Gradient-Aware Model-Based Policy Search

Pierluca D'Oro,* **Alberto Maria Metelli,***
Andrea Tirinzoni, Matteo Papini, Marcello Restelli
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
Piazza Leonardo da Vinci, 32, 20133, Milano, Italy
pierluca.doro@mail.polimi.it, {albertomaria.metelli, andrea.tirinzoni, matteo.papini, marcello.restelli}@polimi.it

Abstract

Traditional model-based reinforcement learning approaches learn a model of the environment dynamics without explicitly considering how it will be used by the agent. In the presence of misspecified model classes, this can lead to poor estimates, as some relevant available information is ignored. In this paper, we introduce a novel model-based policy search approach that exploits the knowledge of the current agent policy to learn an approximate transition model, focusing on the portions of the environment that are most relevant for policy improvement. We leverage a weighting scheme, derived from the minimization of the error on the model-based policy gradient estimator, in order to define a suitable objective function that is optimized for learning the approximate transition model. Then, we integrate this procedure into a batch policy improvement algorithm, named Gradient-Aware Model-based Policy Search (GAMPS), which iteratively learns a transition model and uses it, together with the collected trajectories, to compute the new policy parameters. Finally, we empirically validate GAMPS on benchmark domains analyzing and discussing its properties.

1 Introduction

Model-Based Reinforcement Learning (MBRL, Sutton and Barto 2018; Nguyen-Tuong and Peters 2011) approaches use the interaction data collected in the environment to estimate its dynamics, with the main goal of improving the sample efficiency of Reinforcement Learning (RL, Sutton and Barto 2018) algorithms. However, modeling 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. Fortunately, in many interesting application domains (e.g., robotics), perfectly capturing the dynamics across the whole state-action space is not necessary for a model to be effectively used by a learning agent (Abbeel, Quigley, and Ng 2006; Nguyen-Tuong, Seeger, and Peters 2009; Levine and Abbeel 2014). Indeed, a wiser approach 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 (Deisenroth et al. 2013). Nonetheless, the relative importance of the different aspects of the dynamics greatly depends on the underlying decision problem, on the control approach, and, importantly, on the policy played by the agent. Recent work (Farahmand, Barreto, and Nikovski 2017; Farahmand 2018) shows that, in the context of value-based reinforcement learning methods, it is possible to derive a *decision-aware* loss function for model learning that compares favorably against maximum likelihood. However, there exists no equivalent for policy-based methods, that are often preferred in the case of continuous observation/action spaces. Moreover, previous work fails at incorporating the influence of the current agent's behavior for evaluating the relative importance of the different aspects of the world dynamics. 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 model learning that explicitly accounts for the current policy and for how it will be improved can outperform traditional maximum likelihood estimation.

In this paper, motivated by these observations, we propose a model-based policy search (Deisenroth et al. 2013; Sutton et al. 2000) 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 in model learning. We choose to focus our discussion on the batch setting (Lange, Gabel, and Riedmiller 2012),

*Equal contribution.

due to its particular real-world importance. Nonetheless, extensions to the interactive scenario can be easily derived. The contributions of this paper are theoretical, algorithmic and experimental. After having introduced our notation and the required mathematical preliminaries (Section 2), we formalize the concept of *Model-Value-based Gradient* (MVG), an approximation of the policy gradient that combines real trajectories along with a value function derived from an estimated model (Section 3). MVG allows finding a compromise between the large variance of a Monte Carlo gradient estimate and the bias of a full model-based estimator. Contextually, we present a bound on how the bias of the MVG is related to the choice of an estimated transition model. In Section 4, we derive from this bound an optimization problem to be solved, using samples, to obtain a gradient-aware forward model. Then, we integrate it into a batch policy optimization algorithm, named *Gradient-Aware Model-based Policy Search* (GAMPS), that iteratively uses samples to learn the approximate forward model and to estimate the gradient, used to perform the policy improvement step. After that, we present a finite-sample analysis for the single step of GAMPS (Section 5), that highlights the advantages of our approach when considering simple model classes. Finally, after reviewing related work in model-based policy search and decision-aware MBRL areas (Section 6), we empirically validate GAMPS against model-based and model-free baselines, and discuss its peculiar features (Section 7). The proofs of all the results presented in the paper are reported in Appendix.[1]

2 Preliminaries

A discrete-time Markov Decision Process (MDP, Puterman 2014) 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, $r(s,a)$ is the reward received by executing action a in state s, $p(\cdot|s,a)$ is the transition model that provides the distribution of the next state when performing action a in state s, μ is the distribution of the initial state and $\gamma \in [0,1)$ is a discount factor. When needed, we assume that r is known, as common in domains where MBRL is employed (e.g., robotic learning (Deisenroth and Rasmussen 2011)), and that rewards are uniformly bounded by $|r(s,a)| \leq R_{\max} < +\infty$. 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. Given a state-action pair (s,a) we define the action-value function (Sutton and Barto 2018), or Q-function, as $Q^{\pi,p}(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} p(s'|s,a) \int_{\mathcal{A}} \pi(a'|s') Q^{\pi,p}(s',a') \mathrm{d}s' \mathrm{d}a'$ and the state-value function, or V-function, as $V^{\pi,p}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi,p}(s,a)]$, where we made explicit the dependence on the policy π and on the transition model p. The goal of the agent is to find an optimal policy π^*, i.e., a policy that maximizes the *expected return*: $J^{\pi,p} = \mathbb{E}_{s_0\sim\mu}[V^{\pi,p}(s_0)]$.

We consider a batch setting (Lange, Gabel, and Riedmiller 2012), in which the learning is performed on a previously collected dataset $\mathcal{D} = \left\{\tau^i\right\}_{i=1}^N = \left\{\left(s_0^i, a_0^i, s_1^i, a_1^i, ..., s_{T_i-1}^i, a_{T_i-1}^i, s_{T_i}^i\right)\right\}_{i=1}^N$ of N independent trajectories τ^i, each composed of T_i transitions, and further interactions with the environment are not allowed. The experience is generated by an agent that interacts with the environment, following a *known* behavioral policy π_b. We are interested in learning a parameterized policy $\pi_{\boldsymbol{\theta}}$ (for which we usually omit the parameter subscript in the notation) that belongs to a parametric space of stochastic differentiable policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d\}$. In this case, the gradient of the expected return w.r.t. $\boldsymbol{\theta}$ is provided by the *policy gradient theorem* (PGT) (Sutton et al. 2000; Sutton and Barto 2018):

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

where $\delta_\mu^{\pi,p}(s,a)$ is the γ-discounted state-action distribution (Sutton et al. 2000), defined as $\delta_\mu^{\pi,p}(s,a) = (1-\gamma) \sum_{t=0}^{+\infty} \gamma^t \Pr(s_t = s, a_t = a|\mathcal{M}, \pi)$. We call $\nabla_{\boldsymbol{\theta}} \log \pi(a|s)$ the *score* of the policy π when executing action a in state s. Furthermore, we denote with $\delta_{s',a'}^{\pi,p}(s,a)$ the state-action distribution under policy π and model p when the environment is deterministically initialized by executing action a' in state s' and with $\zeta_\mu^{\pi,p}(\tau)$ the probability density function of a trajectory τ. In batch policy optimization, the policy gradient is typically computed for a policy π that is different from the policy π_b having generated the data (*off-policy* estimation (Precup, Sutton, and Singh 2000)). To correct for the distribution mismatch, we employ *importance sampling* (Kahn and Marshall 1953; Owen 2013), re-weighing the transitions based on the probability of being observed under the policy π. Namely, we 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 $\rho_{\pi/\pi_b}(\tau_{t':t''}) = \frac{\zeta_\mu^{\pi,p}(\tau_{t':t''})}{\zeta_\mu^{\pi_b,p}(\tau_{t':t''})} = \prod_{t=t'}^{t''} \frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}$.

3 Model-Value-based Gradient

The majority of the model-based policy search approaches employ the learned forward model for generating rollouts, which are used to compute an improvement direction $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ either via likelihood-ratio methods or by propagating gradients through the model (Deisenroth and Rasmussen 2011). Differently from these methods, we consider an approximation of the gradient, named *Model-Value-based Gradient* (MVG), defined as follows.

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

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{MVG}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) \nabla_{\boldsymbol{\theta}} \log \pi(a|s) \times Q^{\pi,\widehat{p}}(s,a) \mathrm{d}s \mathrm{d}a. \tag{2}$$

Thus, the MVG employs experience collected in the real environment p, i.e., sampling from $\delta_\mu^{\pi,p}(s,a)$, and uses the

[1] The full version of the paper, including the appendix, is available at *https://arxiv.org/abs/1909.04115*.

generative power of the estimated transition kernel $\widehat{p}$ in the computation of an approximate state-action value function $Q^{\pi,\widehat{p}}$ only. In this way, it is possible to find a compromise between a full model-based estimator, in which the experience is directly generated from $\delta_\mu^{\pi,\widehat{p}}$ (Deisenroth and Rasmussen 2011; Deisenroth et al. 2013), and a Monte Carlo estimator (e.g., GPOMDP (Baxter and Bartlett 2001)) in which also the Q-function is computed from experience collected in the real environment. Therefore, the MVG limits the bias effect of $\widehat{p}$ to the Q-function approximation $Q^{\pi,\widehat{p}}$.[2] At the same time, it enjoys a smaller variance w.r.t. a Monte Carlo estimator, especially in an off-policy setting, as the Q-function is no longer estimated from samples but just approximated using $\widehat{p}$. Existing approaches can be interpreted as MVG. For instance, the ones based on model-based value expansion (Feinberg et al. 2018; Buckman et al. 2018), that use a fixed-horizon unrolling of an estimated forward model for obtaining a better value function in an actor-critic setting.

A central question concerning Definition 3.1, 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 3.2. *Let $q \in [1, +\infty]$ and $\widehat{p} \in \mathcal{P}$. Then, the L^q-norm of the difference between the policy gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ and the corresponding MVG $\nabla_{\boldsymbol{\theta}}^{\mathrm{MVG}} J(\boldsymbol{\theta})$ can be upper bounded as:*

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

where

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

is a probability distribution over $\mathcal{S}\times\mathcal{A}$, with $\nu_\mu^{\pi,p}(s',a') = \frac{1}{Z}\delta_\mu^{\pi,p}(s',a')\left\|\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a'|s')\right\|_q$ and $Z = \int_{\mathcal{S}}\int_{\mathcal{A}}\delta_\mu^{\pi,p}(s',a')\left\|\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a'|s')\right\|_q \mathrm{d}s'\mathrm{d}a'$ both not depending on $\widehat{p}$.[3]

Proof Sketch. Since $Q^{\pi,p}(s,a) = \int \delta_{s,a}^{\pi,p}(s',a')r(s',a')\,\mathrm{d}s'\mathrm{d}a'$, we bound the Q-function difference with $\left\|\delta_{s,a}^{\pi,p} - \delta_{s,a}^{\pi,\widehat{p}}\right\|_1$. The latter is upper bounded with $\left\|p(\cdot|s',a') - \widehat{p}(\cdot|s',a')\right\|_1$. The result follows from Pinsker's inequality. □

Similarly to what was noted for other forms of decision-aware MBRL (Farahmand, Barreto, and Nikovski 2017), a looser bound in which the expectation on the KL-divergence is taken under $\delta_\mu^{\pi,p}$ can be derived (see Appendix). This motivates the common maximum likelihood approach. However, our bound is tighter and clearly shows that not all collected transitions have the same relevance when learning a model that is used in estimating the MVG. Overall, the most important (s,a) pairs are those that are *likely to be reached from the policy starting from high gradient-magnitude state-action pairs*.

4 Gradient-Aware Model-based Policy Search

Inspired by Theorem 3.2, 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 model $\widehat{p}$ (Section 4.1), computing the value function $Q^{\pi,\widehat{p}}$ (Section 4.2) and updating the policy using the estimated gradient $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ (Section 4.3).

4.1 Learning the Transition Model

To learn $\widehat{p}$, we aim at minimizing the bound in Theorem 3.2, 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 3.2, 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 general result.

Lemma 4.1. *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:*

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

To specialize Lemma 4.1 for our specific case, we just 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 ignored in the minimization procedure. Furthermore, minimizing the KL-divergence is equivalent to maximizing the log-likelihood of the observed transitions. Putting everything together, we obtain the objective:

$$\widehat{p} \in \arg\max_{\overline{p}\in\mathcal{P}} \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_i-1}\omega_t^i \log\overline{p}\left(s_{t+1}^i|s_t^i,a_t^i\right), \qquad \omega_t^i = \gamma^t\rho_{\pi/\pi_b}(\tau_{0:t}^i)\sum_{l=0}^{t}\left\|\nabla_{\boldsymbol{\theta}}\log\pi(a_l^i|s_l^i)\right\|_q. \tag{3}$$

[2] 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.

[3] 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 $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{0}$ and there is no need to define $\eta_\mu^{\pi,p}$ in this case.

The factors contained in the weight ω_t^i accomplish three goals in weighting the transitions. 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 (Deisenroth et al. 2013), an accurate approximation of the dynamics for the regions that are rarely reached by the current policy is not useful. Lastly, the factor $\sum_{l=0}^{t} \left\| \nabla_{\boldsymbol{\theta}} \log \pi(a_l^i|s_l^i) \right\|_q$ favors those 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 (Baxter and Bartlett 2001). 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.

4.2 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. This 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 either in closed form or in an iterative manner via dynamic programming (Bellman and others 1954; Sutton and Barto 2018). For continuous MDPs, $Q^{\pi,\widehat{p}}$ cannot, in general, be represented exactly. A first approach consists of employing a function approximator $\widehat{Q} \in \mathcal{Q}$ and applying approximate dynamic programming (Bertsekas et al. 1995). 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 $\widehat{p}$ (Ernst, Geurts, and Wehenkel 2005; Riedmiller 2005), possibly introducing further bias.

We instead encourage the use of $\widehat{p}$ as a generative model for the sole purpose of approximating $Q^{\pi,\widehat{p}}$. 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 $Q^{\pi,\widehat{p}}$ on the fly, in an unbiased way, averaging the return from a (possibly large) number M of imaginary trajectories obtained from the estimated model $\widehat{p}$:

$$\widehat{Q}(s,a) = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=0}^{T_j-1} \gamma^t r(s_t^j, a_t^j), \quad \tau^j \sim \zeta_{s,a}^{\pi,\widehat{p}}. \tag{4}$$

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

4.3 Estimating the policy gradient

After computing $Q^{\pi,\widehat{p}}$ (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 (2) we obtain:

$$\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}^i) \nabla_{\boldsymbol{\theta}} \log \pi(a_t^i|s_t^i) \times Q^{\pi,\widehat{p}}(s_t^i, a_t^i). \tag{5}$$

For performing batch policy optimization, we 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 of 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 1.

Algorithm 1 Gradient-Aware Model-based Policy Search

Input: Trajectory dataset $\mathcal{D}$, behavior policy π_b, initial parameters $\boldsymbol{\theta}_0$, step size schedule $(\alpha_k)_{k=0}^{K-1}$

for $k = 0, 1, ..., K-1$ **do**
- ▷ Learn $\widehat{p}$ (Section 4.1)
- $\omega_{t,k}^i \leftarrow \gamma^t \rho_{\pi_{\boldsymbol{\theta}_k}/\pi_b}(\tau_{0:t}^i) \sum_{l=0}^{t} \|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_l^i|s_l^i)\|_q$
- $\widehat{p}_k \leftarrow \arg\max_{\overline{p} \in \mathcal{P}} \frac{1}{N} \sum_{i=1}^{N} \sum_t \omega_{t,k}^i \log \overline{p}(s_{t+1}^i|s_t^i, a_t^i)$
- ▷ Compute $\widehat{Q}$ (Section 4.2)
- Generate M trajectories for each (s,a) using $\widehat{p}_k$
- $\widehat{Q}_k(s,a) = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=0}^{T_j-1} \gamma^t r(s_t^j, a_t^j)$
- ▷ Improve Policy (Section 4.3)
- $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k) \leftarrow \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi_{\boldsymbol{\theta}_k}/\pi_b}(\tau_{0:t}^i) \times \times \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t^i|s_t^i) \widehat{Q}_k(s_t^i, a_t^i)$
- $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k)$

end for

5 Theoretical Analysis

In this section, we provide a finite-sample bound for the gradient estimation of Equation (5), assuming to have the exact value of $Q^{\pi,\widehat{p}}$. This corresponds to the analysis of a single iteration of GAMPS. We first define the following functions. Let τ be a trajectory, $\pi \in \Pi_\Theta$ and $\overline{p} \in \mathcal{P}$. We define $l^{\pi,\overline{p}}(\tau) = \sum_{t=0}^{+\infty} \omega_t \log \overline{p}(s_{t+1}|s_t, a_t)$ and $\mathbf{g}^{\pi,\overline{p}}(\tau) = \sum_{t=0}^{+\infty} \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}) \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t) Q^{\pi,\overline{p}}(s_t, a_t)$. To obtain our result, we need 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 for all $\overline{p} \in \mathcal{P}$ and $\pi \in \Pi_\Theta$:*

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

$$\max \left\{ \left\| \mathop{\mathbb{E}}_{\tau \sim \zeta_\mu^{\pi_b,p}} \left[\mathbf{g}^{\pi,\overline{p}}(\tau)^2 \right] \right\|_\infty ,$$

$$\left\| \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}^{\pi,\overline{p}}(\tau^i)^2 \right\|_\infty \Bigg\} \leq R_{\max}^2 c_2^2.$$

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* $v < +\infty$.

Assumption 1 is requiring that the overall effect of the importance weight ρ_{π/π_b}, the score $\nabla_{\boldsymbol{\theta}} \log \pi$ 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 2 is necessary to state learning theory guarantees. We are now ready to present the main result, which employs the learning theory tools of Cortes, Greenberg, and Mohri (2013).

Theorem 5.1. *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* (3)*, 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:*

$$\left\| \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \right\|_q \leq \underbrace{2R_{\max} \left(d^{\frac{1}{q}} c_2 \epsilon + \frac{\gamma \sqrt{2Z c_1 \epsilon}}{1-\gamma} \right)}_{\textit{estimation error}}$$

$$+ \underbrace{\frac{\gamma \sqrt{2} Z R_{\max}}{(1-\gamma)^2} \inf_{\overline{p} \in \mathcal{P}} \sqrt{\mathop{\mathbb{E}}_{s,a \sim \eta_\mu^{\pi,p}} [D_{KL}(p(\cdot|s,a) \| \overline{p}(\cdot|s,a))]}}_{\textit{approximation error}},$$

where $\epsilon = \sqrt{\frac{v \log \frac{2eN}{v} + \log \frac{8(d+1)}{\delta}}{N}} \Gamma\left(\sqrt{\frac{v \log \frac{2eN}{v} + \log \frac{8(d+1)}{\delta}}{N}} \right)$ *and* $\Gamma(\xi) \coloneqq \frac{1}{2} + \sqrt{1 + \frac{1}{2} \log \frac{1}{\xi}}$.

The theorem justifies the intuition behind the gradient estimation based on MVG. A model $\overline{p}$ is good when it achieves a reasonable trade-off between the errors in approximation and estimation.[4] In the 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 of the pseudo-dimension v. However, this carries the risk of being unable to approximate the original model. Nonetheless, the approximation error depends on an expected value under $\eta_\mu^{\pi,p}$. 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.

6 Related Works

We now revise prior work in MBRL, focusing on policy search methods and those that include some level of awareness of the underlying control problem into model learning.

Policy Search with MBRL The standard approach consists in using a maximum likelihood estimation of the environment dynamics to perform simulations (or *imaginary rollouts*) through which a policy can be improved without further or with limited interactions with the environment (Deisenroth et al. 2013). This approach has taken different forms, with the use of tabular models (Wang and Dietterich 2003), least-squares density estimation techniques (Tangkaratt et al. 2014) or, more recently, combinations of variational generative models and recurrent neural networks employed in world models based on mixture density networks (Ha and Schmidhuber 2018). Several methods incorporate the model uncertainty into policy updates, by using Gaussian processes and moment matching approximations (Deisenroth and Rasmussen 2011), Bayesian neural networks (Gal, McAllister, and Rasmussen 2016) or ensembles of forward models (Chua et al. 2018; Kurutach et al. 2018; Janner et al. 2019; Buckman et al. 2018). MBRL works that are particularly related to GAMPS are those employing estimated forward models that are accurate only locally (Abbeel, Quigley, and Ng 2006; Nguyen-Tuong, Seeger, and Peters 2009; Levine and Abbeel 2014), or using a *model-value based gradient* formulation (Abbeel, Quigley, and Ng 2006; Feinberg et al. 2018; Buckman et al. 2018; Heess et al. 2015) as described in Section 3.

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 decision-aware approaches for model learning (Farahmand, Barreto, and Nikovski 2017). While one of the first examples was a financial application (Bengio 1997), the idea was introduced into MBRL (Joseph et al. 2013; Bansal and others 2017) and the related adaptive optimal control literature (Piroddi and Spinelli 2003) by using actual evaluations of a control policy in the environment as a performance index for model learning. More similarly to our approach, but in the context of value-based methods, a theoretical framework called *value-aware* MBRL (Farahmand, Barreto, and Nikovski 2017) was proposed, in which the model is estimated by minimizing the expected error on the Bellman operator, explicitly considering its actual use in the control algorithm. Starting from this, further theoretical considerations and approaches have been proposed (Farahmand 2018; Asadi et al. 2018). Awareness of the final task to be performed has been also incorporated into stochastic dynamic programming (Donti, Amos, and Kolter 2017; Amos et al. 2018) and, albeit implicitly, into neural network-based works (Oh, Singh, and Lee 2017; Silver and others 2017; Luo et al. 2019), in which value functions and models consistent with each other are learned.

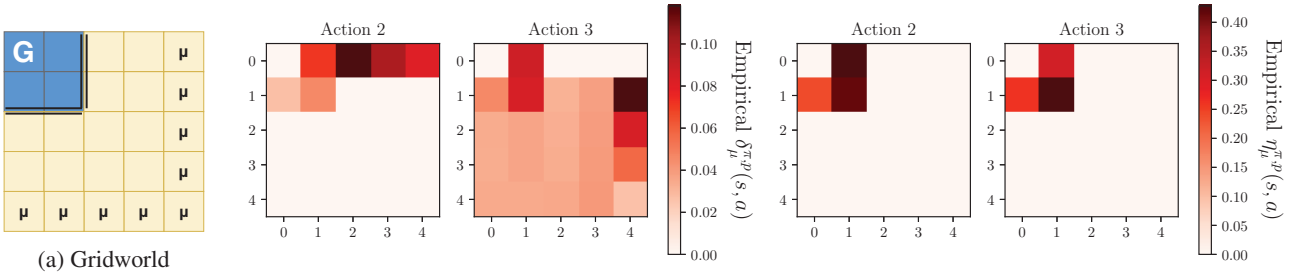
7 Experiments

We now 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.

7.1 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 is a 5×5 gridworld, divided into two areas (lower

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



(a) Gridworld

(b) State-action distribution weights

(c) Gradient-aware weights

Figure 1: (a): 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. (b) and (c): Normalized values of the empirical state-action distribution and the weighting factor for GAMPS. Each grid represents every state of the environment for the two most representative actions.

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 to go back in the upper one (Figure 1a). We collect experience with a linear policy π_b that is deterministic on the lower area and randomly initialized in the upper area, which is also used as initial policy for learning.

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. Hence, we use a forward model which, given the sole action executed by the agent (i.e., without knowledge of the current state), predicts the effect that it will cause on the position of the agent (i.e., up, down, left, right, stay). This model class cannot perfectly represent the whole environment dynamics at the same time, as it 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, weighs the transitions just upon visitation, regardless of 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 1b and 1c) show that our method is able, in a totally automatic way, to avoid assigning importance to the transitions in which the policy cannot be improved.

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 (Williams 1992) and PGT (Sutton et al. 2000). To adapt the latter two to the batch setting, we employ importance sampling in the same way as described in Equation (5), but estimating the Q-function using the same trajectories (and importance sampling as well). The results obtained by collecting different numbers of trajectories and evaluating on the environment are shown in Figure 2. When the data are too scarce, all compared algorithms struggle in converging towards good policies, experiencing high variance.[5] It is worth noting that, with any amount of data we tested, the GAMPS learning curve is consistently above the others, showing superior performance even when considering the best iteration of all algorithms.

7.2 Continuous Control

To show that our algorithm is able to perform well also on continuous domains, we test its performance on a simulated Minigolf environment (Lazaric, Restelli, and Bonarini 2008; Tirinzoni, Salvini, and Restelli 2019) and the 3-link Swimmer robot control benchmark based on Mujoco (2012).

In the minigolf game, the agent hits a ball using a flat-faced golf club (the putter) with the goal of reaching a hole in the minimum number of strokes. Given only the distance to the hole, the agent chooses, at each step, the angular velocity of the putter which determines the next position of the ball. The episode terminates when the ball enters the hole, with reward 0, or when the agent overshoots, with reward -100. In all other cases, the reward is -1 and the agent can try other hits. We further suppose that the minigolf course is divided into two areas, one twice larger than the other, with different terrains: the first, nearest to the hole and biggest, has the friction of a standard track; the second has very high friction, comparable to the one of an area with sand. We use Gaussian policies that are linear on six radial basis function features. The model predicts the difference from the previous state by sampling from a Gaussian distribution with linearly parameterized mean and standard deviation.

For the Mujoco swimmer the goal of the agent is to swim forward in a fluid. The policy is linear in the state features and the forward model is a 2-layer neural networks with 32 hidden neurons and tanh activation.

We evaluate GAMPS against the same baselines employed for the previous experiment. We collect a dataset of 50 and 100 trajectories for minigolf and swimmer respec-

[5] In batch learning, performance degradation when the current policy π becomes too dissimilar 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 (Owen 2013) can be employed.

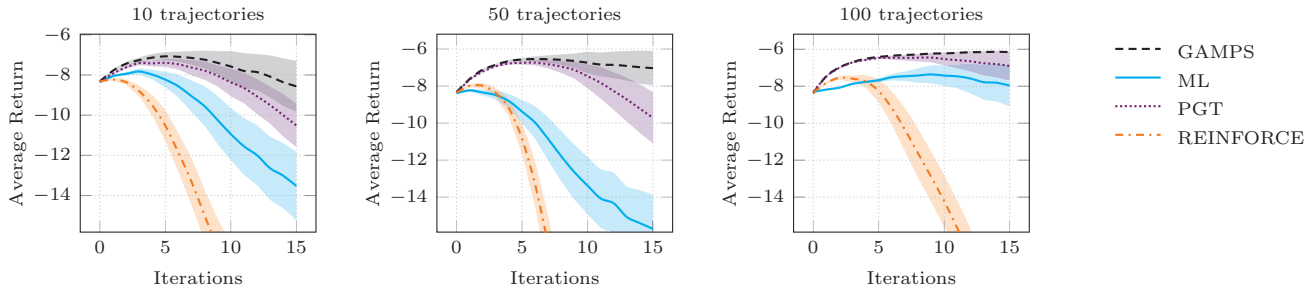


Figure 2: Average return on the gridworld. ML employs maximum-likelihood model estimation (20 runs, mean $\pm$ std).

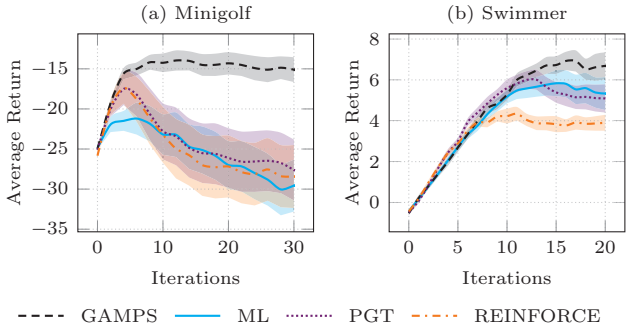


Figure 3: Average return in the minigolf domain using 50 trajectories and in the swimmer environment using 100 trajectories (10 runs, mean $\pm$ std).

tively, using an explorative policy, and then run the algorithms for 30 and 20 iterations. Results in terms of average return are reported in Figure 3, showing that GAMPS outperforms all the other algorithms both in terms of terminal and maximum performance. In the minigolf domain, it is less susceptible to overfitting compared to the baselines, and it obtains a final policy able to reach the hole most of the time. In the swimmer environment, where powerful models are used, GAMPS still shows superior performance, validating the insight provided by Theorem 3.2 even in this setting.

8 Discussion and Conclusions

In this paper, we presented GAMPS, a batch gradient-aware model-based policy search algorithm. GAMPS leverages the knowledge about the policy that is being optimized for learning the transition model, by giving more importance to the aspects of the dynamics that are more relevant for improving its performance. We derived GAMPS from the minimization of the bias of the model-value-based gradient, an approximation for the policy gradient that mixes trajectories collected in the real environment together with a value function computed with the estimated model. Our theoretical analysis validates the intuition that, when dealing with low capacity models, it is convenient to focus their representation capabilities on the portions of the environment that are most crucial for improving the policy. The empirical validation demonstrates that, even when extremely simple model classes are considered, GAMPS is able to outperform the baselines.

The main limitations of GAMPS are in the need to reuse all the samples in model learning at each iteration and in the compounding errors during rollouts. Future work could focus on mitigating these issues, as well as on adapting GAMPS to the interactive scenario, by leveraging existing on/off-policy approaches (Metelli et al. 2018), and on different ways to exploit gradient-aware model learning.

Acknowledgments

This work has been partially supported by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Market".

References

Abbeel, P.; Quigley, M.; and Ng, A. Y. 2006. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, 1–8. ACM.

Amos, B.; Jimenez, I.; Sacks, J.; Boots, B.; and Kolter, J. Z. 2018. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, 8289–8300.

Asadi, K.; Cater, E.; Misra, D.; and Littman, M. L. 2018. Equivalence between wasserstein and value-aware model-based reinforcement learning. *arXiv preprint arXiv:1806.01265*.

Bansal, S., et al. 2017. Goal-driven dynamics learning via bayesian optimization. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* 5168–5173.

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

Bellman, R., et al. 1954. The theory of dynamic programming. *Bulletin of the American Mathematical Society* 60(6):503–515.

Bengio, Y. 1997. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems*.

Bertsekas, D. P.; Bertsekas, D. P.; Bertsekas, D. P.; and Bertsekas, D. P. 1995. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Buckman, J.; Hafner, D.; Tucker, G.; Brevdo, E.; and Lee, H. 2018. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 8224–8234.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 4754–4765.

Cortes, C.; Greenberg, S.; and Mohri, M. 2013. Relative deviation learning bounds and generalization with unbounded loss functions. *arXiv preprint arXiv:1310.5796*.

Deisenroth, M., and Rasmussen, C. E. 2011. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 465–472.

Deisenroth, M. P.; Neumann, G.; Peters, J.; et al. 2013. A survey on policy search for robotics. *Foundations and Trends® in Robotics* 2(1–2):1–142.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, 5484–5494.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6(Apr):503–556.

Farahmand, A.-m.; Barreto, A.; and Nikovski, D. 2017. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, 1486–1494.

Farahmand, A.-m. 2018. Iterative value-aware model learning. In *Advances in Neural Information Processing Systems*, 9072–9083.

Feinberg, V.; Wan, A.; Stoica, I.; Jordan, M. I.; Gonzalez, J. E.; and Levine, S. 2018. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*.

Gal, Y.; McAllister, R.; and Rasmussen, C. E. 2016. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4.

Ha, D., and Schmidhuber, J. 2018. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2450–2462.

Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Erez, T.; and Tassa, Y. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, 2944–2952.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*.

Joseph, J. M.; Geramifard, A.; Roberts, J. W.; How, J. P.; and Roy, N. 2013. Reinforcement learning with misspecified model classes. *2013 IEEE International Conference on Robotics and Automation* 939–946.

Kahn, H., and Marshall, A. W. 1953. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America* 1(5):263–278.

Kurutach, T.; Clavera, I.; Duan, Y.; Tamar, A.; and Abbeel, P. 2018. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*.

Lange, S.; Gabel, T.; and Riedmiller, M. 2012. Batch reinforcement learning. In *Reinforcement learning*. Springer. 45–73.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Reinforcement learning in continuous action spaces through sequential monte carlo methods. In *Advances in neural information processing systems*, 833–840.

Levine, S., and Abbeel, P. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, 1071–1079.

Luo, Y.; Xu, H.; Li, Y.; Tian, Y.; Darrell, T.; and Ma, T. 2019. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*.

Metelli, A. M.; Papini, M.; Faccio, F.; and Restelli, M. 2018. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, 5442–5454.

Nguyen-Tuong, D., and Peters, J. 2011. Model learning for robot control: a survey. *Cognitive processing* 12(4):319–340.

Nguyen-Tuong, D.; Seeger, M.; and Peters, J. 2009. Model learning with local gaussian process regression. *Advanced Robotics* 23(15):2015–2034.

Oh, J.; Singh, S.; and Lee, H. 2017. Value prediction network. In *Advances in Neural Information Processing Systems*, 6118–6128.

Owen, A. B. 2013. *Monte Carlo theory, methods and examples*.

Piroddi, L., and Spinelli, W. 2003. An identification algorithm for polynomial narx models based on simulation error minimization. *International Journal of Control* 76(17):1767–1781.

Precup, D.; Sutton, R. S.; and Singh, S. P. 2000. Eligibility traces for off-policy policy evaluation. In Langley, P., ed., *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, 759–766.

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

Riedmiller, M. 2005. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, 317–328. Springer.

Silver, D., et al. 2017. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3191–3199. JMLR. org.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1057–1063.

Tangkaratt, V.; Mori, S.; Zhao, T.; Morimoto, J.; and Sugiyama, M. 2014. Model-based policy gradients with parameter-based exploration by least-squares conditional density estimation. *Neural networks* 57:128–140.

Tirinzoni, A.; Salvini, M.; and Restelli, M. 2019. Transfer of samples in policy search via multiple importance sampling. In *Proceedings of the 36th International Conference on Machine Learning*, 6264–6274.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Wang, X., and Dietterich, T. G. 2003. Model-based policy gradient reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 776–783.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.