Robust Reinforcement Learning using Adversarial Populations

Anonymous Authors[1]

Abstract

Reinforcement Learning (RL) is an effective tool for controller design but can struggle with issues of robustness, failing catastrophically when the underlying system dynamics are perturbed. The Robust RL formulation tackles this by adding worst-case adversarial noise to the dynamics and constructing the noise distribution as the solution to a zero-sum minimax game. However, existing work on learning solutions to the Robust RL formulation has primarily focused on training a single RL agent against a single adversary. In this work, we demonstrate that using a single adversary does not consistently yield good generalization to new dynamics. Instead, we propose randomly initializing a population of adversaries and sampling from the population uniformly during training. We empirically validate across robotics benchmarks that the single adversary approach results in overfitting to the current adversary whereas the multiple adversaries approach generalizes better to new environments.

1. Introduction

Reinforcement Learning (RL) is a powerful tool for control design but generally relies upon simulators that may contain inaccurate dynamics models. To make (RL) control design viable for deployment, it is necessary to develop techniques that can guarantee robust performance despite misspecified models. One approach is to formulate the problem as a zero-sum game and learn an adversary that perturbs the transition dynamics (Tessler et al., 2019; Kamalaruban et al., 2020; Pinto et al., 2017). If a global Nash equilibrium of this problem is found, then that equilibrium provides a lower bound on the performance of the policy under some bounded set of perturbations. In addition to the benefit of removing user design once the perturbation mechanism is specified, this approach is maximally conservative, which is useful for safety critical applications.

However, the aforementioned literature on learning an adversary predominantly uses a single adversary which we contend can decrease robustness. Consider a robot trying to learn to walk forwards while an adversary outputs a force representing wind. For a fixed, deterministic adversary the agent can learn to cancel the wind by applying a countervailing force. Once the adversary is removed, the robot will still apply the compensatory forces and possibly become unstable. Stochastic Gaussian policies (which are ubiquitous in continuous control) offer little improvement: low entropy policies can be counteracted whereas high entropy policies endow the robot with the prior that the wind cancels on average. Under these standard policy parametrizations, we cannot use an adversary to endow the agent with a prior that a persistent, strong wind could come from any direction.

To tackle this, we introduce **RAP** (Robustness via Adversary Pools): a randomly initialized set of adversaries that we sample from at each rollout and train alongside the agent. If the robot learns to cancel any one of the adversaries effectively, that opens a niche for an adversary to exploit by applying forces in another direction. As the number of adversaries increases, the robot is eventually endowed with the prior that a strong wind could come from any direction and that it must walk carefully to avoid being toppled over.

Our contributions are as follows:

- Using a set of continuous control tasks, we provide evidence that a single adversary does not have a consistent positive impact on the robustness of an RL policy while the use of an adversary population provides improved robustness across all considered examples.
- We investigate the source of the robustness and show that the single adversary policy is exploitable by new adversaries whereas policies trained against randomly initialized adversaries leads to reduced exploitability.
- We demonstrate that adversary populations can be competitive with domain randomization while avoiding potential failure modes of domain randomization.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Formalism

Notation. In this work we use the framework of a multi-agent, finite-horizon, discounted, Markov Decision Process (MDP) (Puterman, 1990). The goal for a given MDP is to find a policy π_θ parametrized by θ that maximizes the expected cumulative discounted reward $J^\theta = \mathbb{E}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t)|\pi_\theta\right]$. The conditional in this expression is a short-hand to indicate that the actions in the MDP are sampled via $a_t \sim \pi_\theta(s_t)$. We denote the agent policy parametrized by weights θ as π_θ and the policy of adversary i as $\bar{\pi}_{\phi_i}$. Actions sampled from the adversary policy $\bar{\pi}_{\phi_i}$ will be written as $\bar{a}_t^i$. We use ξ to denote the parametrization of the system dynamics (e.g. different values of friction, mass, wind, etc.) and the system dynamics for a given state and action as $s_{t+1} \sim f_\xi(s_t, a_t)$.

2.1. Baselines

2.1.1. Single Minimax Adversary

Related Work. The Robust Markov Decision Process (R-MDP) formulation views robustness as maximizing performance subject to uncertainty sets on the transition dynamics of an MDP (Nilim & El Ghaoui, 2005; Lim et al., 2013). One prominent variant of the R-MDP literature is to interpret the perturbations as an adversary and attempt to learn the distribution of the perturbation under a minimax objective (Pinto et al., 2017; Tessler et al., 2019).

Objective. In this formulation, the minimax objective becomes

$$\min_{\phi}\max_{\theta}\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t + \alpha\bar{a}_t)|\pi_\theta,\ \bar{\pi}_\phi\right]$$
$$s_{t+1} \sim f_\phi(s_t, a_t + \alpha\bar{a}_t)$$

where α is a hyperparameter controlling the adversary strength. This formulation is identical to that in (Tessler et al., 2019; Kamalaruban et al., 2020) where they call it the *Noisy Action Robust MDP*. Since the adversary is only able to attack the agent through the actions, there is a restricted class of dynamical systems that it can represent; this set of dynamical systems may not necessarily align with those accessible via domain randomization.

2.1.2. Domain Randomization

Related Work. Domain randomization asks a designer to explicitly define a distribution of environment dynamics that the agent should be robust to. For example, (Peng et al., 2018) varies simulator friction, mass, table height, and controller gain etc. to train a robot to robustly push a puck to a target location. Additionally, domain randomization has been successfully used to build accurate object detectors solely from simulated data (Tobin et al., 2017) and to zero-shot transfer a quadcopter flight policy from simulation (Sadeghi & Levine, 2016).

Objective. We denote the domain over which ξ is drawn from as Ξ and use $\mathcal{P}(\Xi)$ to denote some probability distribution over ξ. The domain randomization objective is

$$\max_{\theta}\mathbb{E}_{\xi\sim\mathcal{P}(\Xi)}\left[\mathbb{E}_{s_{t+1}\sim f_\xi(s_t,a_t)}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t)|\pi_\theta\right]\right]$$

Most commonly, and in this work, the parameters ξ are sampled uniformly over Ξ.

2.2. RAP: Robustness via Adversary Pools

RAP simply extends the minimax objective by adding uniform sampling over a set of adversaries. Denoting $\bar{\pi}_{\phi_i}$ as the i-th adversary and $i \sim U(1, n)$ as the discrete uniform distribution defined on 1 through n, the objective becomes

$$\min_{\phi_1,\ldots,\phi_n}\ \max_{\theta}\mathbb{E}_{i\sim U(1,n)}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t, \alpha\bar{a}_t^i)|\pi_\theta,\ \pi_{\phi_i}\right]$$

Algorithm 1 Robustness via Adversary Pools

```
Initialize $\theta, \phi_1 \cdots \phi_n$ using Xavier initialization (Glorot & Bengio, 2010)
while not converged do
    for rollout j=1...J do
        sample adversary $i \sim U(1, n)$ run policies $\pi_\theta, \pi_{\phi_i}$ in environment till termination
    end
    update $\theta, \phi_1 \cdots \phi_n$ using PPO (Schulman et al., 2017)
end
```

Related Work. The use of pools of agents/adversaries is a standard technique in multi-agent settings. Alphastar uses a population of ”exploiter” agents that fine-tune against the bot to prevent it from developing exploitable strategies (Vinyals et al., 2019). (Czarnecki et al., 2020) establishes conditions under which learning in games can often fail to converge without populations. Closest to our work, Active Domain Randomization (Mehta et al., 2019) uses a pool of ”adversaries” to select domain randomization parameters. However, they use a Stein Variation Policy Gradient (Liu et al., 2017) to ensure diversity in their adversaries and a discriminator reward instead of the negative of the agent reward.

3. Experiments

We present experiments on continuous control tasks from the OpenAI Gym Suite (Brockman et al., 2016; Todorov et al., 2012). We compare with the existing literature and evaluate the efficacy of a pool of learned adversaries across a wide range of state and action space sizes. We investigate the following hypotheses:

H1. Agents are more likely to overfit to a single adversary than a pool of adversaries.

H2. **RAP** leads to increased robustness over training with a minimax adversary.

H3. **RAP** has comparable robustness effects to domain randomization while avoiding optimization challenges caused by the DR objective.

3.1. Experimental Setup and Hyperparameter Selection

We experiment with the Hopper, Ant, and HalfCheetah continuous control environments. Since robustness may not correspond with optimal training time reward, we adopt the train-validate-test split from supervised learning. To generate the validation set, we predefine ranges of mass and friction coefficients as follows: for Hopper, mass $\in [0.7, 1.3]$ and friction $\in [0.7, 1.3]$; Half Cheetah and Ant, mass $\in [0.5, 1.5]$ and friction $\in [0.1, 0.9]$. These values are used to scale mass and friction of all the joints. We compare the robustness of agents trained via **RAP** against agents trained with: 1) a single minimax adversary, 2) domain randomization, and 3) PPO without an adversary. The domain randomization agent is trained via uniform sampling of mass, friction coefficients from the predefined ranges. To generate the test set, we take both the highest and lowest friction coefficients from the validation range and apply them to different combinations of individual geoms.

3.2. Results

H1. Analysis of Overfitting

A globally minimax optimal adversary should be unexploitable and perform equally well against any adversary of equal strength. We investigate the optimality of our policy by asking whether the minimax agent is robust to swaps of adversaries from different training runs, i.e. different seeds. Fig. 1 shows the result of these swaps for the one adversary and three adversary case. The diagonal corresponds to playing against the adversaries the agent was trained with while every other square corresponds to playing against adversaries from a different seed. The agents trained against 3 adversaries are significantly more robust on average under swaps. This suggests that random adversary initializations are sufficient here to generate less exploitable agents.

H2. Adversary Pool Performance

Fig.2 shows the average reward (the average of ten seeds across the validation or test sets respectively) for each environment. Table 1 gives the corresponding numerical values and the percent change of each policy from the baseline.

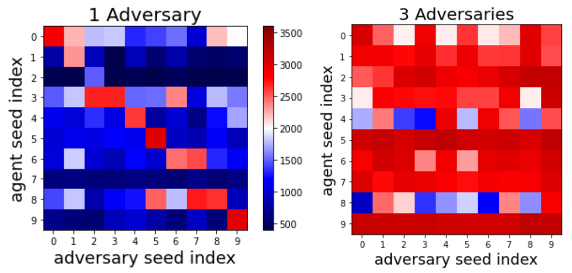

Figure 1. Average cumulative reward under swaps for one adversary training (left) and three-adversary training (right). Each square corresponds to 20 trials. In the three adversary case, each square is the average performance against the adversaries from that seed.

Standard deviations are omitted on the test set due to wide variation in task difficulty; the standard deviation across tasks is not a particularly meaningful quantity. In all environments we achieve a higher reward using **RAP** of size three and/or five when compared to the single minimax adversary case.

For a detailed comparison of robustness across the validation set, Fig. 3 and Fig. 4 show heatmaps of the performance across all the mass, friction coefficient combinations for Hopper and Cheetah. Fig. 3 is an example of a case where a single adversary actually reduces the robustness of the resultant policy whereas additional adversaries consistently increase robustness.

Ant	0 Adv	DR	1 Adv	3 Adv	5 Adv
Mean Rew.	2908	3613	3206	**3272**	3203
% Change		24.3	10.2	**12.5**	10.2
Hopper	**0 Adv**	**DR**	**1 Adv**	**3 Adv**	**5 Adv**
Mean Rew.	472	1636	913	**1598**	1565
% Change		246	93.4	**238**	231
Cheetah	**0 Adv**	**DR**	**1 Adv**	**3 Adv**	**5 Adv**
Mean Rew.	5592	3656	5664	6046	**6406**
% Change		-35	1.3	8.1	**14.6**

Table 1. Average reward and % change from vanilla PPO (0 Adv) for Ant, Hopper, and Cheetah environments across ten seeds averaged over the holdout test set. Across all environments, we see consistently higher robustness using **RAP** than the minimax adversary. Most robust adversarial approach is bolded.

H3. Effect of Domain Randomization Parametrization

From Fig. 2, we see that in the Ant and Hopper domains, the oracle achieves the highest transfer reward in the validation set as expected. Interestingly, we found that the domain randomization policy performed much worse on the Half Cheetah validation set, despite being trained on the validation set. Looking at the performance for each mass and friction combination in Fig. 4, we found that the DR agent was able to perform much better at the low friction coefficients and learned to prioritize those values at the cost of significantly worse performance on average. Naive DR

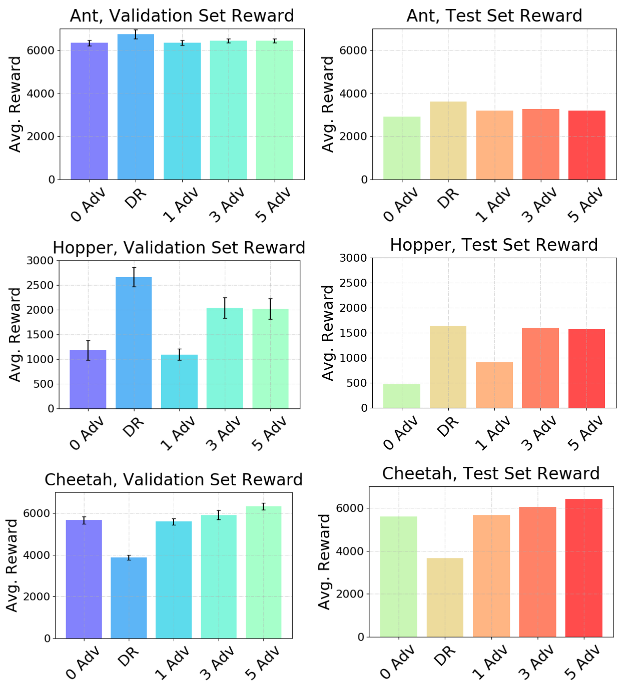

Figure 2. Average reward for Ant (top), Hopper (middle), and Cheetah (left) environments across ten seeds and across the validation set (left column) and across the holdout test set (right column). We compare vanilla PPO (0 adv), the domain randomization oracle, and the minimax adversary against **RAP** of size three and five. Bars represent the mean and the arms represent the std. deviation. Both are computed over 20 rollouts for each test-set sample.

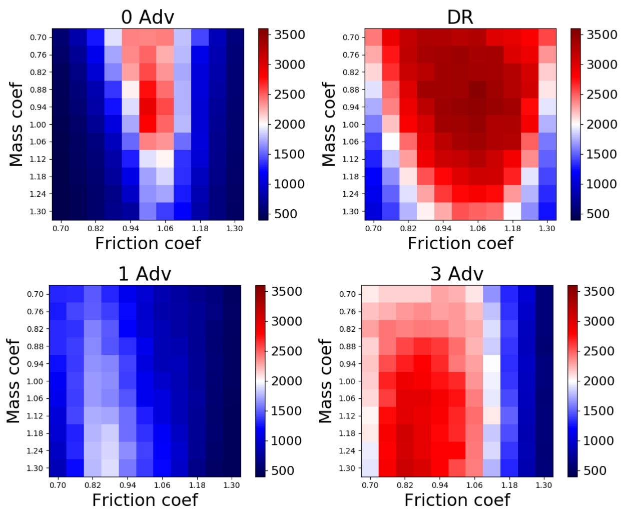

Figure 3. Hopper scores across validation set with friction on x-axis and mass on y-axis. No adversaries (upper left), domain randomization (upper right), one adversary (bottom left), three adversaries (bottom right).

parametrizations can cause the policy to exploit subsets of the randomized domain and lead to a brittle policy. Furthermore, since the validation set is the training set for domain randomization, its failure to outperform **RAP** there suggests additional optimization challenges are occurring. Notably, the adversary-based methods are not susceptible to the same parametrization issues.

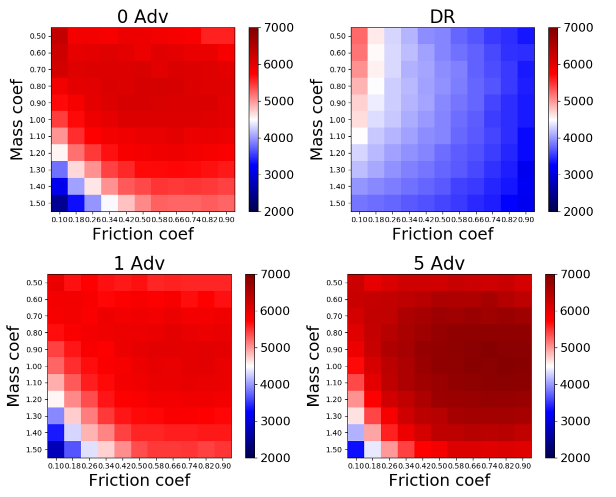

Figure 4. Half Cheetah scores across validation set with friction on x-axis and mass on y-axis. No adversaries (upper left), domain randomization (upper right), one adversary (bottom left), three adversaries (bottom right).

4. Conclusions

In this work we demonstrate that the use of a single adversary to approximate the solution to a minimax problem does not consistently lead to improved robustness. We propose a solution through the use of multiple adversaries (**RAP**), and demonstrate that this provides robustness across a variety of robotics benchmarks. We also compare **RAP** with domain randomization and demonstrate that while DR can lead to a more robust policy, it requires careful parametrization of the domain. **RAP** is more general, allowing for use in domains where appropriate tuning requires extensive prior knowledge or expertise.

There are some interesting extensions of this work that we would like to pursue. Our agents are currently memory-less and therefore cannot perform adversary identification; it would be worthwhile to see if identification of the adversaries improves performance. Our adversaries can also be viewed as forming a task distribution, allowing them to be used in continual learning approaches like MAML (Nagabandi et al., 2018) where domain randomization is frequently used to construct task distributions. Finally, we would like to construct a theoretical characterization explaining why multiple adversaries are beneficial.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Czarnecki, W. M., Gidel, G., Tracey, B., Tuyls, K., Omidshafiei, S., Balduzzi, D., and Jaderberg, M. Real world games look like spinning tops. *arXiv preprint arXiv:2004.09468*, 2020.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V. Robust reinforcement learning via adversarial training with langevin dynamics. *arXiv preprint arXiv:2002.06063*, 2020.

Lim, S. H., Xu, H., and Mannor, S. Reinforcement learning in robust markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 701–709, 2013.

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L. Active domain randomization. *arXiv preprint arXiv:1904.04762*, 2019.

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018.

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–8. IEEE, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.