Reinforcement Learning with Langevin Dynamics

Parameswaran Kamalaruban
LIONS, EPFL
kamalaruban.parameswaran@epfl.ch

Doga Tekin
EPFL
doga.tekin@epfl.ch

Paul Rolland
LIONS, EPFL
paul.rolland@epfl.ch

Volkan Cevher
LIONS, EPFL
volkan.cevher@epfl.ch

Abstract

We re-think the exploration-exploitation trade-off in reinforcement learning (RL) as an instance of a distribution sampling problem in infinite dimensions. Using the powerful Stochastic Gradient Langevin Dynamics, we propose a new RL algorithm, which is a sampling variant of the Twin Delayed Deep Deterministic Policy Gradient (TD3) method. Our new algorithm consistently outperforms existing exploration strategies for TD3 based on heuristic noise injection strategies on several MuJoCo environments.

1 Introduction

Recent advances in policy gradient methods [1, 2, 3, 4] and deep reinforcement learning (RL) have demonstrated impressive performance in games [5, 6], continuous control [7], and robotics [8]. Despite this success, understanding the so-called exploration-exploitation trade-off remains a key challenge in deep RL.

In this setting, we study the exploration-exploitation challenge with a new sampling twist. In particular, we recast the standard RL problem as a distribution learning problem over deterministic policies. This re-formulation results in a sampling problem albeit in infinite dimensions. To our knowledge, our sampling perspective of RL is new. Unlike [9, 10], we do not impose any parametric assumptions.

In order to solve this sampling problem, we use the well-known Stochastic Gradient Langevin Dynamics (SGLD) [11, 12]. This method iterates similarly as Stochastic Gradient Descent in optimization, but adds Gaussian noise to the gradient in order to sample. This sampling approach is understood as a way of performing exploration in the case of RL. The temperature parameter controlling the injected noise allows for trading exploration and exploitation, and decreases towards 0 throughout the iterations.

As an off-policy actor-critic method, Deep Deterministic Policy Gradient (DDPG) uses the Bellman equations for optimizing the value function and the policy gradient method to optimize the actor policy. DDPG's use of deterministic policies mitigates the problem of variance in the gradient, but it still lacks efficient exploration. In this paper, we will focus on a recent extension of DDPG, called Twin Delayed DDPG (or TD3) which has shown wide success [13].

Recent works have proposed several approaches to encourage exploration in DDPG via injecting noise in the action space [14, 7] or the parameter space [15]. It is still unclear whether these exploration strategies can always lead to desirable learning of the deterministic actor policy.

In parallel, sampling methods based on SGLD have recently shown great success in non-convex optimization [16], [17]. Due to their inherent noise injection, such methods naturally allow for

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

efficient exploration. In particular, SGLD has been found to improve learning for deep neural networks and other non-convex models [18, 19, 20, 21, 22, 23].

Based on this we propose a novel sampling variant of TD3 algorithm called "TD3-Annealing Langevin Dynamics" (TD3-ALD), which uses SGLD in order to optimize the actor. In practice, for complex RL environments, such as MuJoCo, we use a RMSProp preconditioned version of SGLD. We perform extensive experiments on several MuJoCo [24] environments and observed that our TD3-ALD method consistently outperforms other existing noisy exploration versions of TD3.

2 Background on Reinforcement Learning

Consider a Markov Decision Process (MDP) represented by $\mathcal{M} := (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, R)$, where the state and action spaces are denoted by $\mathcal{S}$ and $\mathcal{A}$ respectively. We focus on continuous control tasks, where the actions are real-valued, *i.e.*, $\mathcal{A} = \mathbb{R}^d$. $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ captures the state transition dynamics, *i.e.*, $T(s' \mid s, a)$ denotes the probability of landing in state s' by taking action a from state s. Here γ is the discounting factor, $P_0 : \mathcal{S} \to [0, 1]$ is the initial distribution over states $\mathcal{S}$, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward.

An agent executing a deterministic policy $\mu : \mathcal{S} \to \mathcal{A}$ in the environment $\mathcal{M}$ obtains a random cumulative return $Z = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$, where $r_t = R(s_t, a_t)$. Then, for this agent, we define the state and state-action value functions:

$$\begin{aligned} V^{\mu}(s) &:= \mathbb{E}[Z \mid S_1 = s; \mu, \mathcal{M}] \\ Q^{\mu}(s, a) &:= \mathbb{E}[Z \mid S_1 = s, A_1 = a; \mu, \mathcal{M}], \end{aligned}$$

where $Q^{\mu}(s, a)$ denotes the expected return after executing an action a in a state s and following the policy μ afterwards. Note that Q^{μ} is the unique solution to the Bellman equation:

$$Q^{\mu}(s, a) = R(s, a) + \gamma \sum_{s'} T(s' \mid s, a) Q^{\mu}(s', \mu(s')). \tag{1}$$

We also define the unnormalized discounted state distribution as $\rho^{\mu}(s) := \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{P}[S_t = s; \mu, \mathcal{M}]$, where $\mathbb{P}[S_t = s; \mu, \mathcal{M}]$ denotes the probability of visiting the state s at time step t by following the policy μ in the MDP $\mathcal{M}$.

Deterministic policy gradient In an off-policy setting, we consider the following performance objective:

$$\begin{aligned} J_b(\mu) &= \int_{\mathcal{S}} \rho^b(s) V^{\mu}(s) \, ds \\ &= \int_{\mathcal{S}} \rho^b(s) Q^{\mu}(s, \mu(s)) \, ds, \end{aligned} \tag{2}$$

which is the value function of the target policy μ, averaged over the state distribution of a (stochastic) behaviour policy $b : \mathcal{S} \times \mathcal{A} \to [0, 1]$. In policy search methods, we consider parameterized policies $\{\mu_\theta : \theta \in \Theta\}$, and search for θ to maximize the performance objective $J_b(\mu_\theta)$. By an abuse of notation, we denote $J_b(\theta) = J_b(\mu_\theta)$. If the policy parametrization μ_θ is differentiable, under some regularity conditions on the MDP, the gradient of $J_b(\theta)$ can be expressed as

$$\begin{aligned} \nabla_\theta J_b(\theta) &\approx \int_{\mathcal{S}} \rho^b(s) \nabla_\theta \mu_\theta(a \mid s) Q^{\mu}(s, a) \, ds \\ &= \mathop{\mathbb{E}}_{s \sim \rho^b} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s, a) \mid_{a = \mu_\theta(s)} \right] \end{aligned} \tag{3}$$

The above equation is referred to as *off-policy deterministic policy gradient* [2].

In this work, we focus on the deep deterministic policy gradient (DDPG) method [7], which is applicable to continuous action spaces. DDPG is an actor-critic method that maintains a parameterized actor function μ_θ which specifies the current deterministic policy. The critic $Q^w(s, a)$ is parameterized by w and learned using the Bellman equation (1), by minimizing the empirical Bellman residual $L(w) = \mathop{\mathbb{E}}_{s_t \sim \rho^b, a_t \sim b} \left[\left(Q^w(s_t, a_t) - y_t \right)^2 \right]$, where $y_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1}))$, and b is a behaviour policy used to collect samples.

Twin Delayed DDPG (TD3) It is well known that Q-learning and deterministic policy gradients suffer from an overestimation bias due to the noise in the value estimates. To address this issue, [13] proposed a variant of DDPG built on Double Q-learning, by making the following changes:

1. Maintains a pair of critics along with a single actor.
2. Clipped action exploration: noise added like DDPG but bounded to fixed range

$$a'(s') \;=\; \mathrm{clip}\left(\mu_{\theta_{\mathrm{targ}}}(s') + \mathrm{clip}\left(\epsilon, -c, c\right), a_{\mathrm{Low}}, a_{\mathrm{High}}\right), \quad \epsilon \sim \mathcal{N}(0, \bar{\sigma})$$

3. Updates of Critic are more frequent than of policy.

3 Stochastic Gradient Langevin Dynamics (SGLD)

Stochastic Gradient Langevin Dynamics (SGLD) is a popular variant of Stochastic Gradient Descent (SGD), where in each step it injects appropriately scaled Gaussian noise to the update. Given a possibly non-convex function $f : \mathbb{R}^d \to \mathbb{R}$, SGLD performs the iterative update:

$$\theta_{t+1} \;\leftarrow\; \theta_t - \eta_t \frac{1}{\beta_t} \widehat{\nabla_\theta f\left(\theta_t\right)} + \sqrt{2\eta_t}\xi \tag{4}$$

where $\widehat{\nabla_\theta f\left(\theta_t\right)}$ is an unbiased estimate of the gradient $\left[\nabla_\theta f\left(\theta\right)\right]_{\theta=\theta_t}$, ξ is a standard Gaussian random vector in $\mathbb{R}^d$, $\{\eta_t\}$ is a sequence of step-sizes, and $\{\beta_t\}$ is a sequence of inverse temperature parameters. [25, 16, 26] have shown that SGLD globally optimizes f whenever the objective is *dissipative* ($\langle \nabla f(\theta), \theta \rangle \geq a \left\|\theta\right\|_2^2 - b$ for $a, b > 0$) with a Lipschitz gradient ($\left\|\nabla f\left(\theta\right) - \nabla f\left(\theta'\right)\right\| \leq L \left\|\theta - \theta'\right\|$ for $L > 0$). Under some additional regularity conditions, [27] show that SGLD can escape strict saddle points in polynomial time.

Sampling Interpretation When η and β are kept fixed, the updates do not converge to a global minimizer, but it can be shown that SGLD asymptotically converges to a stationary distribution $p_\beta\left(\theta\right) \propto \exp\left(-\frac{f(\theta)}{\beta}\right)$. As $\beta \to 0$, the probability mass of p_β concentrates on the global minimum of f, and the algorithm asymptotically converges to a neighborhood of the global minimum. One could consider an annealing scheme given by $\beta_t = \beta \cdot t^{-r}$, where $r \in [0, 1]$. In the continuous control experiments; we found that $r = 0.25$ is a reasonable choice.

Preconditioned SGLD In some cases, the convergence rate of SGLD can be improved by scaling the noise using a positive-definite symmetric matrix C. We thus define a preconditioned variant of the above update (4) as follows:

$$\theta_{t+1} \;\leftarrow\; \theta_t - \eta_t C_t^{-1} g_t + \sqrt{2\eta_t} C_t^{-\frac{1}{2}} \xi, \tag{5}$$

where $g_t = \frac{1}{\beta_t}\widehat{\nabla_\theta f\left(\theta_t\right)}$. In this setting, the Fisher Information Matrix (FIM) given by $F\left(\theta_t\right) = \mathbb{E}\left[g_t g_t^\top\right]$ has been suggested as an efficient preconditioner C_t [28, 29, 30]. In practice, it is impossible to invert or even store this large dimensional matrix. As approximating FIM by any positive-definite symmetric matrix does not invalidate the asymptotic convergence of the SGLD, we consider the preconditioner from RMSProp algorithm as an approximation to FIM. Specifically, we set C_t as follows:

$$f_t \;\leftarrow\; \gamma f_{t-1} + (1-\gamma)\, g_t \odot g_t \tag{6}$$

$$C_t \;\leftarrow\; \mathrm{diag}\left(\sqrt{f_t + \delta}\right) \tag{7}$$

where the operator $\odot$ represents element-wise vector product, and $V = \mathrm{diag}\left(v\right)$ denotes a diagonal matrix with $V_{i,i} = v_i$.

4 Reinforcement Learning with SGLD

A fundamental aspect of RL is the exploration-exploitation dilemma: in order to maximize cumulative reward, agents need to trade-off what is expected to be best at the moment, (i.e., exploitation), with

potentially sub-optimal exploratory actions. The DDPG method is effective in continuous control tasks mainly due to its low variance nature as compared to stochastic policy search methods. However, DDPG lacks efficient state space exploration due to the absence of stochasticity in the policy. The following two techniques are typically used in practice to encourage exploration while collecting the episodes for the replay buffer:

1. Noise injection in the action space [2, 7]: either uncorrelated noise $\epsilon \sim \mathcal{N}\left(0, \sigma^2 I\right)$ or correlated noise $\epsilon \sim \mathrm{OU}\left(0, \sigma^2\right)$ is added to policy output, *i.e.*, $a = \mu_\theta\left(s\right) + \epsilon$. [31] noted that this action space noise may be insufficient in some problems.
2. Noise injection in the parameter space [15]: Gaussian noise with adaptive scale is added to the current policy parameter, *i.e.*, $\tilde{\theta} = \theta + \mathcal{N}\left(0, \sigma^2 I\right)$.

Let us denote the off-policy RL objective as follows:

$$\underset{\theta}{\text{maximize}} \quad J_b\left(\theta\right). \tag{8}$$

We translate the above non-convex problem into an infinite dimensional convex-problem (linear in p) by considering a distribution over deterministic policies as follows:

$$\underset{p \in \mathcal{M}(\Theta)}{\text{maximize}} \quad \mathop{\mathbb{E}}_{\theta \sim p}\left[J_b\left(\theta\right)\right]. \tag{9}$$

where $\mathcal{M}\left(\Theta\right)$ is the (infinite dimensional) space of all probability distributions on Θ. More precisely, we consider that at the beginning of each episode, deterministic policy parameters θ are sampled from a distribution $p \in \mathcal{M}\left(\Theta\right)$. Note that $p^* = \arg\max_p \mathop{\mathbb{E}}_{\theta \sim p}\left[J_b\left(\theta\right)\right]$ is a delta measure centered at $\theta^* = \arg\max_\theta J_b\left(\theta\right)$, so this reformulated sampling problem is indeed equivalent to the initial optimization problem. [9, 10] have also studied the problem (9), but they have only considered parametric distributions over Θ, specifically θ is sampled from a Gaussian distribution $p\left(\theta\right) = \mathcal{N}\left(\mu, \sigma^2 I\right)$. Whereas we do not impose any parametric assumption on the probability distribution, and our approach to solve (9) is completely different from theirs.

We then smooth this problem by adding an entropy regularizer as follows [32]:

$$\underset{p \in \mathcal{M}(\Theta)}{\text{maximize}} \quad \mathop{\mathbb{E}}_{\theta \sim p}\left[J_b\left(\theta\right)\right] + \beta H\left(p\right), \tag{10}$$

where $H\left(p\right) = \mathop{\mathbb{E}}_{\theta \sim p}\left[-\log p\left(\theta\right)\right]$ is the entropy of the distribution p. It can be shown that the optimal solution to the above problem takes the form:

$$p^*_\beta\left(\theta\right) \propto \exp\left(\frac{1}{\beta} J_b\left(\theta\right)\right). \tag{11}$$

When $\beta \to 0$, samples drawn from $p^*_\beta\left(\theta\right)$ will be around global optima of the expected return $J_b\left(\theta\right)$. However, as $\beta \to 0$, the distribution p^*_β becomes less and less smooth, and so it becomes harder and harder to sample from it. Consequently, we have to choose a suitable β to provide a good tradeoff between smoothness of p_β and convergence of the samples to the global optima of J_b. Then, for a given β, Stochastic Gradient Langevin Dynamics (SGLD) [11] can be used to draw samples from $p^*_\beta\left(\theta\right)$.

Our Approach Inspired by the above discussion on the entropy regularized objective, we propose a sampling scheme to draw policy parameters θ from $p_t\left(\theta\right) \propto \exp\left(\frac{1}{\beta_t} J_b\left(\theta\right)\right)$ with $\lim_{t \to \infty} \beta_t = 0$, which gradually converges to the optimal solution of problem (9). Specifically, we use the following update rule:

$$\theta_{t+1} \leftarrow \theta_t + \eta_t g_t + \sqrt{2\eta_t}\xi_t, \tag{12}$$

where η_t is the step-size, $g_t = \frac{1}{\beta_t}\widehat{\nabla_\theta J_b\left(\theta_t\right)}$ is an unbiased gradient estimator, *i.e.*, $\mathbb{E}\left[\widehat{\nabla_\theta J_b\left(\theta_t\right)}\right] = \nabla_\theta J_b\left(\theta_t\right)$, and $\xi_t \sim \mathcal{N}\left(0, I\right)$ is a standard normal vector, independently drawn across iterations. In the experiments, we use the RMSProp preconditioned variant of the above update as in (5), (6), and (7). We note that the baselines also take advantage of RMSProp optimizer. By incorporating this sampling scheme for the actor with standard TD3, we obtain our final algorithm called "TD3-ALD" (i.e., TD3 with Annealing Langevin Dynamics). See Appendix A for full algorithm.

Note that the existing exploration schemes (action noise [7] and parameter noise [15]) encourage exploration during the episode collection phase (their training updates for the network are unchanged), whereas in our method we alter actor update and encourage exploration in the policy optimization phase. In the standard TD3 implementation, the algorithm alternates between episode collection and policy update at a particular frequency. Thus, our TD3-ALD algorithm without exploration noise suffers from collecting episodes using deterministic policies. In practice, we found that injecting action or parameter noise (of small magnitude compared to [7, 15]) to TD3-ALD boosts its performance considerably.

As an illustrative example, consider a continuous bandit problem, where we want to maximize an unknown reward function $r(a)$. We can tackle this problem via a simple actor-critic method, where we maintain an actor $a = \mu_\theta(s) = \theta$ and a critic $r_w(\cdot)$. Based on the transitions $(a, r(a))$ obtained from the interaction with the environment, the critic r_w can be trained using a simple regression algorithm. In this setting, both action noise and parameter noise strategies are the same, i.e., they both use a perturbed action $\theta + \epsilon$ while collecting the transitions. Instead, we can update the actor using SGLD, and collect the transitions without perturbing the action.

5 Experiments

We evaluate the performance of our TD3-ALD algorithm on standard continuous control benchmarks available on OpenAI Gym [33] utilizing the MuJoCo environment [24]. Specifically, we benchmark on six tasks: Walker, Hopper, Reacher, Half-Cheetah, Swimmer, and Ant. Details of these environments can be found in [33] and on the GitHub website.

We compare the results from the following configurations: (a) TD3 with action noise [7, 13] (b) TD3 with parameter noise [15] (c) TD3-ALD with no noise (d) TD3-ALD with action noise (e) TD3-ALD with parameter noise. Note that the action or parameter noise is injected while collecting transitions for the replay buffer. In [13], authors noted that the action noise drawn from the Ornstein-Uhlenbeck [14] process offered no performance benefits. Thus we also consider uncorrelated Gaussian noise.

Setup The TD3-ALD implementation is based on the available codebase from OpenAI Baselines. For all the algorithms, we use a two-layer feedforward neural network structure of (64, 64, tanh) for both actor and critic. The optimizer we use to update the critic is Adam [34] with a learning rate of 10^{-3}. The target networks are soft-updated with $\tau = 0.001$. For the TD3-specific hyper-parameters (target action noise, clipping range, and policy update delay), we use the values that are reported in [13].

For the baselines (a) and (b), the actor is trained with RMSProp optimizer. For our algorithms (c), (d), and (e), the actor is updated according to (5), (6), and (7) with fixed annealing rate $r = 0.25$ for the inverse temperature parameter $\beta_t = \beta \cdot t^{-r}$. The hyperparameters that are not related to exploration (see Table 1 and Table 2 in Appendix A) are identical across all the algorithms that are compared.

And we tuned only the exploration-related hyper-parameters (for all the algorithms) by grid search: (a) and (b) $\sigma \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1\}$; (c) $\beta \in \{e-4, e-5, e-6, e-7, e-8, e-9, e-10\}$; (d) and (e) $(\beta, \sigma) \in \{e-4, e-5, e-6, e-7, e-8, e-9, e-10\} \times \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$. For each algorithm-environment pair, we identified the best performing exploration hyperparameter configuration (see Tables 3, 4, 5, 6, 7 in Appendix A and the figures in Appendix B).

Each agent is trained for 2M steps, and the performance of the agent (measured in terms of the average cumulative reward) is evaluated every 10k steps for 10 episodes. The exploration noise is turned off for evaluation. We run five experiment trials for each evaluation, each with a different preset random seed. To ensure fairness, the same set of random seeds is used for all configurations and environments.

Evaluation In this section, we analyze the main experimental results. In Figure 1, we plot the mean performance with the standard deviation across random seeds. Our algorithms (d) and (e) consistently achieve better or comparable performance to the baselines (a) and (b) across all tasks. The results also show that TD3-ALD combined with action or parameter noise (small in magnitude compared to the baselines) provides considerably improved performance over TD3-ALD without exploration noise during the data collection phase. The baselines (a) and (b) fail to make any progress on Ant-v2,

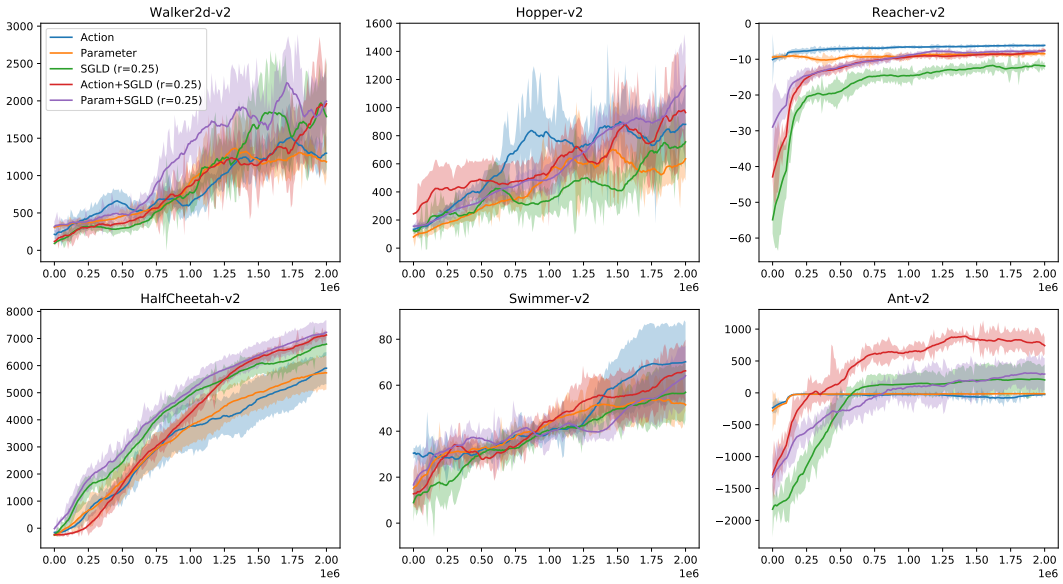

Figure 1: Comparison of best performing hyperparameter options of different noise types when using dynamic ALD with decay $r = 0.25$. Best hyperparameters for a noise type can be different for each environment. Plots show (smoothed) mean rewards over 10 evaluation episodes every 10000 timesteps, the shaded area shows half a standard deviation on both sides of the curve.

whereas all our algorithms (c)-(e) result in improved performance. Note that we use smaller (64,64) networks in our experiments, whereas [13] have used larger (400,300) networks, this could be a reason for the failure of baselines on Ant-v2.

Ablation As noted in [35], tuned hyperparameters play a significant role in eliciting the best results from many deep RL algorithms. We have performed a detailed investigation on how sensitive the algorithms (a)-(e) are to their corresponding exploration hyper-parameters. The detailed results are given in Appendix B. For each algorithm, we identified a single exploration hyper-parameter configuration that performs well across all the environments (see Table 8 in Appendix A). The mean performance (with the standard deviation) under this hyper-parameter choice is presented in Figure 2. We have also examined the impact of annealing the inverse temperature and noise parameters on the performance of all the algorithms (see Appendix A).

We note that we have done similar experiments with DDPG as well and obtained qualitatively similar results that we have presented for TD3.

6 Conclusion

In this work, we first introduced a maximum entropy policy optimization framework. By modeling the policy as a distribution over deterministic policy parameters, we rephrased the initial optimization problem as an entropy regularized optimization problem over distributions, of which we can obtain an analytical solution. Then, to sample from this solution distribution, we use Stochastic Gradient Langevin Dynamics. By making the entropy regularization parameter go to 0, the samples obtained from this distribution become closer to the global optima of the objective function. This framework explicitly encourages exploration in the parameter space while also optimizing the expected utility of policies generated from this distribution. In our experiments, we evaluated our algorithm on several continuous control tasks and found that they greatly improved the performance of the original off-policy gradient methods.

References

[1] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[2] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[3] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[5] 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, 2015.

[6] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[7] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[9] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[10] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 10–15 Jul 2018.

[11] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[12] Arnak S Dalalyan and Avetik G Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *arXiv preprint arXiv:1710.00095*, 2017.

[13] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[14] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[15] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[16] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*, 2017.

[17] Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1702.05575*, 2017.

[18] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[19] Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

[20] Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. *arXiv preprint arXiv:1511.06392*, 2015.

[21] Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*, 2015.

[22] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2462–2466. IEEE, 2017.

[23] Arnak S Dalalyan. Further and stronger analogy between sampling and optimization: Langevin monte carlo and gradient descent. *arXiv preprint arXiv:1704.04752*, 2017.

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

[25] Saul B Gelfand and Sanjoy K Mitter. Recursive stochastic algorithms for global optimization in rˆd. *SIAM Journal on Control and Optimization*, 29(5):999–1018, 1991.

[26] Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3122–3133, 2018.

[27] Xi Chen, Simon S Du, and Xin T Tong. Hitting time of stochastic gradient langevin dynamics to stationary points: A direct analysis. *arXiv preprint arXiv:1904.13016*, 2019.

[28] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[29] Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

[30] Sam Patterson and Yee Whye Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*, pages 3102–3110, 2013.

[31] Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1):14–24, 2010.

[32] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[33] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[34] Diederik P Kingma and Jimmy Ba. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.

[35] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[36] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.