TD-Regularized Actor-Critic Methods

Simone Parisi*[1], Voot Tangkaratt[2], Jan Peters[1,3], and Mohammad Emtiyaz Khan[2]

[1] *Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany*
[2] *RIKEN Center for Advanced Intelligence Project , 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan*
[3] *Max Planck Institute for Intelligent Systems, Spemannstr. 41, 72076 Tübingen, Germany*

Machine Learning Journal (in press)

Abstract

Actor-critic methods can achieve incredible performance on difficult reinforcement learning problems, but they are also prone to instability. This is partly due to the interaction between the actor and the critic during learning, e.g., an inaccurate step taken by one of them might adversely affect the other and destabilize the learning. To avoid such issues, we propose to regularize the learning objective of the actor by penalizing the temporal difference (TD) error of the critic. This improves stability by avoiding large steps in the actor update whenever the critic is highly inaccurate. The resulting method, which we call the *TD-regularized* actor-critic method, is a simple plug-and-play approach to improve stability and overall performance of the actor-critic methods. Evaluations on standard benchmarks confirm this.
Source code can be found at `https://github.com/sparisi/td-reg`

Keywords: reinforcement learning, actor-critic, temporal difference

1 Introduction

Actor-critic methods have achieved incredible results, showing super-human skills in complex real tasks such as playing Atari games and the game of Go [Silver et al., 2016, Mnih et al., 2016]. Unfortunately, these methods can be extremely difficult to train and, in many cases, exhibit unstable behavior during learning. One of the reasons behind their instability is the interplay between the actor and the critic during learning, e.g., a wrong step taken by one of them might adversely affect the other and can destabilize the learning [Dai et al., 2018]. This behavior is more common when nonlinear approximators, such as neural networks, are employed, but it could also arise even when simple linear functions are used[1]. Figure 1 (left) shows such an example where a linear function is used to model the critic but the method fails in three out of ten learning trajectories. Such behavior is only amplified when deep neural networks are used to model the critic.

In this paper, we focus on developing methods to improve the stability of actor-critic methods. Most of the existing methods have focused on stabilizing either the actor or the critic. For example, some recent works improve the stability of the critic by using a slowly-changing critic [Lillicrap et al., 2016, Mnih et al., 2015, Hessel et al., 2018], a low-variance critic [Munos et al., 2016, Gruslys et al., 2018], or two separate critics to reduce their bias [van Hasselt, 2010, Fujimoto et al., 2018]. Others have proposed to stabilize the actor instead, e.g., by constraining its update using entropy or the Kullback-Leibler (KL) divergence [Peters et al., 2010, Schulman et al., 2015, Akrour et al., 2016, Achiam et al., 2017, Nachum et al., 2018, Haarnoja et al., 2018]. In contrast to these approaches that focus on stabilizing either the actor or the critic, we focus on stabilizing the *interaction* between them.

*Corresponding author: `parisi@ias.tu-darmstadt.de`

[1] Convergence is assured when special types of linear functions known as *compatible* functions are used to model the critic [Sutton et al., 1999, Peters and Schaal, 2008]. Convergence for other types of approximators is assured only for some algorithms and under some assumptions [Baird, 1995, Konda and Tsitsiklis, 2000, Castro et al., 2008].

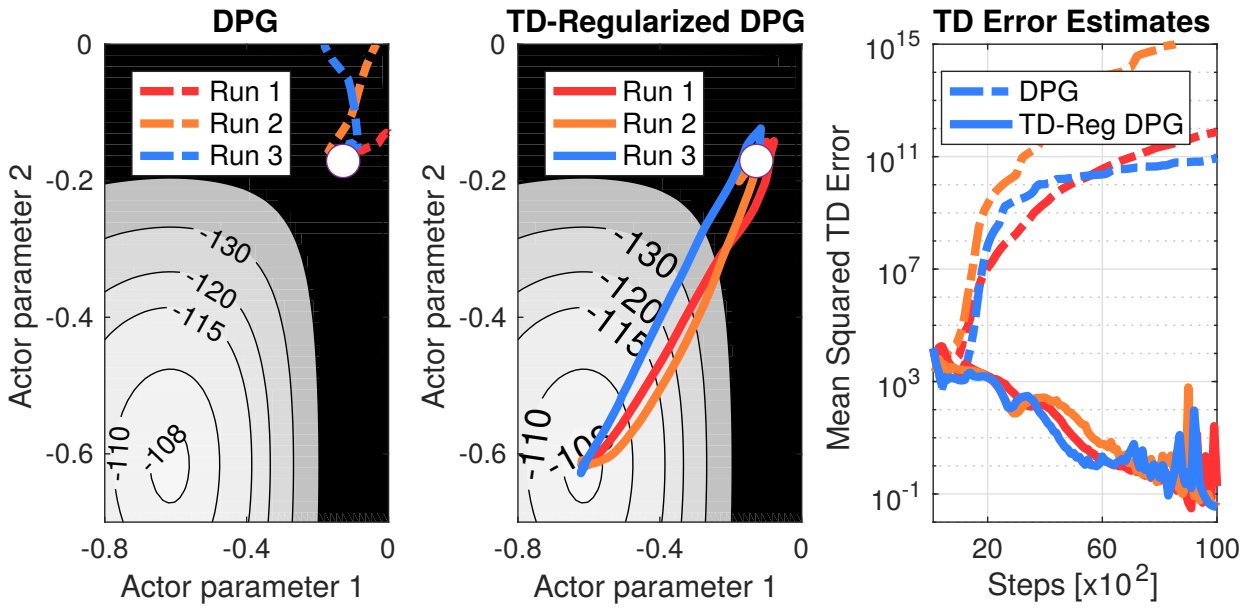


Figure 1: Left figure shows three runs that failed to converge out of ten runs for an actor-critic method called deterministic policy gradient (DPG). The contour lines show the true expected return for the two parameters of the actor, while the white circle shows the starting parameter vector. For DPG, we approximate the value function by an incompatible linear function (details in Section 5.1). None of the three runs make it to the maximum which is located at the bottom-left corner. By contrast, as shown in the middle figure, adding the TD-regularization fixes the instability and all the runs converge. The rightmost figure shows the estimated TD error for the two methods. We clearly see that TD-regularization reduces the error over time and improves not only stability and convergence but also the overall performance.

Our proposal is to stabilize the actor by penalizing its learning objective whenever the critic's estimate of the value function is highly inaccurate. We focus on critic's inaccuracies that are caused by severe violation of the Bellman equation, as well as large temporal difference (TD) error. We penalize for such inaccuracies by adding the critic's TD error as a regularization term in the actor's objective. The actor is updated using the usual gradient update, giving us a simple yet powerful method which we call the *TD-regularized actor-critic method*. Due to this simplicity, our method can be used as a plug-and-play method to improve stability of existing actor-critic methods together with other critic-stabilizing methods. In this paper, we show its application to stochastic and deterministic actor-critic methods [Sutton et al., 1999, Silver et al., 2014], trust-region policy optimization [Schulman et al., 2015] and proximal policy optimization [Schulman et al., 2017], together with Retrace [Munos et al., 2016] and double-critic methods [van Hasselt, 2010, Fujimoto et al., 2018]. Through evaluations on benchmark tasks, we show that our method is complementary to existing actor-critic methods, improving not only their stability but also their performance and data efficiency.

1.1 Related Work

Instability is a well-known issue in actor-critic methods, and many approaches have addressed it. The first set of methods do so by stabilizing the critic. For instance, the so-called target networks have been regularly used in deep reinforcement learning to improve stability of TD-based critic learning methods [Mnih et al., 2015, Lillicrap et al., 2016, van Hasselt, 2010, Gu et al., 2016b]. These target networks are critics whose parameters are slowly updated and are used to provide stable TD targets that do not change abruptly. Similarly, Fujimoto et al. [2018] proposed to take the minimum value between a pair of critics to limit overestimation, and delay to update the policy parameters so that the per-update error is reduced. Along the same line, Munos et al. [2016] proposed to use truncated importance weighting to compute low-variance TD targets to stabilize critic learning. Instead, to avoid sudden changes in the critic, Schulman et al. [2016] proposed to constrain the learning of the value function such that the average Kullback-Leibler divergence between the previous and the current value function is sufficiently small. All of these methods can be categorized as methods that improve the stability by stabilizing the critic.

An alternative approach is to stabilize the actor by forcing it to not change abruptly. This is often done by incorporating a Kullback-Leibler divergence constraint to the actor learning objective. This constraint ensures that the actor does not take a large update step, ensuring safe and stable actor learning [Peters et al., 2010, Schulman et al., 2015, Akrour et al., 2016, Achiam et al., 2017, Nachum et al., 2018].

Our approach differs from both these approaches. Instead of stabilizing either the actor or the critic, we focus on stabilizing the interaction between the two. We do so by penalizing the mistakes made by the critic during the learning of the actor. Our approach directly addresses the instability arising due to the interplay between the actor and the critic.

Prokhorov and Wunsch [1997] proposed a method in a spirit similar to our approach where they only update the actor when the critic is sufficiently accurate. This *delayed* update can stabilize the actor, but it might require many more samples to ensure an accurate critic, which could be time consuming and make the method very slow. Our approach does not have this issue. Another recent approach proposed by Dai et al. [2018] uses a dual method to address the instability due to the interplay between the actor and the critic. In their framework the actor and the critic have competing objectives, while ours encourages cooperation between them.

2 Actor-Critic Methods and Their Instability

We start with a description of the reinforcement learning (RL) framework, and then review actor-critic methods. Finally, we discuss the sources of instability considered in this paper for actor-critic methods.

2.1 Reinforcement Learning and Policy Search

We consider RL in an environment governed by a Markov Decision Process (MDP). An MDP is described by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mu_1 \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^{d_{\mathcal{S}}}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^{d_{\mathcal{A}}}$ is the action space, $\mathcal{P}(s'|s,a)$ defines a Markovian transition probability density between the current s and the next state s' under action a, $\mathcal{R}(s,a)$ is the reward function, and μ_1 is initial distribution for state s_1. Given such an environment, the goal of RL is to learn to act. Formally, we want to find a *policy* $\pi(a|s)$ to take an appropriate action when the environment is in state s. By following such a policy starting at initial state s_1, we obtain a sequence of states, actions and rewards $(s_t, a_t, r_t)_{t=1\ldots T}$, where $r_t = \mathcal{R}(s_t, a_t)$ is the reward at time t (T is the total timesteps). We refer to such sequences as *trajectories* or *episodes*. Our goal is to find a policy that maximizes the expected return of such trajectories,

$$\max_{\pi} \; \mathbb{E}_{\mu_\pi(s)\pi(a|s)}[Q^\pi(s,a)]\,, \tag{1}$$

$$\text{where} \;\; Q^\pi(s_t,a_t) := \mathbb{E}_{\prod_{i=t+1}^{T}\pi(a_i|s_i)\mathcal{P}(s_{i+1}|s_i,a_i)}\left[\sum_{i=t}^{T}\gamma^{i-t}r_{i+1}\right], \tag{2}$$

where $\mu_\pi(s)$ is the state distribution under π, i.e., the probability of visiting state s under π, $Q^\pi(s,a)$ is the action-state value function (or Q-function) which is the expected return obtained by executing a in state s and then following π, and, finally, $\gamma \in [0,1)$ is the discount factor which assigns weights to rewards at different timesteps.

One way to solve the optimization problem of Eq. (2) is to use policy search [Deisenroth et al., 2013], e.g., we can use a parameterized policy function $\pi(a|s;\boldsymbol{\theta})$ with the parameter $\boldsymbol{\theta}$ and take the following gradients steps

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_{\boldsymbol{\theta}}\, \nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{\mu_\pi(s)\pi(a|s;\boldsymbol{\theta})}[Q^\pi(s,a)]\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}\,, \tag{3}$$

where $\alpha_{\boldsymbol{\theta}} > 0$ is the stepsize and i is a learning iteration. There are many ways to compute a stochastic estimate of the above gradient, e.g., we can first collect one trajectory starting from $s_1 \sim \mu_1(s)$, compute Monte Carlo estimates $\widehat{Q}^\pi(s_t,a_t)$ of $Q^\pi(s_t,a_t)$, and then compute the gradient using REINFORCE [Williams, 1992] as

$$\nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{\mu_\pi(s)\pi(a|s;\boldsymbol{\theta})}[Q^\pi(s,a)] \approx \sum_{t=1}^{T}\left[\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t;\boldsymbol{\theta})\right]\widehat{Q}^\pi(s_t,a_t), \tag{4}$$

$$\text{where} \quad \widehat{Q}^\pi(s_t,a_t) = \mathcal{R}(s_t,a_t) + \gamma\widehat{Q}^\pi(s_{t+1},a_{t+1}), \quad \forall t = 1,2,\ldots,T-1,$$

and $\widehat{Q}^{\pi}(s_T, a_T) := \mathcal{R}(s_T, a_T) = r_T$. The recursive update to estimate $\widehat{Q}^{\pi}$ is due to the definition of Q^{π} shown in Eq. (2). The above stochastic gradient algorithm is guaranteed to converge to a locally optimal policy when the stepsize are chosen according to Robbins-Monro conditions [Robbins and Monro, 1985]. However, in practice, using one trajectory might have high variance and the method requires averaging over many trajectories which could be inefficient. Many solutions have been proposed to solve this problem, e.g., baseline substraction methods [Greensmith et al., 2004, Gu et al., 2016a, Wu et al., 2018]. Actor-critic methods are one of the most popular methods for this purpose.

2.2 Instability of Actor-Critic Methods

In actor-critic methods, the value function is estimated using a parameterized function approximator, i.e., $Q^{\pi}(s, a) \approx \widehat{Q}(s, a; \boldsymbol{\omega})$, where $\boldsymbol{\omega}$ are the parameters of the approximator such as a linear model or a neural network. This estimator is called the *critic* and can have much lower variance than traditional Monte Carlo estimators. Critic's estimate are used to optimize the policy $\pi(a|s; \boldsymbol{\theta})$, also called the *actor*.

Actor-critic methods alternate between updating the parameters of the actor and the critic. Given the critic parameters $\boldsymbol{\omega}_i$ at iteration i and its value function estimate $\widehat{Q}(s, a; \boldsymbol{\omega}_i)$, the actor can be updated using a policy search step similar to Eq. (3),

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_{\boldsymbol{\theta}} \quad \nabla_{\boldsymbol{\theta}} \, \mathbb{E}_{\mu_\pi(s)\pi(a|s;\boldsymbol{\theta})} \Big[\widehat{Q}(s, a; \boldsymbol{\omega}_i) \Big] \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} . \tag{5}$$

The parameters $\boldsymbol{\omega}$ are updated next by using a gradient method, e.g., we can minimize the *temporal difference* (TD) error δ_Q using the following update:

$$\begin{aligned} \boldsymbol{\omega}_{i+1} &= \boldsymbol{\omega}_i + \alpha_{\boldsymbol{\omega}} \, \mathbb{E}_{\mu_\pi(s), \pi(a|s;\boldsymbol{\theta}), \mathcal{P}(s'|s,a)} \Big[\delta_Q(s, a, s'; \boldsymbol{\theta}_{i+1}, \boldsymbol{\omega}_i) \nabla_{\boldsymbol{\omega}} \widehat{Q}(s, a; \boldsymbol{\omega}) |_{\boldsymbol{\omega}=\boldsymbol{\omega}_i} \Big] , \\ \text{where} \quad \delta_Q(s, a, s'; \boldsymbol{\theta}, \boldsymbol{\omega}) &:= \mathcal{R}(s, a) + \gamma \, \mathbb{E}_{\pi(a'|s', \boldsymbol{\theta})} \Big[\widehat{Q}(s', a'; \boldsymbol{\omega}) \Big] - \widehat{Q}(s, a; \boldsymbol{\omega}). \end{aligned} \tag{6}$$

The above update is approximately equivalent to minimizing the mean square of the TD error [Baird, 1995]. The updates (5) and (6) together constitute a type of actor-critic method. The actor's goal is to optimize the expected return shown in Eq. (2), while the critic's goal is to provide an accurate estimate of the value function.

A variety of options are available for the actor and the critic, e.g., stochastic and deterministic actor-critic methods [Sutton et al., 1999, Silver et al., 2014], trust-region policy optimization methods [Schulman et al., 2015], and proximal policy optimization methods [Schulman et al., 2017]. Flexible approximators, such as deep neural networks, can be used for the actor and the critic. Actor-critic methods exhibit lower variance than the policy gradient methods that use Monte Carlo methods to estimate of the Q-function. They are also more sample efficient. Overall, these methods, when tuned well, can perform extremely well and achieved state-of-the-art performances on many difficult RL problems [Mnih et al., 2015, Silver et al., 2016].

However, one issue with actor-critic methods is that they can be unstable, and may require careful tuning and engineering to work well [Lillicrap et al., 2016, Dai et al., 2018, Henderson et al., 2017]. For example, deep deterministic policy gradient (DDPG) [Lillicrap et al., 2016] requires implementation tricks such as target networks, and it is known to be highly sensitive to its hyperparameters [Henderson et al., 2017]. Furthermore, convergence is guaranteed only when the critic accurately estimates the value function [Sutton et al., 1999], which could be prohibitively expensive. In general, stabilizing actor-critic methods is an active area of research.

One source of instability, among many others, is the interaction between the actor and the critic. The algorithm alternates between the update of the actor and the critic, so inaccuracies in one update might affect the other adversely. For example, the actor relies on the value function estimates provided by the critic. This estimate can have lower variance than the Monte Carlo estimates used in Eq. (4). However, Monte Carlo estimates are unbiased, because they maintain the recursive relationship between $\widehat{Q}^{\pi}(s_t, a_t)$ and $\widehat{Q}^{\pi}(s_{t+1}, a_{t+1})$ which ensures that the expected value of $\widehat{Q}^{\pi}(s_t, a_t)$ is equal to the true value function (the expectation is taken with respect to the trajectories). When we use function approximators, it is difficult to satisfy such recursive properties of the value function estimates. Due to this reason, critic estimates are often biased. At times, such inaccuracies might push the actor into wrong directions, from which the actor may never recover. In this paper, we propose a new method to address instability caused by such bad steps.

3 TD-Regularized Actor-Critic

As discussed in the previous section, the critic's estimate of the value function $\widehat{Q}(s,a;\boldsymbol{\omega})$ might be biased, while Monte Carlo estimates can be unbiased. In general, we can ensure the unbiased property of an estimator if it satisfies the Bellman equation. This is because the following recursion ensures that each $\widehat{Q}(s,a;\boldsymbol{\omega})$ is equal to the true value function in expectation, as shown below

$$\widehat{Q}(s,a;\boldsymbol{\omega}) = \mathcal{R}(s,a) + \gamma\, \mathbb{E}_{\mathcal{P}(s'|s,a),\pi(a'|s';\boldsymbol{\theta})}\left[\widehat{Q}(s',a';\boldsymbol{\omega})\right], \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{7}$$

If $\widehat{Q}(s',a';\boldsymbol{\omega})$ is unbiased, $\widehat{Q}(s,a;\boldsymbol{\omega})$ will also be unbiased. Therefore, by induction, all estimates are unbiased. Using this property, we modify actor's learning goal (Eq. (2)) as the following constrained optimization problem

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\left[\widehat{Q}(s,a;\boldsymbol{\omega})\right], \tag{8}$$

$$\text{s.t. } \widehat{Q}(s,a;\boldsymbol{\omega}) = \mathcal{R}(s,a) + \gamma\, \mathbb{E}_{\mathcal{P}(s'|s,a),\pi(a'|s';\boldsymbol{\theta})}\left[\widehat{Q}(s',a';\boldsymbol{\omega})\right], \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{9}$$

We refer to this problem as the *Bellman-constrained policy search*. At the optimum, when $\widehat{Q} = Q^\pi$, the constraint is satisfied, therefore the optimal solution of this problem is equal to the original problem of Eq. (3). For a suboptimal critic, the constraint is not satisfied and constrains the maximization of the expected return proportionally to the deviation in the Bellman equation. We expect this to prevent a large update in the actor when the critic is highly inaccurate for some state-action pairs. The constrained formulation is attractive but computationally difficult due to the large number of constraints, e.g., for continuous state and action space this number might be infinite. In what follows, we make three modifications to this problem to obtain a practical method.

3.1 Modification 1: Unconstrained Formulation Using the Quadratic Penalty Method

Our first modification is to reformulate the constrained problem as an unconstrained one by using the quadratic penalty method [Nocedal and Wright, 2006]. In this method, given an optimization problem with equality constraints

$$\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \quad \text{s.t. } h_j(\boldsymbol{\theta}) = 0, \quad j = 1, 2, \ldots, M \tag{10}$$

we optimize the following function

$$\widehat{f}(\boldsymbol{\theta}, \eta) := f(\boldsymbol{\theta}) - \eta \sum_{j=1}^{M} b_j h_j^2(\boldsymbol{\theta}), \tag{11}$$

where b_j are the weights of the equality constraints and can be used to trade-off the effect of each constraint, and $\eta > 0$ is the parameter controlling the trade-off between the original objective function and the penalty function. When $\eta = 0$, the constraint does not matter, while when $\eta \to \infty$, the objective function does not matter. Assuming that $\boldsymbol{\omega}$ is fixed, we propose to optimize the following quadratic-penalized version of the Bellman-constrained objective for a given η

$$\begin{aligned} L(\boldsymbol{\theta}, \eta) := \ & \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\left[\widehat{Q}(s,a;\boldsymbol{\omega})\right] \\ & - \eta \iint b(s,a) \left(\mathcal{R}(s,a) + \gamma\, \mathbb{E}_{\mathcal{P}(s'|s,a),\pi(a'|s';\boldsymbol{\theta})}\left[\widehat{Q}(s',a';\boldsymbol{\omega})\right] - \widehat{Q}(s,a;\boldsymbol{\omega})\right)^2 \mathrm{d}s\mathrm{d}a, \end{aligned} \tag{12}$$

where $b(s,a)$ is the weight of the constraint corresponding to the pair (s,a).

3.2 Modification 2: Reducing the Number of Constraints

The integration over the entire state-action space is still computationally infeasible. Our second modification is to focus on few constraints by appropriately choosing the weights $b(s,a)$. A natural choice is to use the state distribution $\mu_\pi(s)$ and the current policy $\pi(a|s;\boldsymbol{\theta})$ to sample the candidate state-action pairs whose constraints we will focus on. In this way, we get the following objective

$$\begin{aligned} L(\boldsymbol{\theta}, \eta) := \ & \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\left[\widehat{Q}(s,a;\boldsymbol{\omega})\right] \\ & - \eta\, \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\left[\left(\mathcal{R}(s,a) + \gamma\, \mathbb{E}_{\mathcal{P}(s'|s,a),\pi(a'|s';\boldsymbol{\theta})}\left[\widehat{Q}(s',a';\boldsymbol{\omega})\right] - \widehat{Q}(s,a;\boldsymbol{\omega})\right)^2\right]. \end{aligned} \tag{13}$$

The expectations in the penalty term in Eq. (13) can be approximated using the observed state-action pairs. We can also use the same samples for both the original objective and the penalty term. This can be regarded as a *local* approximation where only a subset of the infinitely many constraint are penalized, and the subset is chosen based on its influence to the original objective function.

3.3 Modification 3: Approximation Using the TD Error

The final difficulty is that the expectation over $\mathcal{P}(s'|s,a)$ is inside the square function. This gives rise to a well-known issue in RL, called the *double-sampling* issue [Baird, 1995]. In order to compute an unbiased estimate of this squared expectation over $\mathcal{P}(s'|s,a)$, we require two sets of independent samples of s' sampled from $\mathcal{P}(s'|s,a)$. The independence condition means that we need to independently sample many of the next states s' from an identical state s. This requires the ability to reset the environment back to state s after each transition, which is typically impossible for many real-world systems. Notice that the squared expectation over $\pi(a'|s';\boldsymbol{\theta})$ is less problematic since we can always internally sample many actions from the policy without actually executing those actions on the environment.

To address this issue, we propose a final modification where we pull the expectation over $\mathcal{P}(s'|s,a)$ outside the square

$$L(\boldsymbol{\theta},\eta) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\widehat{Q}(s,a;\boldsymbol{\omega})\Big] \tag{14}$$
$$- \eta \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}\Big[\underbrace{\big(\mathcal{R}(s,a) + \gamma\, \mathbb{E}_{\pi(a'|s';\boldsymbol{\theta})}\Big[\widehat{Q}(s',a';\boldsymbol{\omega})\Big] - \widehat{Q}(s,a;\boldsymbol{\omega})\big)}_{:=\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})}^2\Big].$$

This step replaces the Bellman constraint of a pair (s,a) by the *temporal difference* (TD) error $\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})$ defined over the tuple (s,a,s'). We can estimate the TD error by using TD(0) or a batch version of it, thereby resolving the double-sampling issue. To further reduce the bias of TD error estimates, we can also rely on TD(λ) (more details in Section 4.5). Note that this final modification only approximately satisfies the Bellman constraint.

3.4 Final Algorithm: the TD-Regularized Actor-Critic Method

Eq. (14) is the final objective we will use to update the actor. For the ease of notation, in the rest of the paper, we will refer to the two terms in $L(\boldsymbol{\theta},\eta)$ using the following notation

$$L(\boldsymbol{\theta},\eta) := J(\boldsymbol{\theta}) - \eta G(\boldsymbol{\theta}), \quad \text{where} \tag{15}$$
$$J(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}[\widehat{Q}(s,a;\boldsymbol{\omega})], \tag{16}$$
$$G(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}[\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})^2]. \tag{17}$$

We propose to replace the usual policy search step (Eq. (5)) in the actor-critic method by a step that optimizes the TD-regularized objective for a given η_i in iteration i,

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_{\boldsymbol{\theta}} \left(\nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{\mu_\pi(s)\pi(a|s;\boldsymbol{\theta})}\Big[\widehat{Q}(s,a;\boldsymbol{\omega}_i)\Big]\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} - \eta_i \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} \right). \tag{18}$$

The blue term is the extra penalty term involving the TD error, where we allow η_i to change with the iteration. We can alternate between the above update of the actor and the update of the critic, e.g., by using Eq. (6) or any other method. We call this method the *TD-regularized* actor-critic. We use the terminology "regularization" instead of "penalization" since it is more common in the RL and machine learning communities.

A simple interpretation of Eq. (14) is that the actor is penalized for increasing the squared TD error, implying that the update of the actor favors policies achieving a small TD error. The main objective of the actor is still to maximize the estimated expected returns, but the penalty term helps to avoid bad updates whenever the critic's estimate has a large TD error. Because the TD error is an approximation to the deviation from the Bellman equation, we expect that the proposed method helps in stabilizing learning whenever the critic's estimate incurs a large TD error.

In practice, the choice of penalty parameter is extremely important to enable a good trade-off between maximizing the expected return and avoiding the bias in the critic's estimate. In a typical optimization problem where the constraints are only functions of $\boldsymbol{\theta}$, it is recommended to slowly increase η_i with i. This

way, as the optimization progresses, the constraints become more and more important. However, in our case, the constraints also depend on $\boldsymbol{\omega}$ which changes with iterations, therefore the constraints also change with i. As long as the overall TD error of the critic decreases as the number of iterations increase, the overall penalty due the constraint will eventually decrease too. Therefore, we do not need to artificially make the constraints more important by increasing η_i. In practice, we found that if the TD error decreases over time, then η_i can, in fact, be decreased with i. In Section 5, we use a simple decaying rule $\eta_{i+1} = \kappa\eta_i$ where $0 < \kappa < 1$ is a decay factor.

4 Application of TD-Regularization to Actor-Critic Methods

Our TD-regularization method is a general plug-and-play method that can be applied to any actor-critic method that performs policy gradient for actor learning. In this section, we first demonstrate its applications to popular actor-critic methods including DPG [Silver et al., 2014], TRPO [Schulman et al., 2015] and PPO [Schulman et al., 2017]. Subsequently, building upon the TD(λ) error, we present a second regularization that can be used by actor-critic methods doing advantage learning, such as GAE [Schulman et al., 2016]. For all algorithms, we show that our method only slightly increases computation time. The required gradients can be easily computed using automatic differentiation, making it very easy to apply our method to existing actor-critic methods. Empirical comparison on these methods are given in Section 5.

4.1 TD-Regularized Stochastic Policy Gradient (SPG)

For a stochastic policy $\pi(a|s;\boldsymbol{\theta})$, the gradient of Eq. (14) can be computed using the chain rule and log-likelihood ratio trick [Williams, 1992, Sutton and Barto, 1998]. Specifically, the gradients of TD-regularized stochastic actor-critic are given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta}) \widehat{Q}(s,a;\boldsymbol{\omega})\Big], \tag{19}$$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}\Big[&\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta}) \delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})^2 \\ &+ 2\gamma \mathbb{E}_{\pi(a'|s';\boldsymbol{\theta})}\Big[\nabla_{\boldsymbol{\theta}} \log \pi(a'|s';\boldsymbol{\theta}) \delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega}) \widehat{Q}(s',a';\boldsymbol{\omega})\Big]\Big].\end{aligned} \tag{20}$$

When compared to the standard SPG method, TD-regularized SPG requires only extra computations to compute $\nabla_{\boldsymbol{\theta}} \log \pi(a'|s';\boldsymbol{\theta})$ and $\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})$.

4.2 TD-Regularized Deterministic Policy Gradient (DPG)

DPG [Silver et al., 2014] is similar to SPG but learns a deterministic policy $a = \pi(s;\boldsymbol{\theta})$. To allow exploration and collect samples, DPG uses a behavior policy $\beta(a|s)$[2]. Common examples are ϵ-greedy policies or Gaussian policies. Consequently, the state distribution $\mu_\pi(s)$ is replaced by $\mu_\beta(s)$ and the expectation over the policy $\pi(a|s;\boldsymbol{\theta})$ in the regularization term is replaced by an expectation over a behavior policy $\beta(a|s)$. The TD error does not change, but the expectation over $\pi(a'|s',\boldsymbol{\theta})$ disappears. TD-regularized DPG components are

$$J_{\text{DPG}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\beta(s)}\Big[\widehat{Q}(s,\pi(s;\boldsymbol{\theta});\boldsymbol{\omega})\Big], \tag{21}$$

$$G_{\text{DPG}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\beta(s),\beta(a|s),\mathcal{P}(s'|s,a)}\big[\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})^2\big], \tag{22}$$

$$\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega}) := \mathcal{R}(s,a) + \gamma\widehat{Q}(s',\pi(s';\boldsymbol{\theta});\boldsymbol{\omega}) - \widehat{Q}(s,a;\boldsymbol{\omega}). \tag{23}$$

Their gradients can be computed by the chain rule and are given by

$$\nabla_{\boldsymbol{\theta}} J_{\text{DPG}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\beta(s)}\Big[\nabla_{\boldsymbol{\theta}} \pi(s;\boldsymbol{\theta}) \nabla_a \widehat{Q}(s,a;\boldsymbol{\omega})\big|_{a=\pi(s;\boldsymbol{\theta})}\Big], \tag{24}$$

$$\nabla_{\boldsymbol{\theta}} G_{\text{DPG}}(\boldsymbol{\theta}) = 2\gamma\, \mathbb{E}_{\mu_\beta(s),\beta(a|s),\mathcal{P}(s'|s,a)}\Big[\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega}) \nabla_{\boldsymbol{\theta}} \pi(s';\boldsymbol{\theta}) \nabla_{a'} \widehat{Q}(s',a';\boldsymbol{\omega})\big|_{a'=\pi(s';\boldsymbol{\theta})}\Big]. \tag{25}$$

The gradient of the regularization term requires extra computations to compute $\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega})$, $\nabla_{\boldsymbol{\theta}} \pi(s';\boldsymbol{\theta})$, and $\nabla_{a'} \widehat{Q}(s',a';\boldsymbol{\omega})$.

[2]Silver et al. [2014] showed that DPG can be more advantageous than SPG as deterministic policies have lower variance. However, the behavior policy has to be chosen appropriately.

4.3 TD-Regularized Trust-Region Policy Optimization (TRPO)

In the previous two examples, the critic estimates the Q-function. In this section, we demonstrate an application to a case where the critic estimates the V-function. The V-function of π is defined as $V^\pi(s) := \mathbb{E}_{\pi(a|s)}[Q^\pi(s,a)]$, and satisfies the following Bellman equation

$$V^\pi(s) = \mathbb{E}_{\pi(a|s)}[\mathcal{R}(s,a)] + \gamma\, \mathbb{E}_{\pi(a|s),\mathcal{P}(s'|s,a)}[V(s')]\,, \qquad \forall s \in \mathcal{S}. \tag{26}$$

The TD error for a critic $\widehat{V}(s;\boldsymbol{\omega})$ with parameters $\boldsymbol{\omega}$ is

$$\delta_V(s,s';\boldsymbol{\omega}) := \mathcal{R}(s,a) + \gamma \widehat{V}(s';\boldsymbol{\omega}) - \widehat{V}(s;\boldsymbol{\omega}). \tag{27}$$

One difference compared to previous two sections is that $\delta_V(s,s';\boldsymbol{\omega})$ does not directly contain $\pi(a|s;\boldsymbol{\theta})$. We will see that this greatly simplifies the update. Nonetheless, the TD error still depends on $\pi(a|s;\boldsymbol{\theta})$ as it requires to sample the action a to reach the next state s'. Therefore, the TD-regularization can still be applied to stabilize actor-critic methods that use a V-function critic.

In this section, we regularize TRPO [Schulman et al., 2015], which uses a V-function critic and solves the following optimization problem

$$\max_{\boldsymbol{\theta}}\ L_{\text{TRPO}}(\boldsymbol{\theta},\eta) := J_{\text{TRPO}}(\boldsymbol{\theta}) - \eta G_{\text{TRPO}}(\boldsymbol{\theta}), \qquad \text{s.t. } \mathbb{E}_{\mu_\pi(s)}[\mathrm{KL}(\pi||\pi_{\text{old}})] \le \epsilon, \tag{28}$$

$$\text{where} \quad J_{\text{TRPO}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\rho(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega})\Big]\,, \tag{29}$$

$$G_{\text{TRPO}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}\left[\rho(\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2\right], \tag{30}$$

where $\rho(\boldsymbol{\theta}) = \pi(a|s;\boldsymbol{\theta})/\pi(a|s;\boldsymbol{\theta}_{\text{old}})$ are importance weights. KL is the Kullback-Leibler divergence between the new learned policy $\pi(a|s;\boldsymbol{\theta})$ and the old one $\pi(a|s;\boldsymbol{\theta}_{\text{old}})$, and helps in ensuring small policy updates. $\widehat{A}(s,a;\boldsymbol{\omega})$ is an estimate of the advantage function $A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s)$ computed by learning a V-function critic $\widehat{V}(s;\boldsymbol{\omega})$ and approximating $Q^\pi(s,a)$ either by Monte Carlo estimates or from $\widehat{V}(s;\boldsymbol{\omega})$ as well. We will come back to this in Section 4.5. The gradients of $J_{\text{TRPO}}(\boldsymbol{\theta})$ and $G_{\text{TRPO}}(\boldsymbol{\theta})$ are

$$\nabla_{\boldsymbol{\theta}} J_{\text{TRPO}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\rho(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega})\Big]\,, \tag{31}$$

$$\nabla_{\boldsymbol{\theta}} G_{\text{TRPO}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}\left[\rho(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2\right]. \tag{32}$$

The extra computation for TD-regularized TRPO only comes from computing the square of the TD error $\delta_V(s,s';\boldsymbol{\omega})^2$.

Notice that, due to linearity of expectations, TD-regularized TRPO can be understood as performing the standard TRPO with a TD-regularized advantage $\widehat{A}_\eta(s,a;\boldsymbol{\omega}) := \widehat{A}(s,a;\boldsymbol{\omega}) - \eta\mathbb{E}_{\mathcal{P}(s'|s,a)}[\delta_V(s,s';\boldsymbol{\omega})^2]$. This greatly simplifies implementation of our TD-regularization method. In particular, TRPO performs natural gradient ascent to approximately solve the KL constraint optimization problem[3]. By viewing TD-regularized TRPO as TRPO with regularized advantage, we can use the same natural gradient procedure for TD-regularized TRPO.

4.4 TD-Regularized Proximal Policy Optimization (PPO)

PPO [Schulman et al., 2017] simplifies the optimization problem of TRPO by removing the KL constraint, and instead uses clipped importance weights and a pessimistic bound on the advantage function

$$\max_{\boldsymbol{\theta}}\ \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\min\{\rho(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega}),\ \rho_\varepsilon(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega})\}\Big]\,, \tag{33}$$

where the $\rho_\varepsilon(\boldsymbol{\theta})$ is the importance ratio $\rho(\boldsymbol{\theta})$ clipped between $[1-\varepsilon, 1+\varepsilon]$ and $0 < \varepsilon < 1$ represents the update stepsize (the smaller ε, the more conservative the update is). By clipping the importance ratio, we remove the incentive for moving $\rho(\boldsymbol{\theta})$ outside of the interval $[1-\varepsilon, 1+\varepsilon]$, i.e., for moving the new policy far from the old one. By taking the minimum between the clipped and the unclipped advantage, the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective.

Similarly to TRPO, the advantage function $\widehat{A}(s,a;\boldsymbol{\omega})$ is computed using a V-function critic $\widehat{V}(s;\boldsymbol{\omega})$, thus we could simply use the regularization in Eq. (30). However, the TD-regularization would not benefit

[3]Natural gradient ascent on a function $f(\boldsymbol{\theta})$ updates the function parameters $\boldsymbol{\theta}$ by $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \boldsymbol{F}^{-1}(\boldsymbol{\theta})\boldsymbol{g}(\boldsymbol{\theta})$, where $g(\boldsymbol{\theta})$ is the gradient and $\boldsymbol{F}(\boldsymbol{\theta})$ is the Fisher information matrix.

from neither importance clipping nor the pessimistic bound, which together provide a way of performing small safe policy updates. For this reason, we propose to modify the TD-regularization as follows

$$\max_{\boldsymbol{\theta}} L_{\text{PPO}}(\boldsymbol{\theta}, \eta) := J_{\text{PPO}}(\boldsymbol{\theta}) - \eta G_{\text{PPO}}(\boldsymbol{\theta}), \qquad \text{where} \tag{34}$$

$$J_{\text{PPO}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}\Big[\min\{\rho(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega}),\ \rho_\varepsilon(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega})\}\Big], \tag{35}$$

$$G_{\text{PPO}}(\boldsymbol{\theta}) := \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta}),\mathcal{P}(s'|s,a)}\big[\max\{\rho(\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2,\ \rho_\varepsilon(\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2\}\big], \tag{36}$$

i.e., we apply importance clipping and the pessimistic bound also to the TD-regularization. The gradients of $J_{\text{PPO}}(\boldsymbol{\theta})$ and $G_{\text{PPO}}(\boldsymbol{\theta})$ can be computed as

$$\nabla_{\boldsymbol{\theta}} J_{\text{PPO}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}[f(\boldsymbol{\theta})] \qquad \text{where} \tag{37}$$

$$f(\boldsymbol{\theta}) := \begin{cases} \rho(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega}) & \text{if } \rho(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega}) < \rho_\varepsilon(\boldsymbol{\theta})\widehat{A}(s,a;\boldsymbol{\omega}) \\ 0 & \text{otherwise,} \end{cases}$$

$$\nabla_{\boldsymbol{\theta}} G_{\text{PPO}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s),\pi(a|s;\boldsymbol{\theta})}[g(\boldsymbol{\theta})] \qquad \text{where} \tag{38}$$

$$g(\boldsymbol{\theta}) := \begin{cases} \rho(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \pi(a|s;\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2 & \text{if } \rho(\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2 > \rho_\varepsilon(\boldsymbol{\theta})\delta_V(s,s';\boldsymbol{\omega})^2 \\ 0 & \text{otherwise.} \end{cases}$$

4.5 GAE-Regularization

In Sections 4.3 and 4.4, we have discussed how to apply the TD-regularization when a V-function critic is learned. The algorithms discussed, TRPO and PPO, maximize the advantage function $\widehat{A}(s,a;\boldsymbol{\omega})$ estimated using a V-function critic $\widehat{V}(s,\boldsymbol{\omega})$. Advantage learning has a long history in RL literature [Baird, 1993] and one of the most used and successful advantage estimator is the generalized advantage estimator (GAE) [Schulman et al., 2016]. In this section, we build a connection between GAE and the well-known TD(λ) method [Sutton and Barto, 1998] to propose a different regularization, which we call the *GAE-regularization*. We show that this regularization is very convenient for algorithms already using GAE, as it does not introduce any computational cost, and has interesting connections with other RL methods.

Let the n-step return $R_{t:t+n}$ be the sum of the first n discounted rewards plus the estimated value of the state reached in n steps, i.e.,

$$R_{t:t+n} := r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n \widehat{V}(s_{t+n};\boldsymbol{\omega}), \quad 0 \le t \le T-n, \tag{39}$$

$$R_{t:T+1} := \sum_{i=t}^{T} \gamma^{i-t} r_i. \tag{40}$$

The full-episode return $R_{t:T+1}$ is a Monte Carlo estimate of the value function. The idea behind TD(λ) is to replace the TD error target $r_t + \gamma\widehat{V}(s_{t+1},\boldsymbol{\omega})$ with the average of the n-step returns, each weighted by λ^{n-1}, where $\lambda \in [0,1]$ is a decay rate. Each n-step return is also normalized by $1-\lambda$ to ensure that the weights sum to 1. The resulting TD(λ) targets are the so-called λ-returns

$$R_t^\lambda := (1-\lambda)\sum_{i=t}^{T-1} \lambda^{i-t} R_{t:i+1} + \lambda^{T-t} R_{t:T+1}, \tag{41}$$

and the corresponding TD(λ) error is

$$\delta_V^\lambda(s_t, s_{t+1};\boldsymbol{\omega}) := R_t^\lambda - \widehat{V}(s_t;\boldsymbol{\omega}). \tag{42}$$

From the above equations, we see that if $\lambda = 0$, then the λ-return is the TD target $R_t^0 = r_t + \gamma\widehat{V}(s_{t+1};\boldsymbol{\omega})$. If $\lambda = 1$, then $R_t^1 = R_{t:T+1}$ as in Monte Carlo methods. In between are intermediate methods that control the bias-variance trade-off between TD and Monte Carlo estimators by varying λ. As discussed in Section 2, in fact, TD estimators are biased, while Monte Carlo are not. The latter, however, have higher variance.

Motivated by the same bias-variance trade-off, we propose to replace δ_V with δ_V^λ in Eq. (30) and (36), i.e., to perform TD(λ)-regularization. Interestingly, this regularization is equivalent to regularize with the GAE advantage estimator, as shown in the following. Let δ_V be an approximation of the advantage function

[Schulman et al., 2016]. Similarly to the λ-return, we can define the n-step advantage estimator

$$\begin{aligned} A_{t:t+n} &:= \delta_V(s_t, s_{t+1}; \boldsymbol{\omega}) + \gamma\delta_V(s_{t+1}, s_{t+2}; \boldsymbol{\omega}) + \ldots + \gamma^{n-1}\delta_V(s_{t+n-1}, s_{t+n}; \boldsymbol{\omega}), \\ &= R_{t:t+n} - \widehat{V}(s_t; \boldsymbol{\omega}), \qquad 0 \leq t \leq T-n, \end{aligned} \tag{43}$$

$$A_{t:T+1} := R_{t:T+1} - \widehat{V}(s_t; \boldsymbol{\omega}). \tag{44}$$

Following the same approach of TD(λ), GAE advantage estimator uses exponentially weighted averages of n-step advantage estimators

$$\widehat{A}^{\lambda}(s_t, a_t; \boldsymbol{\omega}) := (1-\lambda)\sum_{i=t}^{T-1} \lambda^{i-t} A_{t:i+1} + \lambda^{T-t} A_{t:T+1}, \tag{45}$$

From the above equation, we see that GAE estimators are discounted sums of TD errors. Similarly to TD(λ), if $\lambda = 0$ then the advantage function estimate is just the TD error estimate, i.e., $\widehat{A}^0(s_t, a_t; \boldsymbol{\omega}) = r_t + \widehat{V}(s_{t+1}; \boldsymbol{\omega}) - \widehat{V}(s_t; \boldsymbol{\omega})$. If $\lambda = 1$ then the advantage function estimate is the difference between the Monte Carlo estimate of the return and the V-function estimate, i.e., $\widehat{A}^1(s_t, a_t; \boldsymbol{\omega}) = R_{t:T+1} - \widehat{V}(s_t; \boldsymbol{\omega})$. Finally, plugging Eq. (43) into Eq. (45), we can rewrite the GAE estimator as

$$\begin{aligned} \widehat{A}^{\lambda}(s_t, a_t; \boldsymbol{\omega}) &:= (1-\lambda)\sum_{i=t}^{T-1} \lambda^{i-t}(R_{t:i+1} - \widehat{V}(s_t; \boldsymbol{\omega})) + \lambda^{T-t}(R_{t:T+1} - \widehat{V}(s_t; \boldsymbol{\omega})) \\ &= (1-\lambda)\sum_{i=t}^{T-1} \lambda^{i-t} R_{t:i+1} + \lambda^{T-t} R_{t:T+1} - \widehat{V}(s_t; \boldsymbol{\omega}) \\ &= R_t^{\lambda} - \widehat{V}(s_t; \boldsymbol{\omega}) = \delta_V^{\lambda}(s_t, s_{t+1}; \boldsymbol{\omega}), \end{aligned} \tag{46}$$

i.e., the GAE advantage estimator is equivalent to the TD(λ) error estimator. Therefore, using the TD(λ) error to regularize actor-critic methods is equivalent to regularize with the GAE estimator, yielding the following quadratic penalty

$$G_{\text{GAE}}(\boldsymbol{\theta}) = \mathbb{E}_{\mu_\pi(s), \pi(a|s;\boldsymbol{\theta})}\left[\widehat{A}^{\lambda}(s, a; \boldsymbol{\omega})^2\right], \tag{47}$$

which we call the *GAE-regularization*. The GAE-regularization is very convenient for methods which already use GAE, such as TRPO and PPO[4], as it does not introduce any computational cost. Furthermore, the decay rate λ allows to tune the bias-variance trade-off between TD and Monte Carlo methods[5]. In Section 5 we present an empirical comparison between the TD- and GAE-regularization.

Finally, the GAE-regularization has some interesting interpretations. As shown by Belousov and Peters [2017], minimizing the squared advantage function is equivalent to maximizing the average reward with a penalty over the Pearson divergence between the new and old state-action distribution $\mu_\pi(s)\pi(a|s;\boldsymbol{\theta})$, and a hard constraint to satisfy the stationarity condition $\iint \mu_\pi(s)\pi(a|s;\boldsymbol{\theta})\mathcal{P}(s'|s,a)\,\mathrm{d}s\,\mathrm{d}a = \mu_\pi(s'), \forall s'$. The former is to avoid overconfident policy update steps, while the latter is the dual of the Bellman equation (Eq. (7)). Recalling that the GAE-regularization approximates the Bellman equation constraint with the TD(λ) error, the two methods are very similar. The difference in the policy update is that the GAE-regularization replaces the stationarity condition with a soft constraint, i.e., the penalty[6].
Interestingly, Eq. (47) is also equivalent to minimizing the variance of the centered GAE estimator, i.e., $\mathbb{E}[(\widehat{A}^{\lambda}(s, a; \boldsymbol{\omega}) - \mu_{\widehat{A}})^2] = \mathrm{Var}[\widehat{A}^{\lambda}(s, a; \boldsymbol{\omega})]$. Maximizing the mean of the value function estimator and penalizing its variance is a common approach in risk-averse RL called *mean-variance* optimization [Tamar et al., 2012]. Similarly to our method, this can be interpreted as a way to avoid overconfident policy updates when the variance of the critic is high. By definition, in fact, the expectation of the true advantage function of any policy is zero[7], thus high-variance is a sign of an inaccurate critic.

[4] For PPO, we also apply the importance clipping and pessimistic bound proposed in Eq. (36).

[5] We recall that, since GAE approximates $A^\pi(s,a)$ with the TD(λ) error, we are performing the same approximation presented in Section 3.3, i.e., we are still approximately satisfying the Bellman constraint.

[6] For the critic update, instead, Belousov and Peters [2017] learn the V-function parameters together with the policy rather than separately as in actor-critic methods.

[7] $\mathbb{E}_{\pi(a|s)}[A^\pi(s,a)] = \mathbb{E}_{\pi(a|s)}[Q^\pi(s,a) - V^\pi(s)] = \mathbb{E}_{\pi(a|s)}[Q^\pi(s,a)] - V^\pi(s) = V^\pi(s) - V^\pi(s) = 0.$

5 Evaluation

We propose three evaluations. First, we study the benefits of the TD-regularization in the 2-dimensional linear-quadratic regulator (LQR). In this domain we can compute the true Q-function, expected return, and TD error in closed form, and we can visualize the policy parameter space. We begin this evaluation by setting the initial penalty coefficient to $\eta_0 = 0.1$ and then decaying it according to $\eta_{i+1} = \kappa\eta_i$ where $\kappa = 0.999$. We then investigate different decaying factors κ and the behavior of our approach in the presence of non-uniform noise. The algorithms tested are DPG and SPG. For DPG, we also compare to the twin delayed version proposed by Fujimoto et al. [2018], which achieved state-of-the-art results.

The second evaluation is performed on the single- and double-pendulum swing-up tasks [Yoshikawa, 1990, Brockman et al., 2016]. Here, we apply the proposed TD- and GAE-regularization to TRPO together with and against Retrace [Munos et al., 2016] and double-critic learning [van Hasselt, 2010], both state-of-the-art techniques to stabilize the learning of the critic.

The third evaluation is performed on OpenAI Gym [Brockman et al., 2016] continuous control benchmark tasks with the MuJoCo physics simulator [Todorov et al., 2012] and compares TRPO and PPO with their TD- and GAE-regularized counterparts. Due to time constraints, we were not able to evaluate Retrace on this tasks as well. Details of the hyperparameter settings are given in Appendix C.

For the LQR and the pendulum swing-up tasks, we tested each algorithm over 50 trials with fixed random seeds. At each iteration, we turned the exploration off and evaluated the policy over several episodes. Due to limited computational resources, we tested MuJoCo experiments over five trials with fixed random seeds. For TRPO, the policy was evaluated over 20 episodes without exploration noise. For PPO, we used the same samples collected during learning, i.e., including exploration noise.

5.1 2D Linear Quadratic Regulator (LQR)

The LQR problem is defined by the following discrete-time dynamics

$$s' = As + Ba + \mathcal{N}(0, 0.1^2), \qquad a = Ks, \qquad \mathcal{R}(s,a) = -s^\mathsf{T} X s - a^\mathsf{T} Y a,$$

where $A, B, X, Y \in \mathbb{R}^{d\times d}$, X is a symmetric positive semidefinite matrix, Y is a symmetric positive definite matrix, and $K \in \mathbb{R}^{d\times d}$ is the control matrix. The policy parameters we want to learn are $\boldsymbol{\theta} = \mathrm{vec}(K)$. Although low-dimensional, this problem presents some challenges. First, the policy can easily make the system unstable. The LQR, in fact, is stable only if the matrix $(A + BK)$ has eigenvalues of magnitude smaller than one. Therefore, small stable steps have to be applied when updating the policy parameters, in order to prevent divergence. Second, the reward is unbounded and the expected negative return can be extremely large, especially at the beginning with an initial random policy. As a consequence, with a common zero-initialization of the Q-function, the initial TD error can be arbitrarily large. Third, states and actions are unbounded and cannot be normalized in [0,1], a common practice in RL.

Furthermore, the LQR is particularly interesting because we can compute in closed form both the expected return and the Q-function, being able to easily assess the quality of the evaluated algorithms. More specifically, the Q-function is quadratic in the state and in the action, i.e.,

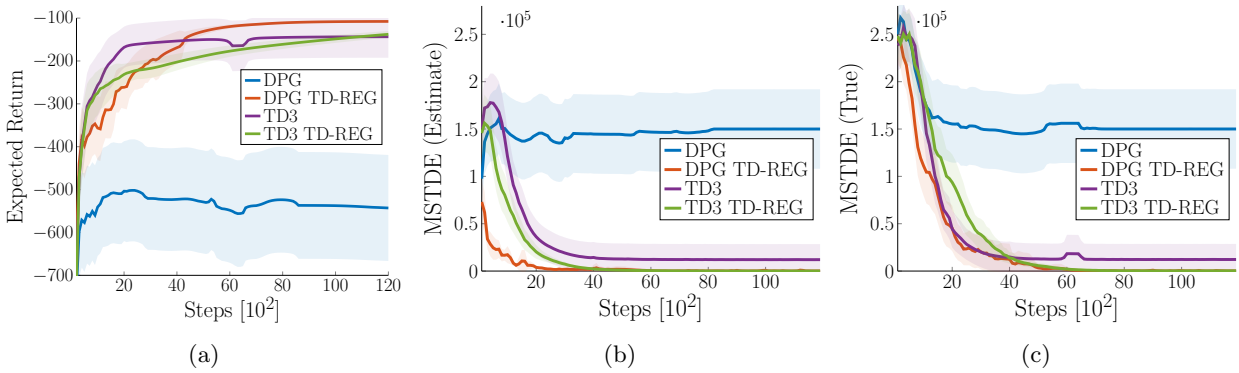
$$Q^\pi(s,a) = Q_0 + s^\mathsf{T} Q_{ss} s + a^\mathsf{T} Q_{aa} a + s^\mathsf{T} Q_{sa} a,$$

where $Q_0, Q_{ss}, Q_{aa}, Q_{sa}$ are matrices computed in closed form given the MDP characteristics and the control matrix K. To show that actor-critic algorithms are prone to instability in the presence of function approximation error, we approximate the Q-function linearly in the parameters $\widehat{Q}(s,a;\boldsymbol{\omega}) = \phi(s,a)^\mathsf{T}\boldsymbol{\omega}$, where $\phi(s,a)$ includes linear, quadratic and cubic features.

Along with the expected return, we show the trend of two mean squared TD errors (MSTDE): one is estimated using the currently learned $\widehat{Q}(s,a;\boldsymbol{\omega})$, the other is computed in closed form using the true $Q^\pi(s,a)$ defined above. It should be noticed that $Q^\pi(s,a)$ is not the optimal Q-function (i.e., of the optimal policy), but the true Q-function with respect to the current policy. For details of the hyperparameters and an in-depth analysis, including an evaluation of different Q-function approximators, we refer to Appendix A.

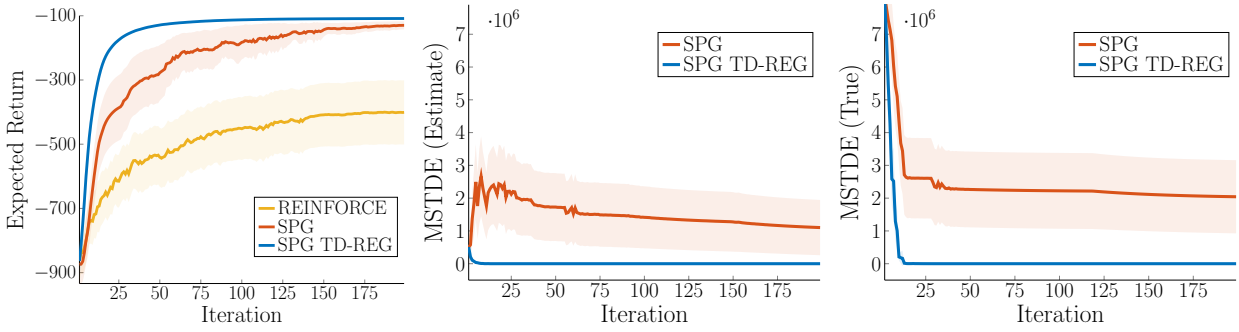
5.1.1 Evaluation of DPG and SPG

DPG and TD-regularized DPG (DPG TD-REG) follow the equations presented in Section 4.2. The difference is that DPG maximizes only Eq. (21), while DPG TD-REG objective includes Eq. (22). TD3 is the twin delayed version of DPG presented by Fujimoto et al. [2018], which uses two critics and delays policy



(a) (b) (c)

Figure 2: DPG comparison on the LQR. Shaded areas denote 95% confidence interval. DPG diverged 24 times out of 50, thus explaining its very large confidence interval. TD3 diverged twice, while TD-regularized algorithms never diverged. Only DPG TD-REG, though, always learned the optimal policy within the time limit.



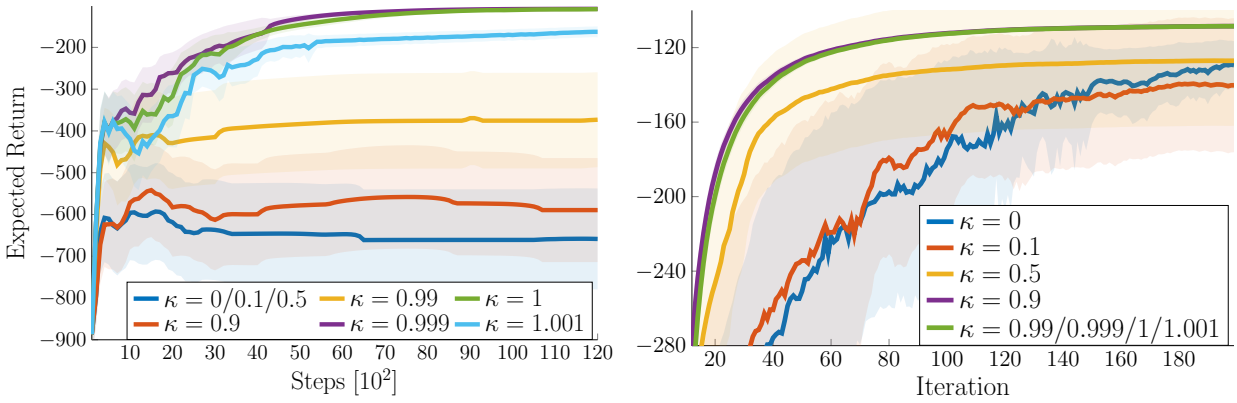
(a) (b) (c)

Figure 3: SPG comparison on the LQR. One iteration corresponds to 150 steps. REINFORCE does not appear in the TD error plots as it does not learn any critic. SPG TD-REG shows an incredibly fast convergence in all runs. SPG-TD, instead, needs much more iterations to learn the optimal policy, as its critic has a much larger TD error. REINFORCE diverged 13 times out of 50, thus explaining its large confidence interval.

updates. TD3 TD-REG is its TD-regularized counterpart. For all algorithms, all gradients are optimized by ADAM [Kingma and Ba, 2014]. After 150 steps, the state is reset and a new trajectory begins.

As expected, because the Q-function is approximated with also cubic features, the critic is prone to overfit and the initial TD error is very large. Furthermore, the true TD error (Figure 2c) is more than twice the one estimated by the critic (Figure 2b), meaning that the critic underestimates the true TD error. Because of the incorrect estimation of the Q-function, vanilla DPG diverges 24 times out of 50. TD3 performs substantially better, but still diverges two times. By contrast, TD-REG algorithms never diverges. Interestingly, only DPG TD-REG always converges to the true critic and to the optimal policy within the time limit, while TD3 TD-REG improves more slowly. Figure 2 hints that this "slow learning" behavior may be due to the delayed policy update, as both the estimated and the true TD error are already close to zero by mid-learning. In Appendix A we further investigate this behavior and show that TD3 policy update delay is unnecessary if TD-REG is used. The benefits of TD-REG in the policy space can also be seen in Figure 1. Whereas vanilla DPG falls victim to the wrong critic estimates and diverges, DPG TD-REG enables more stable updates.

The strength of the proposed TD-regularization is also confirmed by its application to SPG, as seen in Figure 3. Along with SPG and SPG TD-REG, we evaluated REINFORCE [Williams, 1992], which does not learn any critic and just maximizes Monte Carlo estimates of the Q-function, i.e., $\widehat{Q}^\pi(s_t, a_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i$. For all three algorithms, at each iteration samples from only one trajectory of 150 steps are collected and used to compute the gradients, which are then normalized. For the sake of completeness, we also tried to collect more samples per iteration, increasing the number of trajectories from one to five. In this case,



(a) DPG (b) SPG

Figure 4: Comparison of different values of κ. Shaded areas denote 95% confidence interval. In order to provide enough regularization, κ must be sufficiently large during the whole learning. With small values, in fact, η vanishes and the TD-regularization is not in effect anymore.

all algorithms performed better, but still neither SPG nor REINFORCE matched SPG TD-REG, as they both needed several samples more than SPG TD-REG. More details in Appendix A.2.

5.1.2 Analysis of the TD-Regularization Coefficient η

In Section 3.4 we have discussed that Eq. (14) is the result of solving a constrained optimization problem by penalty function methods. In optimization, we can distinguish two approaches to apply penalty functions [Boyd and Vandenberghe, 2004]. *Exterior* penalty methods start at optimal but infeasible points and iterate to feasibility as $\eta \to \infty$. By contrast, *interior* penalty methods start at feasible but sub-optimal points and iterate to optimality as $\eta \to 0$. In actor-critic, we usually start at infeasible points, as the critic is not learned and the TD error is very large. However, unlike classical constrained optimization, the constraint changes at each iteration, because the critic is updated to minimize the same penalty function. This trend emerged from the results presented in Figures 2 and 3, showing the change of the mean squared TD error, i.e., the penalty.

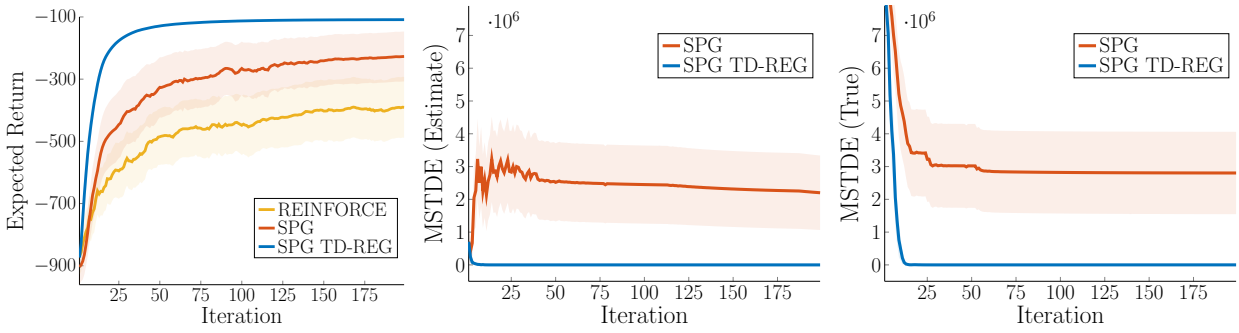
In the previous experiments we started with a penalty coefficient $\eta_0 = 0.1$ and decreased it at each policy update according to $\eta_{t+1} = \kappa\eta_t$, with $\kappa = 0.999$. In this section, we provide a comparison of different values of κ, both as decay and growth factor. In all experiments we start again with $\eta_0 = 0.1$ and we test the following κ: 0, 0.1, 0.5, 0.9, 0.99, 0.999, 1, 1.001.

As shown in Figures 4a and 4b, results are different for DPG TD-REG and SPG TD-REG. In DPG TD-REG, 0.999 and 1 allowed to always converge to the optimal policy. Smaller κ did not provide sufficient help, up to the point where 0.1 and 0.5 did not provide any help at all. However, it is not true that larger κ yield better results, as with 1.001 performance decreases. This is expected, since by increasing η we are also increasing the magnitude of the gradient, which then leads to excessively large and unstable updates.

Results are, however, different for SPG TD-REG. First, 0.99, 0.999, 1, 1.001 all achieve the same performance. The reason is that gradients are normalized, thus the size of the update step cannot be excessively large and $\kappa > 1$ does not harm the learning. Second, 0.9, which was not able to help enough DPG, yields the best results with a slightly faster convergence. The reason is that DPG performs a policy update at each step of a trajectory, while SPG only at the end. Thus, in DPG η, which is updated after a policy update, will decay too quickly if a small κ is used.

5.1.3 Analysis of Non-Uniform Observation Noise

So far, we have considered the case of high TD error due to an overfitting critic and noisy transition function. However, the critic can be inaccurate also because of noisy or partially observable state. Learning in the presence of noise is a long-studied problem in RL literature. To address every aspect of it and to provide a complete analysis of different noises is out of the scope of this paper. However, given the nature of our approach, it is of particular interest to analyze the effects of the TD-regularization in the presence of non-uniformly distributed noise in the state space. In fact, since the TD-regularization penalizes for



(a) (b) (c)

Figure 5: SPG comparison on the LQR with non-uniform noise on the state observation. Shaded areas denote 95% confidence interval. The TD error is not shown for REINFORCE as it does not learn any critic. Once again, SPG TD-REG performs the best and is not affected by the noise. Instead of being drawn to low-noise regions (which are far from the goal and correspond to low-reward regions), its actor successfully learns the optimal policy in all trials and its critic achieves a TD error of zero. Un-regularized SPG, which did not diverge in Figure 3a, here diverges six times.

high TD error, the algorithm could be drawn towards low-noise regions of the space in order to avoid high prediction errors. Intuitively, this may not be always a desirable behavior. Therefore, in this section we evaluate SPG TD-REG when non-uniform noise is added to the observation of the state, i.e.,

$$s_{\text{OBS}} = s_{\text{TRUE}} + \frac{\mathcal{N}(0, 0.05)}{\texttt{clip}(s_{\text{TRUE}}, 0.1, 200)},$$

where s_{OBS} is the state observed by the actor and the critic, and s_{TRUE} is the true state. The clipping between $[0.1, 200]$ is for numerical stability. The noise is Gaussian and inversely proportional to the state. Since the goal of the LQR is to reach $s_{\text{TRUE}} = 0$, near the goal the noise will be larger and, subsequently, the TD error as well. One may therefore expect that SPG TD-REG would lead the actor towards low-noise regions, i.e., away from the goal. However, as shown in Figure 5, SPG TD-REG is the only algorithm learning in all trials and whose TD error goes to zero. By contrast, SPG, which never diverged with exact observations (Figure 3a), here diverged six times out of 50 (Figure 5a). SPG TD-REG plots, instead, are the same in both Figures. REINFORCE, instead, does not significantly suffer from the noisy observations, since it does not learn any critic.

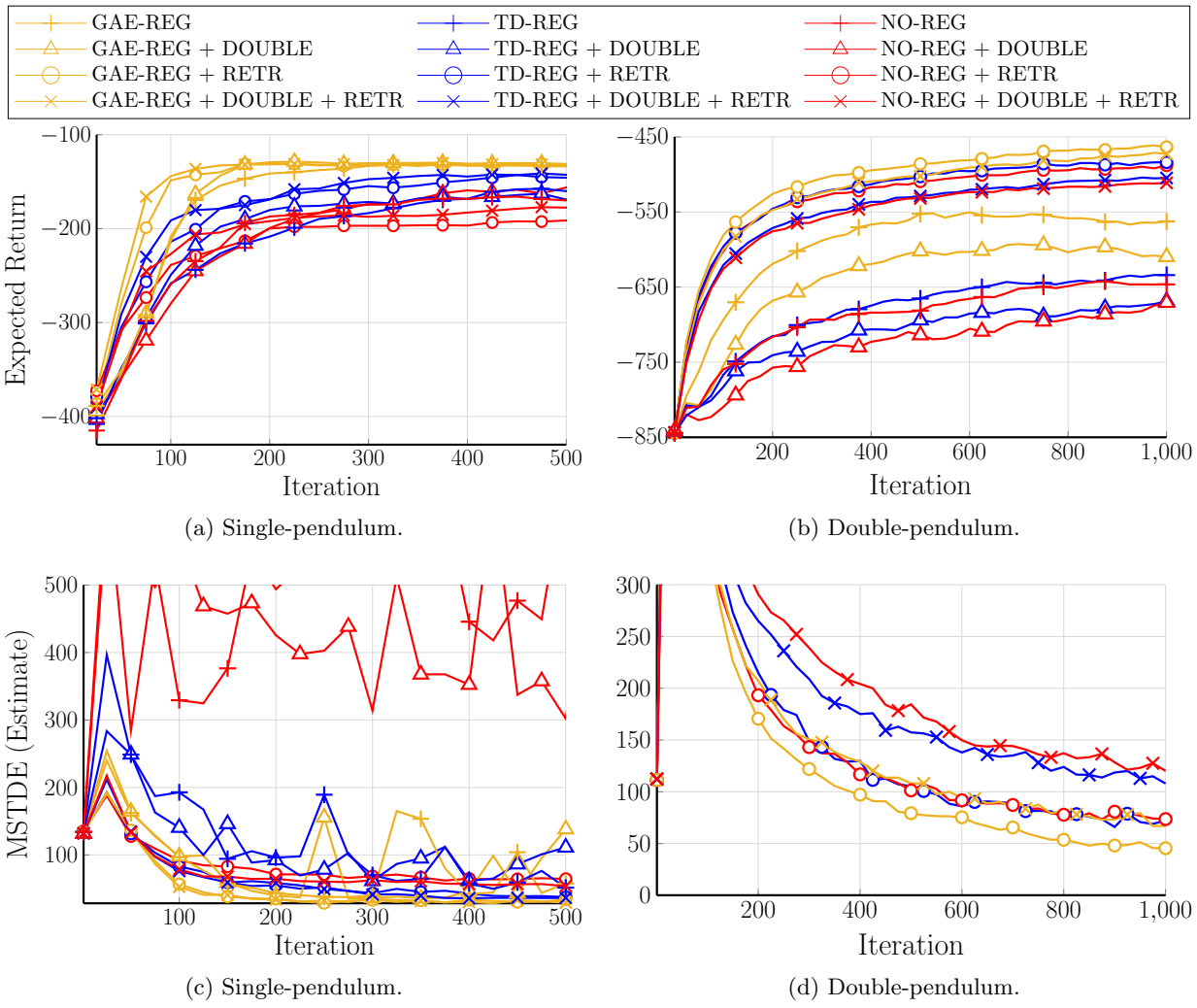
5.2 Pendulum Swing-up Tasks

The pendulum swing-up tasks are common benchmarks in RL. Their goal is to swing-up and stabilize a single- and double-link pendulum from any starting position. The agent observes the current joint position and velocity and acts applying torque on each joint. As the pendulum is underactuated, the agent cannot swing it up in a single step, but needs to gather momentum by making oscillatory movements. Compared to the LQR, these tasks are more challenging –especially the double-pendulum– as both the transition and the value functions are nonlinear.

In this section, we apply the proposed TD- and GAE-regularization to TRPO and compare to Retrace [Munos et al., 2016] and to double-critic learning [van Hasselt, 2010], both state-of-the-art techniques to stabilize the learning of the critic. Similarly to GAE, Retrace replaces the advantage function estimator with the average of n-step advantage estimators, but it additionally employs importance sampling to use off-policy data

$$\widehat{A}^{\lambda}(s_t, a_t; \boldsymbol{\omega}) := (1-\lambda) \sum_{i=t}^{T-1} \lambda^{i-t} \left(\prod_{j=t}^{T-1} w_j \right) A_{t:i+1} + \lambda^{T-t} w_T A_{t:T+1}, \tag{48}$$

where the importance sampling ratio $w_j = \min(1, \pi(a_j|s_j;\boldsymbol{\theta})/\beta(a_j|s_j))$ is truncated at 1 to prevent the "variance explosion" of the product of importance sampling ratios, and $\beta(a|s)$ is the behavior policy used to collect off-policy data. For example, we can reuse past data collected at the i-th iteration by having $\beta(a|s) = \pi(a|s;\boldsymbol{\theta}_i)$.



(a) Single-pendulum.

(b) Double-pendulum.

(c) Single-pendulum.

(d) Double-pendulum.

Figure 6: TRPO results on the pendulum swing-up tasks. In both tasks, GAE-REG + RETR yields the best results. In the single-pendulum, the biggest help is given by GAE-REG, which is the only version always converging to the optimal policy. In the double-pendulum, Retrace is the most important component, as without it all algorithms performed poorly (their TD error is not shown as too large). In all cases, NO-REG always performed worse than TD-REG and GAE-REG (red plots are always below blue and yellow plots with the same markers).

Double-critic learning, instead, employs two critics to reduce the overestimation bias, as we have seen with TD3 in the LQR task. However, TD3 builds upon DPG and modifies the target policy in the Q-function TD error target, which does not appear in the V-function TD error (compare Eq. (6) to Eq. (27)). Therefore, we decided to use the double-critic method proposed by van Hasselt [2010]. In this case, at each iteration only one critic is randomly updated and used to train the policy. For each critic update, the TD targets are computed using estimates from the other critic, in order to reduce the overestimation bias.

In total, we present the results of 12 algorithms, as we tested all combinations of vanilla TRPO (NO-REG), TD-regularization (TD-REG), GAE-regularization (GAE-REG), Retrace (RETR) and double-critic learning (DOUBLE). All results presented below are averaged over 50 trials. However, for the sake of clarity, in the plots we show only the mean of the expected return. Both the actor and the critic are linear function with random Fourier features, as presented in [Rajeswaran et al., 2017]. For the single-pendulum we used 100 features, while for the double-pendulum we used 300 features. In both tasks, we tried to collect as few samples as possible, i.e., 500 for the single-pendulum and 3,000 for the double-pendulum. All algorithms additionally reuse the samples collected in the past four iterations, effectively learning with 2,500 and 15,000 samples, respectively, at each iteration. The advantage is estimated with importance sampling as in Eq. (48), but only Retrace uses truncated importance ratios. For the single-pendulum, the starting

regularization coefficient is $\eta_0 = 1$. For the double-pendulum, $\eta_0 = 1$ for GAE-REG and $\eta_0 = 0.1$ for TD-REG, as the TD error was larger in the latter task (see Figure 6d). In both tasks, it then decays according to $\eta_{t+1} = \kappa\eta_t$ with $\kappa = 0.999$. Finally, both the advantage and the TD error estimates are standardized for the policy update. For more details about the tasks and the hyperparameters, we refer to Appendix B.

Figure 6 shows the expected return and the mean squared TD error estimated by the critic at each iteration. In both tasks, the combination of GAE-REG and Retrace performs the best. From Figure 6a, we can see that GAE-REG is the most important component in the single-pendulum task. First, because only yellow plots converge to the optimal policy. TD-REG also helps, but blue plots did not converge after 500 iterations. Second, because the worst performing versions are the ones without any regularization *but* with Retrace. The reason why Retrace, if used alone, harms the learning can be seen in Figure 6c. Here, the estimated TD error of NO-REG + RETR and of NO-REG + DOUBLE + RETR is rather small, but its poor performance in Figure 6a hints that the critic is affected by overestimation bias. This is not surprising, considering that Retrace addresses the variance and not the bias of the critic.
Results for the double-pendulum are similar but Retrace performs better. GAE-REG + RETR is still the best performing version, but this time Retrace is the most important component, given that all versions without Retrace performed poorly. We believe that this is due to the larger number of new samples collected per iteration.

From this evaluation, we can conclude that the TD- and GAE-regularization are complementary to existing stabilization approaches. In particular, the combination of Retrace and GAE-REG yields very promising results.

5.3 MuJoCo Continuous Control Tasks

We perform continuous control experiments using OpenAI Gym [Brockman et al., 2016] with the MuJoCo physics simulator [Todorov et al., 2012]. For all algorithms, the advantage function is estimated by GAE. For TRPO, we consider a deep RL setting where the actor and the critic are two-layer neural networks with 128 hyperbolic tangent units in each layer. For PPO, the units are 64 in each layer. In both algorithms, both the actor and the critic gradients are optimized by ADAM [Kingma and Ba, 2014]. For the policy update, both the advantage and the TD error estimates are standardized. More details of the hyperparameters are given in Appendix C.

From the evaluation on the pendulum tasks, it emerged that the value of the regularization coefficient η strongly depends on the magnitude of the advantage estimator and the TD error. In fact, for TD-REG we had to decrease η_0 from 1 to 0.1 in the double-pendulum task because the TD error was larger. For this reason, for both PPO and TRPO we tested different initial regularization coefficients η_0 and decay factors κ, choosing among all combinations of $\eta_0 = 0.1, 1$ and $\kappa = 0.99, 0.9999$.

Figure 7 shows the expected return against training iteration for PPO. On Ant, HalfCheetahm, Walker2d, Humanoid and HumanoidStandup both TD-REG and GAE-REG performed substantially better, especially on Ant-v2, where PPO performed very poorly. On Swimmer and Hopper, TD-REG and GAE-REG also outperformed vanilla PPO, but the improvement was less substantial. On Reacher all algorithms performed the same. This behavior is expected, since Reacher is the easiest of MuJoCo tasks, followed by Swimmer and Hopper. On Ant and Walker, we also notice the "slow start" of TD-regularized algorithms already experienced in the LQR. For the first 1,000 iterations, in fact, vanilla PPO expected return increased faster than PPO TD-REG and PPO GAE-REG, but then it also plateaued earlier.

Results for TRPO (Figure 8) are the same except for Humanoid and HumanoidStandup. TD-REG and GAE-REG always outperformed or performed as well as TRPO, and the ranking of the algorithm (first, second and third best performing) is the same of Figure 7. On Ant, HalfCheetah and Walker2d, GAE-REG performed better than TD-REG, while on Swimmer GAE-REG surpassed TD-REG. On Hopper, Reacher, Humanoid, and HumanoidStandup, all algorithms performed the same.

It is also interesting to notice that both GAE-REG and TD-REG shared the best performing η_0 and κ. For instance, with PPO on Ant they both performed best with $\eta_0 = 1.0$ and $\kappa = 0.99999$. Only on HalfCheetah (for PPO) and Swimmer (for TRPO) we had to use different η_0 and κ between GAE-REG and TD-REG. On the contrary, the same values do not work for both PPO and TRPO, as for instance with TRPO on Ant the best performing values were $\eta_0 = 0.1$ and $\kappa = 0.99$.

From this evaluation it emerged that both TD-REG and GAE-REG can substantially improve the performance of TRPO and PPO. However, as for any other method based on regularization, the tuning of the regularization coefficient is essential for their success.

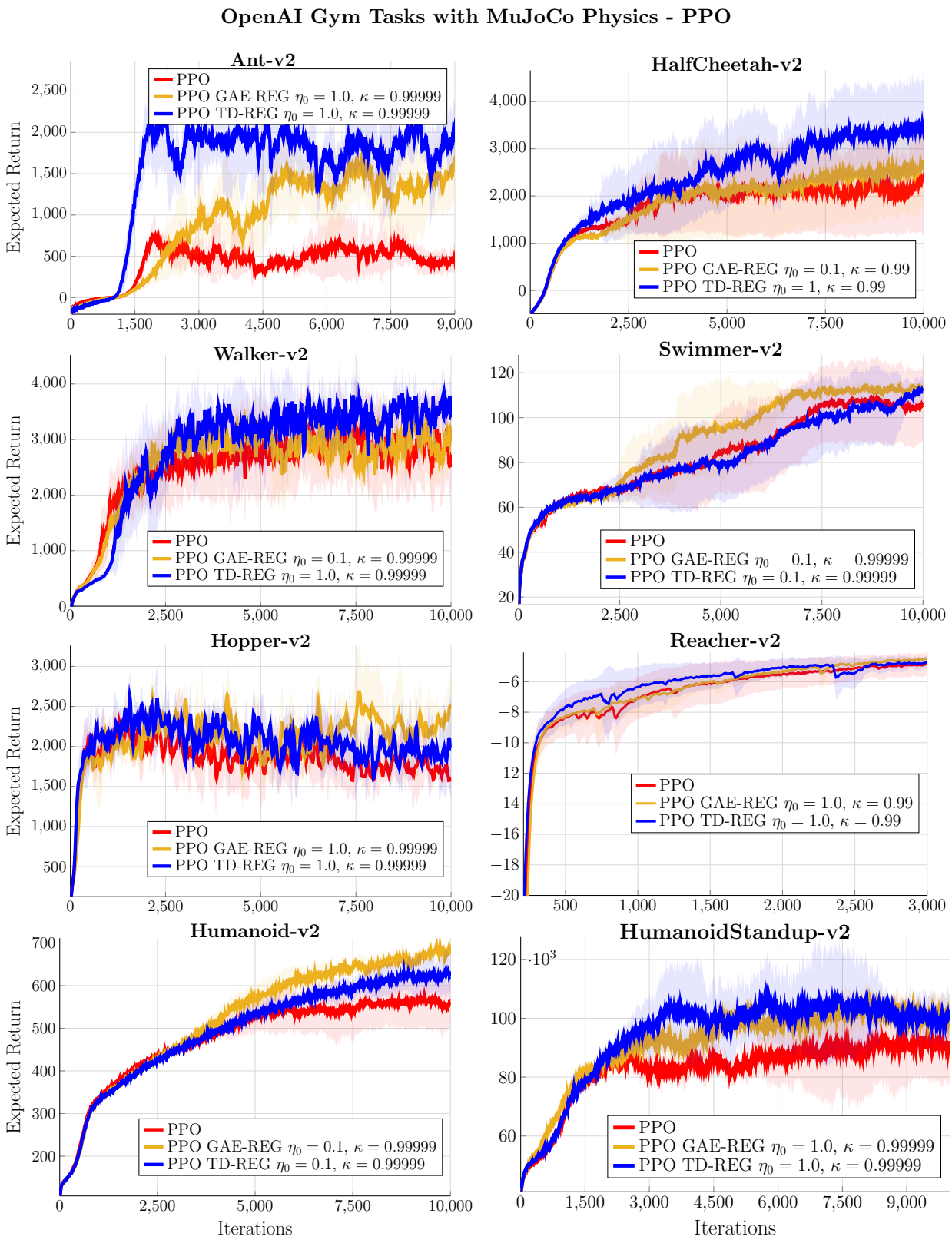


Figure 7: Results averaged over five runs, shaded areas denote 95% confidence interval.

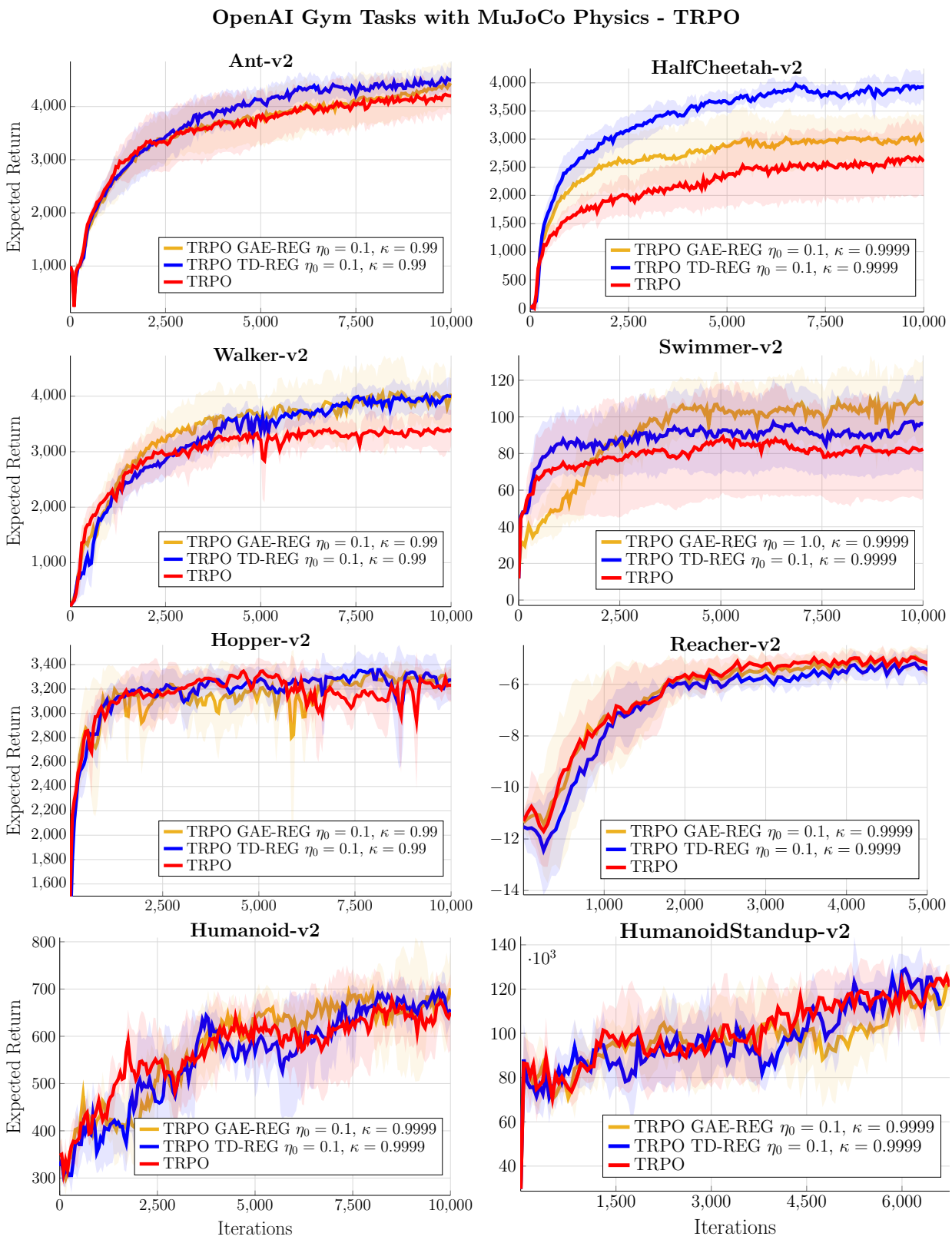


Figure 8: Results averaged over five runs, shaded areas denote 95% confidence interval.

6 Conclusion

Actor-critic methods often suffer from instability. A major cause is the function approximation error in the critic. In this paper, we addressed the stability issue taking into account the relationship between the critic and the actor. We presented a TD-regularized approach penalizing the actor for breaking the critic Bellman equation, in order to perform policy updates producing small changes in the critic. We presented practical implementations of our approach and combined it together with existing methods stabilizing the critic. Through evaluation on benchmark tasks, we showed that our TD-regularization is complementary to already successful methods, such as Retrace, and allows for more stable updates, resulting in policy updates that are less likely to diverge and improve faster.

Our method opens several avenues of research. In this paper, we only focused on direct TD methods. In future work, we will consider the Bellman-constrained optimization problem and extend the regularization to residual methods, as they have stronger convergence guarantees even when nonlinear function approximation is used to learn the critic [Baird, 1995]. We will also study equivalent formulations of the constrained problem with stronger guarantees. For instance, the approximation of the integral introduced by the expectation over the Bellman equation constraint could be addressed by using the representation theorem. Furthermore, we will also investigate different techniques to solve the constrained optimization problem. For instance, we could introduce slack variables or use different penalty functions. Another improvement could address techniques for automatically tuning the coefficient η, which is crucial for the success of TD-regularized algorithms, as emerged from the empirical evaluation. Finally, it would be interesting to study the convergence of actor-critic methods with TD-regularization, including cases with tabular and compatible function approximation, where convergence guarantees are available.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Riad Akrour, Abbas Abdolmaleki, Hany Abdulsamad, and Gerhard Neumann. Model-Free trajectory optimization for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Leemon Baird. Advantage updating. Technical report, Wright-Patterson Air Force Base Ohio: Wright Laboratory, 1993.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine learning (ICML)*, 1995.

Boris Belousov and Jan Peters. f-Divergence constrained policy improvement, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Dotan D. Castro, Dmitry Volkinshtein, and Ron Meir. Temporal difference based actor critic learning - convergence and neural implementation. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Bo Dai, Albert Shaw, Niao He, Lihong Li, and Le Song. Boosting the actor with dual critic. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

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

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in Actor-Critic methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 5(Nov):1471–1530, 2004.

Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G. Bellemare, and Remi Munos. The reactor: A fast and sample-efficient Actor-Critic agent for reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016a.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep Q-Learning with Model-based acceleration. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016b.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine learning (ICML)*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2017.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

Rmi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient Off-Policy reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-PCL: An Off-Policy trust region method for continuous control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2010.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *Transactions on Neural Networks*, 8(5): 997–1007, 1997.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, et al. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman1, Dominik Grewe1, John Nham, Nal Kalchbrenner1, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel1, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

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 (NIPS)*, 1999.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Policy gradients with variance related risk criteria. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Hado van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Ronald J. Williams. Simple statistical Gradient-Following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Tsuneo Yoshikawa. *Foundations of robotics: analysis and control*. Mit Press, 1990.

A 2D Linear-Quadratic Regulator Experiments

The LQR problem is defined by the following discrete-time dynamics

$$s' = As + Ba + \mathcal{N}(0, 0.1^2), \qquad a = Ks, \qquad \mathcal{R}(s,a) = -s^\mathsf{T} X s - a^\mathsf{T} Y a,$$

where $A, B, X, Y \in \mathbb{R}^{d\times d}$, X is a symmetric positive semidefinite matrix, Y is a symmetric positive definite matrix, and $K \in \mathbb{R}^{d\times d}$ is the control matrix. The policy parameters we want to learn are $\boldsymbol{\theta} = \text{vec}(K)$. For simplicity, dynamics are not coupled, i.e., A and B are identity matrices, and both X and Y are identity matrices as well.

Although it may look simple, the LQR presents some challenges. First, the system can become easily unstable, as the control matrix K has to be such that the matrix $(A + BK)$ has eigenvalues of magnitude smaller than one. Therefore, policy updates cannot be too large, in order to prevent divergence. Second, the reward is unbounded and the expected return can be very large, especially at the beginning with an initial random policy. As a consequence, the initial TD error can be very large as well. Third, states and actions are unbounded and cannot be normalized in [0,1], a common practice in RL.

However, we can compute in closed form both the expected return and the Q-function, being able to easily assess the quality of the evaluated algorithms. More specifically, the Q-function is quadratic in the state and in the action, i.e.,

$$Q^\pi(s,a) = Q_0 + s^\mathsf{T} Q_{ss} s + a^\mathsf{T} Q_{aa} a + s^\mathsf{T} Q_{sa} a,$$

where $Q_0, Q_{ss}, Q_{aa}, Q_{sa}$ are matrices computed in closed form given the MDP characteristics and the control matrix K. It should be noted that the linear terms are all zero.

In the evaluation below, we use a 2-dimensional LQR, resulting in four policy parameters. The Q-function is approximated by $\widehat{Q}(s,a;\boldsymbol{\theta}) = \phi(s,a)^\mathsf{T}\boldsymbol{\omega}$, where $\phi(s,a)$ are features. We evaluate two different features: polynomial of second degree (*quadratic features*) and polynomial of third degree (*cubic features*). We know that the true Q-function is quadratic without linear features, therefore quadratic features are sufficient. By contrast, cubic features could overfit. Furthermore, quadratic features result in 15 parameters $\boldsymbol{\omega}$ to learn, while the cubic one has 35.

In our experiments, the initial state is uniformly drawn in the interval $[-10, 10]$. The Q-function parameters are initialized uniformly in $[-1, 1]$. The control matrix is initialized as $K = -K_0^\mathsf{T} K_0$ to enforce negative semidefiniteness, where K_0 is drawn uniformly in [-0.5, -0.1].

Along with the expected return, we show the trend of two mean squared TD errors (MSTDE) of the critic $\widehat{Q}(s,a;\boldsymbol{\omega})$: one is estimated using the TD error, the other is computed in closed form using the true $Q^\pi(s,a)$ defined above. It should be noticed that $Q^\pi(s,a)$ is not the optimal Q-function (i.e., of the optimal policy), but the true Q-function with respect to the current policy. We also show the learning of the diagonal entries of K in the policy parameter space. These parameters, in fact, are the most relevant because the optimal K is diagonal as well, due to the reward and transition functions characteristics ($A = B = X = Y = I$).

All results are averaged over 50 trials. In all trials, the random seed is fixed and the initial parameters are the same (all random). In expected return plots, we bounded the expected return to -10^3 and the MSTDE to $3{\cdot}10^5$ for the sake of clarity, since in the case of unstable policies (i.e., when the matrix $(A+BK)$ has eigenvalues of magnitude greater than or equal to one) the expected return and the TD error are $-\infty$.

A.1 DPG Evaluation on the LQR

In this section, we evaluate five versions of deterministic policy gradient [Silver et al., 2014]. In the first three, the learning of the critic happens in the usual actor-critic fashion. The Q-function is learned independently from the policy and a target Q-function of parameters $\bar{\boldsymbol{\omega}}$, assumed to be independent from the critic, is used to improve stability, i.e.,

$$\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega},\bar{\boldsymbol{\omega}}) = r + \gamma\widehat{Q}(s',\pi(s';\boldsymbol{\theta});\bar{\boldsymbol{\omega}}) - \widehat{Q}(s,a;\boldsymbol{\omega}). \tag{49}$$

Under this assumption, the critic is updated by following the SARSA gradient

$$\nabla_{\boldsymbol{\omega}} \mathbb{E}_{\mu_\beta(s),\beta(a|s),\mathcal{P}(s'|s,a)}\left[\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega},\bar{\boldsymbol{\omega}})^2\right] = \mathbb{E}_{\mu_\beta(s),\beta(a|s),\mathcal{P}(s'|s,a)}\left[\delta_Q(s,a,s';\boldsymbol{\theta},\boldsymbol{\omega},\bar{\boldsymbol{\omega}})\nabla_{\boldsymbol{\omega}}\widehat{Q}(s,a;\boldsymbol{\omega})\right], \tag{50}$$

where $\beta(a|s)$ is the behavior policy used to collect samples. In practice, $\bar{\boldsymbol{\omega}}$ is a copy of $\boldsymbol{\omega}$. We also tried a soft update, i.e., $\bar{\boldsymbol{\omega}}_{t+1} = \tau_{\boldsymbol{\omega}}\boldsymbol{\omega}_t + (1-\tau_{\boldsymbol{\omega}})\bar{\boldsymbol{\omega}}_t$, with $\tau_{\boldsymbol{\omega}} \in (0,1]$, as in DDPG [Lillicrap et al., 2016], the

deep version of DPG. However, the performance of the algorithms decreased (TD-regularized DPG still outperformed vanilla DPG). We believe that, since for the LQR we do not approximate the Q-function with a deep network, the soft update just restrains the convergence of the critic.

These three versions of DPG differ in the policy update. The first algorithm (**DPG**) additionally uses a target actor of parameters $\bar{\boldsymbol{\theta}}$ for computing the Q-function targets, i.e.,

$$\delta_Q(s, a, s'; \bar{\boldsymbol{\theta}}, \boldsymbol{\omega}, \bar{\boldsymbol{\omega}}) = r + \gamma \widehat{Q}(s', \pi(s'; \bar{\boldsymbol{\theta}}); \bar{\boldsymbol{\omega}}) - \widehat{Q}(s, a; \boldsymbol{\omega}), \tag{51}$$

to improve stability. The policy is updated softly at each iteration, i.e., $\bar{\boldsymbol{\theta}}_{t+1} = \tau_{\boldsymbol{\omega}} \boldsymbol{\theta}_t + (1 - \tau_{\boldsymbol{\theta}}) \bar{\boldsymbol{\theta}}_t$, with $\tau_{\boldsymbol{\theta}} \in (0, 1]$ The second algorithm (**DPG TD-REG**) applies the penalty function $G(\boldsymbol{\theta})$ presented in this paper and does not use the target policy, i.e.,

$$\delta_Q(s, a, s'; \boldsymbol{\theta}, \boldsymbol{\omega}, \bar{\boldsymbol{\omega}}) = r + \gamma \widehat{Q}(s', \pi(s'; \boldsymbol{\theta}); \bar{\boldsymbol{\omega}}) - \widehat{Q}(s, a; \boldsymbol{\omega}), \tag{52}$$

in order to compute the full derivative with respect to $\boldsymbol{\theta}$ for the penalty function (Eq. (24)). The third algorithm (**DPG NO-TAR**) is like DPG, but also does not use the target policy. The purpose of this version is to check that the benefits of our approach do not come from the lack of the target actor, but rather from the TD-regularization.

The last two versions are twin delayed DPG (**TD3**) [Fujimoto et al., 2018], which achieved state-of-the-art results, and its TD-regularized counterpart (**TD3 TD-REG**). TD3 proposes three modifications to DPG. First, in order to reduce overestimation bias, there are two critics. Only the first critic is used to update the policy, but the TD target used to update both critics is given by the minimum of their TD target. Second, the policy is not updated at each step, but the update is delayed in order to reduce per-update error. Third, since deterministic policies can overfit to narrow peaks in the value estimate (a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error) noise is added to the target policy. The resulting TD error is

$$\delta_{Q_i}(s, a, s'; \bar{\boldsymbol{\theta}}, \boldsymbol{\omega}, \bar{\boldsymbol{\omega}}_1, \bar{\boldsymbol{\omega}}_2) = r + \gamma \min_{j=1,2} \widehat{Q}(s', \pi(s'; \bar{\boldsymbol{\theta}}) + \xi; \bar{\boldsymbol{\omega}}_j) - \widehat{Q}(s, a; \boldsymbol{\omega}_i), \tag{53}$$

where the noise $\xi = \texttt{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ is clipped to keep the target close to the original action. Similarly to DPG TD-REG, **TD3 TD-REG** removes the target policy (but keeps the noise ξ) and adds the TD-regularization to the policy update. Since TD3 updates the policy according to the first critic only, the TD-regularization considers the TD error in Eq. (53) with $i = 1$.

Hyperparameters

- Maximum number of steps per trajectory: 150.
- Exploration: Gaussian noise (diagonal covariance matrix) added to the action. The standard deviation σ starts at 5 and decays at each step according to $\sigma_{t+1} = 0.95\sigma_t$.
- Discount factor: $\gamma = 0.99$.
- Steps collected before learning (to initialize the experience replay memory): 100.
- Policy and TD errors evaluated every 100 steps.
- At each step, all data collected (state, action, next state, reward) is stored in the experience replay memory, and a mini-batch of 32 random samples is used for computing the gradients.
- DPG target policy update coefficient: $\tau_{\boldsymbol{\theta}} = 0.01$ (DPG NO-TAR is like DPG with $\tau_{\boldsymbol{\theta}} = 1$). With $\tau_{\boldsymbol{\theta}} = 0.1$ results were worse. With $\tau_{\boldsymbol{\theta}} = 0.001$ results were almost the same.
- ADAM hyperparameters for the gradient of $\boldsymbol{\omega}$: $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. With higher α all algorithms were unstable, because the critic was changing too quickly.
- ADAM hyperparameters for the gradient of $\boldsymbol{\theta}$: $\alpha = 0.0005$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Higher α led all algorithms to divergence, because the condition for stability (magnitude of the eigenvalues of $(A + BK)$ smaller than one) was being violated.
- Regularization coefficient: $\eta_0 = 0.1$ and then it decays according to $\eta_{t+1} = 0.999\eta_t$.
- In TD3 original paper, $\xi \sim \mathcal{N}(0, 0.2)$ and is clipped in [-0.5,0.5]. However, the algorithm was tested on tasks with action bounded in [-1,1]. In the LQR, instead, the action is unbounded, therefore we decided to use $\xi \sim \mathcal{N}(0, 2)$ and to clip it in $[-\sigma_t/2, \sigma_t/2]$, where σ_t is the current exploration noise. We also tried different strategies, but we noticed no remarkable differences. The noise is used only for the policy and critics updates, and it is removed for the evaluation of the TD error for the plots.

- In TD3 and TD3 TD-REG, the second critic parameters are initialized uniformly in $[-1, 1]$.
- In TD3 and TD3 TD-REG, the policy is updated every two steps, as in the original paper.
- The learning of all algorithms ends after $12,000$ steps.

Results

Figures 9 and 11a show the results using quadratic features. Since these features are very unlikely to overfit, all algorithms perform almost the same. However, from Figure 11a we can clearly see that the TD-regularization keeps the policy parameters on a more straight path towards the optimum, avoiding detours. Both the TD-regularization and TD3 also result in smaller, but safer, update steps. This behavior is reflected by the learning curves in Figure 9, as DPG and DPG NO-TAR converge slightly faster. However, using both TD3 and TD-REG at the same time can result in excessively slow learning. The green arrows in Figure 11a are, in fact, the smallest, and in Figure 9 TD3 TD-REG did not converge within the time limits in two runs (but it did not diverge either).
Looking at the TD error plots, it is interesting to notice that the estimated TD error is always smaller than the true TD error, meaning that the critic underestimates the TD error. This is a normal behavior, considering the stochasticity of the behavior policy and of the environment. It is also normal that this overestimation bias is less prominent in TD3, thanks to the use of two critics and to delayed policy updates. However, TD3 TD-REG is surprisingly the only algorithm increasing the estimated TD error around mid-learning. We will further investigate this behavior at the end of this section.

Results are substantially different when cubic features are used (Figure 10). In this case, many features are irrelevant and the model is prone to overfit. As a consequence, the TD error shown in Figure 10 is much larger than the one shown in Figure 9, meaning that it is harder for the critic to correctly approximate the true Q-function. The problem is prominent for DPG and DPG NO-TAR, which cannot learn 24 and 28 times, respectively, out of 50 (thus, the very large confidence interval). Similarly to the previous case, the true TD error is underestimated. Initially their critics estimate a TD error of 10^5 but the true one is $2.5{\cdot}10^5$. This large error guides the actors incorrectly, inevitably leading to divergence. Then, after approximately 3,000 steps, the two TD errors match at $1.5{\cdot}10^5$. Among the two algorithms, DPG NO-TAR performs worse, due to the lack of the target policy.
TD3 performs substantially better, but still diverges two times. By contrast, TD-REG algorithms never diverges. Figure 11b shows the benefits of the TD-regularization in the policy space. Initially, when the TD error is large, the policy "moves around" the starting point. Then, as the critic accuracy increases, the policy goes almost straightforwardly to the goal. This "slow start" behavior is also depicted in Figure 10, where DPG TD-REG expected return initially improves more slowly compared to TD3 and TD3 TD-REG. Finally, we notice once again that the combination of TD3 and TD-REG results in the slowest learning: unlike TD3, TD3 TD-REG never diverged, but it also never converged within the time limit. This behavior is also depicted in Figure 11b, where green arrows (TD3 TD-REG policy updates) are so small that the algorithm cannot reach the goal in time.

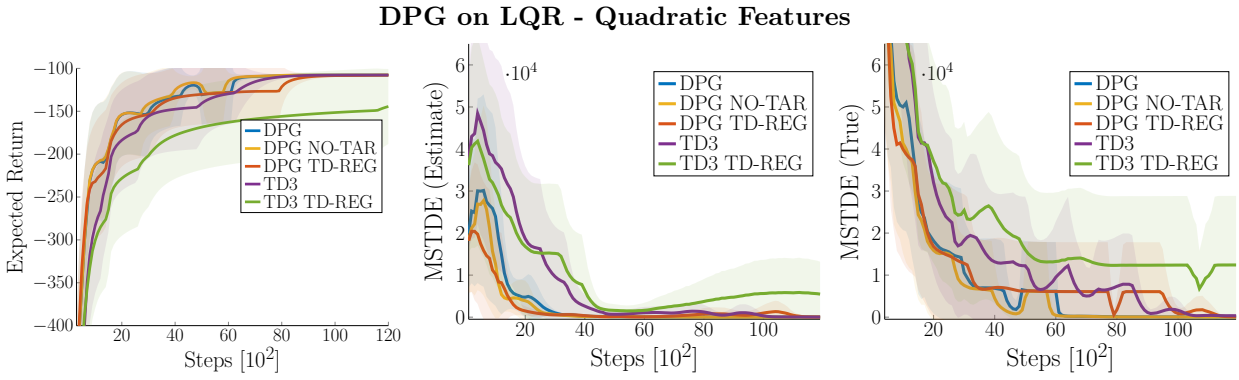


Figure 9: All algorithms perform similarly (DPG and DPG NO-REG almost overlap), because quadratic features are sufficient to approximate the true Q-function. Only TD3 TD-REG did not converge within the time limit in two runs, but it did not diverge either.

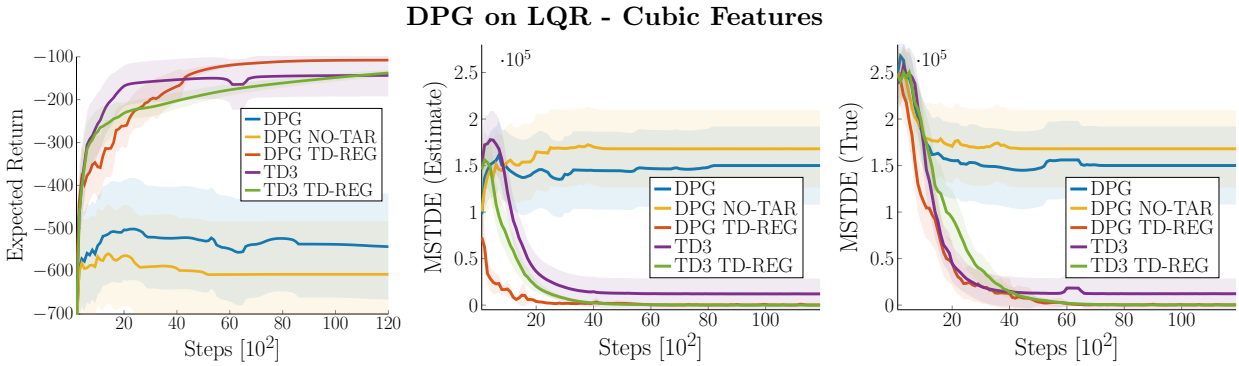
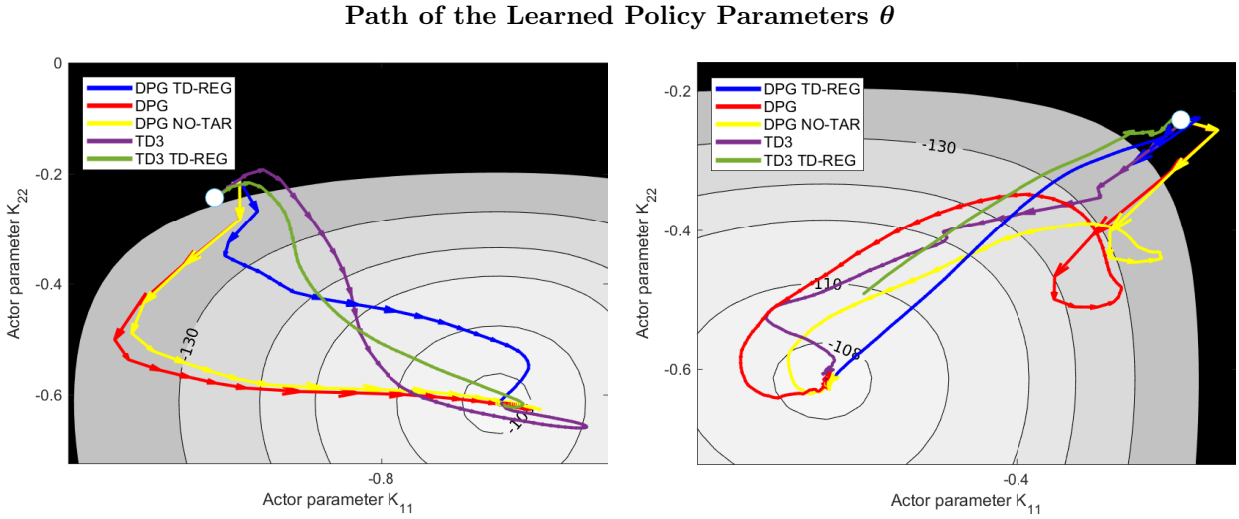


Figure 10: By contrast, with cubic features the critic is prone to overfit, and DPG and DPG NO-TAR diverged 24 and 28 times, respectively. TD3 performed better, but still diverged two times. TD-regularized algorithms, instead, never diverged, and DPG TD-REG always learned the true Q-function and the optimal policy within the time limit. Similarly to Figure 9, TD3 TD-REG is the slowest, and in this case it never converged within the time limit, but did not diverge either (see also Figure 11b).



(a) Quadratic features.

(b) Cubic features.

Figure 11: Paths followed by the policy parameters during runs in which no algorithm diverged. Each arrow represents 100 update steps. Contour denotes the expected return magnitude. The initial policy parameter vector $\boldsymbol{\theta}$ is denoted by the white circle. The TD-regularization enables safer and more stable trajectories, leading the policy parameters more straightforwardly to the optimum. We can also see that blue and green arrows are initially shorter, denoting smaller update steps. In fact, due to the initial inaccuracy of the critic, the TD-regularization gradient "conflicts" with vanilla gradient at early stages of the learning and avoids divergence. However, in the case of TD3 TD-REG (green arrows) the use of both two critics, delayed policy updates and TD-regularization slows down the learning up to the point that the algorithm never converged within the time limit, as in Figure 11b.

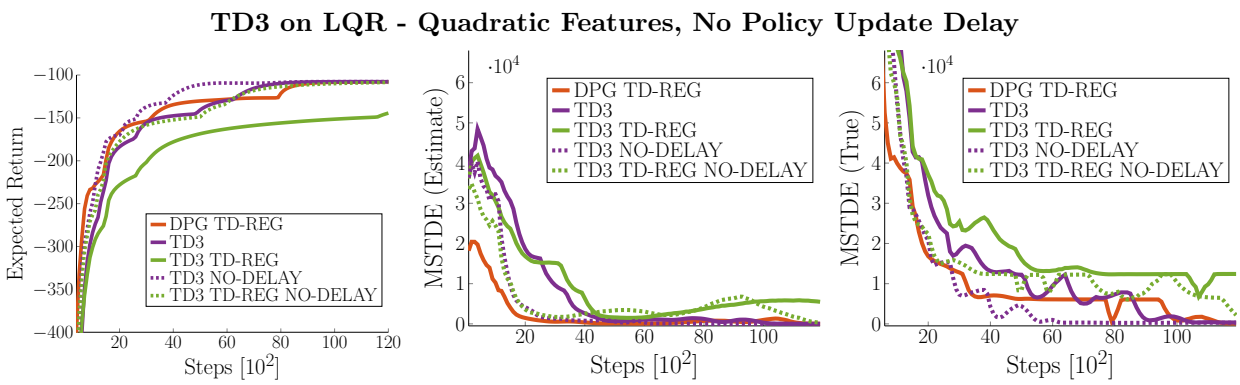


Figure 12: Without policy update delays, both TD3 and TD3 TD-REG converge faster.

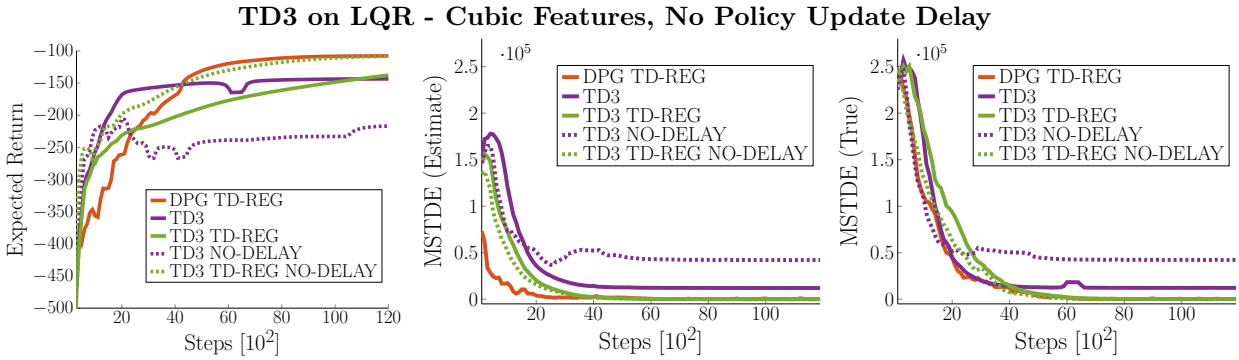


Figure 13: With cubic features, TD3 NO-DELAY performs worse than TD3, diverging six times instead of two. By contrast, TD3 TD-REG NO-DELAY performs better than TD3 TD-REG and it always converges within the time limit.

Figure 10 hints that the "slow learning" behavior of TD3 TD-REG may be due to the delayed policy update, as both the estimated and the true TD error are close to zero by mid-learning. To further investigate this behavior, we performed the same experiments without delayed policy updates for both TD3 and TD3-REG. For the sake of clarity, we report the results on separates plots without DPG and DPG NO-TAR and without shaded areas for confidence interval. In the case of quadratic features (Figure 12) both TD3 and TD3 TD-REG gain from the removal of the policy update delay. However, in the case of cubic features (Figure 13), TD3 NO-DELAY performs worse than TD3, as it diverges six times instead of two. By contrast, TD3 TD-REG NO-DELAY performs better than TD3 TD-REG, and the expected return curve shows traits of both TD3 and TD-REG: initially it improves faster, like TD3, and then it always converges to optimality, like DPG TD-REG. We can conclude that delaying policy updates is not necessary when appropriate features are used and overfitting cannot happen. Otherwise, the combination of two critics and TD-regularization yields the best results, with the TD-regularization providing the most benefits.

A.2 SPG Evaluation on the LQR

DPG is an on-line algorithm, i.e., it performs a critic/actor update at each time step, using mini-batches of previously collected samples. Instead, stochastic policy gradient (SPG) collects complete trajectories with a stochastic policy before updating the critic/actor. In this section, we evaluate SPG, SPG TD-REG, and REINFORCE [Williams, 1992]. The first two maximize Q-function estimates given by a learned critic. SPG TD-REG additionally applies the TD-regularization presented in the paper. REINFORCE, instead, maximizes Monte Carlo estimates of the Q-function, i.e.,

$$\widehat{Q}^{\pi}(s_t, a_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i. \tag{54}$$

The policy is Gaussian, i.e., $\pi(a|s;\boldsymbol{\theta}) = \mathcal{N}(Ks, \Sigma)$, where Σ is a diagonal covariance matrix. K is initialized as for DPG. The diagonal entries of Σ are initialized to five. Six policy parameters are learned, four for K

and two for Σ. For SPG TD-REG, the expectation $\mathbb{E}_{\pi(a'|s';\boldsymbol{\theta})}\left[\widehat{Q}(s',a';\boldsymbol{\omega})\right]$ is approximated with the policy mean Q-value, i.e., $\widehat{Q}(s',Ks';\boldsymbol{\omega})$. For SPG and SPG TD-REG, $\widehat{Q}(s,a;\boldsymbol{\omega})$ is learned by Matlab `fminunc` optimizer using the samples collected during the current iteration.

Hyperparameters

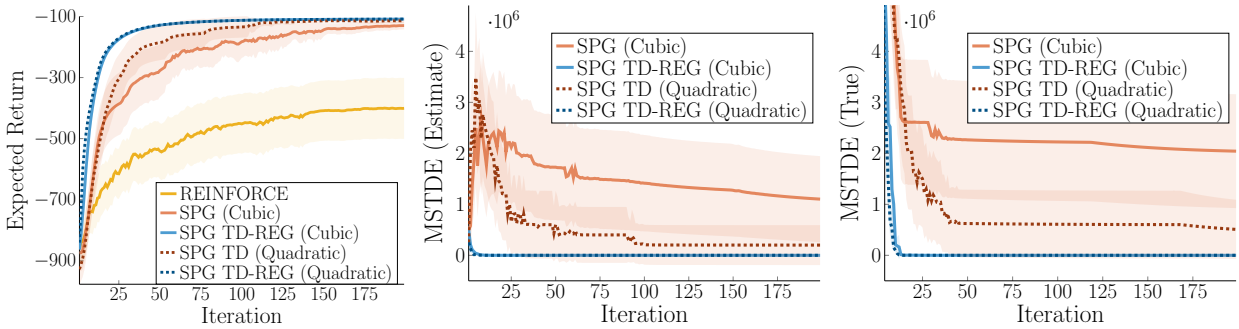
- Trajectories collected per iteration: 1 or 5.
- Steps per trajectory: 150.
- Discount factor: $\gamma = 0.99$.
- Policy and TD error evaluated at every iteration.
- No separate target policy or target Q-function are used for learning the critic, but we still consider $\nabla_{\boldsymbol{\omega}}\widehat{Q}(s',a';\boldsymbol{\omega}) = 0$.
- Policy update learning rate: 0.01. The gradient is normalized if its norm is larger than one.
- The policy update performs only one step of gradient descent on the whole dataset.
- Regularization coefficient: $\eta_0 = 0.1$ and then it decays according to $\eta_{t+1} = 0.999\eta_t$.

Results

Figures 14 and 15 show the results when one or five episodes, respectively, are used to collect samples during one learning iteration. REINFORCE performed poorly in the first case, due to the large variance of Monte Carlo estimates of the Q-function. In the second case, it performed better but still converged slowly. SPG performed well except for two runs in Figure 15, in which it diverged already from the beginning. In both cases, its performance is coupled with the approximator used, with quadratic features yielding more stability. By contrast, SPG TD-REG never failed, regardless of the number of samples and of the function approximator used, and despite the wrong estimate of the true TD error. Similarly to what happened in DPG, in fact, the critic always underestimates the true TD error, as shown in Figures 14b and 15b.

Finally, Figure 16 shows the direction and the magnitude of the gradients. We clearly see that initially the vanilla gradient (which maximizes Q-function estimates, red arrows) points towards the wrong direction, but thanks to the gradient of the TD-regularization (blue arrows) the algorithm does not diverge. As the learning continues, red arrows point towards the optimum and the magnitude of blue arrows decreases, because 1) the critic becomes more accurate and the TD error decreases, and 2) the regularization coefficient η decays. The same behavior was already seen in Figure 11 for DPG, with the penalty gradient dominating the vanilla gradient at early stages of the learning.

SPG on LQR - One Episode per Iteration



(a) (b) (c)

Figure 14: Results averaged over 50 runs, shaded areas denote 95% confidence interval. Only one episode is collected to update the critic and the policy. SPG TD-REG did not suffer from the lack of samples, and it always learned the optimal critic and policy within few iterations, both with quadratic and cubic features (the two blue plots almost overlap). By contrast, vanilla SPG is much more sensitive to the number of samples. Even if it never diverged, its convergence is clearly slower. REINFORCE, instead, diverged 13 times, due to the high variance of Monte Carlo estimates.

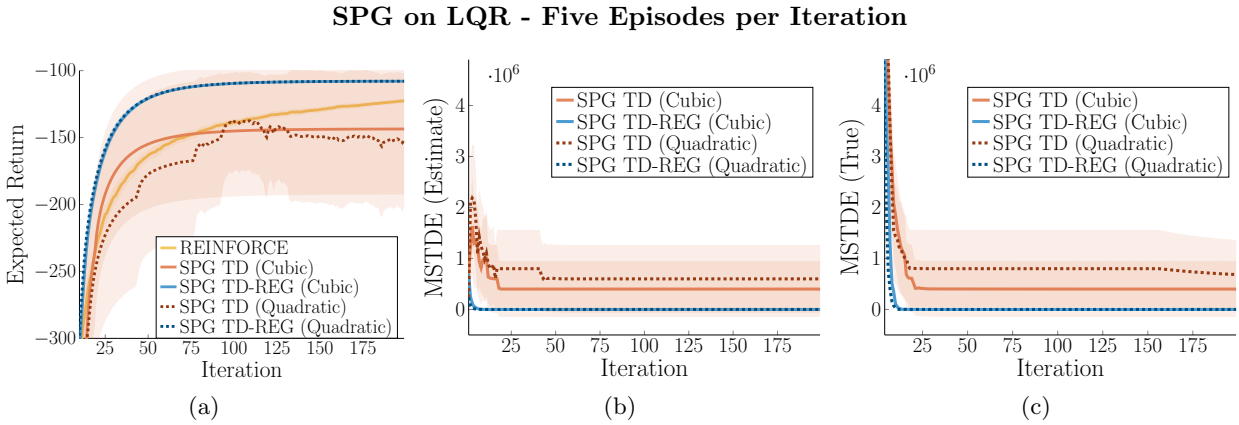


Figure 15: With more samples per iteration REINFORCE was able to converge, as Monte Carlo estimates have lower variance. However, it still converged more slowly than SPG TD-REG. Surprisingly, vanilla SPG diverged two times, thus explaining its slightly larger confidence interval (in Figure 15a the y-axis starts at -300 for the sake of clarity). In the other 48 runs, it almost matched SPG TD-REG.

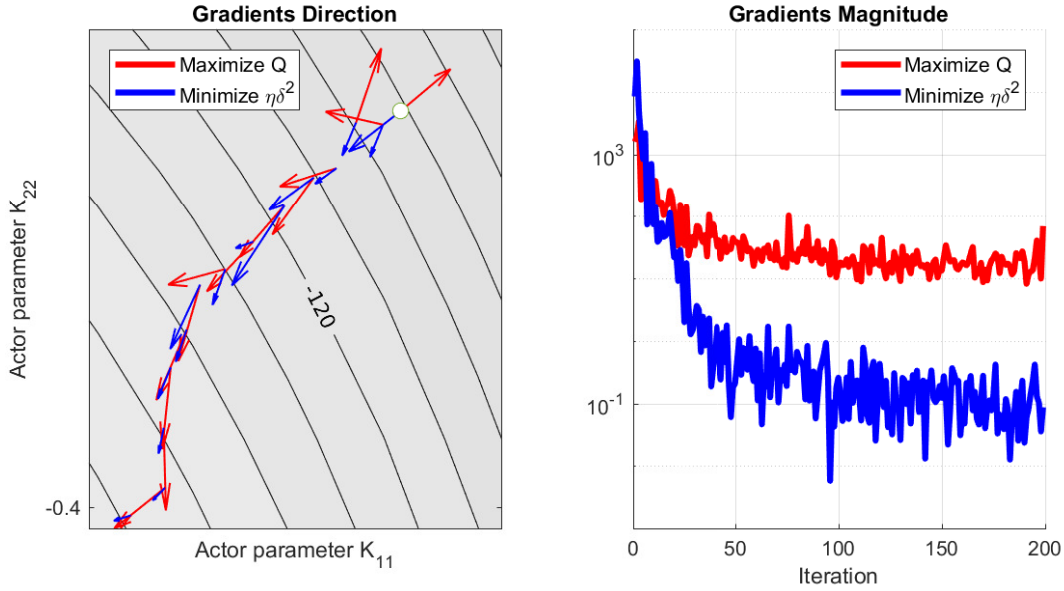
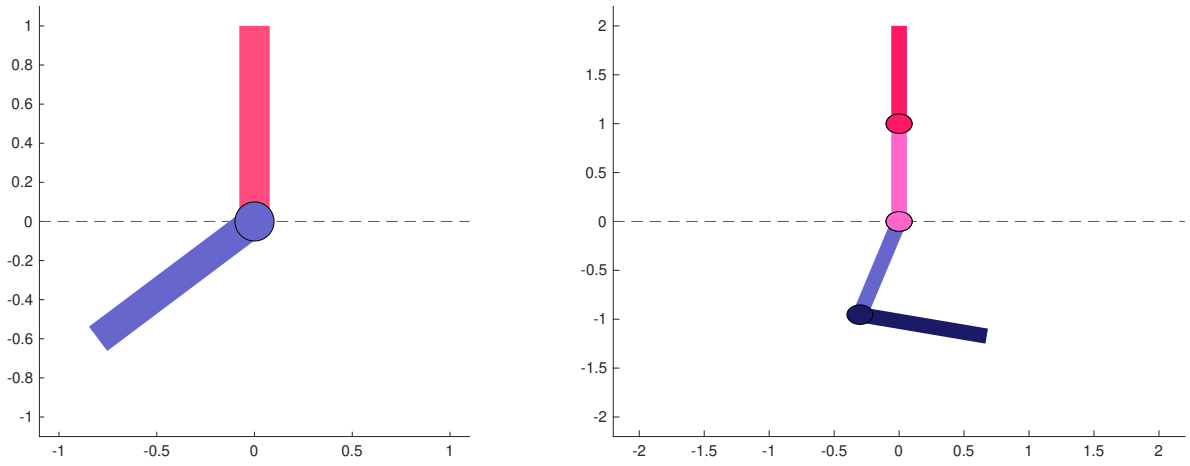


Figure 16: On the left, beginning of the path followed by SPG TD-REG in the policy parameter space during one trial, when samples from only one episode are collected per iteration. The initial policy parameter vector $\boldsymbol{\theta}$ is denoted by the white circle. Each arrow represents one iteration. Red and blue arrows denote the direction of the gradient $\nabla_{\boldsymbol{\theta}} \mathbb{E}[\widehat{Q}(s, a; \boldsymbol{\omega})]$ and $\nabla_{\boldsymbol{\theta}} \mathbb{E}[\eta \delta_Q(s, a, s'; \boldsymbol{\theta}, \boldsymbol{\omega})^2]$, respectively. Contour denotes the expected return magnitude. The magnitude of the arrows has been scaled for better visualization, and the true one is shown on the right. Initially, as the critic estimate is highly inaccurate (see the TD error in Figures 14b and 14c), red arrows point to the wrong direction. However, blue arrows help keeping the policy on track.

B Pendulum Swing-up Tasks Experiments



(a) Single-pendulum. (b) Double-pendulum.

Figure 17: The two pendulum swing-up tasks, with an example of the current state (in blue) and the goal state (in red).

Single-pendulum. This task (Figure 17a) follows the equations presented in OpenAI Gym [Brockman et al., 2016]. The state consists of the angle position q and velocity $\dot{q}$ of the link. The former is bounded in $[-\pi, \pi]$ and is initialized uniformly in $[-\pi, \pi]$. The latter is bounded in $[-8, 8]$ and is initialized in $[-1, 1]$. As the pendulum is underactuated, the action is bounded in $[-2, 2]$. The transition function is

$$\dot{q}_{t+1} = \dot{q}_t - \frac{3g}{2l}\sin(q + \pi) + \frac{3}{ml^2}a\delta_t,$$
$$q_{t+1} = q_t + \dot{q}_{t+1}\delta_t,$$

where $g = 10$ is the gravitational acceleration, $m = 1$ is the link mass, $l = 1$ is the link length, and $\delta_t = 0.05$ is the timestep. The goal state is $q = 0$ and the reward function is

$$r_t = -q_t^2 - 0.1\dot{q}_t^2 - 0.001a_t^2.$$

For learning, at each iteration only 10 episodes of 50 steps are collected. For the expected return evaluation, the policy is tested over 1,000 episodes of 150 steps.
The critic approximates the V-function with a linear function $\widehat{V}(s; \boldsymbol{\omega}) = \phi(s)^\mathsf{T}\boldsymbol{\omega}$ where $\phi(s)$ are 100 random Fourier features, as presented in [Rajeswaran et al., 2017]. The bandwidths are computed as suggested by [Rajeswaran et al., 2017] as the average pairwise distances between 10,000 state observation vectors. These states are collected only once before the learning and are shared across all 50 trials. The phase and the offset of the Fourier features are instead random and different for all trials.
The policy is Gaussian, i.e., $\pi(a|s; \boldsymbol{\theta}) = \mathcal{N}(b + K\phi(s), \sigma^2)$. The same Fourier features of the critic are used for the policy, for a total of 102 policy parameters to learn. K and b are initialized to zero, while σ to four. For the expected return evaluation, the noise of the policy is zeroed.
The critic parameters are initialized uniformly in $[-1, 1]$ and are learned with Matlab `fminunc` optimizer such that they minimize the mean squared TD(λ) error (since we use GAE). GAE hyperparameters are $\gamma = 0.99$ and $\lambda = 0.95$.
The policy is learned with TRPO with a KL bound of 0.01. For computing the natural gradient, both the advantage, the TD error and the regularization are standardized, i.e., $\hat{y} \leftarrow (\hat{y} - \mu_{\hat{y}})/\sigma_{\hat{y}}$, where $\hat{y}$ is either the advantage estimator, the TD error estimator, or the regularization (δ^2 or $\widehat{A}^2$, depending if we use TD-REG or GAE-REG), $\mu_{\hat{y}}$ is the mean of the estimator and $\sigma_{\hat{y}}$ its standard deviation. The conjugate gradient is computed with Matlab `pcg`. Additionally, since TRPO uses a quadratic approximation of the KL divergence, backtracking line search is performed to ensure that the KL bound is satisfied. The starting regularization coefficient is $\eta_0 = 1$ and then decays according to $\eta_{t+1} = \kappa\eta_t$ with $\kappa = 0.999$.

Double-pendulum. This task (Figure 17b) follows the equations presented by Yoshikawa [1990]. The state consists of the angle position $[q_1, q_2]$ and velocity $[\dot{q}_1, \dot{q}_2]$ of the links. Both angles are bounded in $[-\pi, \pi]$ and initialized uniformly in $[-\pi, \pi]$, while both velocities are bounded in $[-50, 50]$ and initialized in $[-1, 1]$. The agent, however, observes the six-dimensional vector $[\sin(q), \cos(q), \dot{q}]$. As the pendulum is underactuated, the action on each link is bounded in $[-10, 10]$. The transition function is

$$\begin{aligned}
\ddot{q}_{t+1} &= M_t^{-1}(a_t - f_{gt} - f_{ct} - f_{vt}),\\
\dot{q}_{t+1} &= \dot{q}_t - \ddot{q}_{t+1}\delta_t,\\
q_{t+1} &= q_t + \dot{q}_{t+1}\delta_t,
\end{aligned}$$

where $\delta_t = 0.02$ is the timestep, M is the inertia matrix, f_g is gravitational, f_c the Coriolis force, and f_v the frictional force. The entries of M are

$$\begin{aligned}
M_{11} &= m_1\left(\frac{l_1}{2}\right)^2 + I_1 + m_2\left(l_1^2 + \left(\frac{l_2}{2}\right)^2 + 2l_1\left(\frac{l_2}{2}\right)^2\cos(q_2)\right) + I_2,\\
M_{12} = M_{21} &= m_2\left(\left(\frac{l_2}{2}\right)^2 + l_1\frac{l_2}{2}\cos(q_2)\right) + I_2,\\
M_{22} &= m_2\left(\frac{l_2}{2}\right)^2 + I_2,
\end{aligned}$$

where $l_i = 1$ is the length of a link, $m_i = 1$ is the mass of the a link, and $I_i = (1 + 0.0001)/3.0$ is the moment of inertia of a link. Gravitational forces are

$$\begin{aligned}
f_{g1} &= m_1 g\frac{l_1}{2}\cos(q_1) + m_2 g\left(l_1\cos(q_1) + \frac{l_2}{2}\cos(q_1 + q_2)\right),\\
f_{g2} &= m_2 g\frac{l_2}{2}\cos(q_1 + q_2),
\end{aligned}$$

where $g = 9.81$ is the gravitational acceleration. Coriolis forces are

$$\begin{aligned}
f_{c1} &= -m_2 l_1\frac{l_2}{2}\sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2),\\
f_{c2} &= m_2 l_1\frac{l_2}{2}\sin(q_2)\dot{q}_1^2.
\end{aligned}$$

Frictional forces are

$$\begin{aligned}
f_{v1} &= v_1\dot{q}_1,\\
f_{v2} &= v_1\dot{q}_2,
\end{aligned}$$

where $v_i = 2.5$ is the viscous friction coefficient. The goal state is $q = [\pi/2, 0]$ and the reward function is

$$r_t = -||\pi - \texttt{abs}(\texttt{abs}(q - q_{\text{GOAL}}) - \pi)||_2^2 - 0.001||a||_2^2,$$

where the first term is the squared distance between the current angles and the goal position $q_{\text{GOAL}} = [\pi/2, 0]$, wrapped in $[-\pi, \pi]$. Note that, compared to the single-pendulum, the frame of reference is rotated of 90°. The first angle q_1 is the angle between the base of the pendulum and the first link. The second angle q_2 is the angle between the two links.

For learning, at each iteration only 6 episodes of 500 steps are collected, because the double-pendulum needs more steps to swing up than the single-pendulum. For the expected return evaluation, the policy is tested over 1,000 episodes of 500 steps. The same actor-critic setup of the single-pendulum is used, except for

- The Fourier features, which are 300,
- The policy, which has a full covariance matrix. Its diagonal entries are initialized to 200 and we learn its Cholesky decomposition, for a total of 605 parameters to learn, and
- The initial regularization coefficient for TD-REG, which is $\eta_0 = 0.1$.

C MuJoCo Continuous Control Tasks Experiments

The continuous control experiments are performed using OpenAI Gym [Brockman et al., 2016] with MuJoCo physics simulator. All environments are version 2 (v2). We use the same hyperparameters and neural network architecture for all tasks. The policy is a Gaussian distribution with a state-independent diagonal covariance matrix. Each algorithm is evaluated over five trials using different random seeds with common seeds for both methods. For TRPO, the policy was evaluated over 20 episodes without exploration noise. For PPO, we used the same samples collected during learning, i.e., including exploration noise.

The actor mean and the critic networks are two-layer neural networks with hyperbolic tangent activation function. Because all tasks actions are bounded in $[-1, 1]$, the actor network output has an additional hyperbolic tangent activation.

At each iteration, we collect trajectories until there are at least 3,000 transition samples in the training batch. The maximum trajectory length is 1,000 steps. Then, we compute the GAE estimator as in Eq. (46) and train the critic to minimize the mean squared TD(λ) error (Eq. (42)) using ADAM [Kingma and Ba, 2014][8]. The actor is then trained using the same training batch, following the algorithm policy update (PPO, TRPO, and their respective regularized versions). For the policy update, both the advantage, the TD error and the regularization are standardized, i.e., $\hat{y} \leftarrow (\hat{y} - \mu_{\hat{y}})/\sigma_{\hat{y}}$, where $\hat{y}$ is either the advantage estimator, the TD error estimator, or the regularization (δ^2 or $\widehat{A}^2$, depending if we use TD-REG or GAE-REG), $\mu_{\hat{y}}$ is the mean of the estimator and $\sigma_{\hat{y}}$ its standard deviation.

TRPO Hyperparameters

- Hidden units per layer 128.
- Policy initial standard deviation 1.
- GAE hyperparameters: $\lambda = 0.97$ and $\gamma = 0.995$
- ADAM hyperparameters: 20 epochs, learning rate 0.0003, mini-batch size 128.
- Policy KL divergence bound $\epsilon = 0.01$.
- We also add a damping factor 0.1 to the diagonal entries of the Fisher information matrix for numerical stability
- The maximum conjugate gradient step is set to 10.

PPO Hyperparameters

- Hidden units per layer 64.
- Policy initial standard deviation 2.
- GAE hyperparameters: $\lambda = 0.95$ and $\gamma = 0.99$
- ADAM hyperparameters (for both the actor and the critic): 20 epochs, learning rate 0.0001, mini-batch size 64.
- Policy clipping value $\varepsilon = 0.05$.

[8] TRPO original paper proposes a more sophisticated method to solve the regression problem. However, we empirically observed that batch gradient descent is sufficient for good performance.