How does explicit exploration influence Deep Reinforcement Learning?

Jakob J. Hollenstein, Erwan Renaudo, Matteo Saveriano, Justus Piater
University of Innsbruck
{jakob.hollenstein,erwan.renaudo,matteo.saveriano,justus.piater}@uibk.ac.at

Abstract. *Most Deep Reinforcement Learning (D-RL) methods perform local search and therefore are prone to get stuck in non-optimal solutions. To overcome this issue, we exploit simulation models and kinodynamic planners as exploration mechanism in a model-based reinforcement learning method. We show that, even on a simple toy domain, D-RL methods are not immune to local optima and require additional exploration mechanisms. In contrast, our planning-based exploration exhibits a better state space coverage which turns into better policies than the ones learned via standard D-RL methods.*

1. Introduction

Deep-Reinforcement Learning (D-RL) has shown promising results in challenging robotics domains (*e.g.* [4]), but can be resource demanding and difficult to train. We assume that part of the difficulty of learning good policies is related to insufficient exploration. Other D-RL methods like [1, 3, 6] partially address the problem by increasing the number of training steps, or by relying on the environment implementation to provide *exploring-starts* to cover a diverse enough state-space region. However, these solutions are impractical and potentially dangerous in robotics applications.

In the robotic context, directed exploration via physically-based simulation appears more promising to find good solutions more reliably and in less time. Therefore, this work proposes the *Planning for Policy Search (PPS)* method that exploits a kinodynamic planner in the exploration phase to collect data which are then used to learn a policy, thereby eliminating the planning time during execution. PPS is tested on the point mass system described in Table 1 and compared with D-RL approaches.

This research has received funding from the European Union's Horizon 2020 research and innovation programme (grant agreement no. 731761, IMAGINE)

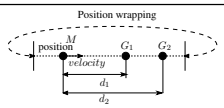
Dynamics
$X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$  $\dot{x} = Ax + Bu$
Reward
$\max((1 - \tanh\lvert X - G_1^*\rvert), 2(1 - \tanh\lvert X - G_2^*\rvert))$ $G_1 = \begin{bmatrix} -2.5 \\ 0.0 \end{bmatrix}$ $G_2 = \begin{bmatrix} 6.0 \\ 0.0 \end{bmatrix}$
Limits
$u \in [-1; 1]$ $x \in [-10; 10]$ $\dot{x} \in [-2.5; 2.5]$

Table 1. Description of the 1D double-integrator test environment: a point mass M can be moved in a one-dimensional space position-velocity $X = [x, \dot{x}]$ by applying a continuous-valued force. Reward is received based on the distance to two possible goal locations (G_1, G_2).

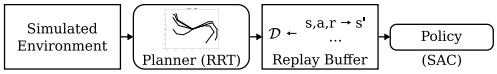


Figure 1. Illustration of PPS Method

2. Planning for Policy Search

The presented PPS implementation (Figure 1) consists of a Linear Quadratic Regulator (LQR)-Rapidly Exploring Random Tree (RRT) [5] to create a tree of data $\mathcal{D} = \{(s, a, r, s'), \ldots\}$ from which Soft-Actor Critic (SAC) [1] learns a policy. In contrast to [5] quadratic programming-based finite-horizon steering is used to extend the tree. In our setup, all the environment interaction data created by RRT are used as training data for the policy rather than using only successful trajectories as expert demonstrations.

3. Evaluation

PPS is evaluated in the one-dimensional goal reaching task presented in Table 1. The environment contains two distinct goal locations. The agent receives a reward based on the distance to the goal

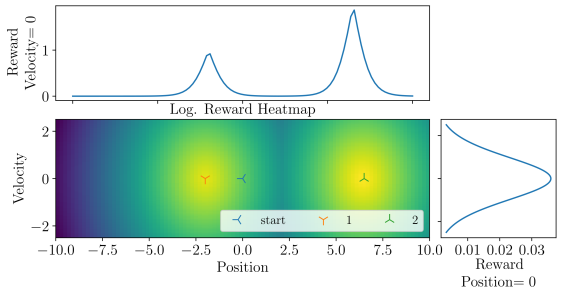


Figure 2. The reward (heatmap) and reward distributions (plots above and on the right of the heatmap) for the double-integrator. The agent starts at $x = 0$ with $\dot{x} = 0$. The reward is based on the distance of the agent to the goal positions 1 and 2.

Alg.	DDPG	PPO	SAC	**PPS (RRT)**
Non-Ex.	15.5%	20.8%	20.4%	**79.3%**
Ex.	59.0%	60.9%	61.0%	-

Table 2. Final coverage as percent of visited bins.

points. The goals ($x_1 = -2.5$ and $x_2 = 6.0$) are chosen such that simply maximizing the reward from the starting position leads to a suboptimal policy, i.e. a local optimum (see Figure 2).

We compare the performance of PPS against the prominent D-RL algorithms Proximal Policy Gradient (PPO) [6], Deep Deterministic Policy Gradient (DDPG) [3], and SAC [1], using the implementation in [2].The algorithms are run for 10^5 environment steps; the D-RL algorithms use 100-step episodes. To have a broader baseline we included an *exploring-starts* mechanism where the initial state of the double integrator is sampled uniformly. However, especially in robotic tasks, exploring starts are impractical and potentially dangerous and should be avoided.

We first compare the state-space coverage obtained from data collected during the exploration phase of the different D-RL approaches. The coverage is calculated as the percentage of non-empty, uniformly-shaped bins. The number of bins is set to $\sqrt{10^5/5}$ in each dimension, i.e. we expect 5 data points in each bin on average. See Table 2 for the final coverages.

Second, Figure 3 depicts boxplots of the evaluation returns achieved by the D-RL algorithms after training for 10^5 steps. DDPG achieves higher rewards without exploring starts, while PPO and SAC profits from exploring starts. Our PPS method shows improved performance compared to non-exploring starts methods. Moreover, the policies learned with PPS achieve performance comparable to the directly-trained SAC policy with exploring starts.

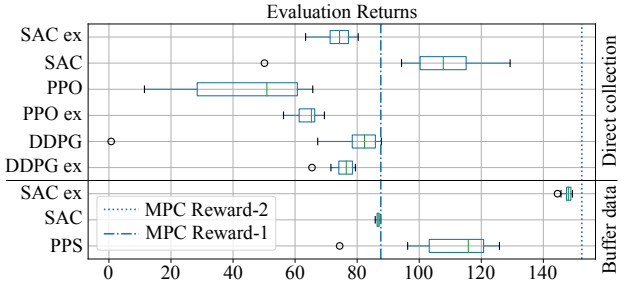


Figure 3. Box plot of the return distributions (11 independent runs); each run consists of the mean of 10 evaluation runs. The evaluation runs are performed towards the end of the training process, equally spaced 10 learning episodes apart.

4. Discussion

In this work, we highlighted that standard D-RL algorithms are not immune to getting stuck in suboptimal policies even in a toy problem with two local optima. The agent controlled by PPS explores a wider part of the state space than D-RL methods that focus on reward accumulation, even with exploring starts. The data gathered by RRT are not biased by reward accumulation and is thus more representative of the environment.

References

[1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Int. Conf. Machine Learning (ICML)*, 2018.

[2] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable Baselines. *GitHub repository*, 2018.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proc. 4th Int. Conf. Learning Representations, (ICLR)*, 2016.

[4] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *I. J. Robotics Res.*, 39(1), 2020.

[5] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE Int. Conf. Robotics and Automation*, 2012.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.