Proximity-constrained reinforcement learning

Abhishek Bhatia
Columbia University
a.bhatia@columbia.edu

Jaan Altosaar
Princeton University
altosaar@princeton.edu

Shixiang Gu
University of Cambridge
Max Planck Institute
sg717@cam.ac.uk

Abstract

Reinforcement learning research is difficult to reproduce (Henderson et al., 2017): the algorithms can be brittle and sensitive to hyperparameters. Our goal is to develop a generic, efficient method to make reinforcement learning algorithms more robust by constraining gradient updates of policy parameters. Inspired by proximal methods in policy optimization (Schulman et al., 2015, 2017) and variational inference (Altosaar et al., 2017), we develop a framework for constraining parameter updates in policy optimization. We carry out an empirical study and demonstrate that our method with an entropy proximity statistic leads to more stable learning and increased exploration using proximal policy optimization (Schulman et al., 2017) in a continuous control task.

1 Introduction

Deep reinforcement learning—neural networks used for solving sequential decision-making tasks—has been successfully deployed in many domains such as the game of Go and robotic manipulation (Silver et al., 2017; Gu et al., 2016). Despite these useful applications, reinforcement learning algorithms are sensitive to hyperparameters and implementation choices (Henderson et al., 2017).

For example, reinforcement learning algorithms are sensitive to initialization. This has led to the development of useful optimization methods such as including the negative entropy in the objective to avoid poor local optima and encourage exploration (Mnih et al., 2016). But this entropy bonus is problematic in contiuous action spaces as it leads to an ill-defined objective function. Specifically, the differential entropy of a Gaussian is maximized at infinity. This is our motivation: to achieve the same benefits as an entropy penalty (exploration) without the downside of an objective that is trivially maximized in continuous action spaces.

To develop our method, we first show that gradient ascent on any objective implicitly uses a Euclidean distance metric to define valid next steps of the parameters. In words, gradient ascent can be viewed as maximizing the first-order Taylor expansion of the objective around the previous parameters, subject to a proximity constraint—a Euclidean ball around the previous parameters (Spall, 2003; Boyd and Vandenberghe, 2004).

A Euclidean proximity constraint is problematic because if we parameterize our policy as a Gaussian, the mean and variance parameters will be equally constrained. For a Gaussian with small variance, small changes in the mean parameter will lead to drastic changes in the support of the distribution. We avoid this by developing proximity constraints, such as the entropy, specific to reinforcement learning algorithms.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Algorithm 1: Proximity-constrained reinforcement learning

Input: Initial parameters θ_0, proximity statistic $f(\theta)$, distance function d
Output: Parameters θ of the policy π_θ that maximize objective L
while L *not converged* **do**
  $\theta_{t+1} \leftarrow \theta_t +$ Noise
  while U *not converged* **do**
    Update $\theta_{t+1} \leftarrow \theta_{t+1} + \rho \nabla_\theta U(\theta_{t+1})$
  end
  $\theta_t \leftarrow \theta_{t+1}$
end
return θ

Our work is complementary to trust region policy optimization (Schulman et al., 2015), proximal policy optimization (PPO) (Schulman et al., 2017), and other gradient-based reinforcement learning methods such as one-step Q-learning (Mnih et al., 2016). In the empirical study, we demonstrate how our algorithm improves the performance of PPO in an environment with a continuous action space.

2 Proximity-constrained reinforcement learning

In reinforcement learning algorithms, an agent interacts with an environment according to its policy π to carry out a sequential decision-making task. This problem can be parameterized in a number of ways. For example, in Q-learning the action-value function Q has parameters θ that may be shared between the policy and the value function. In policy optimization, the policy has parameters θ. A reinforcement learning algorithm is used to derive the update rule for θ (Sutton and Barto, 1998).

Without loss of generality, we will focus on gradient-based reinforcement learning methods that maximize an objective function L (algorithms that minimize a loss can be treated with a sign change), and take θ to be the parameters of the policy π for the rest of the discussion. In the empirical study, we will apply our method to the objective function used in PPO.

We first review the proximity constraint framework introduced in Altosaar et al. (2017) before showing how it applies to reinforcement learning.

2.1 Gradient descent implicitly uses a Euclidean proximity constraint

With θ_t as the parameters of the policy at step t and step size ρ, gradient descent on L maximizes the following update equation:

$$U(\theta_{t+1}) = L(\theta_t) + \nabla L(\theta_t)^\top (\theta_{t+1} - \theta_t) - \frac{1}{2\rho}(\theta_{t+1} - \theta_t)^\top (\theta_{t+1} - \theta_t). \tag{1}$$

This is the first-order Taylor expansion of the objective around the previous parameters θ_t, with the Euclidean proximity constraint. The Euclidean proximity constraint enforces that the next iterates of the parameters will remain close in squared Euclidean distance.

The solution that maximizes Equation (1) is the familiar gradient ascent equation,

$$\theta_{t+1} = \theta_t + \rho \nabla L(\theta_t). \tag{2}$$

2.2 Proximity-constrained update equation

We now develop new constraints in addition to the above Euclidean proximity constraint implicit in gradient ascent. These alter the notion of distance between successive parameter updates, and will enable us to design new proximity constraints for reinforcement learning.

Let $f(\cdot)$ be a *proximity statistic*, and let d be a differentiable distance function that measures distance between proximity statistic iterates. A *proximity constraint* is the combination of a distance function d applied to a proximity statistic f.

Algorithm 2: (Fast) Proximity-constrained reinforcement learning

Input: Initial parameters θ_0, adaptive learning rate optimizer, proximity statistic $f(\theta)$, distance d
Output: Parameters θ of the policy π_θ that maximize objective L
while $L_{proximity}$ *not converged* **do**
  $\theta_{t+1} = \theta_t + \rho(\nabla L(\theta_t) - k \cdot (\nabla d(f(\tilde{\theta}), f(\theta_t))\nabla f(\theta_t))$.
end
return θ

Rather than use the same step size to define the strength of additional proximity constraints, we let k be the scalar magnitude of the proximity constraint. We define the proximity update equation for the variational parameters θ_{t+1} to be

$$U(\theta_{t+1}) = L(\theta_t) + \nabla L(\theta_t)^\top(\theta_{t+1} - \theta_t) - \frac{1}{2\rho}(\theta_{t+1} - \theta_t)^\top(\theta_{t+1} - \theta_t) - k \cdot d(f(\tilde{\theta}), f(\theta_{t+1})). \tag{3}$$

Here, $\tilde{\theta}$ is the policy parameter iterate to which we are measuring closeness. In gradient ascent, this is always $\tilde{\theta} = \theta_t$, but in our algorithm we can set it to be an exponential moving average. This helps make our algorithm robust to one-step errors during optimization.

Maximizing Equation (3) ensures that the parameters of the policy remain close in terms of the proximity statistic $f(\theta)$. How can we choose a good notion of distance? If we set f to be the entropy of the policy and initialize the parameters so the policy has high entropy, this will lead to an algorithm that enforces exploration. This algorithm is described in Algorithm 1. There is but one issue with the inner loop: it is too expensive, as it requires optimizing U anew every iteration.

2.3 Linearizing the update equation for an efficient algorithm

To arrive at an efficient algorithm, we Taylor expand the proximity constraint. The gradient with respect to the second argument of the distance d is denoted ∇d, and the first argument to the distance is $f(\tilde{\theta})$. Computing the first-order expansion around the policy parameters θ_t yields

$$\begin{aligned} U(\theta_{t+1}) = &L(\theta_t) + \nabla L(\theta_t)^\top(\theta_{t+1} - \theta_t) - \frac{1}{2\rho}(\theta_{t+1} - \theta_t)^\top(\theta_{t+1} - \theta_t) \\ &- k \cdot (d(f(\tilde{\theta}), f(\theta_t)) + \nabla d(f(\tilde{\theta}), f(\theta_t))\nabla f(\theta_t)^\top(\theta_{t+1} - \theta_t)). \end{aligned}$$

Setting this to zero and solving for the next parameters gives a closed-form solution for θ_{t+1},

$$\theta_{t+1} = \theta_t + \rho(\nabla L(\theta_t) - k \cdot (\nabla d(f(\tilde{\theta}), f(\theta_t))\nabla f(\theta_t)). \tag{4}$$

Fast proximity-constrained reinforcement learning is shown in Algorithm 2. We have achieved a more efficient solution than Algorithm 1, as we removed the need for the inner optimization loop.

We note that the fast version of our algorithm implies a global objective which varies over time:

$$L_{\text{proximity}}(\theta_{t+1}) = L(\theta_{t+1}) - k \cdot d(f(\tilde{\theta}), f(\theta_{t+1})).$$

This shows that our algorithm is simple to implement, and comparable in terms of efficiency. It requires adding but a single term to the objective function of a given reinforcement learning algorithm. And because d is a distance, it is a valid lower bound on the objective function. This new objective encourages the policy to remain close in terms of f to previous iterations' policies.

3 Empirical study

We developed proximity-constrained reinforcement learning and derived a simple, easy-to-implement algorithm. Although our algorithms apply to various reinforcement learning algorithms, we focus on proximal policy optimization (PPO), whose objective function is a lower bound on the performance of the policy (Schulman et al., 2017). We implement our algorithm starting from the official version of

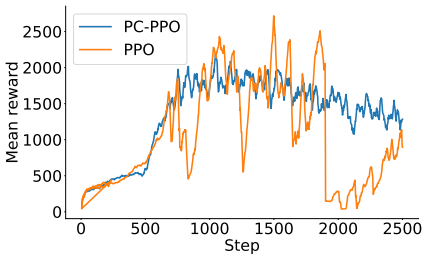

Figure 1: Episode average reward for agents trained in the Hopper-v1 MuJoCo environment. Proximity-constrained proximal policy optimization (PC-PPO) with an entropy proximity statistic yields more stable learning and increased reward.

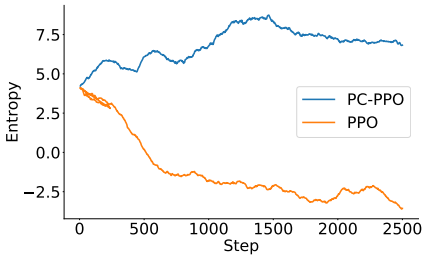

Figure 2: Episode average entropy for agents trained in the Hopper-v1 MuJoCo environment. Proximity-constrained proximal policy optimization (PC-PPO) with an entropy proximity statistic enforces increased exploration.

PPO at https://github.com/openai/baselines. Specifically, for the neural networks parameterizing the policy and value function, we use the following architecture: two fully-connected hidden layers with tanh activation functions.

In Schulman et al. (2017) the authors propose adding an entropy penalty term to the objective as in Mnih et al. (2016). However, this entropy penalty term is only applicable in finite action spaces. In preliminary experiments, we tested the entropy penalty against Algorithm 2 with entropy as the proximity statistic. As expected these two methods performed similarly: they achieved similar reward and enjoyed the benefits of enforced exploration during optimization.

However, the entropy penalty suggested in Schulman et al. (2017) is problematic in continuous control tasks. To see this, consider a continuous action space where the policy is parameterized by a Gaussian: $\pi_\theta = N(\mu, \sigma^2)$ where $\theta = (\mu, \sigma^2)$. The entropy of a Gaussian is $H(\pi_\theta) = \frac{1}{2}\log(2\pi e\sigma^2)$. Naively adding the entropy penalty to the objective results in unstable learning: the variance is quickly set to infinity and we cannot train further. We verified this happens experimentally.

We therefore compare to PPO without the entropy penalty term, and study a simple continuous action space task. For our method, Algorithm 2, we set the proximity constraint magnitude to be $k = 0.01$. The results are shown in Figures 1 and 2.

4 Discussion

We developed proximity-constrained reinforcement learning, a generic framework to constrain gradient updates in reinforcement learning methods. We showed that the Euclidean proximity constraint intrinsic to gradient ascent can lead to poor performance such as lack of exploration in continuous control tasks. We demonstrated that with an entropy proximity statistic, PPO had more stable performance and achieved more exploration and higher average reward. In future work we will test our method with other algorithms besides PPO, with novel proximity statistics such as state occupancy measures or variance of expected reward.

References

Altosaar, J., Ranganath, R., and Blei, D. (2017). Proximity variational inference. abs/1705.08931.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Gu, S., Holly, E., Lillicrap, T. P., and Levine, S. (2016). Deep reinforcement learning for robotic manipulation. *CoRR*, abs/1610.00633.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. *CoRR*, abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354 EP –.

Spall, J. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.