CS7011 : Topics in Reinforcement Learning
Summary of PPO Algorithms : Schulman, Wolski, Dhariwal, Radford, Klimov

By Sahana Ramnath : EE15B109

1. Abstract

This paper (Schulman et al., 2017) proposes a new family of policy gradient methods **Proximal Policy Optimization(PPO)** for RL which alternate between sampling transitions from the environment and optimizing a "surrogate" objective function. Standard PG methods perform one gradient update per sample, but PPO enables multiple epochs of minibatch updates. PPO has some of the benefits that TRPO has over standard PG methods, but they are simpler to implement, more general and have better sample complexity(empirically). PPO was tested on multiple benchmark tasks, including simulated robotic locomotion and ALE games; compared to other PG methods, PPO gave better performance, sample complexity and time.

2. Introduction

PPO is an algorithm that attains the data efficiency and reliability of TRPO but while usisng first-order approximation. The paper introduces various versions of the surrogate objective and compares them using experiments; the best performer is the one with clipped probability ratios which forms a a pessimistic estimate/lower bound on the policy's performance. To optimize policies, PPO algorithms alternate between sampling data from the policy and performing several epochs of optimization on this sampled data.

2.1. Prior Policy Optimization Methods

- Policy Gradient Methods : Work by computing an estimator of the policy gradient and optimizing it using stochastic gradient ascent : loss $L^{PG}(\theta) = \hat{\mathrm{E}}_t[log\pi_\theta(a_t|s_t)\hat{A}_t]$. Multiple steps of optimization on L^{PG} using the same trajectory often leads to destructively large policy updates.
- Trust Region Methods : Here, the surrogate objective function is maximized subject to a constraint on the size of the policy update. This problem can efficiently be approximately solved using the conjugate gradient algorithm, after making a linear approximation to the objective and a quadratic approximation to the constraint. Alternately, TRPO could use a penalty instead of a constraint which is an unconstrained optimization problem : $max_\theta \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]]$

3. Surrogate Objectives for PPO

3.1. Clipped Surrogate Objective

Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. TRPO maximises $L^{CPI}(\theta) = \hat{E}_t[r_t(\theta)\hat{A}_t]$. Without a constraint, maximization L^{CPI} would lead to an excessively large policy update; PPO modifies the objective and penalizes changes to the policy that move $r_t(\theta)$ away from 1. The proposed objective : $L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$. The first term is L^{CPI} and the second term modifies the surrogate objective by clipping the probability ration which prevent r_t from moving outside $[1-\epsilon, 1+\epsilon]$. The minimum of the clipped and unclipped objective is taken to give a final objective which is a lower bound on the unclipped objective.

3.2. Adaptive KL Penalty Coefficient

This uses a penalty on the KL divergence and to adapt the penalty coefficient β so that some target value of the KL divergence d_{targ} is achieved each policy update. The objective : $L^{KLPEN}(\theta) = \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]]$. Compute $d = \hat{E}_t[KL[\pi_\theta(a_t|s_t), \pi_{\theta_{old}}(a_t|s_t)]]$: if $d < d_{targ}/1.5, \beta \leftarrow \beta/2$, if $d > d_{targ}X1.5, \beta \leftarrow \beta X2$. The updated β is used for the next policy update. The algorithm quickly adjusts β whenever needed.

4. Algorithm

The previously defined surrogate losses can be computed and differentiated with a minor change to the usual PG implementation. If using a neural network that shares parameters between the policy and value function, a loss function that combines the policy surrogate and a value function error term is included. Further, to encourage exploration, an entropy term can be added to the loss. Essentially, $L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$ where c_1, c_2 are constants, S is an entropy bonus and L_t^{VF} is the squared error loss $(V_\theta(s_t) - V_t^{targ})^2$. Advantage can be estimated using a truncated version of generalized advantage estimation : $\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ...(\gamma\delta)^{T-t+1}\delta_{T-1}$, where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. Each iteration, each of N(parallel) actors collect T timesteps of data. Then the surrogate loss is constructed on these NT timesteps of data, and optimized it with minibatch SGD, for K epochs.

5. Experiments

5.1. Comparing Surrogate Objectives

The 3 surrogate objectives : no clipping/penalty, with clipping and with fixed/adaptive KL penalty were compared under different hyperparameters for 7 simulated robotics tasks in OpenAI gym implemented using Mujoco. Clipping gave the best performance.

5.2. Comparing with other algos

PPO is compared with TRPO, CEM, vanilla PG with adaptive step size, A2c and A2C with trust region; PPO outperforms them on almost all the continuous control methods.PPO gives (mostly) comparable performances on Atari games, when compared against A2C and ACER.

5.3. Humanoid Running and Steering

PPO is tested on high-dimensional continuous control problems : a set of problems involving a 3D humanoid where the robot must run, steer, and get up off the ground, possibly while being pelted by cubes. It performed well on these.

References

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.