Balancing Learning Speed and Stability in Policy Gradient via Adaptive Exploration

Matteo Papini
Politecnico di Milano

Andrea Battistello
Politecnico di Milano

Marcello Restelli
Politecnico di Milano

Abstract

In many Reinforcement Learning (RL) applications, the goal is to find an optimal deterministic policy. However, most RL algorithms require the policy to be stochastic in order to avoid instabilities and perform a sufficient amount of exploration. Adjusting the level of stochasticity during the learning process is non-trivial, as it is difficult to assess whether the costs of random exploration will be repaid in the long run, and to contain the risk of instability. We study this problem in the context of policy gradients (PG) with Gaussian policies. Using tools from the safe PG literature, we design a surrogate objective for the policy variance that captures the effects this parameter has on the learning speed and on the quality of the final solution. Furthermore, we provide a way to optimize this objective which guarantees a stable improvement of the original performance measure. We evaluate the proposed methods on simulated continuous control tasks.

1 Introduction

Reinforcement learning (RL, Sutton and Barto, 2018) is an approach to adaptive intelligence that employs a reward signal to train an autonomous agent on a general task through direct interaction with an unknown environment. The results recently achieved by RL in challenging games (Mnih et al., 2015; Silver et al., 2017; OpenAI, 2018) are astounding. However, in order to apply RL to real-world scenarios (e.g., robotics, autonomous driving, finance), we have to tackle further challenges. Unlike games, problems involving physical systems are more naturally modeled as continuous control tasks. For this reason, we will focus on policy gradient (PG, Sutton et al., 1999; Deisenroth et al., 2013), an RL technique that employs stochastic gradient ascent to optimize parametric controllers. PG is particularly suitable for continuous tasks due to its robustness to noise, convergence properties, and versatility in policy design (Peters et al., 2005). Another perk of games is that they are easily simulated. Simulations require a reliable model of the environment, which is often not available. Learning online on a physical system (like a robot) sharpens the need for *stable* learning algorithms, as large deviations from known policies may yield unsafe behavior or unrecoverable costs.

Although we normally look for a deterministic controller, PG is only able to stably improve *stochastic* policies. A notable exception is Deterministic Policy Gradient (DPG, Silver et al., 2014; Lillicrap et al., 2016), which optimizes a deterministic policy while collecting data with a noisy version of it. Even in this case, the stochastic nature of the behavioral policy is necessary to maintain a sufficient level of exploration and avoid the local optima. This introduces an inevitable trade-off: policy stochasticity facilitates, stabilizes, and speeds up the learning of other policy parameters (Ahmed et al., 2019). On the other hand, random behavior yields worse *online* performance, and may be unsafe in some applications. A natural solution to this problem is to adapt the *amount of exploration* during the learning process. Unfortunately, this is highly non-trivial, as it falls under the infamous exploration-exploitation dilemma. If exploration is abandoned too soon, the agent may never know all the relevant aspects of the task and get stuck in suboptimal behavior. If the transition to deterministic behavior is delayed too much, the learning process may become unnecessarily long and expensive. This problem has been thoroughly studied in the Multi-Armed Bandit (MAB) literature (Bubeck and Cesa-Bianchi, 2012; Lattimore and Szepesvári, 2019). Adaptive exploration has a long history also in RL, mostly limited to the tabular setting (Kearns and Singh, 2002;

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

Brafman and Tennenholtz, 2002; Strehl et al., 2009; Jaksch et al., 2010; Lattimore and Hutter, 2014; Dann and Brunskill, 2015; Dann et al., 2017; Jin et al., 2018; Ok et al., 2018), with extensions to continuous states (Ortner and Ryabko, 2012; Lakshmanan et al., 2015; Bellemare et al., 2016). In Policy Gradient methods, it is common to learn the amount of stochasticity via gradient ascent on the performance measure, like any other policy parameter (Duan et al., 2016). As randomness typically erodes performance, this *greedy* approach can yield premature convergence to quasi-deterministic policies, causing learning instability and getting stuck to local optima. The current trend is to augment the traditional performance objective (Sutton et al., 1999) with an entropy bonus (Haarnoja et al., 2017, 2018; Nachum et al., 2018; Shani et al., 2018) which favors more stochastic policies. However, the exploration-exploitation trade-off is still unsolved, as one has to decide how much weight to give to the entropy bonus.

In this paper, we propose an alternative approach: we design a separate optimization objective for the policy stochasticity that also accounts for the long-term effects of random exploration. We do so for the special case of Gaussian policies, which are, however, the most used in practice (Duan et al., 2016). In this framework, the amount of exploration can be controlled via the policy variance parameters[1]. To do so, we use insights from the *safe policy gradient* literature (Kakade and Langford, 2002; Pirotta et al., 2013, 2015; Papini et al., 2017, 2019), where the effects of policy variance on performance improvement have been already exposed, but not fully exploited. We hence propose the Meta-Exploring Policy Gradient (MEPG) algorithm, that optimizes the variance parameters via gradient ascent on the new surrogate objective, while concurrently optimizing the other parameters as in vanilla PG.

Although built upon insights from the safe PG literature, the MEPG algorithm is heuristic and comes with no formal guarantees. By developing an adaptive meta-parameter schedule, we devise the Stably Exploring Policy Gradient (SEPG) algorithm, a variant of MEPG with guarantees of monotonic improvement of the original performance objective. To do so, we generalize existing improvement guarantees (Papini et al., 2019) for Gaussian policies to the previously uncharted case of adaptive policy variance. These improvement guarantees come at the cost of worsening learning speed and sample complexity, which can be critical in RL applications, where collecting samples is costly and time-consuming. Although the scope of this work is mostly theoretical, for the sake of applicability, we also relax the classical monotonic improvement constraint (Kakade and Langford, 2002) to a more general *bounded worsening* constraint. This allows the practitioner to specify how much performance he would accept to lose (and with which probability) at any policy update, introducing a way to trade-off the stability of learning with the time required.

The paper is structured as follows: in Section 2, we provide an essential background on PG and review the existing performance improvement guarantees for this framework. In Section 3, we present our novel exploration objective and the MEPG algorithm. In Section 4.1, we extend existing improvement guarantees for Gaussian policies to the adaptive-variance case, providing safe exact-gradient updates. In section 4.2, we generalize these results to the more realistic stochastic-gradient case, and present the SEPG algorithm. Finally, in Section 5, we evaluate the proposed algorithms on simulated control tasks. Proofs of all the formal statements are given in Appendix A.

2 Preliminaries

In this section, we provide an essential background on policy gradient methods, including existing performance-improvement guarantees.

2.1 Policy Gradient Fundamentals

A continuous Markov Decision Process (MDP, Puterman, 1994) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$ is defined by a continuous state space $\mathcal{S} \subseteq \mathbb{R}^d$; a continuous action space $\mathcal{A}$; a Markovian transition kernel $\mathcal{P}$, where $\mathcal{P}(s'|s,a)$ is the transition density from state s to s' under action a; a reward function $\mathcal{R}$, where $\mathcal{R}(s,a) \in [-R_{\max}, R_{\max}]$ is the reward for state-action pair (s,a) and $R_{\max}$ is the maximum absolute-value reward; a discount factor $\gamma \in [0,1)$; and an initial-state distribution ρ on $\mathcal{S}$. An agent's behavior is modeled as a policy π, where $\pi(\cdot|s)$ is the density function over $\mathcal{A}$ in state s. We study episodic MDPs with indefinite horizon. In practice, we consider episodes of length H, the effective horizon of the task. A trajectory τ is a sequence of states and actions $(s_0, a_0, s_1, a_1, \dots, s_{H-1}, a_{H-1})$ observed by following a stationary policy, where $s_0 \sim \rho$ and $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$. The policy induces a measure p_π over trajectories. We denote with $\mathcal{R}(\tau)$ the total discounted reward provided by trajectory τ: $\mathcal{R}(\tau) = \sum_{h=0}^{H-1} \gamma^h \mathcal{R}(s_h, a_h)$. Policies can be ranked based on their expected total reward $J(\pi) = \mathbb{E}_{\tau \sim p_\pi}[\mathcal{R}(\tau)]$. Solving the MDP means finding an optimal policy $\pi^* \in \arg\max_\pi \{J(\pi)\}$.

[1]Extensions to other classes of policies are possible, but require to identify explicit scale parameters e.g., the *temperature* for Softmax policies or the *variance* for (reparametrized) beta policies.

Policy gradient (PG, Sutton et al., 1999; Peters and Schaal, 2008) methods restrict this optimization problem to a class of parametric policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^m\}$, so that $\pi_{\boldsymbol{\theta}}$ is differentiable w.r.t. $\boldsymbol{\theta}$. We denote the performance of a parametric policy $\pi_{\boldsymbol{\theta}}$ with $J(\boldsymbol{\theta})$. A *locally* optimal policy can be found via gradient ascent on the performance measure:

$$\begin{aligned}&\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_t \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t),\\ &\text{where} \quad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau) \mathcal{R}(\tau)\right],\end{aligned} \tag{1}$$

where t denotes the current iteration, $p_{\boldsymbol{\theta}}$ is short for $p_{\pi_{\boldsymbol{\theta}}}$, and $(\alpha_t)_{t=1}^{\infty}$ is a sequence of positive step sizes. In practice, $\nabla_{\boldsymbol{\theta}} J$ is not available, but can be estimated from a batch of trajectories $\mathcal{D}_N = \{\tau_1, \ldots, \tau_N\}$. The GPOMDP (Baxter and Bartlett, 2001) algorithm (a refinement of REINFORCE, Williams, 1992) provides an unbiased gradient estimator:

$$\begin{aligned}\widehat{\nabla}_{\boldsymbol{\theta}}^N J(\boldsymbol{\theta}) = &\frac{1}{N} \sum_{n=1}^{N} \sum_{h=0}^{H-1} \left(\sum_{i=0}^{h} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_i^n | s_i^n) \right)\\ &\cdot \left(\gamma^h \mathcal{R}(s_h^n, a_h^n) - b\right),\end{aligned} \tag{2}$$

where b is a baseline used to reduce variance. Any baseline that does not depend on actions preserves the unbiasedness of the estimator.[2] We employ the variance-minimizing baselines provided by Peters and Schaal (2008).

A widely used (Duan et al., 2016) policy class is the Gaussian[3]:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi}\sigma_\omega} \exp\left\{ -\frac{1}{2} \left(\frac{a - \mu_{\boldsymbol{\upsilon}}(s)}{\sigma_\omega} \right)^2 \right\}, \tag{3}$$

also denoted with $\mathcal{N}(a|\mu_{\boldsymbol{\upsilon}}(s), \sigma_\omega^2)$, where the action space is $\mathcal{A} = \mathbb{R}$, $\mu_{\boldsymbol{\upsilon}}$ is the state-dependent mean and $\sigma_\omega^2 > 0$ is the variance (σ_ω is the standard deviation). The policy parameters consist of a vector of mean parameters $\boldsymbol{\upsilon} \in \Upsilon \subseteq \mathbb{R}^m$ and a variance parameter $\omega \in \Omega \subseteq \mathbb{R}$, i.e., $\boldsymbol{\theta} \equiv [\boldsymbol{\upsilon}^T | \omega]^T$ and $\Theta \equiv \Upsilon \times \Omega \subseteq \mathbb{R}^{m+1}$. We focus on the following, common (Rajeswaran et al., 2017) parametrization:

$$\mu_{\boldsymbol{\upsilon}}(s) = \boldsymbol{\upsilon}^T \boldsymbol{\phi}(s), \qquad \sigma_\omega = e^\omega, \tag{4}$$

where $\boldsymbol{\phi}(\cdot)$ is a vector of m state-features[4]. We also assume that the state features are bounded in Euclidean norm, i.e., $\sup_{s \in \mathcal{S}} \|\boldsymbol{\phi}(s)\| \leqslant \varphi$, and both Υ and Ω are convex sets.

[2]Some action-dependent baselines are also possible. See (Tucker et al., 2018) for a discussion.

[3]We consider scalar actions for simplicity. Multi-dimensional actions are discussed in Appendix D, together with heteroskedastic exploration.

[4]This is a shallow policy since we assume to have the features already available, as opposed to a deep policy, where the features are learned together with $\boldsymbol{\upsilon}$ in an end-to-end fashion. Generalizations to deep neural policies are plausible, but beyond the scope of this paper.

Entropy regularization (Schulman et al., 2017) consists in modifying the reward to favor policy stochasticity:

$$\widetilde{r}_t = (1 - \tau) r_t + \tau \mathbb{H}\left(\pi(\cdot|s_t)\right), \tag{5}$$

where $\mathbb{H}\left(\pi(\cdot|s_t)\right) = \mathbb{E}_{a \sim \pi(\cdot|s_t)}\left[-\log \pi(a|s_t)\right]$ is the entropy of the action distribution at state s_t and $\tau \in [0, 1)$ is a regularization coefficient. In the case of Gaussian policies, the entropy bonus can be computed in closed form and the GPOMDP algorithm can be easily modified to account for the new reward function (e.g., Ahmed et al., 2019). Vanilla policy gradient is recovered for $\tau = 0$.

2.2 Safe Policy Gradients

When learning on-line in the real world, we would like to avoid oscillations in the performance $J(\boldsymbol{\theta})$, as large deviations from previously observed behavior may compromise the safety of the system or deemed unacceptable by stakeholders. A convenient way to enforce this desideratum is through an *improvement constraint* (Thomas et al., 2015):

Definition 2.1. Given a parametric policy $\pi_{\boldsymbol{\theta}}$ with current parameter $\boldsymbol{\theta}_t$, we say that update $\Delta\boldsymbol{\theta} \in \mathbb{R}^{m+1}$ is *safe* w.r.t. requirement $C_t \in \mathbb{R}$ if:

$$J(\boldsymbol{\theta}_{t+1}) - J(\boldsymbol{\theta}_t) \geqslant C_t, \tag{6}$$

where $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}$.

We talk of a *required performance improvement* when C_t is non-negative, otherwise of a *bounded worsening.* The case $C_t \equiv 0$ corresponds to the well-studied *monotonic improvement* constraint (Kakade and Langford, 2002; Pirotta et al., 2013; Papini et al., 2017). This constraint can also be used to enforce an absolute performance threshold $J_{\min}$, by setting $C_t = J_{\min} - J(\boldsymbol{\theta}_t)$. For instance, if we want to guarantee that the performance is never worse than that of the initial policy, $J(\boldsymbol{\theta}_0)$, we set $C_t = J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}_t)$. This can be useful in safety-critical systems where the initial policy is designed to be safe, assuming the performance measure captures all sources of risk (García and Fernández, 2015; Amodei et al., 2016). Recent work (Papini et al., 2019) provides improvement guarantees for a general family of policies. Given positive constants ψ, κ and ξ, a policy class Π_Θ is called (ψ, κ, ξ)-smoothing if:

$$\begin{aligned}&\sup_{s \in \mathcal{S}} \mathbb{E}_{a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)} \Big[\left\|\nabla \log \pi_{\boldsymbol{\theta}}(a|s)\right\| \Big] \leqslant \psi,\\ &\sup_{s \in \mathcal{S}} \mathbb{E}_{a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)} \Big[\left\|\nabla \log \pi_{\boldsymbol{\theta}}(a|s)\right\|^2 \Big] \leqslant \kappa,\\ &\sup_{s \in \mathcal{S}} \mathbb{E}_{a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)} \Big[\left\|\nabla\nabla^T \log \pi_{\boldsymbol{\theta}}(a|s)\right\| \Big] \leqslant \xi,\end{aligned} \tag{7}$$

for all $\boldsymbol{\theta} \in \Theta$, where $\|\cdot\|$ denotes the Euclidean norm for vectors and the spectral norm for matrices. In particular, Gaussian policies *with fixed standard deviation* (constant ω) are (ψ, κ, ξ)-smoothing with the following constants (Papini et al., 2019):

$$\psi = \frac{2\varphi}{\sqrt{2\pi}\sigma_\omega}, \qquad \kappa = \xi = \frac{\varphi^2}{\sigma_\omega^2}, \tag{8}$$

where φ is the Euclidean-norm bound on state features. For a (ψ, κ, ξ)-smoothing policy, the performance improvement yielded by a policy gradient update can be lower-bounded by a function of the step size α as follows (Theorem 9 from Papini et al., 2019):

$$J(\boldsymbol{\theta}_{t+1}) - J(\boldsymbol{\theta}_t) \geqslant \alpha \left\|\nabla J(\boldsymbol{\theta}_t)\right\|^2 - \alpha^2 \frac{L}{2} \left\|\Delta\boldsymbol{\theta}\right\|^2, \tag{9}$$

where $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\nabla J(\boldsymbol{\theta}_t)$ and $L = \frac{R_{\max}}{(1-\gamma)^2}\left(\frac{2\gamma\psi^2}{1-\gamma} + \kappa + \xi\right)$. This allows to select a *safe step size* for improvement constraint C_t, i.e., a step size for which (6) is satisfied by the policy gradient update (1). In this paper, whenever multiple choices of the step size are safe, we decide to employ the *largest* safe step size. This is meant to yield faster convergence (see Section 5 for an empirical substantiation of this claim). In the fixed-variance Gaussian case, we can obtain a safe step size for the mean-parameter update (adaptation of Corollary 10 from Papini et al., 2019):

Lemma 2.1. *Let Π_Υ be the class of Gaussian policies parametrized as in (4), but with* fixed variance parameter *ω. Let $\boldsymbol{\upsilon}_t \in \mathbb{R}^m$ and $\boldsymbol{\upsilon}_{t+1} = \boldsymbol{\upsilon}_t + \alpha_t\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t, \omega)$. For any $C_t \leqslant C_t^*$, the largest step size guaranteeing $J(\boldsymbol{\upsilon}_{t+1}, \omega) - J(\boldsymbol{\upsilon}_t, \omega) \geqslant C_t$ is:*

$$\overline{\alpha}_t := \frac{\sigma_\omega^2}{F}\left(1 + \sqrt{1 - \frac{C_t}{C_t^*}}\right), \tag{10}$$

where $F = \frac{2\varphi^2 R_{\max}}{(1-\gamma)^2}\left(1 + \frac{2\gamma}{\pi(1-\gamma)}\right)$ and $C_t^* = \frac{\sigma_\omega^2\|\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t,\omega)\|^2}{2F}$. □

We have highlighted the role of the policy standard deviation. We can see how a larger σ_ω allows to take larger steps. Moreover, it increases the maximum improvement guarantee that one *can ask for* (although C_t can be selected by the user at each step, its highest feasible value C_t^* depends on the current policy variance and gradient norm). In fact, both $\overline{\alpha}_t$ and C_t^* are $\mathcal{O}(\sigma_\omega^2)$. This is due to the smoothing effect of the policy variance on the optimization landscape, in accordance with the empirical analysis from (Ahmed et al., 2019). In practice, C_t^* is too small to be of any relevance, so we are more interested in the cases when $C_t \leqslant 0$.

3 Adaptive Exploration

In this section, we use some insights from the safe PG literature to devise a heuristic approach to adapt the standard deviation of a Gaussian policy during the learning process. Our desiderata are fast convergence, avoiding instabilities and not getting stuck in local optima. The algorithm we present here is heuristic. A variant with formal improvement guarantees is presented in the next section.

Consider a Gaussian policy $\pi_{\boldsymbol{\upsilon},\omega}(a|s) = \mathcal{N}(a|\mu_{\boldsymbol{\upsilon}}(s), \sigma_\omega)$, parametrized as in (4). As mentioned above, it is common to learn the policy variance parameter via gradient ascent just like any other parameter, i.e., $\omega_{t+1} \leftarrow \omega_t + \beta_t\nabla_\omega J(\boldsymbol{\upsilon}_t, \omega_t)$. However, the effects of σ on the optimization landscape, exposed by Lemma 2.1, suggest to treat it with particular care, both to exploit its potential and to avoid its possible risks. In fact, adjusting the policy variance with policy gradient tends to degenerate too early into quasi-deterministic policies, getting stuck in local optima or even causing divergence issues (see Section 5). We use our understanding of the special nature of this parameter to modify GPOMDP in two ways. First of all, we make the step size *for updating the mean parameters* dependent on the policy variance, like the safe step size from Lemma 2.1. In particular, we use the following:

$$\alpha_t = \frac{\alpha\sigma_{\omega_t}^2}{\|\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t, \omega_t)\|}, \tag{11}$$

to update the mean parameters $\boldsymbol{\upsilon}$, where $\alpha > 0$ is a hyper-parameter. This has both the effect of reducing the step size when a small σ makes the optimization landscape less smooth, preventing oscillations, and increasing it when a large σ allows it to do so, increasing the learning speed. This is not entirely unheard of, as it is exactly what a natural gradient (Kakade, 2001; Amari, 1998) would do in a *pure* Gaussian setting ($1/\sigma^2$ is the Fisher information w.r.t. the mean parameters of a Gaussian distribution, see Sehnke et al., 2008; Miyamae et al., 2010). We also divide the step size by the norm of the gradient. This is a common normalization technique (Peters et al., 2005), and is further motivated by the results of Section 4.2 on stochastic gradient updates.

As for the variance parameter ω, we treat it as a separate *meta-parameter* and we learn it in a meta-gradient fashion (Sutton, 1992; Schraudolph, 1999; Veeriah et al., 2017; Xu et al., 2018). Specifically, we employ a more far-sighted learning objective to avoid premature convergence to deterministic behavior. To do so, we look at the target performance one step in the future:

$$J\left(\boldsymbol{\upsilon}_t + \alpha\sigma_{\omega_t}^2\frac{\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t, \omega_t)}{\|\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t, \omega_t)\|}, \omega_t\right)$$
$$\simeq J(\boldsymbol{\upsilon}_t, \omega_t) + \alpha\sigma_{\omega_t}^2\left\|\nabla_{\boldsymbol{\upsilon}}J(\boldsymbol{\upsilon}_t, \omega_t)\right\| := \mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t), \tag{12}$$

where we performed a first-order approximation. The

Algorithm 1 MEPG

```
1: Input: Initial parameters $\boldsymbol{\upsilon}_0$ and $\omega_0$, step size $\alpha > 0$, meta step size $\eta > 0$, batch size $N$
2: for $t = 1, 2, \dots$ do
3:     Collect a batch of N trajectories with $\pi_{\boldsymbol{\upsilon}_t,\omega_t}$
4:     Estimate $\widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)$, $\widehat{\nabla}_{\omega} J(\boldsymbol{\upsilon}_t, \omega_t)$ and $\widehat{\nabla}_{\omega} \|\nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\|$        ▷ See Appendix B
5:     $\widehat{\nabla}_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t) = \widehat{\nabla}_{\omega} J(\boldsymbol{\upsilon}_t, \omega_t) + \alpha e^{2\omega_t}\left(2\left\|\widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\right\| + \widehat{\nabla}_{\omega}\|\nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\|\right)$
6:     $\omega_{t+1} \leftarrow \omega_t + \eta\widehat{\nabla}_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t) / \left\|\widehat{\nabla}_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t)\right\|$
7:     $\boldsymbol{\upsilon}_{t+1} \leftarrow \boldsymbol{\upsilon}_t + \alpha e^{2\omega_t}\widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t) / \left\|\widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\right\|$
8: end for
```

gradient of $\mathcal{L}$ w.r.t. ω is:

$$\nabla_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t) = \nabla_{\omega} J(\boldsymbol{\upsilon}_t, \omega_t) + 2\alpha\sigma^2_{\omega_t}\|\nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\| + \alpha\sigma^2_{\omega_t}\nabla_{\omega}\|\nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\|. \tag{13}$$

The first term of the sum is the usual policy gradient w.r.t. ω, and accounts for the immediate effect of policy stochasticity on performance. The role of the second term is to increase the step size α_t, more so if the gradient w.r.t. $\boldsymbol{\upsilon}$ is large. The third term is meant to modify the policy variance to increase the gradient norm and can be seen as a way to escape local optima. The last two terms, together, account for the long-term effects of modifying the policy variance. We propose to update ω in the direction of the (normalized) meta-gradient $\nabla_{\omega}\mathcal{L}$ using a meta-step size $\eta > 0$:

$$\omega_{t+1} \leftarrow \omega_t + \eta\frac{\nabla_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t)}{\|\nabla_{\omega}\mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t)\|}. \tag{14}$$

In practice, exact gradients are not available. The policy gradient for the mean-parameter update can be estimated with GPOMDP (2). Computing $\widehat{\nabla}_{\omega}\mathcal{L}$ also requires estimating $\nabla_{\omega}\|\nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)\|$, which is computationally no more expensive, but could suffer from more variance (See Appendix B). The pseudocode for the resulting algorithm, called Meta-Exploring Policy Gradient (MEPG), is provided in Algorithm 1.

4 Stable Exploration

In this section, we extend the performance improvement guarantees reported in Section 2.2 to Gaussian policies with adaptive variance and we use these theoretical results to devise a variant of Algorithm 1 with improvement guarantees.

4.1 Exact framework

Existing guarantees for fixed-variance Gaussian policies are based on the smoothing constants from (Papini et al., 2019), reported in (8). These depend (inversely) on σ^2_{ω}, hence are no longer constant once we allow ω to vary. In particular, they tend to infinity as the policy approaches determinism. Unfortunately, this is enough to invalidate the safety guarantees. A workaround would be to replace σ_{ω} with a lower bound, which can be imposed by constraining the parameter space Ω or by changing the parametrization. However, this would make the improvement bounds unnecessarily conservative, and would prevent the agent to converge to deterministic behavior in the end. For these reasons, we instead propose to update the mean and variance parameters *alternately.* First, we show that Gaussian policies are smoothing w.r.t. the variance parameter *alone*:

Lemma 4.1. *Let Π_{Ω} be the class of Gaussian policies parametrized as in* (4)*, but with* fixed mean parameter $\boldsymbol{\upsilon}$. *Π_{Ω} is $\left(\frac{4}{\sqrt{2\pi e}}, 2, 2\right)$-smoothing.* □

This allows to devise a safe policy-gradient update for the variance parameters:

Theorem 4.2. *Let Π_{Ω} be the class of policies defined in Lemma 4.1. Let $\omega_t \in \Omega$ and $\omega_{t+1} \leftarrow \omega_t + \beta_t\nabla_{\omega} J(\boldsymbol{\upsilon}, \omega_t)$. For any $C_t \leqslant C_t^*$, the largest step-size satisfying* (6) *is:*

$$\overline{\beta}_t = \frac{1}{G}\left(1 + \sqrt{1 - \frac{C_t}{C_t^*}}\right), \tag{15}$$

where $C_t^* = \frac{\|\nabla_{\omega} J(\boldsymbol{\upsilon},\omega_t)\|^2}{2G}$ *and* $G = \frac{4R_{\max}}{(1-\gamma)^2}\left(1 + \frac{4\gamma}{\pi e(1-\gamma)}\right)$. □

Similarly to the mean parameters case, $\beta_t = \frac{1}{G}$ is the greedy-safe step size and $\beta_t = \frac{2}{G}$ is the largest step size guaranteeing monotonic improvement.

Alternately updating the mean parameter as in Lemma 2.1 and the variance parameter as in Theorem 4.2 ensures $J(\boldsymbol{\upsilon}_{t+1}, \omega_{t+1}) - J(\boldsymbol{\upsilon}_t, \omega_t) \geqslant C_t$ for all t.

However, Theorem 4.2 still pertains *naïve* variance updates, which suffer from all the problems discussed in Section 3. The next question is how to optimize the surrogate exploratory objective $\mathcal{L}$ from (12) while satisfying the original constraint (6) on the performance objective J. The following Theorem provides a safe

Algorithm 2 SEPG

```
1: Input: Initial parameters $\boldsymbol{\upsilon}_0$ and $\omega_0$, batch size $N$, improvement thresholds $\{C_t\}_{t=1}^{\infty}$, confidence parameter
   $\delta$, discount factor $\gamma$, maximum reward $R_{\max}$, feature bound $\varphi$
2: for $t = 1, 2, \dots$ do
3:     Collect a batch of N trajectories with $\pi_{\boldsymbol{\upsilon}_t,\omega_t}$
4:     Estimate $\widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)$,
5:     if $t$ is odd then
6:         $\boldsymbol{\upsilon}_{t+1} = \boldsymbol{\upsilon}_t + \widetilde{\alpha}_t \nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t)$        ▷ Safe step size from Section 4.2
7:     else
8:         estimate $\widehat{\nabla}_{\omega} J(\boldsymbol{\upsilon}_t, \omega_t)$ and $\widehat{\nabla}_{\omega} \left\| \widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t) \right\|$        ▷ See Appendix B
9:         $\widehat{\nabla}_{\omega} \mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t) = \widehat{\nabla}_{\omega} J(\boldsymbol{\upsilon}_t, \omega_t) + \widehat{\alpha}_t \left(2 \left\| \widehat{\nabla}_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t) \right\| + \widehat{\nabla}_{\omega} \left\| \nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t) \right\| \right)$
10:        $\omega_{t+1} = \omega_t + \widetilde{\eta}_t \nabla_{\omega} \mathcal{L}(\boldsymbol{\upsilon}_t, \omega_t)$        ▷ Safe meta-step size from Section 4.2
11:    end if
12: end for
```

update for a smoothing policy in the direction of a generic update vector $\boldsymbol{x}_t$ ($\nabla_\omega \mathcal{L}$ in MEPG):

Theorem 4.3. *Let Π_Θ be a (ψ, κ, ξ)-smoothing policy class, $\boldsymbol{\theta}_t \in \Theta$, and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \boldsymbol{x}_t$, where $\boldsymbol{x}_t \in \mathbb{R}^m$ and $\eta_t \in \mathbb{R}$ is a (possibly negative) step size. Let $\lambda_t := \frac{\langle \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t), \boldsymbol{x}_t \rangle}{\|\boldsymbol{x}_t\|}$ be the scalar projection of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$ onto $\boldsymbol{x}_t$. For any $C_t \leqslant C_t^*$, provided $\lambda_t \neq 0$, the largest step size guaranteeing $J(\boldsymbol{\theta}_{t+1}) - J(\boldsymbol{\theta}_t) \geqslant C_t$ is:*

$$\overline{\eta}_t = \frac{|\lambda_t|}{L \|\boldsymbol{x}_t\|} \left(sign(\lambda_t) + \sqrt{1 - \frac{C_t}{C_t^*}} \right), \tag{16}$$

where $C_t^* = \frac{\lambda_t^2}{2L}$ and $L = \frac{R_{\max}}{(1-\gamma)^2} \left(\frac{2\gamma\psi^2}{1-\gamma} + \kappa + \xi \right)$. □

Note that a positive performance improvement up to C_t^* can always be guaranteed, even if the improvement direction $\nabla_{\boldsymbol{\theta}} J$ is not explicitly followed. However, when the scalar projection λ_t is negative, the largest safe step size is negative. This corresponds to the case in which maximizing the surrogate objective *reduces* the original one. For positive values of C_t (required improvement), there may be no way to safely pursue the surrogate objective. In this case, a negative step size is prescribed to follow the direction of $\nabla_{\boldsymbol{\theta}} J$ instead[5]. We can use the step size $\overline{\eta}_t$ from Theorem 4.3 to safely replace $\nabla_\omega J$ with the meta gradient $\nabla_\omega \mathcal{L}$ from (13) in the variance update:

$$\boldsymbol{\upsilon}_{t+1} \leftarrow \boldsymbol{\upsilon}_t + \overline{\alpha}_t \nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}_t, \omega_t), \tag{17}$$

$$\omega_{t+2} \leftarrow \omega_{t+1} + \overline{\eta}_{t+1} \nabla_\omega \mathcal{L}(\boldsymbol{\upsilon}_{t+1}, \omega_{t+1}), \tag{18}$$

where $\omega_{t+1} \equiv \omega_t$ and $\boldsymbol{\upsilon}_{t+2} \equiv \boldsymbol{\upsilon}_{t+1}$, as the two set of parameters cannot be safely updated together.

[5] The special case when the two gradients are orthogonal ($\lambda_t = 0$) is discussed in Appendix A.

4.2 Approximate framework

In practice, exact gradients are not available and must be estimated from data. In this section, we show how to adapt the safe step sizes from Section 4 to take gradient estimation errors into account. Let $\widehat{\nabla}_{\boldsymbol{\upsilon}}^N J$, $\widehat{\nabla}_{\omega}^N J$ and $\widehat{\nabla}_{\omega}^N \mathcal{L}$ be unbiased estimators of $\nabla_{\boldsymbol{\upsilon}} J$, $\nabla_{\boldsymbol{\upsilon}} J$ and $\nabla_\omega \mathcal{L}$, respectively, each using a batch of N trajectories. As for MEPG, the first two can be GPOMDP estimators (2) and meta-gradient estimation is discussed in Appendix B. We make the following assumption on the gradient estimators[6]:

Assumption 4.4. *For every $\delta \in (0,1)$ there exists a non-negative constant ϵ_δ such that, with probability at least $1 - \delta$:*

$$\left\| \nabla_{\boldsymbol{\upsilon}} J(\boldsymbol{\upsilon}, \omega) - \widehat{\nabla}_{\boldsymbol{\upsilon}}^N J(\boldsymbol{\upsilon}, \omega) \right\| \leqslant \frac{\epsilon_\delta}{\sqrt{N}},$$

$$\left\| \nabla_{\omega} J(\boldsymbol{\upsilon}, \omega) - \widehat{\nabla}_{\omega}^N J(\boldsymbol{\upsilon}, \omega) \right\| \leqslant \frac{\epsilon_\delta}{\sqrt{N}},$$

for every $\boldsymbol{\upsilon} \in \Upsilon$, $\omega \in \Omega$ and $N \geqslant 1$. □

Here ϵ_δ represents an upper bound on the gradient estimation error. This can be characterized using various statistical inequalities (Papini et al., 2017). A possible one, based on ellipsoidal confidence regions, is described in Appendix C. Under Assumption 4.4, for a sufficiently large batch size, the safe step size for the mean update can be adjusted as follows:

$$\widetilde{\alpha}_t = \frac{\sigma_\omega^2 \left(\left\| \widehat{\nabla}_{\boldsymbol{\upsilon}}^N J(\boldsymbol{\upsilon}_t, \omega_t) \right\| - \frac{\epsilon_\delta}{\sqrt{N}} \right)}{F \left\| \widehat{\nabla}_{\boldsymbol{\upsilon}}^N J(\boldsymbol{\upsilon}_t, \omega_t) \right\|} \left(1 + \sqrt{1 - \frac{C_t}{C_t^*}} \right),$$

where $C_t^* = \frac{\sigma_{\omega_t}^2 \left(\left\| \widehat{\nabla}_{\boldsymbol{\upsilon}}^N J(\boldsymbol{\upsilon}_t, \omega_t) \right\| - \frac{\epsilon_\delta}{\sqrt{N}} \right)^2}{2F}$ and F is from Lemma 2.1. Similarly, the safe step size for the vari-

[6] We do not need a similar assumption on the meta-gradient estimator $\widehat{\nabla}_\omega \mathcal{L}$, since our improvement requirements are always on the performance J (see Appendix A.2).

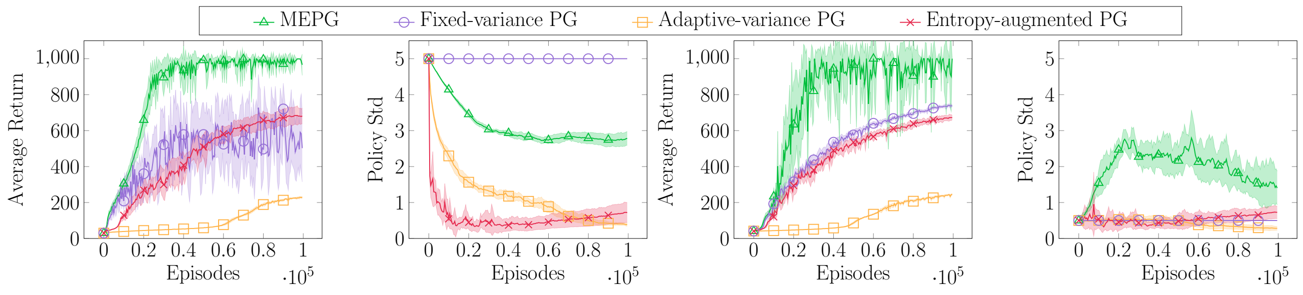

Figure 1: Average return (undiscounted) and policy standard deviation per episode of MEPG, fixed-variance PG, adaptive-variance PG and entropy-augmented PG on the continuous Cart-Pole task, starting from $\sigma = 5$ (left) and $\sigma = 0.5$ (right); averaged over 10 independent runs with 95% Student's t-confidence intervals.

ance update can be adjusted as follows:

$$\widetilde{\eta}_t = \frac{\left|\widehat{\lambda}_t\right| - \frac{\epsilon_\delta}{\sqrt{N}}}{G\left\|\widehat{\nabla}^N_\omega \mathcal{L}(\boldsymbol{v}_t, \omega_t)\right\|}\left(sign\left(\widehat{\lambda}_t\right) + \sqrt{1 - \frac{C_t}{C_t^*}}\right),$$

where $\widehat{\lambda}_t$ is the scalar projection of $\widehat{\nabla}^N_\omega J(\boldsymbol{v}_t, \omega_t)$ onto $\widehat{\nabla}^N_\omega \mathcal{L}(\boldsymbol{v}_t, \omega_t)$, $C_t^* = \frac{\left(\left|\widehat{\lambda}_t\right| - \frac{\epsilon_\delta}{\sqrt{N}}\right)^2}{2G}$ and G is from Theorem 4.2. We call the variant of MEPG that alternates mean and variance updates using these step sizes Stably Exploring Policy Gradient (SEPG), detailed in Algorithm 2. The SEPG algorithm satisfies our bounded-worsening constraint (6) with high probability:

Theorem 4.5. *Under Assumption 4.4, provided $C_t \leqslant 0$ and $N > \epsilon_\delta^2/\widehat{\lambda}_t^2$, Algorithm 2 guarantees $J(\boldsymbol{\theta}_{t+1}) - J(\boldsymbol{\theta}_t) \geqslant C_t$ with probability at least $1 - \delta$ for any $t \geqslant 1$.* □

In Appendix F we prove a stronger version of this theorem that also allows strictly positive improvements. The requirement on the batch size ensures that estimation errors are smaller than the estimates themselves. If this requirement is not satisfied, we can either collect more samples or terminate. This typically happens close to stationary points anyway. The step sizes proposed by SEPG may be excessively conservative. This is similar to what happens in supervised learning, where the convergence-guaranteeing step sizes are rarely used in practice, typically replaced by heuristics (Kingma and Ba, 2015). However, since policy updates in online RL can have concrete consequences, we will use the prescribed step sizes in our experiments, leaving the possibility of explicitly relaxing the safety requirements.

5 Experiments

In this section, we test the proposed methods on simulated continuous control tasks. More details on the simulation environments are provided in Appendix E.

MEPG We test MEPG on a continuous-action version of the Cart-Pole balancing task (Barto et al., 1983). Figure 1 shows the performance (1000 is the maximum) and the policy standard deviation of MEPG and three versions of PG (with the GPOMDP gradient estimator). In fixed-variance PG, the policy variance parameter is kept constant. In adaptive-variance PG, it is learned via gradient ascent as any other parameter. Entropy-augmented PG is the same, but with entropy regularization (5). For each algorithm, the best hyper-parameters (step sizes and entropy coefficient τ) have been selected by grid search (see Appendix F). Two very different initializations are considered for the standard deviation: $\sigma_{\omega_0} = 5$ (on the left of Figure 1) and $\sigma_{\omega_0} = 0.5$ (on the right). As shown by the behavior of fixed-variance GPOMDP, the former constant value is too large to achieve optimal performance at convergence, while the latter is too small to properly explore the environment. As expected, adaptive-variance GPOMDP is too greedy and ends up always reducing the standard deviation. Besides preventing exploration, divergence issues force us to use a smaller step size ($\alpha = 0.01$ instead of 0.1), resulting in slower learning. This problem is fixed by the entropy bonus, which prevents the policy from becoming deterministic and allows to use the larger learning rate ($\alpha = 0.1$). However, entropy-augmented PG does not perform significantly better than its fixed-variance counterpart on this task, as the amount of exploration needed to find the global optimum is not maintained (or is pursued too late). Instead, MEPG is able to settle on an intermediate value with both variance initializations. This allows both to learn faster and to achieve optimal performance, although non-negligible oscillations can be observed. This oscillations are partly due to the variance of the meta-gradient estimator, and can be mitigated by the conservative step sizes prescribed by SEPG.

SEPG Figure 2 shows the performance and the policy standard deviation of SEPG on the one-

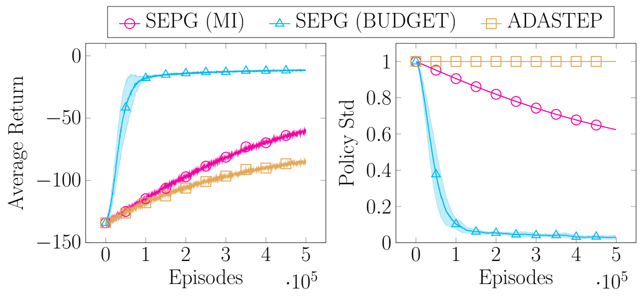

Figure 2: Average return (undiscounted) and standard deviation per episode of SEPG and ADASTEP on the LQG task, averaged over 10 independent runs with 95% Student's t-confidence intervals.

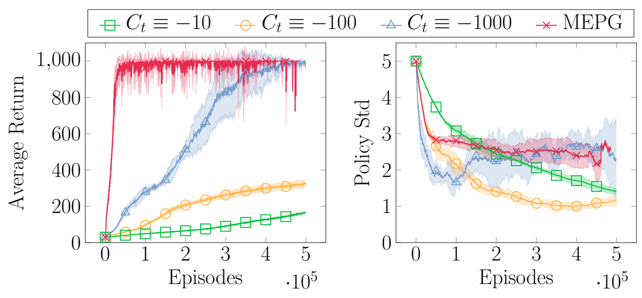

Figure 3: Average return (undiscounted) and standard deviation per episode of SEPG on the Cart-Pole task for different values of C_t, averaged over 5 runs with 95% confidence intervals. The learning curve of MEPG is reported as a reference.

dimensional LQG (Linear-Quadratic Gaussian regulator) task (Peters and Schaal, 2008). SEPG with a monotonic improvement constraint ($C_t \equiv 0$) is compared with the adaptive-step-size algorithm (ADASTEP in the figure) by Pirotta et al. (2013). Starting from $\sigma_{\omega_0} = 1$, SEPG achieves higher returns by safely lowering it, while ADASTEP has no way to safely update this parameter. Both algorithms use $\delta = 0.2$ and a large batch size ($N = 500$). We also consider a looser constraint, already discussed in Section 2.2 (BUDGET in the figure): that of never doing worse than the initial performance ($C_t = J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}_t)$). As expected, this allows faster learning, leading to optimal performance within a reasonable time.

On the Cart-Pole task, MEPG showed an oscillatory behavior. Motivated by this fact, we run SEPG with a fixed, *negative* improvement threshold C_t. Recall that the meaning of such a constraint is to limit per-update performance worsening. Figure 3 shows the results for different values of the threshold, starting from $\sigma_{\omega_0} = 5$ and neglecting the gradient estimation error (i.e., by setting $\delta = 1$). Even under this simplifying assumption, only a very large value of C_t allows to reach optimal performance within a reasonable time. This is due to the over-conservativeness of the step sizes proposed by SEPG, as already discussed. Note how oscillations are reduced w.r.t. MEPG, and how policy standard deviation is first reduced and then *safely increased* again.

6 Discussion and Future Work

We have highlighted the special role of stochasticity in PG with Gaussian policies, complementing the empirical observations from (Ahmed et al., 2019) with theoretical insights from the Safe PG literature (Papini et al., 2019). We have proposed a variant of GPOMDP for this setting, called Meta-Exploring Policy Gradient (MEPG), which is able to adapt the variance parameter in a more far-sighted way than vanilla or entropy-augmented gradient ascent. We have empirically shown the effectiveness of this approach on the Cart-Pole balancing task, where the entropy bonus is able to prevent divergence but not to escape local optima. This should be intended as a proof of concept: entropy augmentation is still a natural choice for most applications due to its simplicity. Future work on MEPG algorithms should study its applicability to larger control problems and overcome its potential bottlenecks, such as the second-order term in (12).

Furthermore, we have generalized the existing performance-improvement bounds for Gaussian policies to the adaptive-variance case and proposed SEPG, a variant of MEPG with guarantees of stable improvement. Experiments confirmed several intuitions provided by the theory. Unfortunately, learning speed is heavily degraded for meaningful values of the improvement requirement C_t, due to over-conservative step sizes. The desired balance between speed and stability can still be achieved by hand-tuning it as a hyper-parameter. Replacing the theoretically sound upper bounds with estimates (Allen-Zhu, 2018) could bridge the gap between theory and practice. Another possible improvement is to employ an adaptive batch size as proposed in (Papini et al., 2017, 2019). Future work should also study in more depth the issue of local optima. Recent work (Agarwal et al., 2019) shows how (natural) PG can achieve global optimality in some cases, and highlights the importance of exploration on this matter. Moreover, adaptively changing the policy variance as in MEPG can be seen as an online version of graduated optimization (Hazan et al., 2016), a popular technique for finding the global optimum of a nonconvex function. Finally, further aspects of *safe exploration* (Hans et al., 2008; Turchetta et al., 2016; Cohen et al., 2018) should be considered for the online selection of policy stochasticity in real-world control tasks.

Acknowledgements

The study was partially funded by Lombardy Region (Announcement PORFESR 2014-2020).

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. (2019). Optimality and approximation with policy gradient methods in markov decision processes. *CoRR*, abs/1908.00261.

Ahmed, Z., Roux, N. L., Norouzi, M., and Schuurmans, D. (2019). Understanding the impact of entropy on policy optimization. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 151–160. PMLR.

Allen-Zhu, Z. (2018). Natasha 2: Faster non-convex optimization than SGD. In *NeurIPS*, pages 2680–2691.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Systems, Man, and Cybernetics*, 13(5):834–846.

Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res.*, 15:319–350.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *NeurIPS*, pages 1471–1479.

Brafman, R. I. and Tennenholtz, M. (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*, abs/1606.01540.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Cohen, A., Yu, L., and Wright, R. (2018). Diverse exploration for fast and safe policy improvement. In *AAAI*, pages 2876–2883. AAAI Press.

Dann, C. and Brunskill, E. (2015). Sample complexity of episodic fixed-horizon reinforcement learning. In *NeurIPS*, pages 2818–2826.

Dann, C., Lattimore, T., and Brunskill, E. (2017). Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. In *NeurIPS*, pages 5713–5723.

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

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1329–1338. JMLR.org.

García, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR.

Hans, A., Schneegaß, D., Schäfer, A. M., and Udluft, S. (2008). Safe exploration for reinforcement learning. In *ESANN*, pages 143–148.

Härdle, W. and Simar, L. (2012). *Applied multivariate statistical analysis*, volume 22007. Springer.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. (2016). On graduated optimization for stochastic non-convex problems. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1833–1841. JMLR.org.

Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is q-learning provably efficient? In *NeurIPS*, pages 4868–4878.

Kakade, S. M. (2001). A natural policy gradient. In *NeurIPS*, pages 1531–1538. MIT Press.

Kakade, S. M. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*, pages 267–274. Morgan Kaufmann.

Kearns, M. J. and Singh, S. P. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Lakshmanan, K., Ortner, R., and Ryabko, D. (2015). Improved regret bounds for undiscounted continuous reinforcement learning. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 524–532. JMLR.org.

Lanczos, C. (1950). *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA.

Lattimore, T. and Hutter, M. (2014). Near-optimal PAC bounds for discounted mdps. *Theor. Comput. Sci.*, 558:125–143.

Lattimore, T. and Szepesvári, C. (2019). *Bandit Algorithms*. Cambridge University Press (preprint).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

Miyamae, A., Nagata, Y., Ono, I., and Kobayashi, S. (2010). Natural policy gradient methods with parameter-based exploration for control tasks. In *NeurIPS*, pages 1660–1668. Curran Associates, Inc.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nachum, O., Norouzi, M., Tucker, G., and Schuurmans, D. (2018). Smoothed action value functions for learning gaussian policies. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3689–3697. PMLR.

Ok, J., Proutière, A., and Tranos, D. (2018). Exploration in structured reinforcement learning. In *NeurIPS*, pages 8888–8896.

OpenAI (2018). Openai five. `https://blog.openai.com/openai-five/`.

Ortner, R. and Ryabko, D. (2012). Online regret bounds for undiscounted continuous reinforcement learning. In *NeurIPS*, pages 1772–1780.

Papini, M., Pirotta, M., and Restelli, M. (2017). Adaptive batch size for safe policy gradients. In *NeurIPS*, pages 3591–3600.

Papini, M., Pirotta, M., and Restelli, M. (2019). Smoothing policies and safe policy gradients. *CoRR*, abs/1905.03231.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697.

Peters, J., Vijayakumar, S., and Schaal, S. (2005). Natural actor-critic. In *ECML*, volume 3720 of *Lecture Notes in Computer Science*, pages 280–291. Springer.

Pirotta, M., Restelli, M., and Bascetta, L. (2013). Adaptive step-size for policy gradient methods. In *NeurIPS*, pages 1394–1402.

Pirotta, M., Restelli, M., and Bascetta, L. (2015). Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2-3):255–283.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S. M. (2017). Towards generalization and simplicity in continuous control. In *NeurIPS*, pages 6550–6561.

Schraudolph, N. N. (1999). Local gain adaptation in stochastic gradient descent.

Schulman, J., Abbeel, P., and Chen, X. (2017). Equivalence between policy gradients and soft q-learning. *CoRR*, abs/1704.06440.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2008). Policy gradients with parameter-based exploration for control. In *ICANN (1)*, volume 5163 of *Lecture Notes in Computer Science*, pages 387–396. Springer.

Shani, L., Efroni, Y., and Mannor, S. (2018). Revisiting exploration-conscious reinforcement learning. *CoRR*, abs/1812.05551.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. (2014). Deterministic policy gradient algorithms. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org.

Strehl, A. L., Li, L., and Littman, M. L. (2009). Reinforcement learning in finite mdps: PAC analysis. *J. Mach. Learn. Res.*, 10:2413–2444.

Sutton, R. S. (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pages 171–176. AAAI Press / The MIT Press.

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

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, pages 1057–1063. The MIT Press.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. (2015). High confidence policy improvement. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2380–2388. JMLR.org.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R. E., Ghahramani, Z., and Levine, S. (2018). The mirage of action-dependent baselines in reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5022–5031. PMLR.

Turchetta, M., Berkenkamp, F., and Krause, A. (2016). Safe exploration in finite markov decision processes with gaussian processes. In *NeurIPS*, pages 4305–4313.

Veeriah, V., Zhang, S., and Sutton, R. S. (2017). Crossprop: Learning representations by stochastic meta-gradient descent in neural networks. In *ECML/PKDD (1)*, volume 10534 of *Lecture Notes in Computer Science*, pages 445–459. Springer.

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

Xu, Z., van Hasselt, H., and Silver, D. (2018). Meta-gradient reinforcement learning. In *NeurIPS*, pages 2402–2413.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. (2011). Analysis and improvement of policy gradient estimation. In *NeurIPS*, pages 262–270.