PPO single agent

Alexandre Popoff

April 18, 2019

A single PPO agent (actor-critic style) is implemented to solve the "reacher" environment (continuous state and actions spaces). The loss is the sum of the clipped PPO loss, the critic loss (Gaussian actor and critic share parameters) and an entropy terms which penalizes small standard deviation. This environment can be solved in 1000 episodes. Convergence highly depends on parameters optimization and empirical best practices.

1 Algorithm

Following [3], the PPO algorithm for one agent is

Algorithm 1 PPO single agent

for *iteration*$= 1, 2, \ldots$ **do**
  Run policy $\pi_{\theta_{\text{old}}}$ in environment in T timesteps;
  Compute advantage estimate $\hat{A}_1, \ldots, \hat{A}_T$;
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leqslant T$;
  $\theta_{old} \leftarrow \theta$
end

The algorithm is typically implemented with the followings optimizations:

1. Normalized advantages
2. Value loss baseline $V^{\pi}_{\theta_{t-1}} + A_t$
3. Clipped value loss around the previous estimate (i.e. minimize $(V_{\theta_t} - V_{targ})^2$ with V_{θ_t} not too far from its previous value)
4. Gradient clipping
5. Annealed Adam learning rate (not used here)
6. Orthogonal initialization (not used here)
7. Reward scaling (not used here)

In practice, vanilla PPO is hard to stabilize. The main idea is to keep the value function V_θ and the policy π_θ close to their previous values (otherwise, the algorithm will unlikely solve the environment). In order to do so, we use a clipped version of the "true" loss and we clip the gradient. Normalized advantages also speeds up training. I have to say that I spent countless hours trying to implement vanilla PPO without any success or small ones (for example, PPO solves the cartpole environment quite easily without any tricks, but it is misleadingly easy). For more details on the intricacies of implementing PPO see [2].

2 Neural network

The actor and critic are fully connected neural networks with two hidden layers (64, 64) units and $\tanh$ activations. They share parameters. The critic network takes the state as an input and then it outputs the value $V_\pi(s)$ of the state s. The actor network takes the state as an input and then it outputs the means of gaussian distributions from which an action is taken. The actor learns the standard deviations so that we add an entropy penalty in the total loss, although the weight of the entropy is hard to adjust according to [1].

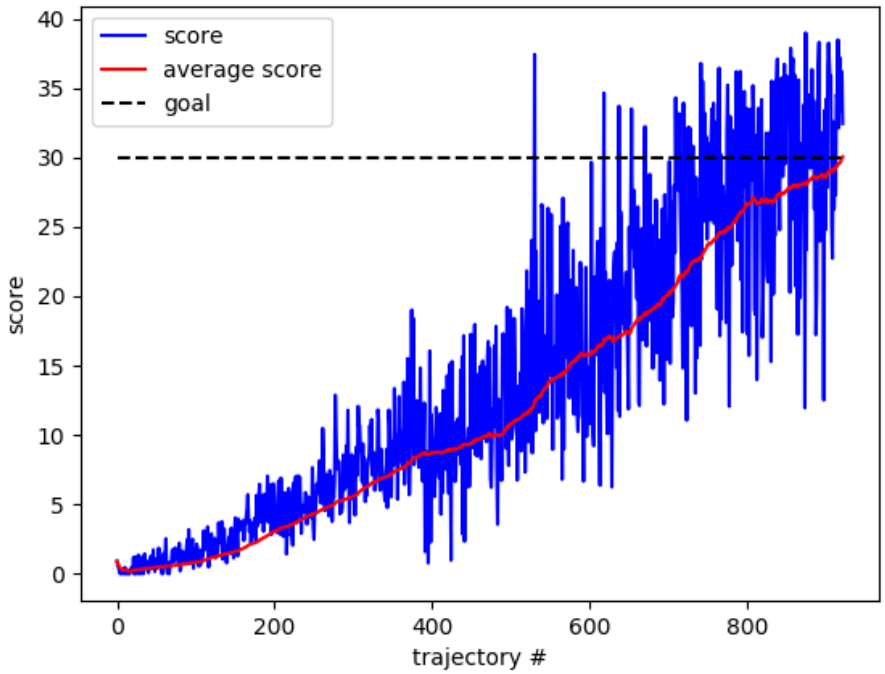
3 Hyperparameters

It is important to recall that hyperparameters and the vanilla PPO algorithm alone won't be sufficient to solve the environment. Normalized advantages, clipped value loss, gradient clipping, etc. seem to be necessary. The following values are pretty standard in literature.

Hyperparameter	Value
Horizon T	2048
PPO clip ϵ	0.2
Num. epochs	8
Adam lr α	3×10^{-4}
Minibatch size	64
Discount γ	0.99
GAE parameter λ	0.95
Maximum budget	2000 episodes
Value loss weight c_1 (same notation as [3])	0.5
Entropy weight β	0.01

4 Plot of rewards

One PPO agent solves the environment in 922 episodes.



5 Future work

Implementing PPO from scratch is tricky since the algorithm alone is not sufficient to obtain good results. Best practices are found in baselines (e.g. OpenAI) and empirical results in [2]. The code reflects all those tests and changes made during this work. Implementing 20 agents in parallel could be the next step, along with the PPO-CMA algorithm [1].

References

[1] Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. PPO-CMA: proximal policy optimization with covariance matrix adaptation. *CoRR*, abs/1810.02541, 2018.

[2] Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are deep policy gradient algorithms truly policy gradient algorithms? *CoRR*, abs/1811.02553, 2018.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.