Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Meta Learning the Step Size in Policy Gradient Methods

Anonymous Author(s) NAME.SURNAME@INSTITUTE.EDU
Anonymous Institute

Abstract

Policy-based algorithms are among the most widely adopted techniques in model-free RL, thanks to their strong theoretical groundings and good properties in continuous action spaces. Unfortunately, these methods require precise and problem-specific hyperparameter tuning to achieve good performance and, as a consequence, they tend to struggle when asked to accomplish a series of heterogeneous tasks. In particular, the selection of the step size has a crucial impact on the ability to learn a highly performing policy, affecting the speed and the stability of the training process, and often being the main culprit for poor results. In this paper, we tackle these issues with a Meta Reinforcement Learning approach, by introducing a new formulation, known as meta-MDP, that can be used to solve any hyperparameter selection problem in RL with contextual processes. After providing a theoretical Lipschitz bound to the performance in different tasks, we adopt the proposed framework to train a batch RL algorithm to dynamically recommend the most adequate step size for different policies and tasks. In conclusion, we present an experimental campaign to show the advantages of selecting an adaptive learning rate in heterogeneous environments.

Keywords: Meta Reinforcement Learning, Hyperparameter tuning, Policy Gradient

1. Introduction

The main goal of reinforcement learning (RL, Sutton and Barto 1998) is to build an agent capable of learning a behavior that maximizes the amount of reward collected while interacting with the environment. Typically, these environments are modelled as Markov decision processes (MDP, Puterman 2014), where all trajectories share the same underlying distribution. Nevertheless, in many real world scenarios there can be exogenous variables affecting the whole process; one might think for example of a car race, where the road temperature, or the tyre choice may require different strategies. One of the most successful stream of model-free RL applications adopts policy based algorithms, that provide strong theoretical groundings and good empirical properties in continuous action spaces. Unfortunately, these methods require precise and problem-specific hyperparameter tuning to achieve good performance, causing them to struggle when applied on a series of heterogeneous tasks. The fundamental parameter to tune is the stepsize, which has a crucial impact on the ability to learn a highly performing policy, affecting the speed and the stability of the training process, and often being the main responsible for poor results. Hyperparameter optimization is an important component of automated machine learning (Bergstra and Bengio, 2012; Snoek et al., 2012; Meier et al., 2018; Zhu et al., 2019). However, these works are seldom adopted in RL (as Paul et al. 2019; Xu et al. 2018; Paine et al. 2020) as they become computationally intractable and sample inefficient. In this work, we consider the specific problem of learning how to dynamically select the best step size for each policy; moreover, we consider the case where the MDP process might differ due to the presence of the exogenous variable, here denoted as "tasks" or "contexts". To start,we propose a formalization of the problem by

©2000 Anonymous Author(s).

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v1/Author.html.

defining a meta-MDP.This general framework allows to solve a set of RL tasks, grouped as a contextual Markov decision process (Hallak et al., 2015). We discuss the main elements of the model, such as the meta objective function, which measures the return variations, and the meta action, consisting in the hyperparameter selection for a policy update. Then we consider Lipschitz meta-MDPs, in which the trajectories sampled from similar contexts have comparable properties, and we derive some guarantees on the expected return. Subsequently, we propose to learn the step size of policy gradient methods in a meta-MDP. We accomplish this through the application of a value based algorithm, known as Fitted Q-Iteration (FQI), used to dynamically recommend the most adequate step size in the current scenario. In conclusion, we evaluate our approach in various simulated environments.

2. Preliminaries

A discrete-time MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space, $\mathcal{P}(\cdot|s,a)$ is the Markovian transition, which assigns to each state-action pair (s,a) the probability of reaching the next state s', $\mathcal{R}$ is the reward distribution, with expected value $r(\cdot|s,a)$, bounded by hypothesis, i.e. $\sup_{s\in\mathcal{S},a\in\mathcal{A}} |\mathcal{R}(s,a)| \leq R_{\max}$. Finally, $\gamma \in [0,1]$ is the discount factor, and μ is the distribution of the initial state. The policy of an agent, denoted as $\pi(\cdot|s)$, assigns to each state s a density distribution over the action space $\mathcal{A}$. We can define the return of a trajectory $\tau \coloneqq (s_0, a_0, s_1, a_1, s_2, a_2, ..., a_{H-1}, s_H)$ with horizon H as the discounted sum of the reward collected: $G_\tau = \sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t)$. Consequently, it is possible to define the expected return j_π as the expected performance under policy π. In an analogous way, we can define the value function $V_\pi(s)$ and the action-value function $Q_\pi(s,a)$ as the expected return obtained by starting respectively from the state s or from the pair (s,a) and then following the policy π. For the rest of the paper, we consider parametric policies, where the policy $\pi_{\boldsymbol{\theta}}$ is parametrized by a vector $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m$. In this case, the goal is to find the optimal parametric policy maximizing the performance, i.e. $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Theta} j(\boldsymbol{\theta})$. Policy-based algorithms adopt a gradient-ascent approach, where the policy gradient $\widehat{\nabla}_N j(\boldsymbol{\theta})$ can be estimated from a batch of N trajectories (Sutton et al., 2000). An important variation on the approach consists in following the steepest ascent direction by means of the natural policy gradient (Kakade, 2001), which includes information regarding the curvature of the return manifold over the policy space in the form of the Fisher information matrix. We denote its estimator as $\widehat{g}_N(\boldsymbol{\theta})$.

Lipschitz MDP. Let $(\mathcal{X}, d_\mathcal{X})$, $(\mathcal{Y}, d_\mathcal{Y})$ be two metric spaces; a function $f : \mathcal{X} \to \mathcal{Y}$ is called L_f-Lipschitz continuous (L_f-LC), if $d_\mathcal{Y}(f(x), f(x')) \leq L_f d_\mathcal{X}(x,x') \forall x, x' \in \mathcal{X}$. Moreover, we define the Lipschitz semi-norm as $\|f\|_L = \sup_{x,x'\in\mathcal{X}: x\neq x'} \frac{d_\mathcal{Y}(f(x),f(x'))}{d_\mathcal{X}(x,x')}$. For real functions, the usual metric is the Euclidean distance while, for distributions, a common metric is the Kantorovich, or L^1-Wasserstein distance $\mathcal{K}$. Rachelson and Lagoudakis (2010b); Pirotta et al. (2015) introduced some notion of smoothness in RL by defining the Lipschitz-MDP. Let $\mathcal{M}$ be an MDP, $\mathcal{M}$ is called (L_P, L_r)-LC if for all $(s,a), (\overline{s}, \overline{a}) \in \mathcal{S} \times \mathcal{A}$:

$$\mathcal{K}\left(P(\cdot|s,a), P(\cdot|\overline{s},\overline{a})\right) \leq L_P \, d_{\mathcal{S}\times\mathcal{A}}\left((s,a),(\overline{s},\overline{a})\right), \; |r(s,a) - r(\overline{s},\overline{a})| \leq L_r \, d_{\mathcal{S}\times\mathcal{A}}\left((s,a),(\overline{s},\overline{a})\right).$$

Moreover, a policy π is called L_π-LC if $\mathcal{K}\left(\pi(\cdot|s), \pi(\cdot|\overline{s})\right) \leq L_\pi \, d_\mathcal{S}\left(s, \overline{s}\right) \forall s, \overline{s} \in \mathcal{S}$. An important result is that, for a $(L_p, L_r) - LC$ MDP, and a $L_\pi - LC$ policy, also the expected return, the $Q-$function and the gradient components are Lipschitz continuous w.r.t. $\boldsymbol{\theta}$.

Meta Reinforcement Learning. As suggested by the name, meta learning implies a higher level of abstraction with respect to regular machine learning. In particular, meta reinforcement learning (meta-RL) consists in applying meta learning techniques to RL tasks. Usually, these tasks are formalized in MDPs by a common set of parameters, known as the *context* ω. The natural candidate to represent the set of RL tasks is the contextual Markov decision process (CMDP), defined by Hallak et al. (2015) as a tuple $(\Omega, \mathcal{S}, \mathcal{A}, \mathcal{M}(\omega))$ where Ω is called the context space, $\mathcal{S}$ and $\mathcal{A}$ are the shared state and action space, and $\mathcal{M}$ is function mapping any context $\omega \in \Omega$ to an MDP, such that $\mathcal{M}(\omega) = \langle \mathcal{S}, \mathcal{A}, P_\omega, R_\omega, \gamma_\omega, \mu_\omega \rangle$. In other words, a CMDP includes in a single entity a group of tasks that share the same state and action space. In the following sections, we will also assume that γ and μ are shared across all the tasks.

3. Meta-MDP

We now present the concept of meta-MDP, a framework to solve meta-RL tasks that extends the CMDP model by including the learning model and the policy parametrization. Similar approaches to the one proposed in this section can be found in Garcia and Thomas (2019). To start, let's consider the various tasks used in a meta-training procedure as a set of MDPs such that each task $\mathcal{M}_\omega$ can be sampled from the distribution ψ defined on the context space Ω. This set can be equivalently seen as a CMDP $\mathscr{M} = \langle \Omega, \mathcal{S}, \mathcal{A}, \mathcal{M}(\omega) \rangle$, where $\mathcal{M}(\omega) = \mathcal{M}_\omega$. Similarly, we define a distribution ρ over the policy space $\boldsymbol{\Theta}$, in such a way that at each iteration in an MDP $\mathcal{M}_\omega$, the policy parameters $\boldsymbol{\theta}_0$ are initialized to a value sampled from ρ. We assume to be able to explicitly represent the task ω.

Definition 3.1 *A meta-MDP is a tuple $\langle \mathcal{X}, \mathcal{H}, \mathcal{L}, \widetilde{\gamma}, (\mathscr{M}, \psi), (\boldsymbol{\theta}, \rho), f \rangle$, where:*

- *$\mathcal{X}$ and $\mathcal{H}$ are respectively the meta observation space and the learning action;*
- *$\mathcal{L} : \boldsymbol{\Theta} \times \Omega \times \mathcal{H} \to \mathbb{R}$ is the meta reward function;*
- *$\widetilde{\gamma}$ is the meta-discount factor;*
- *$(\mathscr{M}, \psi)$ and $(\boldsymbol{\theta}, \rho)$ contain respectively a CMDP $\mathscr{M}$ with distribution over tasks ψ, and the policy space, with initial distribution ρ;*
- *$f : \boldsymbol{\Theta} \times \mathcal{H} \to \boldsymbol{\Theta}$ is the update rule of the learning model chosen.*

In particular, a meta-MDP attempts to enclose the general elements necessary to learn a RL task into a model with similar properties to a classic MDP. The meta observation space $\mathcal{X}$ of a meta-MDP can be considered as the generalization of the observation space in classic MDPs. In order to have a complete information about the current position, it is necessary to include in the state the policy $\boldsymbol{\theta}_t$ and the task ω; moreover, relevant information of the overall process is included in the gradient estimation. Consequently, in the following we will model the meta state as the concatenation of all these features.

Each action $h \in \mathcal{H}$ performed on the meta-MDP determines a specific hyperparameter that regulates the update rule f, i.e. $\boldsymbol{\theta}_{k+1} = f(\boldsymbol{\theta}_k, h_k)$. In particular, in this work we focus on (normalized) natural gradient ascent (NGA), in which the action h determines the step size. Similarly to a standard RL problem, the training of a meta-MDP is accomplished by optimizing a reward function. Meta-learning has the main goal of learning to learn: as a

consequence we want to consider learning as our reward, hence we define $\mathcal{L}(\boldsymbol{\theta}, \omega, h)$ as the performance gain obtained after one update of the policy $\boldsymbol{\theta}$ with action h under context ω: $\mathcal{L}(\boldsymbol{\theta}, \omega, h) := j_\omega(f(\boldsymbol{\theta}, h)) - j_\omega(\boldsymbol{\theta})$.

Differently from a standard MDP, a meta-MDP does not include a Markovian transition model that regulates its dynamics: given x_t, the transition to the next state x_{k+1} is, of course, stochastic, but it implicitly depends only on the distribution of the trajectories induced by the pair $(\theta_k, \mathcal{M}_\omega)$ and on the update rule f. The initial state hence depends on ψ and ρ. The choice of the meta-discount factor $\widetilde{\gamma}$ is critical: meta-learning is very often considered as paired with few-shot learning, where a short horizon is taken into account for the learning process. However, a myopic behavior induced by a low discount factor might lead to prefer actions leading to local optima, while it might be necessary to take more cautious steps in order to get to the global optima of the learning process. For this reason, we set $\widetilde{\gamma} = 1$.

4. Context Lipschitz Continuity

In this section, we consider a meta-MDP in which all inner tasks satisfy the Lipschitz continuity assumption. Under this condition, we are able to derive a bound on the approximation errors obtained by the meta-agent when acting on unseen tasks. Let's suppose to be provided with a CMDP, where all the inner MDPs are Lipschitz. Moreover, let's also assume that the set of MDPs is Lipschitz continuous w.r.t the context ω:

Assumption 4.1 *Let $\mathscr{M}$ be a CMDP. $\mathscr{M}$ is called $(L_{\omega_P}, L_{\omega_r})$-Context Lipschitz Continuous ($(L_{\omega_P}, L_{\omega_r})$-CLC) if for all $(s, a), (\overline{s}, \overline{a}) \in \mathcal{S} \times \mathcal{A}$, $\forall \omega, \widehat{\omega} \in \Omega$:*

$$\mathcal{K}\left(P_\omega(\cdot \mid s, a), P_{\widehat{\omega}}(\cdot \mid s, a)\right)) \leq L_{\omega_P} d_\Omega(\omega, \widehat{\omega}) \qquad \left|R_\omega(s, a) - R_{\widehat{\omega}}(s, a)\right| \leq L_{\omega_r} d_\Omega(\omega, \widehat{\omega});$$

This means that we have some notion of continuity w.r.t. the task: when two MDPs with similar contexts are considered, then their transition and reward processes are similar.

Theorem 4.1 *Let $\mathscr{M}$ be a $(L_{\omega_P}, L_{\omega_r})$-CLC CMDP for which $\mathcal{M}(\omega)$ is $(L_P(\omega), L_r(\omega))$-LC $\forall \omega \in \Omega$. Given a L_π-LC policy π, the action value function $Q^\pi_\omega(s, a)$ is L_{ω_Q}-CLC w.r.t. the context ω, i.e.:*

$$\left|Q^\pi_\omega(s, a) - Q^\pi_{\widehat{\omega}}(s, a)\right| \leq L_{\omega_Q}(\pi) d_\Omega(\omega, \widehat{\omega});$$

where

$$L_{\omega_Q}(\pi) = \frac{L_{\omega_r} + \gamma L_{\omega_p} L_{V^\pi}(\omega)}{1 - \gamma}, \qquad L_{V^\pi}(\omega) = \frac{L_r(\omega)(1 + L_\pi)}{1 - \gamma L_P(\omega)(1 + L_\pi)}$$

Given this result, also the return function $j_\omega(\pi)$ is L_{ω_Q}-CLC. In simpler terms, theorem 4.1 exploits the LC property to derive an upper bound on the distance between the Q^π_ω functions of two tasks $\omega_1, \omega_2 \in \Omega$. This result represents an important guarantee on the generalization capabilities of the approach, as it provides a boundary on the error obtained in testing by making inference on a Q function based on the training tasks.

5. Fitted Q-Iteration on Meta-MDP

We now define our approach to learn a dynamic learning rate in the framework of a meta-MDP. As a meta-RL approach, the objectives of our algorithm are to improve the generalization

capabilities of PG methods and to remove the need of manually tuning the learning rate for each task. The identification of an optimal dynamic step size serves two purposes: it maximizes the convergence speed by performing large updates when allowed and it improves the overall training stability, selecting low values when the return is near to the optimum or the current region is uncertain. To accomplish these goals, we propose the adoption of the Fitted Q-Iteration (FQI) (Ernst et al., 2005) algorithm, which is an off-policy, and offline algorithm designed to learn a good approximation of the optimal action-value function by exploiting the Bellman optimality operator $(T^*Q)(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s',a')\right]$. The approach consists in the application of Supervised Learning techniques as, in our case, Extra Trees (Geurts et al., 2006), in order to generalize the Q estimation over the entire state-action space. The algorithm considers a dataset of tuples representing an interaction with the environment, and at each iteration, the regression algorithm can estimate the value function with a planning horizon increased of one step. In our case, the dataset generation consists in collecting learning trajectories on the meta-MDP, where the context is sampled as well as the initial policy, while the next policies are computed through iterations of NGA with a stepsize sampled from a uniform distribution in $\mathcal{H}$. Once the dataset is generated, the meta-RL version of FQI can be applied, where the goal is to learn the action value function of the Meta-MDP. In this way, for each policy and context, once the natural gradient is estimated, the model can evaluate the best action from a discretization of $\mathcal{H}$.

6. Experimental Evaluation

In this section, we show an empirical analysis of the performance of our approach applied in three different environments. As we shall see, the meta action is capable of choosing the best stepsize, and to dynamically adapt it to fine tune the learning procedure close to the optima. When FQI models are trained, the iterations are compared and the best one is shown in Figure 1, along with NGA performed with fixed stepsize, evaluated on the same 20 random test tasks and initial policies. More details can be found in Appendix C.

For our first evaluation of the approach, we reproduce one of the environments presented in Finn et al. (2017), called Navigation2D. This environment consists of a unit square space in which an agent aims to reach a random goal in the plane, sampled uniformly in the unit square. In our second experiment, inspired by Penner (2002); Tirinzoni et al. (2019), we consider the scenario of a flat mini golf green, in which the agent has to place the ball inside the hole in the minimum number of strokes. The CMDP is built by varying the putter length and the friction coefficient. For the final experiment, we examine the CartPole balancing task (Barto et al., 1990), that consists in a pole attached to a cart, which the agent has to move to balance the pole as long as possible. The CMDP is induced by varying two environment parameters, the gravity and the pole mass.

As we can note in figure 1, the algorithm is able to select the stepsizes with a good return gain without suffering from any drop. The algorithm is able to calibrate its action, starting with larger steps and slowing down once the policies obtain good results. In addition, at least for navigation and minigolf, all trajectories reach convergence in fewer steps than any other method and consistently reach the optimal values with a low variance. The right plots of figure 1 illustrate an evaluation in the CartPole environment. Once again, the algorithm is able to dynamically select high step sizes when the return function offers large margins of

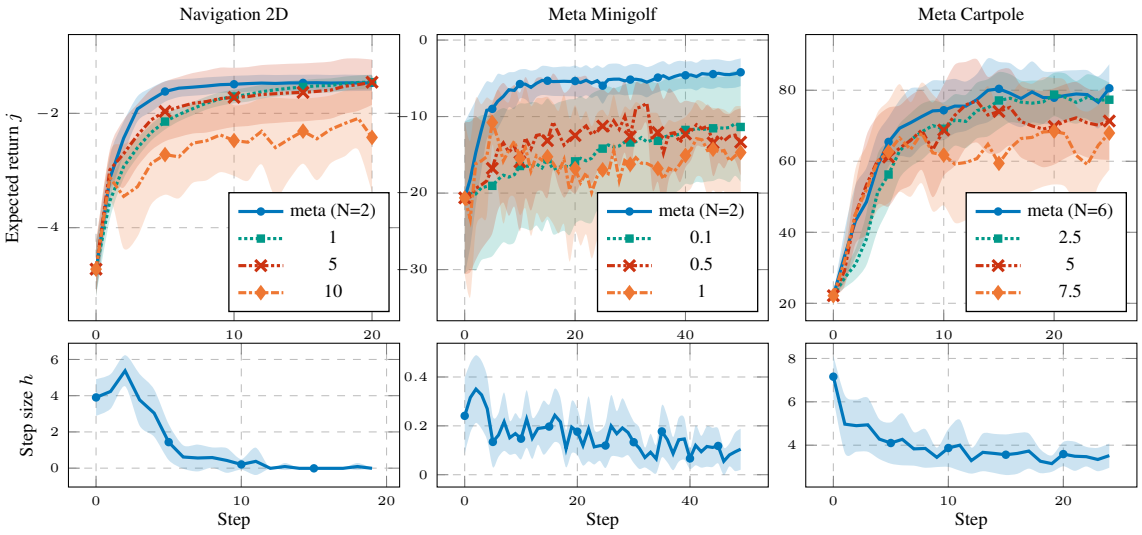

Figure 1: FQI model performance on 20 random test context against fixed step size. The top plots show the avg±std of the expected returns. The bottom plots show the avg±std meta action chosen through learning iterations. N represents the FQI iteration selected.

improvement, to then resort to smaller actions when the optimal area is reached, however in this case, the advantages w.r.t. benchmarks are smaller in terms of performance. Nonetheless, the result can still be considered a success from the perspective of automatic hyperparameter tuning, since the approach is able to autonomously match the performance of the best fixed configurations identified after a manual fine tuning of the parameter.

7. Discussion and Future Work

In this work, we considered the problem of hyperparameter tuning for policy gradient-based algorithms in contextual MDPs, modeling the general problem through the definition of the meta-MDP, for which any policy update rule can be optimized by an agent whose reward is learning. We analyzed the case of Lipschitz meta-MDPs, deriving some general guarantees that hold if the tasks in the model are smooth with respect to the parametrization of the context. Finally, we implemented the FQI algorithm on the meta-MDP where the update rule is the natural gradient ascent, and we used it to choose an adaptive stepsize through the learning process. The approach has been evaluated in different settings, where we observed good generalization capabilities of the model, thanks to which it is possible to reach fast convergence speed and robustness without the need of manual fine hyperparameter tuning. There are many important challenges to be faced in future works, other than the evaluation on more complex environments: in realistic settings, where deep policies are required, the inclusion of the whole policy in the meta state might be inefficient; a solution might consist in learning a compressed representation of the policy through autoencoders.

References

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. *Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems*, page 81–93. IEEE Press, 1990. ISBN 0818620153.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Francisco M Garcia and Philip S Thomas. A meta-mdp approach to exploration for lifelong reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1976–1978, 2019.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Franziska Meier, Daniel Kappler, and Stefan Schaal. Online learning of a memory for learning rates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2425–2432. IEEE, 2018.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/743c41a921516b04afde48bb48e28ce6-Paper.pdf`.

A Raymond Penner. The physics of golf. *Reports on Progress in Physics*, 66(2):131, 2002.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2):255–283, 2015.

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

Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. In *11th International Symposium on Artificial Intelligence and Mathematics (ISIAM 2010)*, pages 1–8, Fort Lauderdale, US, 2010a. URL `https://oatao.univ-toulouse.fr/17977/`.

Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2010, Fort Lauderdale, Florida, USA, January 6-8, 2010*, 2010b.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000. URL `https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

Andrea Tirinzoni, Mattia Salvini, and Marcello Restelli. Transfer of samples in policy search via multiple importance sampling. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6264–6274. PMLR, 09–15 Jun 2019.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

Yingda Zhu, Teruaki Hayashi, and Yukio Ohsawa. Gradient descent optimization by reinforcement learning. In *The 33rd Annual Conference of the Japanese Society for Artificial Intelligence*, 2019.

Appendix A. Proofs

In this part of the appendix, we provide the proofs of the results shown in the main paper.

A.1 Lipschitz continuity of the action-value function

Before describing the proof for Theorem 4.1, we need to recall the Bellman Operator T^π for the Action Value Function Q^π_ω:

$$T^\pi Q^\pi_\omega(s,a) = R_\omega(s,a) + \gamma \int_{\mathcal{S}} P_\omega(s'|s,a) \int_{\mathcal{A}} Q^\pi_\omega(s',a')\pi(a'|s')dads' \tag{1}$$

$$= R_\omega(s,a) + \gamma \int_{\mathcal{S}} P_\omega(s'|s,a) V^\pi_\omega(s')ds' \tag{2}$$

where Q^π_ω is the fixed point.
Moreover, let's consider a preliminary result on the LC-continuity of the value functions:

Lemma 1 (Lipschitz value functions) *Given an (L_P, L_R)-LC MDP and a L_π-LC stationary policy π, if $\gamma L_P(1+L_\pi) < 1$, then the Q-function Q^π is L_{Q^π}-LC and the V function is L_{V^π}-LC w.r.t. the joint state-action space;*

$$L_{Q^\pi} = \frac{L_R}{1-\gamma L_P(1+L_\pi)}; \qquad L_{V^\pi} = L_{Q^\pi}(1+L_\pi) \tag{3}$$

Theorem 4.1 *Let $\mathscr{M}$ be a $(L_{\omega_P}, L_{\omega_r})$-CLC CMDP for which $\mathcal{M}(\omega)$ is $(L_P(\omega), L_r(\omega))$-LC $\forall \omega \in \Omega$. Given a L_π-LC policy π, the action value function $Q^\pi_\omega(s,a)$ is L_{ω_Q}-CLC w.r.t. the context ω, i.e.:*

$$\left| Q^\pi_\omega(s,a) - Q^\pi_{\widehat{\omega}}(s,a) \right| \leq L_{\omega_Q}(\pi) d_\Omega(\omega, \widehat{\omega});$$

where

$$L_{\omega_Q}(\pi) = \frac{L_{\omega_r} + \gamma L_{\omega_p} L_{V^\pi}(\omega)}{1-\gamma}, \qquad L_{V^\pi}(\omega) = \frac{L_r(\omega)(1+L_\pi)}{1-\gamma L_P(\omega)(1+L_\pi)}$$

Proof We follow the same ideas as in Rachelson and Lagoudakis (2010a): first of all, given an L_{ω_Q}-LC continuous Q function Q^π w.r.t. the task space Ω, the related value function V^π_ω is L_{ω_Q}-LC. Indeed,

$$\begin{aligned} \left| V^\pi_\omega(s) - V^\pi_{\widehat{\omega}}(s) \right| &= \left| \int_{\mathcal{A}} \pi(a \mid s) \left(Q^\pi_\omega(s,a) - Q^\pi_{\widehat{\omega}}(s,a) \right) da \right| \\ &\leq \int_{\mathcal{A}} \pi(a \mid s) \left| Q^\pi_\omega(s,a) - Q^\pi_{\widehat{\omega}}(s,a) \right| da \\ &\leq \max_a \left| Q^\pi_\omega(s,a) - Q^\pi_{\widehat{\omega}}(s,a) \right| \leq L_{\omega_Q} d_\Omega(\omega, \widehat{\omega}). \end{aligned} \tag{4}$$

Now, we consider the iterative application of Bellman Operators, in such a way that $Q^{\pi,n+1}_\omega = T^\pi Q^{\pi,n}_\omega$, and we prove that $Q^{\pi,n}_\omega$ is $L^n_{\omega_Q}$-LC continuous, and that satisfies the recurrence relation:

$$L^{n+1}_{\omega_Q} = L_{\omega_r} + \gamma L_\pi L_V(\omega) + \gamma L^n_{\omega_Q}. \tag{5}$$

Indeed, for $n = 1$ the property holds immediately, since:

$$\left|Q_{\omega}^{\pi,1}(s,a) - Q_{\widehat{\omega}}^{\pi,1}(s,a)\right| = \left|R_{\omega}(s,a) - R_{\widehat{\omega}}(s,a)\right| \leq L_{\omega_r} d_{\Omega}(\omega, \widehat{\omega}). \tag{6}$$

Now, let us suppose the property holds for n. Then:

$$\begin{aligned}
&\left|Q_{\omega}^{\pi,n+1}(s,a) - Q_{\widehat{\omega}}^{\pi,n+1}(s,a)\right| = \\
&\left|R_{\omega}(s,a) - R_{\widehat{\omega}}(s,a) + \gamma \int_{\mathcal{S}} P_{\omega}\left(s' \mid s,a\right) V_{\omega}^{\pi,n}\left(s'\right) ds' - \gamma \int_{\mathcal{S}} P_{\widehat{\omega}}\left(s' \mid s,a\right) V_{\widehat{\omega}}^{\pi}\left(s'\right) ds'\right| \\
&\leq L_{\omega_r} d_{\Omega}(\omega, \widehat{\omega}) + \gamma \left|\int_{\mathcal{S}} \left(P_{\omega}\left(s' \mid s,a\right) - P_{\widehat{\omega}}\left(s' \mid s,a\right)\right) V_{\omega}^{\pi,n}\left(s'\right) ds'\right| \\
&+\gamma \left|\int_{\mathcal{S}} P_{\widehat{\omega}}\left(s' \mid s,a\right) \left(V_{\omega}^{\pi,n}\left(s'\right) - V_{\widehat{\omega}}^{\pi,n}\left(s'\right)\right) ds'\right| \\
&\leq L_{\omega_r} d_{\Omega}(\omega, \widehat{\omega}) + \gamma L_V(\omega) \sup_{\|f\|_L \leq 1} \left\{\left|\int_{\mathcal{S}} \left(P_{\omega}\left(s' \mid s,a\right) - P_{\widehat{\omega}}\left(s' \mid s,a\right)\right) f\left(s'\right) ds'\right|\right\} \\
&+\gamma \max_{s'} \left|V_{\omega}^{\pi,n}\left(s'\right) - V_{\widehat{\omega}}^{\pi,n}\left(s'\right)\right| \\
&\leq \left(L_{\omega_r} + \gamma L_{\omega_P} L_V(\omega) + \gamma L_{\omega_Q}^{n}\right) d_{\Omega}(\omega, \widehat{\omega}).
\end{aligned}$$

Now, if the sequence $L_{\omega_Q}^{n}$ is convergent, it converges to the fixed point of the recurrence equation:

$$L_{\omega_Q} = L_{\omega_r} + \gamma L_{\omega_P} L_V(\omega) + \gamma L_{\omega_Q}. \tag{7}$$

Hence the limit point is the one expressed in 4.1, and the sequence can be proven to be convergent since $\gamma < 1$. ■

As a consequence, the Proof that $j_{\omega}(\pi)$ is CLC under ω is immediate:

$$\begin{aligned}
\left|j_{\omega}(\pi) - j_{\widehat{\omega}}(\pi)\right| &= \left|\int_{\mathcal{S}} \mu\left(s_0\right) \left[V_{\omega}^{\pi}\left(s_0\right) - V_{\widehat{\omega}}^{\pi}\left(s_0\right)\right] ds_0\right| \\
&\leq \int_{\mathcal{S}\times\mathcal{A}} \mu\left(s_0\right) \pi\left(a \mid s_0\right) \left|Q_{\omega}^{\pi}\left(s_0,a\right) - Q_{\widehat{\omega}}^{\pi}\left(s_0,a\right)\right| da ds_0 \\
&\leq L_{\omega_Q}(\pi) d_{\Omega}(\omega, \widehat{\omega}).
\end{aligned}$$

Appendix B. FQI: dataset generation

In this section, we provide a synthetic description for FQI algorithm in the Meta-MDP. The algorithm considers a full dataset $\mathcal{F} = \{(x_t^k, h_t^k, l_t^k, x_{t+1}^k)\}_k$, where each tuple represents an interaction with the meta-MDP: in the $k-$th tuple, x_t^k and x_{t+1}^k are respectively the current and next meta-state, h_t^k the meta-action and l_t^k the meta reward function, as described in Section 3. In order to consider each meta-state x, there is the need to sample n trajectories in the inner MDP to estimate return and gradient. At the iteration N of the algorithm, given the (meta) action-value function Q_{N-1}, the training set $TS_N = \{(i^k, o^k)\}_k$ is built, where each input is equivalent to the state-action pair $i^k = (x_t^k, h_t^k)$, and the target is the result of

Algorithm 1 Meta-MDP Dataset Generation for NGA (trajectory method)

Input: CMDP $\mathscr{M}$, distribution over tasks ψ, policy space $\boldsymbol{\Theta}$, distribution over initial policies ρ, number of meta episodes K, number of learning steps T, number of inner trajectories n.
Initialize: $\mathcal{F} = \{\}$,
for $k = 1, \dots, K$ **do**
  Sample context $\omega \sim \psi(\Omega)$ and initial policy $\boldsymbol{\theta}_0 \sim \rho(\boldsymbol{\Theta})$
  Sample n trajectories in task $\mathcal{M}_\omega$ under policy $\pi(\boldsymbol{\theta}_t)$ and Estimate $j_\omega(\boldsymbol{\theta}_0)$, $\widehat{g}_n(\boldsymbol{\theta}_0, \omega)$
  for $t = 0, \dots, T-1$ **do**
    Sample meta-action $h \in \mathcal{H}$ and Update policy $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + h \frac{\widehat{g}_N(\boldsymbol{\theta}_t, \omega)}{\|\widehat{g}_n(\boldsymbol{\theta}_t, \omega)\|}$
    Sample n trajectories in $(\mathcal{M}_\omega, \pi(\boldsymbol{\theta}_t))$ and estimate $j_\omega(\boldsymbol{\theta}_{t+1})$, $\widehat{g}_n(\boldsymbol{\theta}_{t+1}, \omega)$
    Set $x = \langle \boldsymbol{\theta}_t, \widehat{g}_n(\boldsymbol{\theta}_t, \omega), \omega \rangle$, $x' = \langle \boldsymbol{\theta}_{t+1}, \widehat{g}_n(\boldsymbol{\theta}_{t+1}, \omega), \omega \rangle$, $l = j_\omega(\boldsymbol{\theta}_{t+1}) - j_\omega(\boldsymbol{\theta}_t)$.
    Append $\{(x, h, x', l)\}$ to $\mathcal{F}$
  end for
end for
Output: $\mathcal{F}$

Bellman optimal operator: $o^k = l_t^k + \gamma \max_{h \in \mathcal{H}} Q_{N-1}(x_{t+1}^k, h)$. In this way, the regression algorithm adopted is trained on TS to learn Q_N with the learning horizon increased of one step. However, as the regression procedures are iterated, new estimation errors are introduced that might cumulate over time, resulting in a degradation of the performances with N.

In general, the dataset is created by following K learning trajectories over the CMDP: at the beginning of each meta-episode, a new context ω and initial policy $\boldsymbol{\theta}_0$ are sampled from ψ and ρ; then, for each of the T learning steps, the meta action h is randomly sampled to perform the policy update. In this way, the overall dataset is composed of KT tuples. However, if the policy space is small enough, it is possible to explore the overall task-policy space $\Omega \times \boldsymbol{\Theta}$ through a generative approach: instead of following the learning trajectories, both $\omega, \boldsymbol{\theta}_0$ and h are sampled every time. We refer to this method as "generative" approach, while the former will be denoted as "trajectory" approach.

The pseudo code for the dataset generation process is provided in algorithm 1.

Appendix C. Experiment Details

In this section, we provide more details regarding the experimental campaign provided. In the following environments, all the policies considered are Gaussian, and linear w.r.t. the state observed (with bias θ_0), i.e. $\pi_{\boldsymbol{\theta}}(a|s) \sim \mathcal{N}(\theta_0 + \boldsymbol{\theta}^\top s, \sigma^2)$, where σ is fixed standard deviation, with a different setting for each environment.

C.1 Navigation2D Description

The Navigation2D environment consists of a 2-dimensional square space in which an agent, represented as a point, aims to reach a goal in the plane traversing the minimum distance.

At the start of the episode, the agent is placed in the initial position $s_0 = (0, 0)$ of the Cartesian plane. Then, at each step t the agent observes its current position and performs

an action a_t corresponding to movement speeds along the x and y axes:

$$a_t = (v_x, v_y), \text{ where } v_x, v_y \in [-v^{\max}, v^{\max}]. \tag{8}$$

According to this action, the agent can move in every direction of the plane, with a limit on the maximum speed $v^{\max} = 0.1$ allowed in a single step. This parameters determines the minimum number of steps necessary to reach the goal and can be varied to tune the difficulty of the environment.

At each step, the environment produces a reward equal to the negative Euclidean distance from the goal:

$$r_t = \sqrt{(x_t - x_{\text{goal}})^2 + (y_t - y_{\text{goal}})^2}. \tag{9}$$

An episode terminates when the agent is within a threshold distance d_{thresh} from the goal or when the horizon $H = 10$ is reached.

The distribution of tasks is implemented as a CMDP $\mathcal{M}(\omega)$ in which, at each episode, a different goal point is selected at random. The context ω is given by a 2D vector, such that:

$$\omega = (x_{\text{goal}}, y_{\text{goal}}), \text{ where } x_{\text{goal}}, y_{\text{goal}} \sim U(-1, 1). \tag{10}$$

Parameters used for experiments:

- initial policy distribution $\rho = \mathcal{N}(0, 0.1)$;
- discount factor $\gamma = 0.99$;
- policy standard deviation $\sigma = 1.001$;
- task distribution $\psi = \mathcal{U}([-0.5, 0.5]^2)$;
- FQI dataset method: trajectories;
- FQI number of samples: $K = 4000$ with learning horizon $T = 20$;
- inner trajectories $n = 200$ with horizon $H = 10$;
- number of estimators $= 50$, minimum samples split $= 0.01$;
- step size $\mathcal{H} = [0, 8]$;
- step size sampling distribution: uniform in $\mathcal{H}$;
- step size selected in evaluation from an evenly spaced discretization of 101 values in $\mathcal{H}$.

C.2 Minigolf Description

In the minigolf game, the agent has to shoot a ball with radius r inside a hole of diameter D with the smallest number of strokes. The friction imposed by the green surface is modeled by 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 of the ball from the hole, the agent must choose the force a, from which the velocity of the ball v of the ball is determined as $v = al^2(1 + \epsilon)$, where $\epsilon \sim \mathcal{N}(0, 0.25)$ and l is the putter length. For each distance x, the ball falls in the hole if its velocity v ranges from $v_{min} = \sqrt{2dx}$ to $v_{max} = \sqrt{(2D - r)^2 \frac{g}{2r} + v_{min}^2}$. In this case, the episode ends with a reward 0; if $v > v_{max}$ the ball falls outside the green, and the episode ends with a reward -100. Otherwise, if

$v < v_{min}$, the agents gets a reward equal to -1, and the episode goes on from a new position $x_{new} = x_{old} - \frac{v^2}{2d}$. At the beginning of each episode, the initial position is selected from an uniform distribution between 0m and 20m from the hole. The stochasticity of the action implies that the stronger is the action chosen the more uncertain is the outcome, as the effect of r.v. ϵ become more effective. As a result, when it is away from the hole, the agent might not prefer to try to make a hole in one shot, preferring to perform a sequence of closer shots. In this case, the context is given by the friction coefficient $\rho \in [0.065, 0.196]$ and by the putter length $l \in [0.7, 1]m$.

During the experiment, the environment parameters are set to imitate the dynamics of a realistic shot in a minigolf green, within the limits of our simplified simulation. This is the complete configuration adopted:

- horizon $H = 20$;
- discount factor $\gamma = 0.99$;
- angular velocity $\omega \in [1 \times 10^{-5}, 10]$;
- initial distance $x_0 \in [0, 20]$ meters;
- ball radius $r = 0.02135$ meters;
- hole diameter $D = 0.10$ meters;
- gravitational acceleration $g = 9.81 \frac{\text{meters}}{\text{second}^2}$.

The distribution of tasks is built as a CMDP $\mathcal{M}(\omega)$, induced by the pair $\omega = (l, \rho)$. At each meta episode, a new task is sampled from a multivariate uniform distribution within this ranges:

- putter length $l \sim U(0.7, 1)$ meters;
- friction coefficient $\rho \sim U(0.065, 0.196)$.

Parameters used for experiments:

- initial policy distribution $\boldsymbol{\theta} = (w, b) \sim U((-1, 2), (-2, 3.5))$ (2-dimensional policy);
- policy standard deviation $\sigma = 0.1$;
- FQI dataset method: generative;
- FQI number of samples: $K = 10000$;
- inner trajectories $n = 400$ with horizon $H = 20$;
- number of estimators = 50, minimum samples split = 0.01;
- step size space: $\mathcal{H} = [0, 1]$
- step size sampling distribution: uniform in $\mathcal{H}$;
- step size selected in evaluation from an evenly spaced discretization of 101 values in $\mathcal{H}$.

C.3 CartPole description

The CartPole environment (Barto et al., 1990), also known as the Inverted Pendulum problem, consists in a pole attached to a cart by a non actuated joint, making it an inherently unstable system. The cart can move horizontally along a frictionless track to balance the pole. The objective is to maintain the equilibrium as long as possible.

In this implementation, an episode starts with the pendulum in vertical position. At each step, the agent observes the following 4-tuple of continuous values:

- cart position $x_{\text{cart}} \in [-4.8, 4.8]$;
- cart velocity $v_{\text{cart}} \in \mathbb{R}$;
- pole angle $\phi_{\text{pole}} \in [-0.418, 0.418]$ rad;
- pole angular velocity $\omega_{\text{pole}} \in \mathbb{R}$.

Given the state, the agent chooses an action between 0 and 1 to push the cart to the left or to the right. For each step in which the pole is in balance, the environment produces a reward of $+1$. An episode ends when the pole angle from the vertical position is higher than 12 degrees, or the cart moves more than 2.4 units from the center, or the horizon $H = 100$ is reached.

In our experiments, we set the environment parameters to these values:

- mass of the cart $m_{cart} = 1$ kg;
- length of the pole $l_{pole} = 0.5$ m;
- force applied by the cart $F = 10$ N.

The CMDP $\mathcal{M}(\omega)$ is induced by varying two environment parameters, the gravity g and the pole mass m_{pole}, that form the context parameterization $\omega = (g, m_{pole})$. Each task in the meta-MDP is built by sampling ω from a multivariate uniform distribution, within these ranges:

- gravity $g \sim U(9.8, 12.8)$ $\frac{\text{m}}{\text{s}^2}$;
- pole mass $m_{pole} \sim U(0.1, 3)$ kg.

Parameters used for experiments:

- initial policy distribution $\boldsymbol{\theta_d} \sim \mathcal{N}(0, 0.1)$ for each component $\boldsymbol{\theta}_d$;
- policy standard deviation $\sigma = 1.001$;
- FQI dataset method: trajectories;
- FQI number of samples: $K = 10000$ with learning horizon $T = 25$;
- inner trajectories $n = 100$ with horizon $H = 100$;
- number of estimators $= 150$, minimum samples split $= 0.025$;
- step size $\mathcal{H} = [0, 10]$;
- step size sampling distribution: uniform in $\mathcal{H}$;
- step size selected in evaluation from an evenly spaced discretization of 101 values in $\mathcal{H}$.

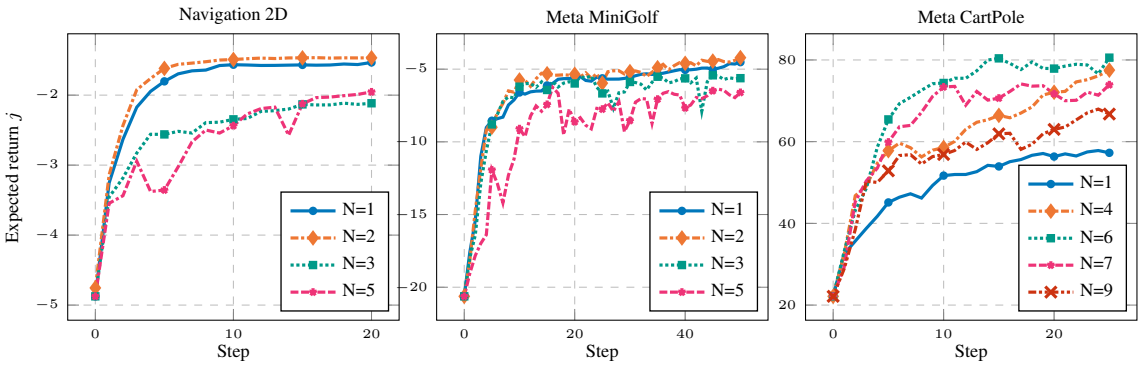

Figure 2: FQI model performance among different iterations of FQI. For the sake of clarity, only the average values are shown.

Appendix D. Other results

In this section of the appendix, we provide more experimental results.

Comparison among FQI Iterations

As said, as the regression procedures are iterated in the application of FQI algorithm, there is a trade-off in performing new iterations, between a larger planning horizon and the accumulation of new regression errors. In figure 2 it is possible to detect the best model to choose: for both Navigation2D and Minigolf, the direct regression on the meta reward provides good results (i.e. one FQI iteration); however the second one seems to reach better performances. From the third iteration, it is possible to see that the results get worse. Finally, CartPole is more complex and with a larger dimension of the policy space: consequently, more iterations are needed to improve the choice of the stepsize. In this case, the best iteration seems to be reached at $N = 6$.

Comparison with learning rate schedules:

In figure 1 we compared our approach with the choice of a fixed stepsize. Other schedules are often considered, as for example a dynamic stepsize $h_{t+1} = \frac{\alpha}{t}$ (a similar case is an exponentially decreasing learning rate $h_{t+1} = \alpha h_t$). We show our results in figure 3, in the same setting as the one provided in section 6. The baselines considered are three different starting stepsizes h_0, where the red line is the one performing best, with respect to the range taken into account.[1]

1. note for reviewers: comparison with Adam and RMSProp schedules are planned to be provided in the eventual camera-ready version. Those schedules have a poor performance when applied to natural gradient updates, hence we consider not fair to show them in this version. We will include their application on the standard gradient ascent approach.

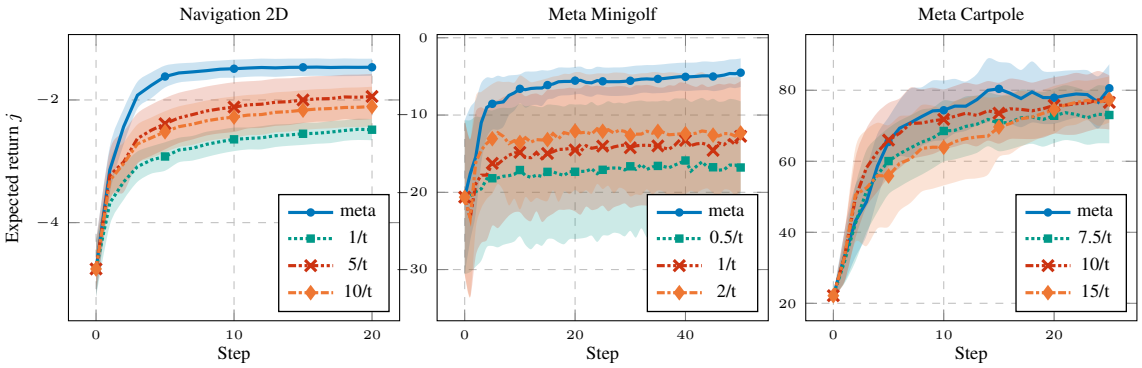

Figure 3: FQI model performance on 20 random test context against a decreasing step size (avg$\pm$*std*).

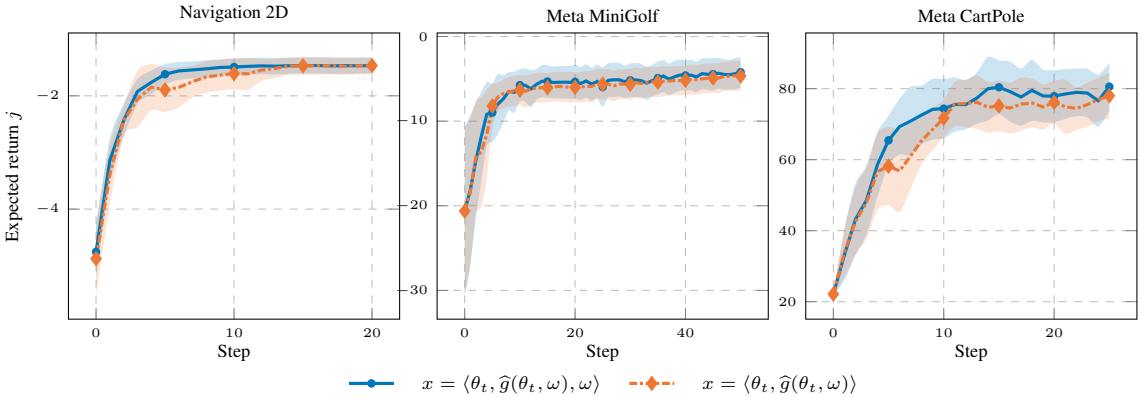

Figure 4: FQI model performance obtained by considering or excluding the explicit task parametrization ω. (avg$\pm$*std*)

Explicit knowledge of the context: is it informative?

In the experiments, we assumed to be able to represent the parametrized context ω, as this information can be used to achieve an *implicit task-identification* by the agent. However, in some cases the external variables influencing the process might be not observable. Hence, the Meta-MDP can be modeled by creating a different task representation. However, the gradient itself already implicitly includes information regarding the transition and reward probabilities: what is lost when we do not consider the explicit parametrization of the task? We address this question by retraining our models, and showing the results in figure 4: in general, there is no big loss in the performance, especially for minigolf environment; however, in CartPole, which is the most difficult, the task parametrization seem to be informative to the choice of the stepsize.