Proximal Policy Optimization: Paper Review

PPO is an Online Reinforcement learning (Policy Gradient) algorithm that is built on top of the Trust Region Policy Optimization (TRPO). PG suffers from being highly sensitive to hyperparameters such as learning rate. TRPO addresses this by forcing the policy updates to be conservative, by adding a KL divergence constraint to the optimization problem. This makes sure that the updated policy lies within a trust region and prevents the updated policy from moving too far from the current policy. However, adding this constraint adds an additional overhead to the optimization process, which is difficult to implement and takes more computation time. PPO addresses this by introducing a much simpler clipped surrogate objective function, which adds it as a penalty in the objective function itself, rather than a complex constraint.
The main contributions of this paper is to strike a balance between the ease of implementation, sample complexity and ease of tuning. This ensures that it tries to compute an update at each step that minimizes the cost function while ensuring that the deviation of the updates policy from the current policy is relatively small.

In cases, where the actions taken in the current policy is more probable than the former policy, this would lead to large policy update and excessive policy update. PPO addresses this by clipping the probability ratio which prevents it from moving outside the interval (1+ ε, 1-ε). The probability ratio is then clipped according to the sign of advantage function. Therefore, If it is much less likely under the new policy than the old, the objective action is flattened out to prevent from overdoing the update once again. This objective can further be augmented by adding an entropy bonus to ensure sufficient exploration.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

The paper further shows the comparison of PPO with different surrogate objectives, and empirically shows that PPO with clipping outperforms the one without clipping as well with adaptive KL divergence penalty coefficient. PPO results outperforms the state of the art algorithms (TRPO, A3C, A2C, CEM, etc.) for both continuous (OpenAI Mujoco domains) and discrete environment (ATARI domain).

Strengths

- The key strength of this paper is the simplicity of the formulated objective function. With no complicated maths involved and just a few lines of code added to the Vanilla PG, it significantly outperforms Vanilla PG.
- This is a robust algorithm which requires little hyper-parameter tuning.
- By adding a soft constraint, it makes the optimization a first order, which achieves the best performance with the most simplicity.
- This algorithm has created a good benchmark for the MUJOCO environment in 2017 (till SAC paper came).

Weeknesses:

- Does not address the problem of efficient exploration. The exploration strategy is not a directed exploration based, rather is it just based on the entropy bonus.
- Since this is an online algorithm, it requires to collect new samples for each gradient step. This quickly becomes extravagantly expensive, as the number of gradient steps and samples per step needed to learn an effective policy increases with task complexity. Also, the old samples are discarded and this reduces the sample efficiency with respect to the Off policy algorithms.
- PPO shall need large batch sizes needs to learn stably on more high-dimensional and complex tasks.
- Though it requires a little hyperparameters tuning, it still needs to tweak other hyperparameters such as as the trust region clipping or learning rate if there is a neural network architecture change.
- Also, the paper does not explicitly mentions about the rewards obtained from the environment in the domains, whether the rewards are dense or sparse.

I am quite curious to apply PPO to environments with delayed rewards or environment with long time horizon which requires efficient exploration strategies. I would also like to identify if modifying the clipped objective function for environments with sparse rewards gives any better performance.
Also, I am curious to know if the noisy value function obtained from the neural networks suffers from overestimation as it does in DQN.