ROBUST REINFORCEMENT LEARNING VIA ADVERSARIAL TRAINING WITH LANGEVIN DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

We re-think the Two-Player 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 two-player RL algorithm, which is a sampling variant of the two-player policy gradient method. Our new algorithm consistently outperforms existing baselines, in terms of generalization across differing training and testing conditions, on several MuJoCo environments.

1 INTRODUCTION

Reinforcement learning (RL) promise automated solutions to many real-world tasks with beyond-human performance. Indeed, recent advances in policy gradient methods (Sutton et al., 2000; Silver et al., 2014; Schulman et al., 2015; 2017) and deep reinforcement learning have demonstrated impressive performance in games (Mnih et al., 2015; Silver et al., 2017), continuous control (Lillicrap et al., 2015), and robotics (Levine et al., 2016) towards this grand challenge.

Despite the success of deep RL, the progress is still upset by the fragility in real-life deployments. In particular, majority of these methods fail to perform well when there is some difference between training and testing scenarios, thereby posting serious safety and security concerns. To this end, learning policies that are *robust* to environmental shifts, mismatched configurations, and even mismatched control actions is becoming increasingly more important.

A powerful framework to learning robust policies is to interpret the changing of the environment as an adversarial perturbation. This notion naturally lends itself to a two-player minimax problem involving a pair of agents, a protagonist and an adversary, where the protagonist learns to fulfill the original task goals while being robust to the disruptions generated by its adversary.

Two prominent examples along this research vein, differing in how they model the adversary, are the Robust Adversarial Reinforcement Learning (RARL) (Pinto et al., 2017) and Noisy Robust Markov Decision Process (NR-MDP) (Tessler et al., 2019). Despite achieving impressive performance in practice, these existing frameworks heavily rely on heuristic algorithms, and hence, suffer from lack of theoretical guarantees, even in idealistic cases with infinite data as well as computational power.

One critical challenge in robust RL setting is that while maximizing rewards is a well-studied subject in classical/deep RL, the needed two-player minimax version is significantly more complicated to solve both in theory and practice. For instance, Tessler et al. (2019) prove that it is in fact strictly suboptimal to directly apply (deterministic) policy gradient steps to their NR-MDP max-min objectives. Owing to the lack of a better algorithm, the policy gradient is nonetheless still employed in their experiments; similar comments also apply to (Pinto et al., 2017).

Our paper precisely bridges this gap between theory and practice in previous works, by proposing the first theoretically convergent algorithm for robust RL. Our key idea is to switch from optimizing the max-min reward to *sampling* from the optimal *randomized policies*, which corresponds to finding a mixed Nash Equilibrium (NE) (Nash et al., 1950) in the max-min objective.

It is a classical fact in game theory that, while deterministic minimax objective is often ill-posed, the mixed NE is well-behaved under very mild assumptions. Furthermore, algorithmic approaches to finding mixed NE with finite strategies have been studied extensively (Freund & Schapire, 1999; Nemirovski, 2004) and is recently extended to the case of infinite strategies (Hsieh et al., 2019). In particular, (Hsieh et al., 2019) show that, by using the Stochastic Gradient Langevin Dynamics

(SGLD) (Welling & Teh, 2011) to take samples from randomized strategies, one can find a mixed NE of Generative Adversarial Networks (Goodfellow et al., 2014).

Our work introduces the same mixed NE perspective to the max-min objectives in robust RL, and substantiate the ensuing theoretical framework with extensive experimental evidence. We apply the new sampling framework to the well-known Deep Deterministic Policy Gradient (DDPG) method in the scope of NR-MDP. We demonstrate that the new algorithm achieves clearly superior performance in its generalization capabilities.

Intriguingly, we also observe that the idea of mixed strategy in single-player RL (i.e., the non-robust or one-player formulation) can lead to substantially more robust policies over the standard DDPG algorithm. More precisely, we represent the agent's policy as a distribution over deterministic policies, and aim to sample from the distribution $\mu(\pi) \propto \exp(-\frac{1}{\sigma}J(\pi))$ where π denotes a deterministic policy, J the associated expected reward, and σ a temperature parameter going to 0 during training.

Our numerical evidence demonstrates that DDPG combined with this sampling approach for the actor update leads to learned policies that generalize better to unseen MDPs, when compared to the state-of-the-art DDPG variant of Tessler et al. (2019) while using similar computational resources.

2 Background

2.1 Stochastic Gradient Langevin Dynamics (SGLD)

For any probability distribution $p(z) \propto \exp(-g(z))$, the Stochastic Gradient Langevin Dynamics (SGLD) Welling & Teh (2011) iterates as

$$z_{k+1} \leftarrow z_k - \gamma \left[\widehat{\nabla_z g(z)}\right]_{z=z_k} + \sqrt{2\gamma}\epsilon\xi_k, \tag{1}$$

where γ is the step-size, $\widehat{\nabla_z g(z)}$ is an unbiased estimator of $\nabla_z g(z)$, $\epsilon > 0$ is a temperature parameter, and $\xi_k \sim \mathcal{N}(0, I)$ is a standard normal vector, independently drawn across different iterations. 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 equation 1 as follows:

$$z_{k+1} \leftarrow z_k - \gamma C^{-1} \left[\widehat{\nabla_z g(z)}\right]_{z=z_k} + \sqrt{2\gamma}\epsilon C^{-\frac{1}{2}}\xi_k. \tag{2}$$

In the experiments, we use a RMSProp-preconditioned version of the SGLD (Li et al., 2016).

2.2 Infinite-Dimensional Bi-Linear Games

In this section, we review some of the key results from (Hsieh et al., 2019). We denote the set of all probability measures on $\mathcal{Z}$ by $\mathcal{P}(\mathcal{Z})$, and the set of all functions on $\mathcal{Z}$ by $\mathcal{F}(\mathcal{Z})$. Given a (sufficiently regular) function $h : \Theta \times \Omega \to \mathbb{R}$, consider the following objective (a two-player game with *infinitely* many strategies):

$$\max_{p\in\mathcal{P}(\Theta)} \min_{q\in\mathcal{P}(\Omega)} f(p,q) := \mathop{\mathbb{E}}_{\theta\sim p}\left[\mathop{\mathbb{E}}_{\omega\sim q}\left[h(\theta,\omega)\right]\right]. \tag{3}$$

A pair (p^*, q^*) achieving the max-min value in equation 3 is called a *mixed Nash Equilibrium* (NE). Define the operator $G : \mathcal{P}(\Omega) \to \mathcal{F}(\Theta)$, and its adjoint operator $G^\dagger : \mathcal{P}(\Theta) \to \mathcal{F}(\Omega)$ as follows:

$$\begin{aligned} Gq(\theta) &:= \mathop{\mathbb{E}}_{\omega\sim q}\left[h(\theta,\omega)\right] \in \mathcal{F}(\Theta) \\ G^\dagger p(\omega) &:= \mathop{\mathbb{E}}_{\theta\sim p}\left[h(\theta,\omega)\right] \in \mathcal{F}(\Omega). \end{aligned}$$

Denoting $\langle p, g\rangle := \mathop{\mathbb{E}}_{z\sim p}\left[g(z)\right]$ for any probability measure p and function g on $\mathcal{Z}$, we can write f as $f(p,q) = \langle p, Gq\rangle = \langle G^\dagger p, q\rangle$. Furthermore, the derivative (the analogue of gradient in infinite dimension) of $f(p,q)$ with respect to p is simply Gq, and the derivative of $f(p,q)$ with respect to q is $G^\dagger p$; i.e., $\nabla_p f(p,q) = Gq$, and $\nabla_q f(p,q) = G^\dagger p$.

Algorithm 1 Approximate Infinite-Dimensional Entropic Mirror Descent

Input: $\omega_1, \theta_1 \leftarrow$ random initialization, SGLD step-size $\{\gamma_t\}_{t=1}^T$, thermal noise of SGLD $\{\epsilon_t\}_{t=1}^T$, warmup steps for SGLD $\{K_t\}_{t=1}^T$, exponential damping factor β, standard normal noise ξ_k, ξ'_k.
for $t = 1, 2, \ldots, T-1$ **do**
  $\bar{\omega}_t, \omega_t^{(1)} \leftarrow \omega_t \; ; \; \bar{\theta}_t, \theta_t^{(1)} \leftarrow \theta_t$
  for $k = 1, 2, \ldots, K_t$ **do**
    $\theta_t^{(k+1)} \leftarrow \theta_t^{(k)} + \gamma_t \left[\nabla_\theta \widehat{h(\theta, \omega_t)}\right]_{\theta=\theta_t^{(k)}} + \sqrt{2\gamma_t}\epsilon_t \xi_k$
    $\omega_t^{(k+1)} \leftarrow \omega_t^{(k)} - \gamma_t \left[\nabla_\omega \widehat{h(\theta_t, \omega)}\right]_{\omega=\omega_t^{(k)}} + \sqrt{2\gamma_t}\epsilon_t \xi'_k$
    $\bar{\omega}_t \leftarrow (1-\beta)\,\bar{\omega}_t + \beta\omega_t^{(k+1)} \; ; \; \bar{\theta}_t \leftarrow (1-\beta)\,\bar{\theta}_t + \beta\theta_t^{(k+1)}$
  end for
  $\omega_{t+1} \leftarrow (1-\beta)\,\omega_t + \beta\bar{\omega}_t \; ; \; \theta_{t+1} \leftarrow (1-\beta)\,\theta_t + \beta\bar{\theta}_t$
end for
Output: ω_T, θ_T.

Conceptually, problem (3) can be solved via the so-called infinite-dimensional entropic mirror descent; see Algorithm 2 in the appendix. Hsieh et al. (2019) have proved the convergence (to the mixed NE) rate of Algorithm 2. But this algorithm is infinite-dimensional and requires infinite computational power to implement. For practical interest, by leveraging the SGLD sampling techniques and using some practical relaxations, Hsieh et al. (2019) have proposed a simplified variant of Algorithm 2. The pseudocode for their resulting algorithm can be found in Algorithm 1.

3 Two-Player Markov Games

Markov Decision Process: We consider a Markov Decision Process (MDP) represented by $\mathcal{M}_1 := (\mathcal{S}, \mathcal{A}, T_1, \gamma, P_0, R_1)$, 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.*, $\mathcal{A} = \mathbb{R}^d$. $T_1 : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ captures the state transition dynamics, *i.e.*, $T_1(s' \mid s, a)$ 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 $R_1 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward.

Two-Player Zero-Sum Markov Games: Consider a two-player zero-sum Markov game Littman (1994); Perolat et al. (2015), where at each step of the game, both players simultaneously choose an action. The reward each player gets after one step depends on the state and the joint action of both players. Furthermore, the transition kernel of the game is controlled jointly by both the players. In this work, we only consider simultaneous games, not the turn-based games.

This game can be described by an MDP $\mathcal{M}_2 = (\mathcal{S}, \mathcal{A}, \mathcal{A}', T_2, \gamma, R_2, P_0)$, where $\mathcal{A}$ and $\mathcal{A}'$ are the continuous set of actions the players can take, $T_2 : \mathcal{S} \times \mathcal{A} \times \mathcal{A}' \times \mathcal{S} \to \mathbb{R}$ is the state transition probability, and $R_2 : \mathcal{S} \times \mathcal{A} \times \mathcal{A}' \to \mathbb{R}$ is the reward for both players. Consider an agent executing a policy $\mu : \mathcal{S} \to \mathcal{A}$, and an adversary executing a policy $\nu : \mathcal{S} \to \mathcal{A}'$ in the environment $\mathcal{M}$. At each timestep t, both players observe the state s_t and take actions $a_t = \mu(s_t)$ and $a'_t = \nu(s_t)$. In the zero-sum game, the agent gets a reward $r_t = R_2(s_t, a_t, a'_t)$ while the adversary gets a negative reward $-r_t$.

This two-player zero-sum Markov game formulation has been used to model the following robust RL settings:

- Robust Adversarial Reinforcement Learning (RARL) (Pinto et al., 2017), where the power of the adversary is limited by the action space $\mathcal{A}'$ of the adversary.
- Noisy Robust Markov Decision Process (NR-MDP) (Tessler et al., 2019), where $\mathcal{A}' = \mathcal{A}$, $T_2(s_{t+1} \mid s_t, a_t, a'_t) = T_1(s_{t+1} \mid s_t, \bar{a}_t)$, and $R_2(s_t, a_t, a'_t) = R_1(s_t, \bar{a}_t)$, with $\bar{a}_t = (1-\delta)a_t + \delta a'_t$, for $\delta \in (0, 1)$. The power of the adversary is limited by δ.

In our adversarial game, we consider the following performance objective:

$$J(\mu,\nu) \;=\; \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_t \;\middle|\; \mu,\nu,\mathcal{M}\right],$$

where $\sum_{t=1}^{\infty}\gamma^{t-1}r_t$ be the random cumulative return. In particular, we consider the parameterized policies $\{\mu_\theta : \theta\in\Theta\}$, and $\{\nu_\omega : \omega\in\Omega\}$. By an abuse of notation, we denote $J(\theta,\omega) = J(\mu_\theta,\nu_\omega)$. We consider the following objective:

$$\max_{\theta\in\Theta}\min_{\omega\in\Omega}\; J(\theta,\omega)\,. \tag{4}$$

Note that J is non-convex/concave in both θ and ω. Instead of solving equation 4 directly, we focus on the mixed strategy formulation of equation 4. In other words, we consider the set of all probability distributions over Θ and Ω, and we search for the optimal distribution that solves the following program:

$$\max_{p\in\mathcal{P}(\Theta)}\min_{q\in\mathcal{P}(\Omega)}\; f(p,q) \;:=\; \mathop{\mathbb{E}}_{\theta\sim p}\left[\mathop{\mathbb{E}}_{\omega\sim q}\left[J(\theta,\omega)\right]\right]. \tag{5}$$

Then, we can use the techniques from Section 2.2 to solve the above problem.

4 EXPERIMENTS

In this section, we demonstrate the effectiveness of using infinite-dimensional sampling techniques to solve the robust RL problem.

Two-Player DDPG: As a case study, we consider NR-MDP setting with $\delta = 0.1$ (as recommended in Section 6.3 of (Tessler et al., 2019)). We design a two-player variant of DDPG (Lillicrap et al., 2015) algorithm by adapting the Algorithm 1. As opposed to standard DDPG, in two-player DDPG two actor networks output two deterministic policies, the protagonist and adversary policies, denoted by μ_θ and ν_ω. The critic is trained to estimate the Q-function of the joint-policy. The gradients of the protagonist and adversary parameters are given in Proposition 5 of (Tessler et al., 2019). The resulting algorithm is given in Algorithm 3.

We compare the performance of our algorithm against the baseline algorithm proposed in (Tessler et al., 2019) (see Algorithm 4). (Tessler et al., 2019) have suggested a training ratio of $1:1$ for actors and critic updates. Note that the action noise is injected while collecting transitions for the replay buffer. In Fujimoto et al. (2018), authors noted that the action noise drawn from the Ornstein-Uhlenbeck Uhlenbeck & Ornstein (1930) process offered no performance benefits. Thus we also consider uncorrelated Gaussian noise.

Setup: We evaluate the performance of Algorithm 3 and Algorithm 4 on standard continuous control benchmarks available on OpenAI Gym Brockman et al. (2016) utilizing the MuJoCo environment Todorov et al. (2012). Specifically, we benchmark on eight tasks: Walker, Hopper, HalfCheetah, Ant, Swimmer, Reacher, Humanoid, and InvertedPendulum. Details of these environments can be found in Brockman et al. (2016) and on the GitHub website.

The Algorithm 3 implementation is based on the codebase from (Tessler et al., 2019). For all the algorithms, we use a two-layer feedforward neural network structure of (64, 64, tanh) for both actors (agent and adversary) and critic. The optimizer we use to update the critic is Adam Kingma & Ba (2015) with a learning rate of 10^{-3}. The target networks are soft-updated with $\tau = 0.999$.

For the baseline, the actors are trained with RMSProp optimizer. For our algorithm, the actors are updated according to Algorithm 1 with warmup steps $K_t = \min\left\{15, \lfloor(1+10^{-5})^t\rfloor\right\}$, and thermal noise $\sigma_t = \sigma_0 \times (1 - 5\times 10^{-5})^t$. The hyperparameters that are not related to exploration (see Table 1) are identical to both algorithms that are compared.

And we tuned only the exploration-related hyper-parameters (for both algorithms) by grid search: (a) Algorithm 3 with $(\sigma_0,\sigma) \in \left\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\right\} \times \{0, 0.01, 0.1, 0.2, 0.3, 0.4\}$; (b) Algorithm 4 with $\sigma \in \{0, 0.01, 0.1, 0.2, 0.3, 0.4\}$. For each algorithm-environment pair, we identified the best performing exploration hyperparameter configuration (see Tables 2 and 3).

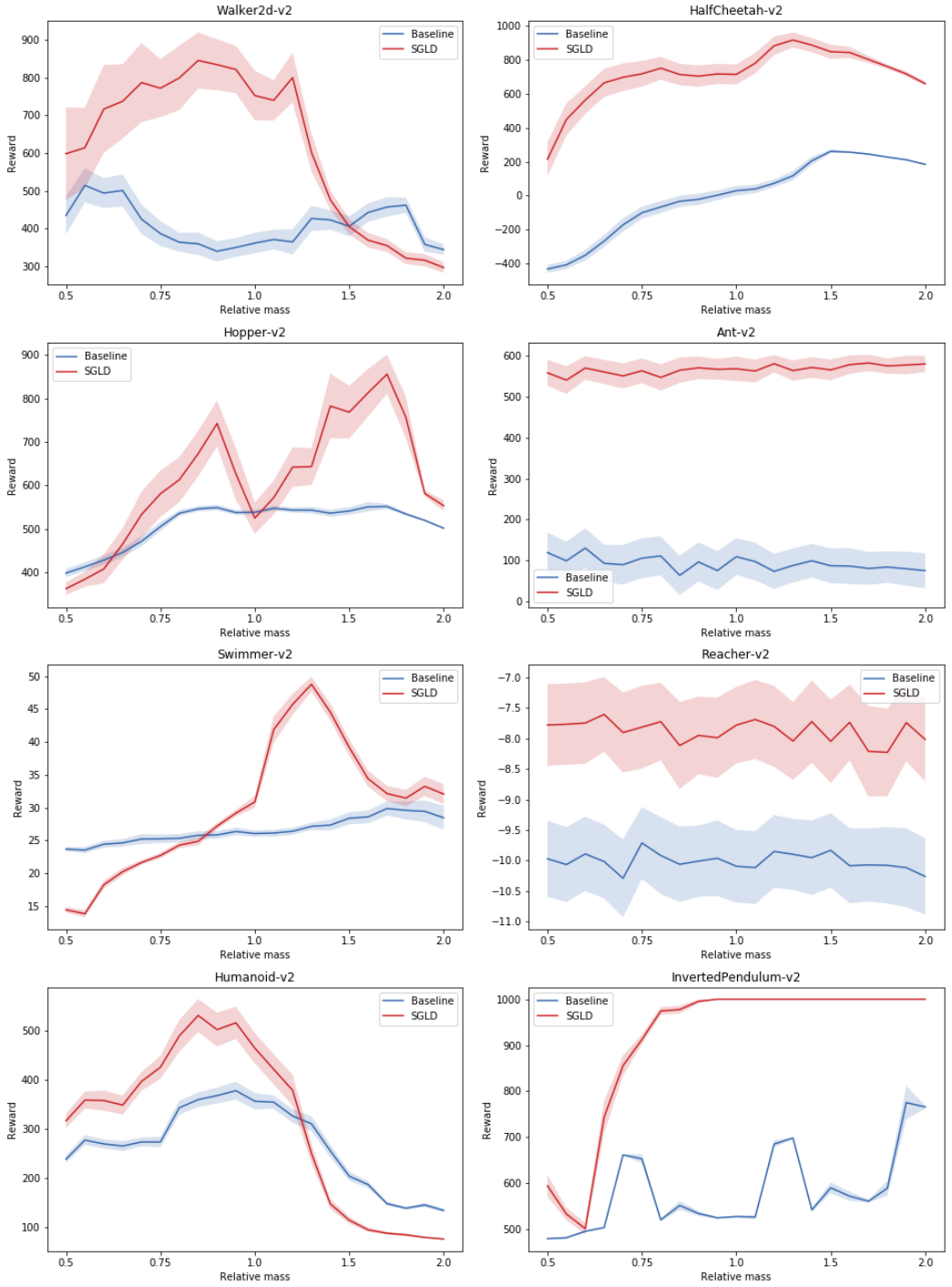

Figure 1: Average performance (over 5 seeds) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0.1$. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training.

Each algorithm is trained on 0.5M samples (i.e., 0.5M time steps in the environment). We run our experiments, for each environment, with 5 different seeds. The exploration noise is turned off for evaluation.

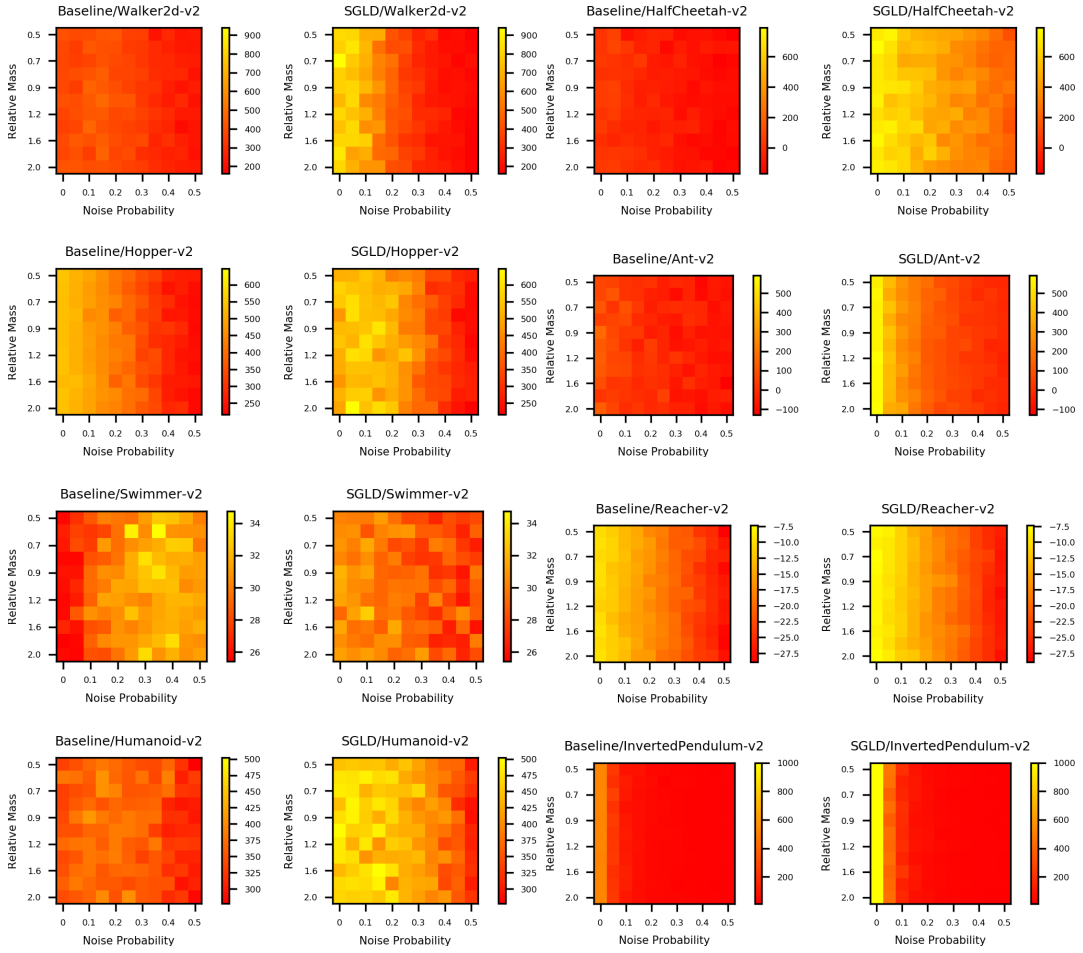

Figure 2: Average performance (over 5 seeds) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0.1$. The evaluation is performed on a range of noise probability and mass values not encountered during training.

Evaluation: We evaluate the robustness of both algorithms under different testing conditions, and in the presence of adversarial disturbances in the testing environment. We train both algorithms with the standard mass variables in OpenAI Gym. At test time, we evaluate the learned policies by changing the mass values (without adversarial perturbations) and estimating the cumulative rewards. As shown in Figure 1, our Algorithm 3 outperforms the baseline Algorithm 4 in terms of robustness. We also evaluate the robustness of the learned policies under both test condition changes, and adversarial disturbances (see Figure 2).

One-Player DDPG: We evaluate the robustness of one-player variants of Algorithm 3, and Algorithm 4, i.e., we consider the NR-MDP setting with $\delta = 0$. In this case, we set $K_t = 1$ for Algorithm 3 (this choice of K_t makes the computational complexity of both algorithms equal). The results are presented in Figures 3 and 4.

Here, we remark that Algorithm 3 with $\delta = 0$, and $K_t = 1$ is simply the standard DDPG with actor being updated by preconditioned version of SGLD. Thus we achieve robustness under different testing conditions with just a simple change in the DDPG algorithm and without additional computational cost.

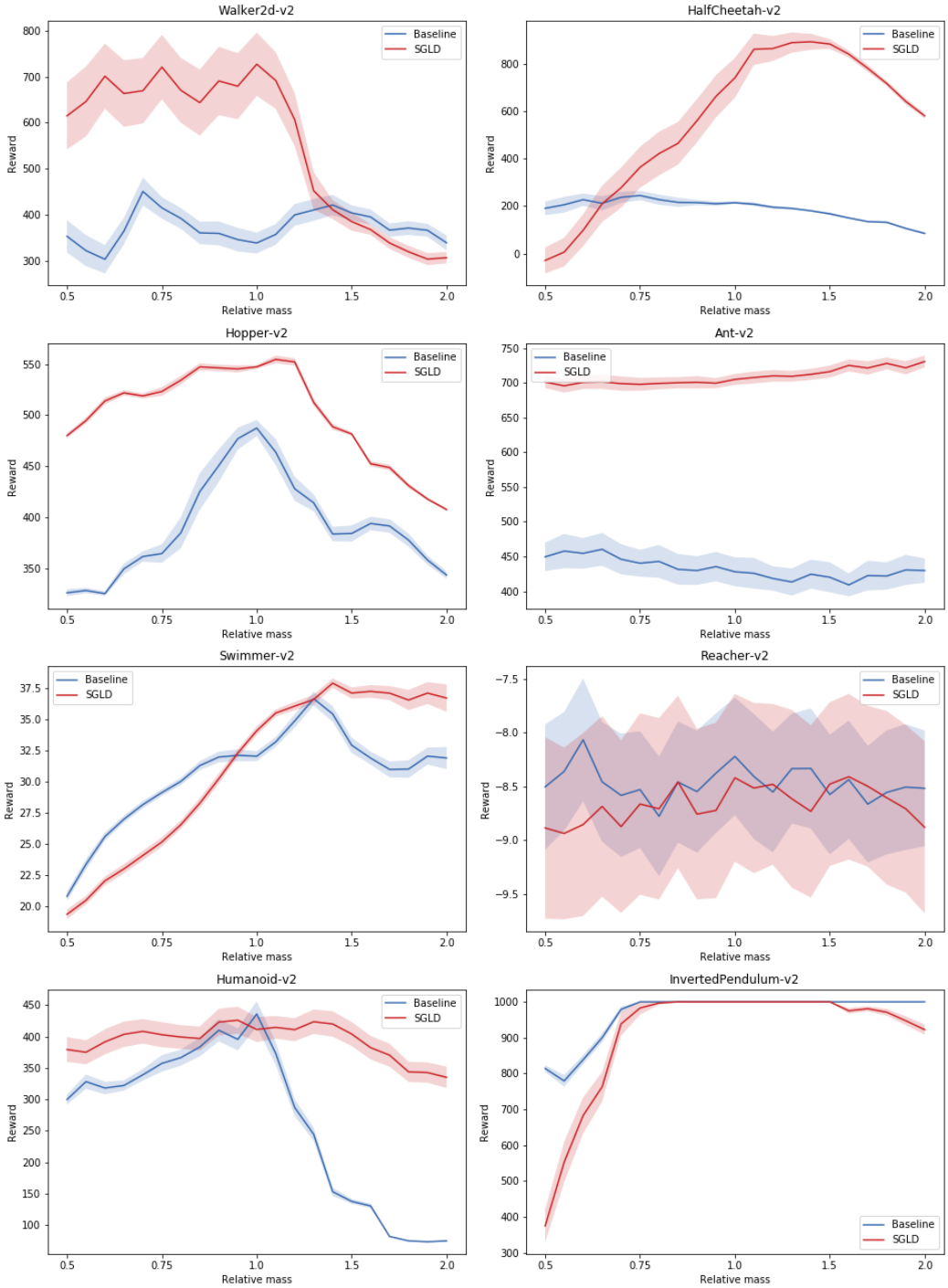

Figure 3: Average performance (over 5 seeds) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0$. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training.

5 Conclusion

In this work, we studied the robust reinforcement learning problem. By adapting the approximate infinite-dimensional entropic mirror descent from (Hsieh et al., 2019), we design a robust variant of

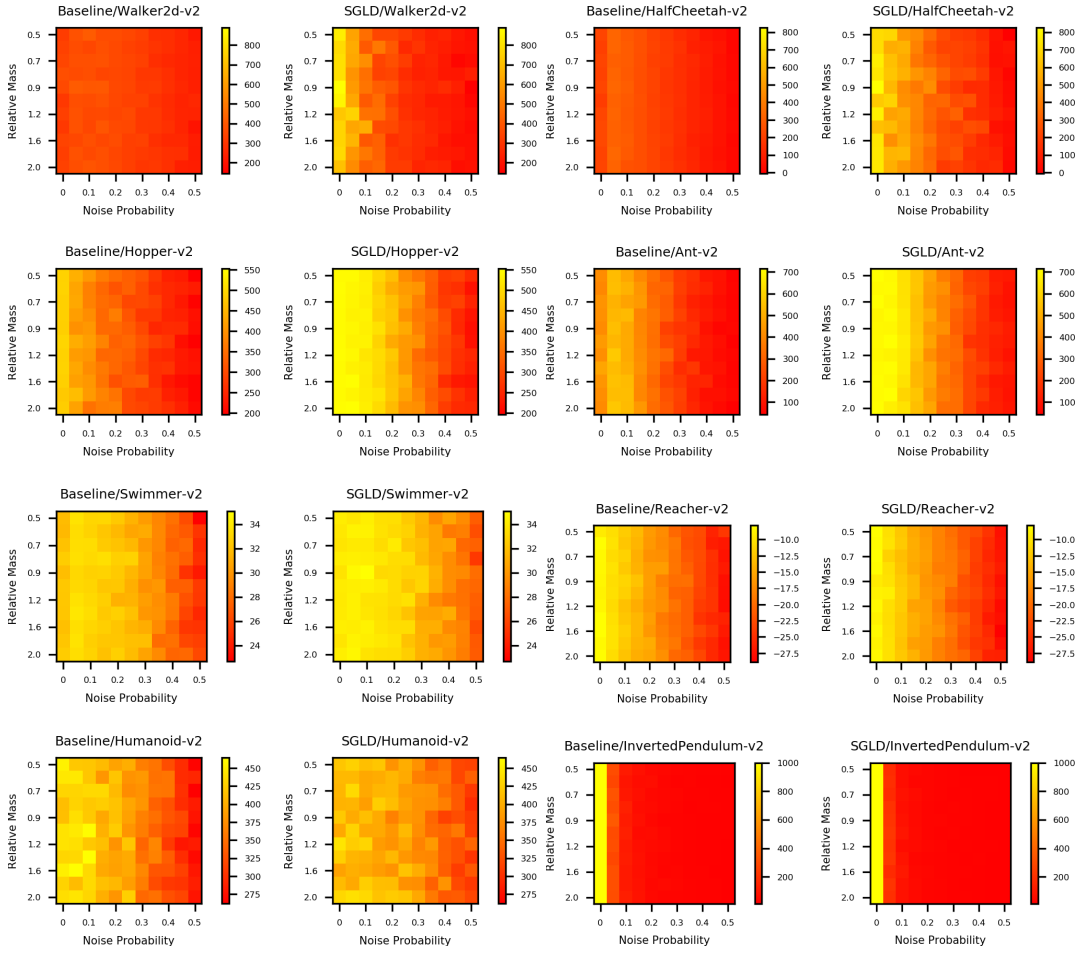

Figure 4: Average performance (over 5 seeds) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0$. The evaluation is performed on a range of noise probability and mass values not encountered during training.

DDPG algorithm, under the NR-MDP setting. In our experiments, we evaluated the robustness of our algorithm on several continuous control tasks, and found that our algorithm clearly outperformed the baseline algorithm from (Tessler et al., 2019).

REFERENCES

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

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.

Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ya-Ping Hsieh, Chen Liu, and Volkan Cevher. Finding mixed nash equilibria of generative adversarial networks. In *International Conference on Machine Learning*, pp. 2810–2819, 2019.

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

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.

Chunyuan Li, Changyou Chen, David E Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI*, volume 2, pp. 4, 2016.

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.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*. Elsevier, 1994.

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.

John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

Arkadi Nemirovski. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum Markov games. In *International Conference on Machine Learning*, 2015.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2017.

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

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

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

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.

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*, pp. 1057–1063, 2000.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

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*, pp. 5026–5033. IEEE, 2012.

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

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)*, pp. 681–688, 2011.

A ADDITIONAL RESULTS

The results for the best performing seeds are presented in Figures 5 and 6.

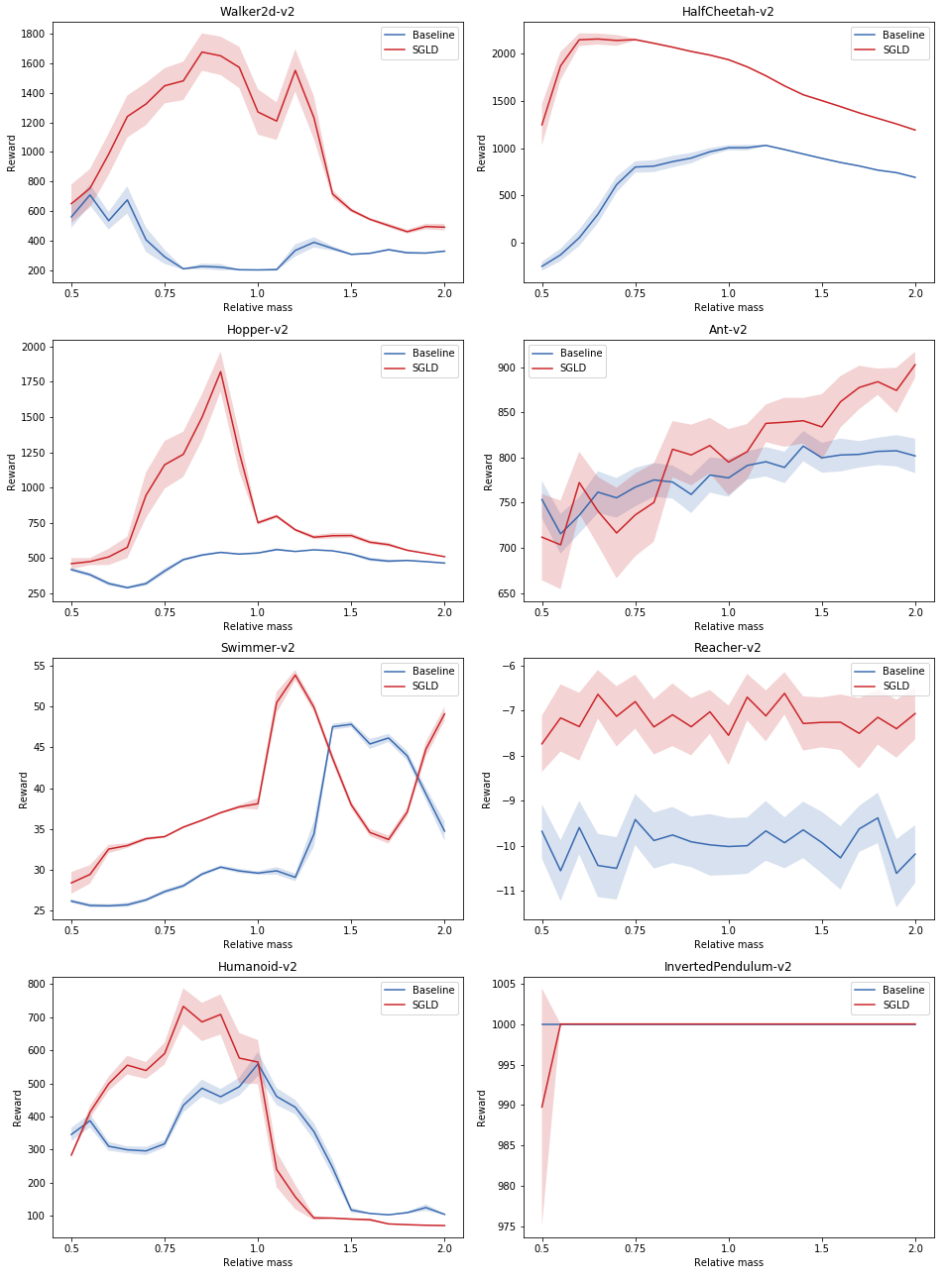

Figure 5: Average performance (of the best performing seed) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0.1$. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training.

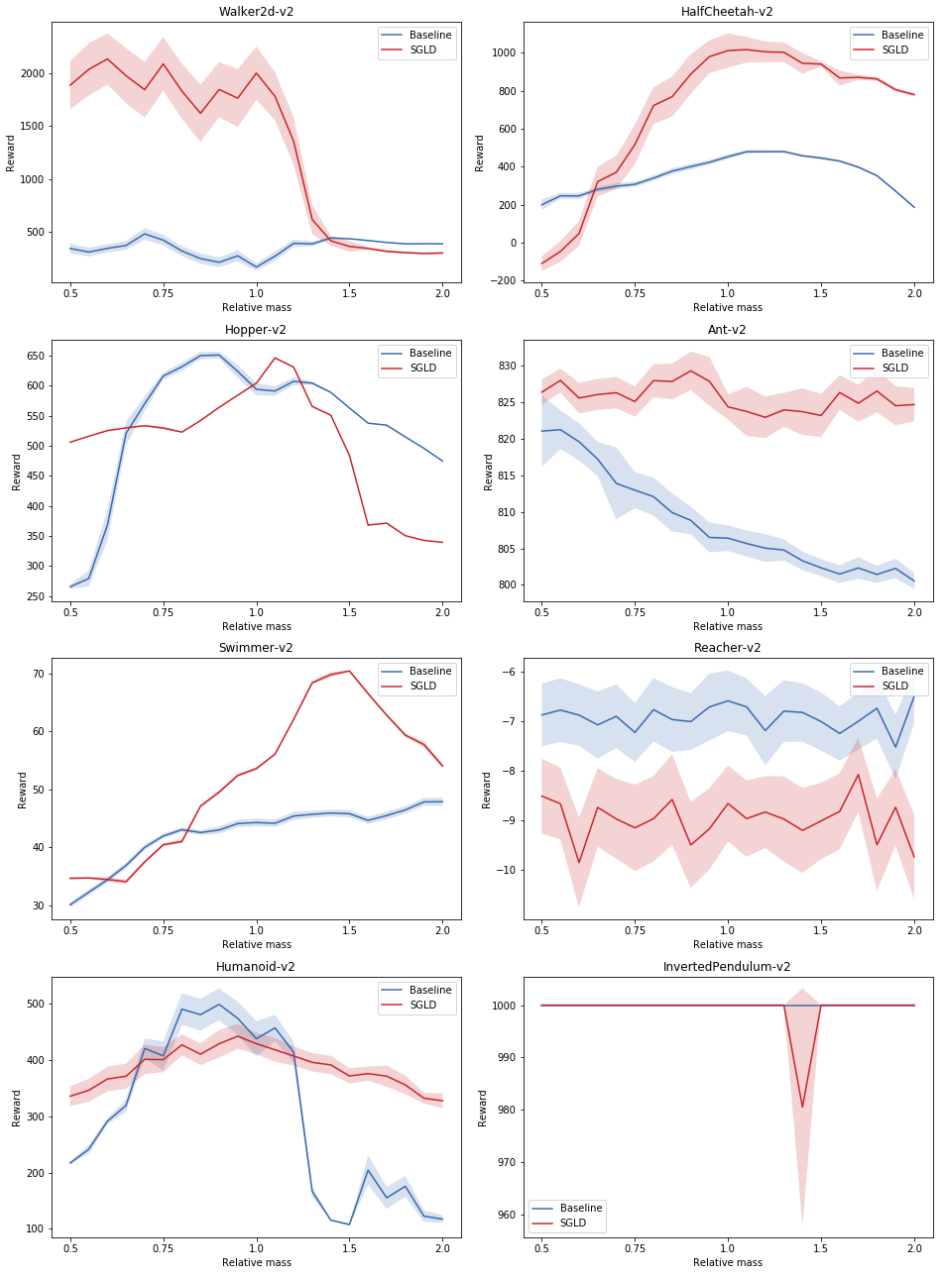

Figure 6: Average performance (of the best performing seed) of Algorithm 3 (—), and Algorithm 4 (—), under the NR-MDP setting with $\delta = 0$. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training.

B Algorithms and Hyperparameter Details

Most of the values for the hyperparameters (in Table 1) are chosen from Dhariwal et al. (2017). In addition, for Algorithm 3, we set

1. thermal noise $\sigma_t = \sigma_0 \times (1 - 5 \times 10^{-5})^t$, where $\sigma_0 \in \left\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\right\}$.
2. warmup steps $K_t = \min\left\{15, \lfloor (1 + 10^{-5})^t \rfloor\right\}$

The best performing values for every environment are presented in Tables 2 and 3.

Algorithm 2 Infinite-Dimensional Entropic Mirror Descent-v2

Input: Initial distributions p_1, q_1, and learning rate η
for $t = 1, 2, \ldots, T-1$ **do**
$p_{t+1}(\theta) \propto \exp\left(+\eta \sum_{s \le t} G q_s(\theta)\right)$
$q_{t+1}(\omega) \propto \exp\left(-\eta \sum_{s \le t} G^{\dagger} p_s(\omega)\right)$
end for
Output: $\bar{p}_T = \frac{1}{T}\sum_{t=1}^{T} p_t$ and $\bar{q}_T = \frac{1}{T}\sum_{t=1}^{T} q_t$

Table 1: Table of Hyperparameters

Hyperparameter	Value
critic optimizer	Adam
critic learning rate	10^{-3}
target update rate τ	0.999
mini-batch size N	128
discount factor γ	0.99
damping factor β	0.9
replay buffer size	10^6
action noise parameter σ	$\{0, 0.01, 0.1, 0.2, 0.3, 0.4\}$
RMSProp parameter α	0.999
RMSProp parameter ϵ	10^{-8}
RMSProp parameter η	10^{-4}

Table 2: Hyperparameters chosen via grid search (for NR-MDP setting with $\delta = 0.1$)

	Algorithm 3: (σ_0, σ)	Algorithm 4: σ
Walker-v2	$(10^{-2}, 0.01)$	0
HalfCheetah-v2	$(10^{-2}, 0)$	0.01
Hopper-v2	$(10^{-3}, 0.2)$	0.2
Ant-v2	$(10^{-4}, 0.2)$	0.4
Swimmer-v2	$(10^{-5}, 0.4)$	0.4
Reacher-v2	$(10^{-3}, 0.2)$	0.4
Humanoid-v2	$(10^{-4}, 0.01)$	0
InvertedPendulum-v2	$(10^{-3}, 0.01)$	0.1

Table 3: Hyperparameters chosen via grid search (for NR-MDP setting with $\delta = 0$)

	Algorithm 3: (σ_0, σ)	Algorithm 4: σ
Walker-v2	$(10^{-2}, 0.1)$	0.01
HalfCheetah-v2	$(10^{-2}, 0.01)$	0.2
Hopper-v2	$(10^{-5}, 0.3)$	0.4
Ant-v2	$(10^{-2}, 0.4)$	0.4
Swimmer-v2	$(10^{-2}, 0.2)$	0.3
Reacher-v2	$(10^{-3}, 0.2)$	0.3
Humanoid-v2	$(10^{-2}, 0.1)$	0
InvertedPendulum-v2	$(10^{-3}, 0)$	0.01

Algorithm 3 Two-Player DDPG with SGLD Actors (SGLD)

Hyperparameters: see Table 1
Initialize (randomly) policy parameters ω_1, θ_1, and Q-function parameter ϕ.
Initialize the target network parameters $\omega_{\text{targ}} \leftarrow \omega_1$, $\theta_{\text{targ}} \leftarrow \theta_1$, and $\phi_{\text{targ}} \leftarrow \phi$.
Initialize replay buffer $\mathcal{D}$.
Initialize $m \leftarrow \mathbf{0}$; $m' \leftarrow \mathbf{0}$.
$t \leftarrow 1$.
repeat
  Observe state s, and select actions $a = \mu_{\theta_t}(s) + \xi$; $a' = \nu_{\omega_t}(s) + \xi'$, where $\xi, \xi' \sim \mathcal{N}(0, \sigma I)$
  Execute the action $\bar{a} = (1-\delta)a + \delta a'$ in the environment.
  Observe reward r, next state s', and done signal d to indicate whether s' is terminal.
  Store $(s, \bar{a}, r, s', d)$ in replay buffer $\mathcal{D}$.
  If s' is terminal, reset the environment state.
  if it's time to update **then**
    for however many updates **do**
      $\bar{\omega}_t, \omega_t^{(1)} \leftarrow \omega_t$; $\bar{\theta}_t, \theta_t^{(1)} \leftarrow \theta_t$
      for $k = 1, 2, \ldots, K_t$ **do**
        Sample a random minibatch of N transitions $B = \{(s, \bar{a}, r, s', d)\}$ from $\mathcal{D}$.
        Compute targets $y(r, s', d) = r + \gamma(1-d) Q_{\phi_{\text{targ}}}\left(s', (1-\delta)\mu_{\theta_{\text{targ}}}(s') + \delta\nu_{\omega_{\text{targ}}}(s')\right)$.

        Update critic by one step of (preconditioned) gradient descent using $\nabla_\phi L(\phi)$, where

$$L(\phi) = \frac{1}{N} \sum_{(s,\bar{a},r,s',d)\in B} \left(y(r, s', d) - Q_\phi(s, \bar{a})\right)^2.$$

        Compute the (agent and adversary) policy gradient estimates:

$$\widehat{\nabla_\theta J(\theta, \omega_t)} = \frac{1-\delta}{N} \sum_{s\in\mathcal{D}} \nabla_\theta \mu_\theta(s) \nabla_{\bar{a}} Q_\phi(s, \bar{a}) |_{\bar{a}=(1-\delta)\mu_\theta(s)+\delta\nu_{\omega_t}(s)}$$

$$\widehat{\nabla_\omega J(\theta_t, \omega)} = \frac{\delta}{N} \sum_{s\in\mathcal{D}} \nabla_\omega \nu_\omega(s) \nabla_{\bar{a}} Q_\phi(s, \bar{a}) |_{\bar{a}=(1-\delta)\mu_{\theta_t}(s)+\delta\nu_\omega(s)}.$$

        $g \leftarrow \left[\widehat{\nabla_\theta J(\theta, \omega_t)}\right]_{\theta=\theta_t^{(k)}}$; $m \leftarrow \alpha m + (1-\alpha) g \odot g$; $C \leftarrow \text{diag}\left(\sqrt{m+\epsilon}\right)$
        $\theta_t^{(k+1)} \leftarrow \theta_t^{(k)} + \eta C^{-1} g + \sqrt{2\eta}\sigma_t C^{-\frac{1}{2}}\xi$, where $\xi \sim \mathcal{N}(0, I)$
        $g' \leftarrow \left[\widehat{\nabla_\omega J(\theta_t, \omega)}\right]_{\omega=\omega_t^{(k)}}$; $m' \leftarrow \alpha m' + (1-\alpha) g' \odot g'$; $D \leftarrow \text{diag}\left(\sqrt{m'+\epsilon}\right)$
        $\omega_t^{(k+1)} \leftarrow \omega_t^{(k)} - \eta D^{-1} g' + \sqrt{2\eta}\sigma_t D^{-\frac{1}{2}}\xi'$, where $\xi' \sim \mathcal{N}(0, I)$
        $\bar{\omega}_t \leftarrow (1-\beta)\bar{\omega}_t + \beta\omega_t^{(k+1)}$; $\bar{\theta}_t \leftarrow (1-\beta)\bar{\theta}_t + \beta\theta_t^{(k+1)}$
        Update the target networks:

$$\begin{aligned}
\phi_{\text{targ}} &\leftarrow \tau\phi_{\text{targ}} + (1-\tau)\phi \\
\theta_{\text{targ}} &\leftarrow \tau\theta_{\text{targ}} + (1-\tau)\theta_t^{(k+1)} \\
\omega_{\text{targ}} &\leftarrow \tau\omega_{\text{targ}} + (1-\tau)\omega_t^{(k+1)}
\end{aligned}$$

      end for
      $\omega_{t+1} \leftarrow (1-\beta)\omega_t + \beta\bar{\omega}_t$; $\theta_{t+1} \leftarrow (1-\beta)\theta_t + \beta\bar{\theta}_t$
      $t \leftarrow t + 1$.
    end for
  end if
until convergence
Output: ω_T, θ_T.

Algorithm 4 Two-Player DDPG with RMSProp Actors (Baseline)

Hyperparameters: see Table 1
Initialize (randomly) policy parameters ω_1, θ_1, and Q-function parameter ϕ.
Initialize the target network parameters $\omega_{\text{targ}} \leftarrow \omega_1$, $\theta_{\text{targ}} \leftarrow \theta_1$, and $\phi_{\text{targ}} \leftarrow \phi$.
Initialize replay buffer $\mathcal{D}$.
Initialize $m \leftarrow \mathbf{0}$; $m' \leftarrow \mathbf{0}$.
$t \leftarrow 1$.
repeat
  Observe state s, and select actions $a = \mu_{\theta_t}(s) + \xi$; $a' = \nu_{\omega_t}(s) + \xi'$, where $\xi, \xi' \sim \mathcal{N}(0, \sigma I)$
  Execute the action $\bar{a} = (1-\delta)a + \delta a'$ in the environment.
  Observe reward r, next state s', and done signal d to indicate whether s' is terminal.
  Store $(s, \bar{a}, r, s', d)$ in replay buffer $\mathcal{D}$.
  If s' is terminal, reset the environment state.
  if it's time to update **then**
    for however many updates **do**
      Sample a random minibatch of N transitions $B = \{(s, \bar{a}, r, s', d)\}$ from $\mathcal{D}$.
      Compute targets $y(r, s', d) = r + \gamma(1-d) Q_{\phi_{\text{targ}}}\left(s', (1-\delta)\mu_{\theta_{\text{targ}}}(s') + \delta\nu_{\omega_{\text{targ}}}(s')\right)$.

      Update critic by one step of (preconditioned) gradient descent using $\nabla_\phi L(\phi)$, where

$$L(\phi) \;=\; \frac{1}{N} \sum_{(s,\bar{a},r,s',d)\in B} \left(y(r,s',d) - Q_\phi(s,\bar{a})\right)^2.$$

      Compute the (agent and adversary) policy gradient estimates:

$$\nabla_\theta \widehat{J(\theta, \omega_t)} \;=\; \frac{1-\delta}{N} \sum_{s\in\mathcal{D}} \nabla_\theta \mu_\theta(s) \nabla_{\bar{a}} Q_\phi(s,\bar{a})\,|_{\bar{a}=(1-\delta)\mu_\theta(s)+\delta\nu_{\omega_t}(s)}$$

$$\nabla_\omega \widehat{J(\theta_t, \omega)} \;=\; \frac{\delta}{N} \sum_{s\in\mathcal{D}} \nabla_\omega \nu_\omega(s) \nabla_{\bar{a}} Q_\phi(s,\bar{a})\,|_{\bar{a}=(1-\delta)\mu_{\theta_t}(s)+\delta\nu_\omega(s)}.$$

      $g \leftarrow \left[\nabla_\theta \widehat{J(\theta, \omega_t)}\right]_{\theta=\theta_t}$; $m \leftarrow \alpha m + (1-\alpha) g \odot g$; $C \leftarrow \text{diag}\left(\sqrt{m+\epsilon}\right)$
      $\theta_{t+1} \leftarrow \theta_t + \eta C^{-1} g$
      $g' \leftarrow \left[\nabla_\omega \widehat{J(\theta_t, \omega)}\right]_{\omega=\omega_t}$; $m' \leftarrow \alpha m' + (1-\alpha) g' \odot g'$; $D \leftarrow \text{diag}\left(\sqrt{m'+\epsilon}\right)$
      $\omega_{t+1} \leftarrow \omega_t - \eta D^{-1} g'$
      Update the target networks:

$$\begin{aligned}
\phi_{\text{targ}} &\leftarrow \tau\phi_{\text{targ}} + (1-\tau)\phi \\
\theta_{\text{targ}} &\leftarrow \tau\theta_{\text{targ}} + (1-\tau)\theta_{t+1} \\
\omega_{\text{targ}} &\leftarrow \tau\omega_{\text{targ}} + (1-\tau)\omega_{t+1}
\end{aligned}$$

      $t \leftarrow t + 1$.
    end for
  end if
until convergence
Output: ω_T, θ_T.