Toward Robust Deep RL via Better Benchmarks: Identifying Neglected Problem Dimensions

Olov Andersson
Department of Computer Science
Linköping University
581 83, Linköping, Sweden
olov.a.andersson@liu.se

Patrick Doherty
Department of Computer Science
Linköping University
581 83, Linköping, Sweden
patrick.doherty@liu.se

Abstract

We consider a wider sense of reproducibility *in practice* for deep reinforcement learning on control tasks, where it holds promise as a general purpose solution. However, unlike other branches of machine learning, for practical reasons only simulation data is used in RL benchmarks. This puts greater responsibility on experiment design for advances on benchmarks to translate to real-world performance. There has been a string of successes with solving ever more impressive-looking problems in the Mujoco physics simulator. However, while involving control with different agent dynamics, there is reason to believe that these benchmarks are in other respects fairly homogeneous, e.g. having deterministic state transitions with quadratic objectives. For instance, [9] recently showed that many of them are solvable by a simple linear policy and suggest widening the initial state distribution to avoid fragile "trajectory-centric" policies. We take this further and argue that the environments themselves should have both an *inherent* uncertainty, and more complex objectives. Autonomous robots and vehicles have to deal with a complex uncertain world that changes over time. To exemplify this we introduce a seemingly simple benchmark where randomly moving obstacles have to be avoided. This both precludes a trajectory-centric solution and involves a more difficult objective than commonly used. We show that this toy example actually has higher sample complexity than the Mujoco benchmarks. We further find that the level of robustness expected of real-world autonomous agents leads to requirements on tail-end convergence that is rarely given much consideration in benchmarks, resulting in difficult tuning problems. While some environment variation has recently been shown for e.g. complex humanoid agents in Mujoco, with training times increasingly requiring massive cloud infrastructure, we finally suggest that such toy examples designed to test different problem dimensions have an important role to play in research.

1 Introduction

Reinforcement learning is a very general approach for control and planning under uncertainty. However, as real-world agents such as robots and autonomous vehicles are cumbersome to work with, by tradition only simulation benchmarks are used in the ML community. Simulation is a powerful tool for speeding up development, but it also carries a risk of neglecting important aspects of reality. The heavy reliance on simulation benchmarks makes reinforcement learning research particularly vulnerable to this.

For e.g. game AI, the game environment is already formalized, with few choices in how to codify it. However, when constructing benchmarks for continuous-domain control tasks, we argue that they

2nd Reproducibility in Machine Learning Workshop at ICML 2018, Stockholm, Sweden.

should capture sufficient real-world complexity to be a useful stand-in for real-world autonomous robots or vehicles. For example, the standard benchmarks for control tasks consist of mainly static locomotion tasks in the Mujoco physics simulator [13]. These tasks involve complex legged agents with contacts and non-linear dynamics. If more inspired by biology than commonly used robots and vehicles, up to the fidelity of the simulator, these are at least potentially challenging control problems. However, learning how to move from one fixed state to another is a much simpler task than learning what to do over a wide state distribution. In fact, in a fully deterministic environment, the optimal policy in the former case can be reduced to a trajectory between the two states. Real autonomous robots on the other hand need to work robustly in a wide range of situations that may be difficult to predict in advance.

Rajeswaran et al. [9] recently did perturbation testing of learned policies in the standard Mujoco benchmarks and found them fragile. They suggested widening the start distribution and changing the termination conditions. They further argued that in a trajectory context, the results were not as impressive when compared to prior work with trajectory optimization methods. We take this recommendation further and argue for more varied environments and tasks, better reflecting the reality faced by autonomous robot in the real world.

1.1 Environments with Inherent Uncertainty

We argue that environments should have inherent uncertainty, for example such that the underlying dynamics $p(x_{t+1}|x_t, a_t)$ are stochastic. Popoular benchmarks often use entirely deterministic dynamics, e.g. $x_{t+1} = f(x_t, a_t)$, even if a priori unknown. Partial observability is another, more indirect, way of introducing inherent uncertainty. The real world also exhibits changes over time, which is how humans learn to generalize, and it is after all rarely entirely predictable. Introducing inherent uncertainty is a natural way of making environments more realistic and guarantees that the policy will never entirely collapse to a trajectory.

For future reference, we introduce a taxanomy with four levels of complexity for generating the state distribution in robot control environments:

1. Fully deterministic and observable state
2. Randomized start or goal state only
3. Inherently uncertain dynamics or state
4. Real-world in-the-loop

The standard Mujoco benchmarks (see e.g. their Gym implementation [3]), fall mostly in the first category with only little randomness in initial state.

Some further steps in this direction have been taken, e.g. OpenAI recently suggested manipulation tasks with randomized goals [8]. Individual papers have also drawn on all of these levels, particularly on the robotics side, but such environments are rarely published or used in RL research. DeepMind showed humanoids trained on an obstacle course with some elements of randomization [5], but they do not appear to report robustness or environments in sufficient detail to deduce how well it represents real-world situations. Manual perturbations, which can be seen as an informal source of inherent uncertainty for exploratory robustness analysis, were also used in both [5] and [9].

We note that for episodic environments, randomized initial state can still be a powerful way to manually widen the average state distribution, if reasonable values for such is known in advance. However, the policy and dynamics will tend to induce some stationary distribution in longer episodes, such that the initial values only end up having a small effect on the average state. Further, a wide variety of real-world problems are inherently uncertain by nature, e.g. what distance to keep in traffic, or how to navigate a blind intersection. Non-deterministic environments such as these are challenging for trajectory optimization methods, often requiring approximations to deal with uncertainty [2].

1.2 Non-quadratic Objectives

The focus for benchmark difficulty has traditionally been on introducing agents with more complex, if deterministic, dynamics. The environments are mostly empty, the objectives are simple, and often static. The complexity of the reward function is a problem dimension which coupled with the state distribution has a large impact on overall task complexity. Typically this is taken to be a quadratic goal deviation, with similar penalty on action. This is a very well-behaved objective, which under linear dynamics is even convex. In practice, it may be more important for a real robot to avoid bad outcomes rather than to reach its goal. As an example, take autonomous vehicles navigating difficult traffic. Avoiding a crash is paramount, and an agent may even end up in situations where all its options are bad, such that it has to weigh them against each other.

We demonstrate these concepts by introducing a seemingly simply robot navigation problem. This uses the simplest possible robot dynamics, but includes inherent uncertainty in how obstacles move, and the penalty on obstacle collisions make the objective more challenging. We show that this seemingly simple problem has higher data complexity than standard Mujoco benchmarks.

2 Introducing a Simple Benchmark with Stochastic Obstacles

We introduce a toy problem with an autonomous robot and randomly moving obstacles. We let the agent dynamics be governed by an M-dimensional Nth-order discrete-time integrator, a simplification of any system expressible as a system of linear differential equations. These are well studied in the control literature and, depending on any additional constraints, may be convex or even have simple analytic solutions. A second-order integrator acts as a pointmass under controlled forces, and e.g. quadcopter dynamics linearized around hover can be seen as a 4th or 5th order integrator. For this paper we only need a first-order integrator, defined as $x_{t+1} = x_t + 0.1a$, where in the following we assume the state is two-dimensional such that it moves in the plane. E.g. it can directly control velocity in 10Hz, which we constrain to $[-2, 2]$ m/s.

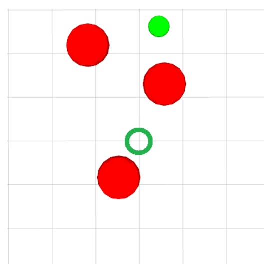


Figure 1: Simple stochastic environment. Red spheres are moving obstacles, the green sphere is the agent. Agent destination is in the dark green circle.

This undeniably simple convex problem is than augmented with up to O=3 spherical obstacles with 1m diameter as in Figure 1. These are given random destinations in a 3x3 square, governed by a simple proportional controller with a max acceleration and velocity of 1m/s. The agent is penalized both with distance to its preferred position in the center of the square, as well as heavily penalized for running into any obstacle (or vice versa). The final cost function is chosen as

$$r(s, a) = -\|p_a - p_D\|^2 - 500 \sum_{i=0}^{O} \mathtt{col}(p_a, p_{o_i}), \tag{1}$$

where p_a, p_D and p_{o_i} are the positions of the agent, its destination, and obstacle i respectively. The collision function `col` is defined as the normalized penetration distance into an obstacle, where 1 is complete overlap. While for simplicity we ignore contact dynamics, we stress that it could not be a replacement for a contact penalty, as it could be optimal to bounce or decelerate against obstacles. The penalty is chosen such that displacement from the center should always be preferable to any significant collision. With such a simple penalty, we expect that very small touches may still be optimal, and for real applications one may want to add a small safety margin or use a more sophisticated penalty method.

The real difficulty in this environment comes from the randomly moving obstacles. Even if we here assume their current position and velocity are fully observable, their future motion is uncertain. Inherently uncertain problems such as these require a non-linear feedback policy and often cannot be solved by classical trajectory optimization methods without approximations, e.g. [2]. Deep RL therefore offers a promising avenue for solving such problems. In this simple example, the agent is both faster and more nimble than the obstacles (it has no inertia). It should only need to plan 10-20 steps ahead and the discount factor is accordingly set to $\gamma = 0.95$. Using a higher-order integrator will impose more dynamical constraints on agent motion, also requiring harder safety considerations to efficiently account for obstacle uncertainty.

3 Experiments

We use the popular PPO algoritm, a trust-region type of policy gradient approach. Policy parameters θ_i are optimized on a surrogate surface constructed from previous trajectories via an importance sampling approximation [7, 10],

$$L(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta_i}(a_t|s_t)}{\pi_{\theta_{i-1}}(a_t|s_t)} \hat{A}_t \right]. \tag{2}$$

This surrogate surface allows reusing data by running several policy gradient epochs within a trust region. In PPO this is enforced implicitly via a clipped objective controlled by ϵ, see [12] for details. The advantage function $\hat{A}_t$ is estimated via generalized advantage estimation (GAE) [11].

We used the `PPO2` implementation from the popular OpenAI Baselines library [4]. As deep reinforcement learning can be sensitive to hyperparameter choices [6], our strategy was to use the same normalization of state and rewards as the Mujoco defaults in Baselines, hoping to allow use of parameters close to those in [12]. After extensive experimentation with individual parameters, we found that the performance of PPO on our domain was most sensitive to step size α, followed by the number of training epochs per batch of data. We tried adjusting ϵ, which regulates the trust-region via clipping, but results were inconsistent. We also attempted to change the λ in GAE within the suggested range of $[0.9, 1.0]$, but it did not appear to improve final performance. Parameters used in the results were defaults for step size $\alpha = 3 \times 10^{-4}$, $\lambda = 0.95$, batch size 2048. However, 5 (vs. 10) epochs per batch was as high we could go without severe oscillations. The obstacle avoidance environment[1] itself is implemented in OpenAI Gym and has 128 steps per episode.

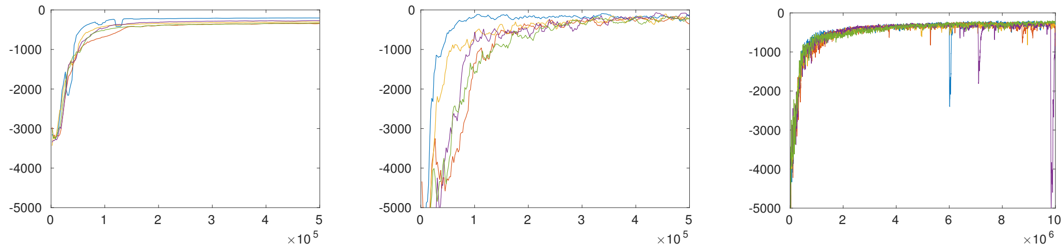
3.1 The Difficulty of Stochastic Obstacles

We test three scenarios of increasing difficulty in this domain, with five environment seeds each. We begin with a very narrow initial state distribution for the agent like in the standard Mujoco benchmarks, augmented with three static obstacles whose positions are drawn and fixed in the first episode. As seen in Figure 2a, this is very easy to solve. Faster than most results reported for PPO on the standard Mujoco benchmarks [12], where 1M steps was used as the upper limit. In 2b we instead draw the initial agent position uniformly from the 3x3m center area, also with static obstacles. Interestingly it is only slightly slower. We suspect this is because policy noice forced the agent to explore much of the space during training anyway, and this problem is sufficiently simple where forgetting is not an issue. Since the obstacle positions are fixed per seed, it only needs to learn to navigate this fixed world. This is verified by observing that it does not generalize well to a different environment seed. Finally, we extend this to also have the obstacles move randomly. We see in Figure 2c that it takes at least an order of magnitude more experience (note scale difference) to handle all the situations in this case, underscoring that it has learned a much harder problem than when just randomizing start position. Finally, in Figure 3a we verify that contrary to some other benchmarks, a linear policy is in fact unable to solve this problem. Interestingly, with a static environment it can still attain a result that is within 100 reward from the NN policy. Although this does not necessarily make it a good policy, without sufficient randomization linear policies can perform unexpectedly well (c.f. [9]).

3.2 Convergence and Robustness

While it is easy to think it has converged at 5M steps by looking at the learning curve in Figure 2c, even with policy noise set to zero, it is actually still making inexplicable choices resulting in collisions. By instead looking at log-plot of the learning curve in Figure 3b, we can see that some runs may be stuck, most likely due to problematic exploration noise, others are still converging — but very slowly — out at 40M. The sample complexity is here getting in range of the full humanoid scenarios

[1]We plan to make the code available.

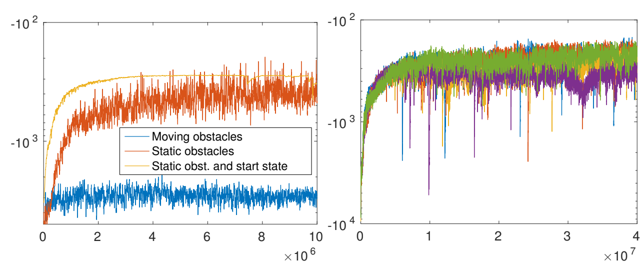


(a) (b) (c)

Figure 2: Reward curves for learning with a) Static initial state and obstacles (0.5M steps) (b) Random initial state but static obstacles (0.5M steps) (c) Random initial state and moving obstacles (10M steps).

in [12], a vastly more complex dynamical system. Some error is expected as modern deep learning algorithms like PPO rely on several approximations, but the rate of hard collisions (>20% overlap) stubbornly remains around 0.1-0.2/min after 40M steps. We tried reducing learning rates accordingly. The occasional oscillations seen in learning curves are partially a result of tuning for as high learning speed as possible without visibly impacting convergence. We tried runs up to 120M with more conservative settings on α, epochs, and ϵ, with only small improvements in collision rate. As neural network policies trained via imitation learning have previously solved more difficult variations of this problem[1], it should be possible to find such a policy. This does not necessarily mean that PPO cannot converge on this domain, but it requires navigating a potentially difficult interaction of four parameters governing step selection and exploration noise.

What is clear is that convergence is an aspect rarely given much attention. As robustness is important for a range of applications, we believe that testing convergence on a diverse set of toy problems such as this has an important role in algorithm development.



(a) (b)

Figure 3: a) Linear policy log-learning curve for different scenarios. b) Log-plot for the NN runs from Figure 2c out to 40M steps. Runs are either still slowly converging or seemingly stuck.

4 Discussion and Conclusions

We argue that while simulation benchmarks in deep RL have tried to capture realism by moving toward ever more complex agent dynamics, the state distribution over which the policy works, as well as the complexity of the reward function, have been mostly neglected. Real-world control applications introduce requirements along both of these problem dimensions. We suggested environments with inherent uncertainty to better reflect real-world conditions for autonomous robots and vehicles. The relevance of these problem dimensions were demonstrated by introducing a toy example with an agent with simplest possible dynamics, but stochastic moving obstacles and penalties for collisions. This toy problem was actually harder to solve than most standard Mujoco benchmarks, which feature more complex dynamics. This shows the importance of being very clear about under what state distribution a benchmark result is expected to work. Further, while not as visually impressive as e.g. humanoids, with computational requirements exploding (e.g. [5] train for 64x100 CPU hours), we suggest toy examples carefully constructed to test different problem dimensions still have a role to play in research.

While sensitivity to hyperparameters has been previously reported [6], we encountered difficulties particularly with tail-end convergence, which resulted in difficult tuning problems. The agent would occasionally make poor choices leading to collisions. Such robustness is crucial in real-world applications such as autonomous robots and vehicles, where a mistake may lead to injury. This aspect also probably merits more representation in RL benchmarks, where initial speed in learning curves often drown out tail-end performance. Perhaps via log-plots or via task-specific success criteria.

5 Acknowledgments

This work is partially supported by grants from the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, the Swedish Foundation for Strategic Research (SSF) project Symbicloud, the Swedish Research Council (VR) Linnaeus Center CADICS, and the ELLIIT Excellence Center at Linköping-Lund for Information Technology.

References

[1] O. Andersson, M. Wzorek, and P. Doherty. Deep Learning Quadcopter Control via Risk-Aware Active Learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 3812–3818. AAAI Press, 2017.

[2] O. Andersson, M. Wzorek, P. Rudol, and P. Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4597–4604, May 2016.

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

[4] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

[5] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2018.

[7] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[8] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

[9] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6553–6564, 2017.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[11] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

[13] E. Todorov, T. Erez, and Y. 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.