Gradient-Aware Model-based Policy Search

P. D'Oro[*], **A.M. Metelli**[*], **A. Tirinzoni, M. Papini, M. Restelli**
DEIB, Politecnico di Milano
pierluca.doro@mail.polimi.it
{albertomaria.metelli,andrea.tirinzoni,matteo.papini,marcello.restelli}@polimi.it

Abstract

Traditional MBRL 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 information is ignored. In this paper, we introduce a model-based policy search approach that, by meta-learning, 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 integrate gradient-aware model learning 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 update the policy. Finally, we empirically validate GAMPS on benchmark domains analyzing and discussing its properties.

1 Introduction

Model-Based Reinforcement Learning (MBRL) [24, 20] 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) [24] algorithms. Policy search MBRL methods [6, 25] typically use this estimate to perform simulations (or *imaginary rollouts*) through which a policy is improved with limited interactions with the environment [6]. This approach has taken different forms, with the use of generative models [28, 26, 11, 13] , or techniques such as Gaussian processes [5], Bayesian neural networks [10] or ensembles of forward models [4, 15, 3].

However, 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. 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, 21, 18]. Indeed, a wiser approach consists in using simpler model classes, whose estimation requires few data, and focus their limited capacity on the most relevant parts of the environment. These parts could present a local dynamics that is simpler than the global one, or easier to model using prior knowledge.

The majority of MBRL methods employs maximum likelihood estimation for learning the model [6]. 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. For taking advantage of this knowledge, an agent that uses both a model and a policy as internal modules must *meta-learn*, intervening in its own learning process: in other words, it has to learn the model in a way that maximizes the possibility for its policy to be learned.

In this paper, we propose a MBRL policy search [6, 25] method that leverages awareness of the current agent's policy in the estimation of a model, used to perform policy optimization. Unlike

[*]Equal contribution.

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada

existing decision-aware approaches [8, 7], which typically ignore all the knowledge available on the policy, we incorporate it into a weighting scheme for the objective function used in model learning.

2 Model-Value-based Gradient

A Markov Decision Process (MDP) [23] is described by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \mu, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r(s,a)$ is a known, uniformly bounded, reward function, $p(\cdot|s,a)$ is the transition model, μ is the initial state distribution and $\gamma \in [0,1)$ is a discount factor. The behavior of an agent is described by a policy $\pi(\cdot|s)$. We define the action-value function [24] 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 as $V^{\pi,p}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi,p}(s,a)]$. The goal of the agent is to find an optimal policy π^*, that maximizes $J^{\pi,p} = \mathbb{E}_{s_0\sim\mu}\left[V^{\pi,p}(s_0)\right]$.

We consider a batch setting [16], in which learning is performed on a dataset $\mathcal{D} = \left\{\tau^i\right\}_{i=1}^N$ of N independent trajectories τ^i, previously collected according to $\zeta_\mu^{\pi_b,p}(\tau)$, each composed of T_i transitions (s_t^i, a_t^i, r_t^i), and further interactions with the environment are not allowed. Although data was collected by a behavior policy π_b, we are interested in improving a different policy π. Therefore, we use the importance weights $\rho_{\pi/\pi_b}(\tau_{t':t''}) = \frac{\zeta_\mu^{\pi,p}(\tau_{t':t''})}{\zeta_\mu^{\pi_b,p}(\tau_{t':t''})}$, to correct for the distribution mismatch.

We consider an interesting [1, 9, 3, 12] approximation of the gradient, named *Model-Value-based Gradient* (MVG). It is defined, starting from the policy gradient theorem (PGT) [25, 24], as follows.

Definition 2.1. *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:*

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

$\delta_\mu^{\pi,p}(s,a) = (1-\gamma) \sum_{t=1}^{+\infty} \gamma^t \Pr\left((s_t,a_t) = (s,a) | \mathcal{M}, \pi\right)$ is the γ-discounted state-action distribution [25], and $\nabla_{\boldsymbol{\theta}} \log \pi(a|s)$ is the *score* of policy π. Thus, the MVG employs experience collected in the real environment p, i.e., sampling from $\delta_\mu^{\pi,p}(s,a)$, and uses $\widehat{p}$ in the computation of an approximated $Q^{\pi,\widehat{p}}$ only. Therefore, the MVG limits the bias effect of $\widehat{p}$ while enjoying smaller variance w.r.t. Monte Carlo estimators.

3 Gradient-Aware Model-based Policy Search

A central question concerning Definition 2.1, is how the choice of $\widehat{p}$ affects the quality of the gradient approximation, i.e., how much bias an MVG introduces. To this end, we bound the approximation error by the expected KL-divergence between p and $\widehat{p}$. The proof can be found in Appendix A.1 and a comparison with classical maximum-likelihood estimation is presented in Appendix A.2.

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

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

where $\eta_\mu^{\pi,p}(s,a) = \frac{1}{Z} \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\| \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a'|s') \right\|_q \delta_{s',a'}^{\pi,p}(s,a) \mathrm{d}s' \mathrm{d}a'$ is a probability distribution over $\mathcal{S} \times \mathcal{A}$, with $Z = \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\| \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a'|s') \right\|_q \mathrm{d}s' \mathrm{d}a'$.

Our bound shows that not all collected transitions have the same relevance when learning a model used in estimating the MVG. Overall, the most important (s,a) pairs to be considered are those that are *likely to be reached from the policy starting from high gradient-magnitude state-action pairs*.

Inspired by Theorem 3.1, 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 model $\widehat{p}$, computing the state-action value function $Q^{\pi,\widehat{p}}$ and updating the policy using the estimated gradient $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.

Algorithm 1 Gradient-Aware Model-based Policy Search (GAMPS)

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**
  $\omega_{t,k}^i \leftarrow \gamma^t \rho_{\pi_{\boldsymbol{\theta}_k}/\pi_b}(\tau_{0:t}^i) \sum_{l=0}^t \|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_l^i|s_l^i)\|_q$
  $\widehat{p}_k \leftarrow \arg\max_{\overline{p}\in\mathcal{P}} \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} \omega_{t,k}^i \log \overline{p}(s_{t+1}^i|s_t^i, a_t^i)$
  Generate a dataset of M trajectories for each (s,a) simulating $\widehat{p}_k$
  $\widehat{Q}_k(s,a) = \frac{1}{M} \sum_{j=1}^M \sum_{t=0}^{T_j-1} \gamma^t r(s_t^j, a_t^j)$
  $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k) \leftarrow \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi_{\boldsymbol{\theta}_k}/\pi_b}(\tau_{0:t}^i) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t^i|s_t^i) \widehat{Q}_k(s_t^i, a_t^i)$
  $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k)$
end for

To learn $\widehat{p}$, we aim at minimizing the bound in Theorem 3.1, over a class of transition models $\mathcal{P}$, using the trajectories $\mathcal{D}$ collected with $\zeta_\mu^{\pi_b,p}$. We can thus get to the objective:

$$\widehat{p} = \arg\max_{\overline{p}\in\mathcal{P}} \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} \omega_t^i \log \overline{p}\left(s_{t+1}^i|s_t^i, a_t^i\right), \quad \omega_t^i = \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}^i) \sum_{l=0}^t \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_l^i|s_l^i)\right\|_q. \quad (2)$$

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 π. Especially, 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, while the factor $\sum_{l=0}^t \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_l^i|s_l^i)\right\|_q$ is related to its *opportunity to be improved*, i.e., the possibility to change the probability of actions. A more theoretically grounded justification of our weighting scheme is reported in Appendix A.3.

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 $\widehat{p}$, via dynamic programming [2, 24] for discrete MDPs or in approximate manner for continuous MDPs. For the latter, we use $\widehat{p}$ as a generative model to obtain a Monte Carlo approximation of $Q^{\pi,\widehat{p}}$ on the fly, in an unbiased way, as $\widehat{Q}(s,a) = \frac{1}{M} \sum_{j=1}^M \sum_{t=0}^{T_j-1} \gamma^t r(s_t^j, a_t^j)$, with $\tau^j \sim \zeta_{s,a}^{\pi,\widehat{p}}$, averaging the return from a (possibly large) number of imaginary trajectories obtained from the estimated model.

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 considering the sample version of Equation (1) we estimate the gradient as $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} \gamma^t \rho_{\pi/\pi_b}(\tau_{0:t}^i) \nabla_{\boldsymbol{\theta}} \log \pi(a_t^i|s_t^i) Q^{\pi,\widehat{p}}(s_t^i, a_t^i)$.

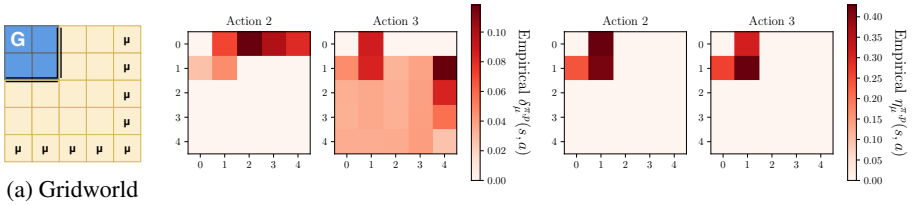
For performing batch policy optimization, we iteratively repeat three steps using the data collected by the behavior policy π_b. At each iteration, we fit the model with the weights relative to the current policy, we employ it in the computation of the value function and we improve the policy with one or more steps of gradient ascent. The overall procedure is summarized in Algorithm 1.

4 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.

Two-areas Gridworld. The environment, depicted in Figure 1a, is a 5×5 gridworld, divided into two areas with switched movement dynamics: the effect of a movement action of the agent is reversed in one area w.r.t. the other. We collect 1000 trajectories using an initial policy π_b, to be improved, that is deterministic in the lower area and randomly initialized in the upper one.

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 linear in the action, able to represent only one of the two parts of the environment. To experimentally assess how our approach leverages the available knowledge about



(a) Gridworld

(b) State-action distribution weights

(c) Gradient-aware weights

Figure 1: (a): Gridworld. **G** is the goal, μ are possible initial states. 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.

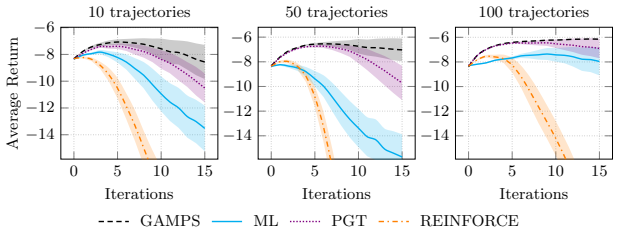


Figure 2: Average return in the gridworld with different dataset size. ML is uses maximum likelihood model estimation (20 runs, mean $\pm$ std).

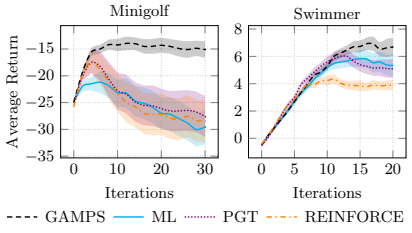


Figure 3: Average return in the minigolf domain using 50 trajectories and in the swimmer environment using 100 trajectories (10 runs, mean $\pm$ std).

the policy, we compare the ML and the gradient-aware weighting factors, $\delta^{\pi,p}_{\mu}(s,a)$ and $\eta^{\pi,p}_{\mu}(s,a)$. The results (Figure 1b and 1c) show that our method is able not to assign importance to the transitions with which the policy cannot be improved.

We further investigate the performance of GAMPS compared to batch learning with the maximum likelihood model (ML) and two model-free algorithms [29, 25] adapted to the batch setting via importance sampling. The results obtained by collecting different numbers of trajectories and evaluating on the environment are shown in Figure 2.[2] Note that the GAMPS learning curve is consistently above the others, showing superior performance on the best iteration of any algorithm.

Continuous Control. To asses how our algorithm generalizes to continuous domains, we employ two different benchmarks. The first one is a one-dimensional minigolf game, where the agent hits a ball with the goal of reaching a hole in the minimum number of strokes [17], in which we use linear-Gaussian policies and models; the second one is the Swimmer Mujoco [27] control task, in which we employ 2-layer neural networks with 32 hidden neurons as models and linear policies.

We evaluate GAMPS on datasets of 50 and 100 trajectories, for 30 and 20 iterations in the two domains respectively. The results, in Figure 3, show that GAMPS outperforms in both cases the baselines, which are also more prone to overfitting in the minigolf environment.

5 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. Future work could focus on adapting GAMPS to the interactive scenario, in order to make it scale on complex high-dimensional environments.

[2] 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 [22] can be employed.

References

[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*, pages 1–8. ACM, 2006.

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

[3] 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*, pages 8224–8234, 2018.

[4] 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*, pages 4754–4765, 2018.

[5] 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)*, pages 465–472, 2011.

[6] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[7] Amir-massoud Farahmand. Iterative value-aware model learning. In *Advances in Neural Information Processing Systems*, pages 9072–9083, 2018.

[8] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pages 1486–1494, 2017.

[9] 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. *arXiv preprint arXiv:1803.00101*, 2018.

[10] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, 2016.

[11] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[12] 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*, pages 2944–2952, 2015.

[13] 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. *arXiv preprint arXiv:1903.00374*, 2019.

[14] 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.

[15] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[16] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[17] 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*, pages 833–840, 2008.

[18] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

[19] Alberto Maria Metelli, Mirco Mutti, and Marcello Restelli. Configurable Markov decision processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3488–3497, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[20] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[21] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[22] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

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

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[25] 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*, pages 1057–1063, 2000.

[26] 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. *Neural networks*, 57:128–140, 2014.

[27] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

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

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

A Proofs and Derivations

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

A.1 Proof of Theorem 3.1

The following lemma is used in proving Theorem 3.1.

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:*

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

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

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

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 [19]:

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

□

Now, we can prove Theorem 3.1.

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

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

where $\eta_\mu^{\pi,p}(s,a) = \frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a'|s')\right\|_q \delta_{s',a'}^{\pi,p}(s,a) \mathrm{d}s'\mathrm{d}a'$ is a probability distribution over $\mathcal{S}\times\mathcal{A}$, with $Z = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s',a') \left\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a'|s')\right\|_q \mathrm{d}s'\mathrm{d}a'$.

Proof.

$$\begin{aligned}
\left\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{\mathrm{MVG}} J(\boldsymbol{\theta})\right\|_q &= \left\|\frac{1}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} (\delta_\mu^{\pi,p}(s,a)(Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a))\nabla_{\boldsymbol{\theta}} \log \pi(a|s) \mathrm{d}s\mathrm{d}a\right\|_q \\
&\leq \frac{1}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_\mu^{\pi,p}(s,a) \left|Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a)\right| \|\nabla_{\boldsymbol{\theta}} \log \pi(a|s)\|_q \mathrm{d}s\mathrm{d}a \quad (3) \\
&= \frac{Z}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a) \left|Q^{\pi,p}(s,a) - Q^{\pi,\widehat{p}}(s,a)\right| \mathrm{d}s\mathrm{d}a \quad (4) \\
&= \frac{Z}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a) \left|\int_{\mathcal{S}}\int_{\mathcal{A}} r(s',a')(\delta_{s,a}^{\pi,p}(s',a') - \delta_{s,a}^{\pi,\widehat{p}}(s',a'))\mathrm{d}s'\mathrm{d}a'\right| \mathrm{d}s\mathrm{d}a \quad (5) \\
&\leq \frac{ZR_{\max}}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a) \left|\int_{\mathcal{S}}\int_{\mathcal{A}} (\delta_{s,a}^{\pi,p}(s',a') - \delta_{s,a}^{\pi,\widehat{p}}(s',a'))\mathrm{d}s'\mathrm{d}a'\right| \mathrm{d}s\mathrm{d}a \quad (6) \\
&\leq \frac{ZR_{\max}}{1-\gamma}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a) \left\|\delta_{s,a}^{\pi,p} - \delta_{s,a}^{\pi,\widehat{p}}\right\|_1 \mathrm{d}s\mathrm{d}a \\
&\leq \frac{ZR_{\max}\gamma}{(1-\gamma)^2}\int_{\mathcal{S}}\int_{\mathcal{A}} \nu_\mu^{\pi,p}(s,a) \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{s,a}^{\pi,p}(s',a') \left\|p(\cdot|s',a') - \widehat{p}(\cdot|s',a')\right\| \mathrm{d}s'\mathrm{d}a'\mathrm{d}s\mathrm{d}a \quad (7)
\end{aligned}$$

$$= \frac{ZR_{\max}\gamma}{(1-\gamma)^2} \int_{\mathcal{S}}\int_{\mathcal{A}} \eta_{\mu}^{\pi,p}(s',a') \int_{\mathcal{S}} \left|p(s''|s',a') - \widehat{p}(s''|s',a')\right| \mathrm{d}s''\mathrm{d}s'\mathrm{d}a'$$

$$\leq \frac{ZR_{\max}\gamma}{(1-\gamma)^2} \int_{\mathcal{S}}\int_{\mathcal{A}} \eta_{\mu}^{\pi,p}(s,a) \sqrt{2D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))}\mathrm{d}s\mathrm{d}a \tag{8}$$

$$\leq \frac{ZR_{\max}\gamma}{(1-\gamma)^2} \sqrt{2\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_{\mu}^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a}, \tag{9}$$

where in Equation (4), we define a new probability distribution $\nu_{\mu}^{\pi,p}(s,a) = \frac{1}{Z}\delta_{\mu}^{\pi,p}(s,a)\|\nabla_{\boldsymbol{\theta}} \log \pi(a|s)\|_q$ by means of an appropriate normalization constant Z, assumed $Z > 0$. In Equation (5), we use the definition of Q-function as $Q^{\pi,p}(s,a) = \int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{s,a}^{\pi,p}(s',a') r(s',a') \mathrm{d}s'\mathrm{d}a'$. After bounding the reward in Equation (6), in Equation (7) we apply Lemma A.1. Then we obtain Equation (8) by employing Pinsker's inequality, defining the overall weighting term $\eta_{\mu}^{\pi,p}(s',a') = \int_{\mathcal{S}}\int_{\mathcal{A}} \nu_{\mu}^{\pi,p}(s,a)\delta_{s,a}^{\pi,p}(s',a')\mathrm{d}s\mathrm{d}a$, and renaming variables for clarity. Last passage follows from Jensen inequality. □

In order to understand how the weighting distribution $\eta_{\mu}^{\pi,p}(s,a)$ enlarges the relative importance of some transitions with respect to others, we can use the auxiliary distribution $\nu_{\mu}^{\pi,p}(s',a') = \frac{1}{Z}\left\|\nabla_{\boldsymbol{\theta}} \log \pi(a'|s')\right\|_q \delta_{\mu}^{\pi,p}(s',a')$ and rewrite the weighting distribution as:

$$\eta_{\mu}^{\pi,p}(s,a) = \int_{\mathcal{S}}\int_{\mathcal{A}} \underbrace{\nu_{\mu}^{\pi,p}(s',a')}_{\substack{\text{gradient magnitude} \\ \text{factor}}} \underbrace{\delta_{s',a'}^{\pi,p}(s,a)}_{\substack{\text{reachability} \\ \text{factor}}} \mathrm{d}s'\mathrm{d}a'. \tag{10}$$

Intuitively, $\nu_{\mu}^{\pi,p}(s',a')$ is high for states and actions that are both likely to be visited executing π and corresponding to high norm of its score; $\delta_{s',a'}^{\pi,p}(s,a)$ is the state-action distribution of (s,a) after executing action a' in state s'. 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).

A.2 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 3.1. For proving the following result, we assume the score is bounded by $\|\nabla_{\boldsymbol{\theta}} \log \pi(a|s)\|_q \leq K$.

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

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

Proof.

$$\left\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{\mathrm{MVG}} J(\boldsymbol{\theta})\right\|_q \leq \frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2} \left(\int_{\mathcal{S}}\int_{\mathcal{A}} \eta_{\mu}^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a\right)^{\frac{1}{2}}$$

$$= \frac{\gamma\sqrt{2}ZR_{\max}}{(1-\gamma)^2} \Bigg(\int_{\mathcal{S}}\int_{\mathcal{A}} \frac{1}{Z}\int_{\mathcal{S}}\int_{\mathcal{A}} \|\nabla_{\boldsymbol{\theta}} \log \pi(a'|s')\|_q \delta_{\mu}^{\pi,p}(s',a') \delta_{s',a'}^{\pi,p}(s,a)\mathrm{d}s'\mathrm{d}a' \tag{11}$$

$$\times D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a\Bigg)^{\frac{1}{2}}$$

$$\leq \frac{\gamma\sqrt{2KZ}R_{\max}}{(1-\gamma)^2} \left(\int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{\mu}^{\pi,p}(s',a') \delta_{s',a'}^{\pi,p}(s,a)\mathrm{d}s'\mathrm{d}a' D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a\right)^{\frac{1}{2}}$$

$$= \frac{\gamma\sqrt{2KZ}R_{\max}}{(1-\gamma)^2} \left(\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{\mu}^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a\right)^{\frac{1}{2}} \tag{12}$$

$$\leq \frac{\gamma\sqrt{2}KR_{\max}}{(1-\gamma)^2} \left(\int_{\mathcal{S}}\int_{\mathcal{A}} \delta_{\mu}^{\pi,p}(s,a) D_{KL}(p(\cdot|s,a)\|\widehat{p}(\cdot|s,a))\mathrm{d}s\mathrm{d}a\right)^{\frac{1}{2}}, \tag{13}$$

where we started from Theorem 3.1. Equation (12) follows from the fact that $\int \delta_{\mu}^{\pi,p}(s',a')\delta_{s',a'}^{\pi,p}(s,a)\mathrm{d}s'\mathrm{d}a' = \delta_{\mu}^{\pi,p}(s,a)$, as we are actually recomposing the state-action distribution that was split at (s',a') and Equation (13) is obtained by observing that $Z \leq K$. □

Therefore, the bound is less tight than the one presented in Theorem 3.1. We can in fact observe that

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

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.

A.3 Weighting Scheme Details

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:*

$$\mathop{\mathbb{E}}_{s,a\sim\delta_\mu^{\pi,p}}[f(s,a)] = (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t \mathop{\mathbb{E}}_{\tau_{0:t}\sim\zeta_\mu^{\pi,p}}[f(s_t,a_t)] = (1-\gamma)\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi,p}}\left[\sum_{t=0}^{+\infty}\gamma^t f(s_t,a_t)\right]. \quad (14)$$

Proof. We denote with $\mathcal{T}$ the set of all possible trajectories. We just apply the definition of $\delta_\mu^{\pi,p}$ [25]:

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

where we exploited the fact that the probability $\Pr\left(s_t=s,\,a_t=a|\mathcal{M},\,\pi\right)$ is equal to the probability that a prefix of trajectory $\tau_{0:t}$ terminates in (s_t,a_t), i.e., $\int_{\mathcal{T}}\zeta_\mu^{\pi,p}(\tau_{0:t})\mathbb{1}\left(s_t=s,\,a_t=a\right)\mathrm{d}\tau_{0:t}$. 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:*

$$\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}[f(s,a)] = \frac{(1-\gamma)^2}{Z}\mathop{\mathbb{E}}_{\tau\sim\zeta_\mu^{\pi,p}}\left[\sum_{t=0}^{+\infty}\gamma^t\sum_{l=0}^{t}\left\|\nabla_{\boldsymbol{\theta}}\log\pi(a_l|s_l)\right\|_q f(s_t,a_t)\right]. \quad (15)$$

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

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

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

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

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

$$\mathop{\mathbb{E}}_{s,a\sim\eta_\mu^{\pi,p}}[f(s,a)] = \frac{(1-\gamma)}{Z} \int_{\mathcal{S}} \int_{\mathcal{A}} \sum_{t=0}^{+\infty} \gamma^t \int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t}) \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t)\right\|_q \delta_{s_t,a_t}^{\pi,p}(s,a) f(s,a) \mathrm{d}\tau_{0:t} \mathrm{d}s \mathrm{d}a$$
$$= \frac{(1-\gamma)}{Z} \sum_{t=0}^{+\infty} \gamma^t \int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t}) \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t)\right\|_q \int_{\mathcal{S}} \int_{\mathcal{A}} \delta_{s_t,a_t}^{\pi,p}(s,a) f(s,a) \mathrm{d}s \mathrm{d}a \mathrm{d}\tau_{0:t}$$
$$= \frac{(1-\gamma)^2}{Z} \sum_{t=0}^{+\infty} \gamma^t \int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau_{0:t}) \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t)\right\|_q \sum_{l=0}^{+\infty} \gamma^l \int_{\mathcal{T}} \zeta_{s_t,a_t}^{\pi,p}(\tau_{0:l}) f(s_l,a_l) \mathrm{d}\tau_{0:l} \mathrm{d}\tau_{0:t}$$
$$= \frac{(1-\gamma)^2}{Z} \int_{\mathcal{T}} \zeta_\mu^{\pi,p}(\tau) \sum_{t=0}^{+\infty} \left\|\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t)\right\|_q \sum_{h=t}^{+\infty} \gamma^h f(s_h,a_h) \mathrm{d}\tau,$$

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

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

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

We are now ready to prove Lemma A.4.

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:*

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

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 A.4 and we just need to apply importance weighting [22]:

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

□

B Experimental details

B.1 Two-areas Gridworld

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 $(\underset{\Uparrow}{p_1}, \underset{\Rightarrow}{p_2}, \underset{\Downarrow}{p_3}, \underset{\Leftarrow}{p_4}, \underset{\nLeftrightarrow}{p_5})$ the probabilities p_1, p_2, p_3, p_4 and p_5 of, respectively, going up, right, down, left and remaining in the same state, the transition model of the environment is defined as follows:

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

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

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.

As model class $\mathcal{P}$, we employ the one in which each $\widehat{p} \in \mathcal{P}$ is such that $\widehat{p}(m|s,a) = \text{softmax}(\mathbb{1}_a^T \mathbf{W})$, where $\mathbf{W}$ is a matrix of learnable parameters, $\mathbb{1}_a$ is the one-hot representation of the action and $m \in \{\Uparrow, \Rightarrow, \Downarrow, \Leftarrow, \nLeftrightarrow\}$ 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.

We learn both the policy and the models by minimizing the corresponding loss function via gradient descent. We use the Adam optimizer [14] with a learning rate of 0.2 for the former and of 0.01 for the latter, together with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These hypeparameters were chosen by trial and error from a range of $(0.001, 0.9)$. In the computation of our gradient-aware weights, we use $\|\nabla_{\boldsymbol{\theta}} \log \pi(a|s)\|_q$ with $q = 2$.

In order to 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 1. 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 gradient, is the only relevant metric. The results, shown in Table 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. The definitions of the metrics used for making the comparison, computed over an hold-out set of 1000 validation trajectories, are now presented. The model accuracy for an estimated model $\hat{p}$ is defined as $\text{acc}(\widehat{p}) = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s')\in\mathcal{D}} \mathbb{1}(s' = \arg\max_{\overline{s}} \widehat{p}(\overline{s}|s,a))$. 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 $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ and the estimated gradient $\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is defined as $\text{sim}(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})) = \frac{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \cdot \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}{\max(\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2 \|\cdot \widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2, \epsilon)}$, where ϵ is set to 10^{-8}.

Table 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	$\widehat{\nabla}_{\boldsymbol{\theta}} J$ 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

B.2 Minigolf

We adopt the version of this domain proposed by [17]. In the following, we report a brief description of the problem.

In the minigolf game, the agent has to shoot a ball with radius r inside a hole of diameter D with the minimum number of strokes. We assume that the ball moves along a level surface with a constant deceleration $d = \frac{5}{7}\rho g$, where ρ is the dynamic friction coefficient between the ball and the ground and g is the gravitational acceleration. Given the distance x_0 of the ball from the hole, the agent must determine the angular velocity ω of the putter that determines the initial velocity $v_0 = \omega l$ (where l is the length of the putter) to put the ball in the hole in one strike. For each distance x_0, the ball falls in the hole if its initial velocity v_0 ranges from $v_{min} = \sqrt{2dx_0}$ to $v_{max} = \sqrt{(2D - r)^2 \frac{g}{2r} + v_{min}^2}$. v_{max} is the maximum allowed speed of the edge of the hole to let the ball enter the hole and not to overcome it. At the beginning of each trial the ball is placed at random, between 2000 cm and 0 cm far from the hole. At each step, the agent chooses an action that determines the initial velocity v_0 of the ball. When the ball enters the hole the episode ends with reward 0. If $v_0 > v_{max}$ the ball is lost and the episode ends with reward -100. Finally, if $v_0 < v_{min}$ the episode goes on and the agent can try another hit with reward -1 from position $x = x_0 - \frac{(v_0)^2}{2d}$. The angular speed of the putter is determined by the action a selected by the agent 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. As a result, the agent is disencumbered by trying to make a hole in one shot when it is away from the hole and will prefer to perform a sequence of approach shots. The state space is divided 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 $\rho_1 = 0.19$.

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. We avoid the predictions of states that are to the right with respect to the current state by using a rectifier function. The overall prediction of the next state by the model is given by $\widehat{s}_{t+1} = s_t - \max(0, \epsilon), \epsilon \sim \mathcal{N}(V_\mu[s_t, a_t], V_\sigma[s_t, a_t])$, where V_μ and V_σ are two learnable parameters.

For all the learning algorithms, we employ a constant learning rate of 0.08 for the Adam optimizer, with $\beta_1 = 0$ and $\beta_2 = 0.999$. For training the model used by GAMPS and ML, we minimize the MSE weighted through our weighting scheme, again using Adam with learning rate 0.02 and default betas. For the estimation of the Q-function, we use the on-the-fly procedure outlined in Section 3, with an horizon of 20 and averaging over 10 rollouts. Also in this experiment we set $q = 2$ for the q-norm $\|\nabla_{\boldsymbol{\theta}} \log \pi(a|s)\|_q$ of the score. We use $\gamma = 0.99$.

B.3 Swimmer

In the swimmer task, a 3-link swimming robot is able to move inside a viscous fluid. The goal is to control the two joints of the robot in order to make it swim forward. The fully-observable state space consists of various positions and velocities, for a total of 8 scalars. The reward function is a linear combination of a *forward term*, determined by how much the action of the agent allowed it to move forward, and a *control term*, consisting of the norm of its actions. The two rewards are combined by means of a control coefficient $\alpha_{\mathrm{ctrl}} = 0.0001$.

For running GAMPS on this task, we chose to make use of more powerful model classes, in order to show that gradient-aware model learning can be effective even in the case of high-capacity regimes. Thus, we use 2-layer neural networks with 32 hidden units, that take current states and actions as inputs. To better model the uncertainty about the true model, we output a parameterized mean and diagonal covariance. Then, we sample from the resulting normal distribution the difference from the previous state. At each iteration, we train the model for 300 epochs with a learning rate of 0.0002 using Adam with default β_1 and β_2. We found beneficial, to reduce the computation burden, to employ a stopping condition on the learning of the model, stopping the training if no improvement in the training loss is detected for 5 epochs.

For computing the approximate $Q^{\pi,\widehat{p}}$, we average the cumulative return obtained by rolling our the model for 20 rollouts composed of 25 steps. The policy is learned using Adam with a learning rate of 0.008 and default βs. We use a $\gamma = 0.99$ and $q = 2$ for the weighted MSE model learning loss.

C Algorithm

C.1 Time complexity of Algorithm 1

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 3. 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 $\mathcal{O}(KNHE + KNMH^2)$.

C.2 Approximation of the value function

We now briefly review in a formal way how $Q^{\pi,\widehat{p}}$ can be estimated. For the discrete case, the standard solution is to find the fixed point of the Bellman equation:

$$\widehat{Q}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s' \sim \widehat{p}(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}} \left[\widehat{Q}(s',a') \right], \tag{16}$$

that can be found either in exact form using matrix inversion or by applying Dynamic Programming. In the continuous case, one can use approximate dynamic programming. For instance, with one step of model unrolling, the state-action value function could be found by iteratively solving the following optimization problem:

$$\widehat{Q} = \arg\min_{Q} \sum_{\tau} \sum_{t} \left(Q(s_t,a_t) - \left(r(s_t,a_t) + \gamma \mathop{\mathbb{E}}_{\substack{s_{t+1} \sim \widehat{p}(\cdot|s_t,a_t) \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[Q(s_{t+1},a_{t+1}) \right] \right) \right)^2. \tag{17}$$

The expected value in Equation (17) can be approximated by sampling from the estimated model $\widehat{p}$ 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.

The third approach, that is the one employed in GAMPS, is to directly use the estimated model for computing the expected cumulative return starting from (s,a). We can therefore use an *ephemeral* Q-function, that is obtained by unrolling the estimated model and computing the reward using the known reward function.