ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

Deep RL for Autonomous Robots: Limitations and Safety Challenges

Olov Andersson[1] and Patrick Doherty[1] *

1- Linköping University - Department of Computer Science
581 83, Linköping - Sweden

Abstract. With the rise of deep reinforcement learning, there has also been a string of successes on continuous control problems using physics simulators. This has lead to some optimism regarding use in autonomous robots and vehicles. However, to successful apply such techniques to the real world requires a firm grasp of their limitations. As recent work has raised questions of how diverse these simulation benchmarks really are, we here instead analyze a popular deep RL approach on toy examples from robot obstacle avoidance. We find that these converge very slowly, if at all, to safe policies. We identify convergence issues on stochastic environments and local minima as problems that warrant more attention for safety-critical control applications.

1 Introduction

Deep reinforcement learning hold promise as a general-purpose solution to planning and control under uncertainty, problems which are also pervasive for real-world autonomous robots and vehicles. However, as real-world agents are cumbersome to work with, the deep RL community relies almost entirely on simulation benchmarks. Sufficiently accurate simulators could potentially also be used as part of the work-flow in real applications, for example to mitigate the most well-known limitation of deep RL, the large number of environment samples needed. However, this requires great care as simulations always carry a risk of neglecting important aspects of reality. The heavy reliance on simulation benchmarks makes reinforcement learning research as a whole particularly vulnerable to this.

Here we focus exclusively on continuous action tasks. The standard benchmarks consist of mainly static locomotion tasks in the Mujoco physics simulator [2]. These tasks involve complex legged agents with non-linearities and contact dynamics. Although inspired more by biology than commonly used robots and vehicles, up to the fidelity of the simulator, these are seemingly 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 wider distribution of states.

Rajeswaran et al. [1] recently did perturbation testing of learned policies in the standard Mujoco benchmarks and found them fragile. They also showed that most are solvable even by linear policies. They suggested widening the distribution of initial states and improving termination conditions. We take this recommendation further and argue for more varied environments and tasks, better reflecting the reality faced by autonomous robots in the real world. In particular, more uncertainty, and more complex objectives where unsafe states also have to be avoided.

*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, and the ELLIIT Excellence Center at Linköping-Lund for Information Technology.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

While the environment state transition in RL is typically seen as an unknown probability density function $p(x_{t+1}|x_t, a_t)$, the underlying dynamics in standard control benchmarks is usually a deterministic function $x_{t+1} = f(x_t, a_t)$. Further, the state is fully observed, and there is little randomization of the start and goal states [3]. This makes the problem much easier. In fact, in a fully deterministic environment, the optimal policy can be reduced to a trajectory from start to goal state. There has only recently been some steps toward improving this, e.g. randomizing start [1] or goal [4] states. Real autonomous robots on the other hand need to both work robustly in a wide range of situations, and under a great deal of uncertainty. It cannot observe what is behind a wall, or the internal state of other agents, which makes the environment appear stochastic.

The focus of RL control benchmarks has instead been on agents with more complex, if deterministic, dynamics. The environments are typically empty and reward functions often static, typically near-quadratic in deviation from a fixed goal state and action penalties. This is a very well-behaved objective, which under linear dynamics is even convex. For real autonomous robots, it may be more important 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 and have to be rationally weighed against each other.

To demonstrate the importance of these factors we consider the problem of robot collision avoidance with moving obstacles [5], like pedestrians or cars in traffic. In contrast to the standard benchmarks, this example intentionally uses the simplest possible robot dynamics, but includes uncertainty in how the obstacles move, and a more challenging objective via a penalty on collisions. We test a popular deep RL approach and find that this seemingly simple problem apparently poses more of a challenge than standard benchmarks, and that learned policies make mistakes that could cause injury on a real autonomous robot.

2 Toy Example: Robot Obstacle Avoidance

We introduce a toy problem with an autonomous robot and randomly moving obstacles. We intentionally use the simplest possible robot motion, a first-order integrator, defined as $x_{t+1} = f(x_t, a_t) = x_t + 0.1a_t$, where the state x_t is the position of the agent. This is equivalent to actions directly controlling its velocity in 10Hz using Euler integration. In the following we assume the state is two-dimensional such that it moves in the plane. Simple integrators are well-studied in the control literature, and in this case could be both convex and admit closed-form solution. However, since real robots have physical limits we constrain actions to $[-2, 2]$ m/s.

This undeniably simple problem is augmented with up to $O = 3$ spherical obstacles with 1m diameter as in Fig. 1a. These are given random destinations in a 3x3m 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 goal position, in the center of the square, as well as heavily penalized for collisions. The cost function is chosen as

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

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

(a) (b)

Fig. 1: a) Simplified RL environment with moving obstacles (velocities represented as arrows) b) Collision example, the agent rams into the top obstacle.

where p_a, p_D and p_{o_i} are the positions of the agent, its destination, and obstacle i respectively. The collision function is defined as the normalized penetration distance into an obstacle, where 1 is complete overlap. For simplicity we ignore contact dynamics, but we stress that it could not be a replacement for a contact penalty, as it could otherwise be optimal to bounce or decelerate against obstacles. The penalty is chosen such that displacement from the destination should always be preferable to any significant collision. With such a simple penalty, we expect that minor touches may still be optimal, and for real applications one may want to add a 5cm safety margin or use a more sophisticated penalty method.

As obstacle motion is stochastic, effective solutions require planning under uncertainty. However, this version is fairly simple as the agent is both faster and more nimble than the obstacles (it has no inertia). The discount factor is set to $\gamma = 0.96$ as it should only need to plan 10-20 steps ahead to avoid collision.

3 Experiments

The policy gradient (PG) family of methods tend to be currently preferred for continuous control problems in RL. These are on-policy, such that the policy π_θ is stochastic. Typically a neural network plus $\mathcal{N}(0, \sigma_\pi^2)$ action "exploration noise" is used. Here we use the popular proximal policy optimization (PPO) algorithm [6], as it is considered to perform well with minimal tuning. In PPO the policy π_θ is optimized on a surrogate surface, constructed from previous trajectories via an importance sampling approximation,

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

This surrogate surface allows reusing data batch i by running several policy optimization epochs within a trust region. In PPO this is enforced implicitly via a clipped objective controlled by ϵ, see [6] for details. Here $\hat{A}_t$ is the improvement in reward, estimated by generalized advantage estimation (GAE) [7].

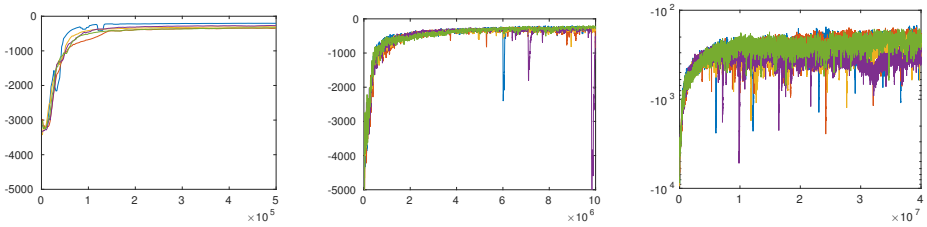
We used the `PPO2` implementation from the popular OpenAI Baselines library [8]. As deep reinforcement learning can be sensitive to hyperparameter choices [9], our strategy was to use the same normalization of state and rewards as the Mujoco defaults in Baselines, hoping to allow use of parameters close to those in [6]. After extensive experimentation with individual parameters, we found that

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

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, and GAE λ within the suggested range of $[0.9, 1.0]$, but found no significant improvement. 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 oscillations. The environment[1] is implemented in OpenAI Gym and has 128 steps per episode.

3.1 Results on Stochastic Obstacle Domain

To establish a baseline we begin with a nearly deterministic scenario like in the standard Mujoco benchmarks. Using a narrow initial state distribution for the agent, augmented with three *static* obstacles whose positions are drawn and fixed in the first episode. The learning curves for five seeds can be seen in Fig. 2a, it is as expected faster than most results reported for PPO on standard Mujoco benchmarks [6]. We then extend this to also have the obstacles move randomly. We see in Fig. 2b that it takes at least an order of magnitude more experience (note axis scale) to handle all the situations in this case, underscoring that it is learning a much harder problem.



(a) (b) (c)

Fig. 2: Reward curves for learning with a) Static initial state and obstacles (0.5M steps) (b) Random initial state and moving obstacles (10M steps) (c) Log-plot out to 40M steps. Runs are either still slowly converging or seemingly stuck.

While it is easy to think it has converged at 5M steps by looking at the learning curve in Fig. 2b, even with policy noise set to zero, it sometimes still makes inexplicable choices as in Fig. 1b. By instead looking at log-plot of the learning curve in Fig. 2c, 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 samples needed is here getting in range of the full humanoid scenarios in [6], a vastly more complex dynamical system. Some small touches are expected as we use a simple penalty function, but the rate of hard collisions ($> 20\%$ overlap) stubbornly remains around 0.1-0.2/min after 40M steps. We tried reducing learning rates, and 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 produced safe policies for more difficult variations of this problem[10], it should be possible to find safe policies here. This by itself does not necessarily mean that it is impossible to get PPO to converge on

[1] Code: `https://github.com/olov-andersson/rl_obstacle_avoidance`

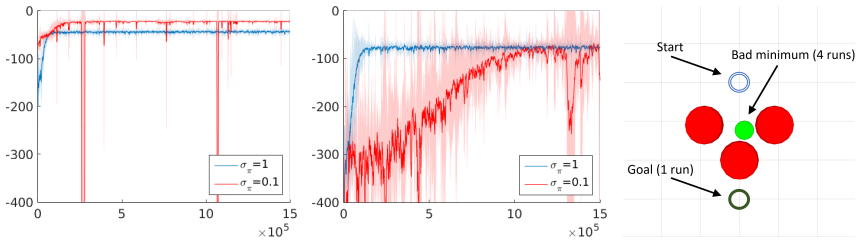
ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

this domain, but it may require an exhaustive search over a non-trivial interaction between four parameters governing step selection and exploration noise.

3.2 Isolating Likely Causes of the Convergence Problems

To better understand why it failed to converge to a safe policy, we try to isolate the causes by simplifying the problem even further. We also turn off PPO features like GAE that could cause bias. We first consider the impact of policy exploration noise on convergence, and then we look for potential issues with local minima.

The first experiment is a 1D version with one static obstacle covering the goal. The objective is to get close to the obstacle without colliding, which incurs a -100 penalty. First with deterministic robot dynamics, then under $\mathcal{N}(0, 0.2^2)$ noise. The exploration noise σ_π is optimized over time in PG approaches. However, the optimal policy on this and many other domains is deterministic. Random moves can push the agent into a collision, which means that $\sigma_\pi \to 0$ at convergence. While exploration noise can simply be set to zero in the final policy, a policy with non-zero noise may not have converged. We have observed that exploration noise seems reluctant to go to zero on stochastic domains. To examine this phenomena we run PPO with fixed policy noise $\sigma_\pi \in \{1, 0.1\}$. On the deterministic domain Fig. 3a shows that σ_π has little effect on convergence rate, and low σ_π enables higher reward, both as expected. Note that exploration is separate from the policy update step size, which is not changed. However, on the stochastic domain, Fig. 3b shows that low σ_π severely reduces the convergence rate. As *reaching* low exploration is required for policy convergence, this could contribute to the observed slow convergence and safety problems above.



(a) (b) (c)

Fig. 3: Exploration noise σ_π vs. convergence rate for a) deterministic environment, b) stochastic environment. c) Experiment testing distribution over local minima.

The second experiment is on local minima, which is what exploration noise is supposed to mitigate. Even with just a few simple obstacles, viewed as a deterministic trajectory optimization problem, there could potentially be local minima. We test PPO by intentionally putting the agent close to the bad minimum shown in Fig. 3c. Even with exploration fixed to a large $\sigma_\pi = 1$ and long planning horizon, we observe that 4/5 times the agent converges to this minimum. In a static environment this means getting stuck. With moving obstacles, bad minima can put the agent in harms way. Moreover, it is not obvious what to do about it within the confines of on-policy policy gradient approaches, which are currently popular on the control benchmarks. Action saturation constraints of real robots imposes hard limits on the effectiveness of simply adding more action noise.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

4 Discussion

For safe application to real-world autonomous robots and vehicles it is important to know the limitations of an approach. Control benchmarks in deep RL have focused on increasing dimensionality and complexity of agent dynamics. Our experiments indicate convergence problems, from e.g. stochastic environments and bad minima, that can jeopardize safety even for simple agent dynamics. While the latter overlaps with exploration, an active research topic in RL, the limitations of PG exploration noise on continuous control tasks merits more attention.

Given the difficulties reported on our toy problems and the gap to more visually impressive results on controlling e.g. humanoids in Mujoco, it is perhaps time to start talking about the risk of trading a curse of dimensionality for a *curse of simulation*. Unless simulation benchmarks are carefully designed to retain all relevant real-world aspects, it is likely that theory will develop toward algorithms that primarily excel in the simulation environment. This is also why real robot experiments is seen as the gold standard in robotics. As the Mujoco simulator and benchmarks were originally designed with local trajectory optimization of deterministic systems in mind [11], it is perhaps not surprising that popular RL approaches also end up suffering from poor convergence on stochastic systems and issues with local minima. A more thorough anchoring in real control applications could serve to mitigate this problem.

As a solution to control under uncertainty, RL is a natural fit for autonomous robots. However, policy convergence is important to respect safety constraints in real-world applications. Even if convergence cannot be proven in deep RL, asymptotic performance can still be measured empirically. We recommend that toy problems with uncertainty and bad minima should be standard benchmarks.

References

[1] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade. Towards generalization and simplicity in continuous control. In *Proc. NIPS*, pages 6553–6564, 2017.

[2] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proc. IROS*, pages 5026–5033. IEEE, 2012.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, abs/1606.01540, 2016.

[4] 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. *arXiv preprint arXiv:1802.09464*, 2018.

[5] O. Andersson, M. Wzorek, P. Rudol, and P. Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In *Proc. ICRA*, pages 4597–4604, May 2016.

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

[7] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proc. ICLR*, 2016.

[8] 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.

[9] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proc. AAAI*, 2018.

[10] O.Andersson, M. Wzorek, and P. Doherty. Deep Learning Quadcopter Control via Risk-Aware Active Learning. In *Proc. AAAI*, pages 3812–3818. AAAI Press, 2017.

[11] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Proc. IROS*, pages 4906–4913. IEEE, 2012.