi(s_{t+1}|s_t, a_t) \tag{B.6}$$

M-step for transformation coefficient:

$$\tau_{k+1} = \arg\max_c \sum_t q_{k+1}(t) u_c(\omega_t) \tag{B.7}$$

Assuming a Gaussian-linear model $\widehat{p} = \mathcal{N}(s_{t+1}|\mu(s_t, a_t), \sigma^2\mathbf{I})$ and a transformation $u_c(x) = c\exp(-cx)$, the update for the model parameters and the transformation parameter is given by:

$$\phi_{k+1} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W Y \tag{B.8}$$

$$\sigma^2_{k+1} = \|Y - \phi^T_{k+1}\Phi\|^2_W \tag{B.9}$$

$$c_{k+1} = \frac{\sum_t u_c(\omega_t)}{\sum_{t'} u_c(\omega_{t'})\omega_{t'}} \tag{B.10}$$

where Φ, Y and W are the matrices containing, respectively, state-action features, successor state features and cumulative score weights on the diagonal.

As in the case of the original RWR, this learned exponentiation of the weights could in practice improve the performance of our algorithm. We leave this direction to future work.