Solving Discrete Gym Environments Using Proximal Policy Optimization

Prasad Madhale
College of Computer and Information Science
Northeastern University
madhale.p@husky.neu.edu

Abstract

There are number of reinforcement learning methodologies that exist to solve Discrete Gym Environments. One of the most common methods is Deep-Q Learning. Although effective Deep-Q learning suffers due to high data requirement and limited capacity of transfer (Gary Marcus 2018). So, we try to implement a relatively new family of policy gradient method of reinforcement learning to solve this problem in a more efficient manner. The method which we use is PPO (Proximal Policy Optimization). PPO is quite similar to the Trust Region Policy Optimization (TRPO) but is much easier to implement. We perform experiments on the Cart Pole gym environment and compare the results with pre-existing stable implementations of PPO and TRPO to obtain sufficient benchmarking.

1 Introduction

Policy Gradient methods have made a lot of progress in terms of solving complex problems. However, vanilla policy gradients still suffer from low data sample efficiency issue. As, vanilla policy gradients perform one epoch of update at each step they do not squeeze the most out of the data sample. On the other hand, PPO performs multiple epochs of optimizations on the data sample. TRPO first introduced the idea of constraining the updates done to a policy by using KL constraint, however TRPO is not compatible with architectures that have noise (John Schulman et al., 2017). Also, TRPO is complex and thus is difficult to implement. PPO builds on the idea of constraining the policy update, but it incorporates the constraining part into the objective itself. PPO uses a clipped objective to achieve constraint on the policy update. The objective used is a ratio of probabilities and it forms a lower bound on the performance of the policy (John Schulman et al., 2017).

Policy gradients methods operate directly on policy unlike traditional reinforcement learning algorithms. As a result, the learning achieved by Policy Gradients is intuitive and thus can be applied to almost any type of problem. The same idea can be extended to PPO. PPO being a policy gradient method can generalize and solve a wide variety of problems. Even though in our implementation we are restricting ourselves to only Discrete environment problems, PPO can be easily applied to other problems such as Continuous environment problems.

In PPO, we keep alternating between two threads one executing a policy generating sequences and calculating advantage and the other performing multiple epochs of optimization.

2 Background

2.1 TRPO

TRPO is reinforcement learning methodology which guarantees monotonic improvements in the performance. TRPO works by establishing Trust Regions for each policy update. Each update is constrained to update the policy only within the bounds of these trust regions.

TRPO achieves this by maximizing a surrogate objective function subject to a constraint on the size of the policy update. This can be demonstrated by the following equation,

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ &\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

TRPO controls the rate of change of the policy by checking the KL divergence between the old and the new policy.

2.2 Clipped Surrogate Objective

The Clipped Surrogate Objective forms the most crucial part of the PPO algorithm. The symbol $r_t(\theta)$ denotes a ratio of the probabilities. So, $r_t(\theta)$ can be given by

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

And the surrogate objective for TRPO is

$$L^{CPI} = \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\,\hat{A}_t]$$

PPO on the other hand, uses a clipped surrogate objective given by,

$$L^{CLIP} = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

So, PPO uses its objective to force the constraint for policy update. Having the advantage term multiplied creates an effect of increasing the probability of picking an action when the advantage is positive and decreasing the probability of picking an action when the advantage is negative.

3 Related Work

3.1 Deep Q-Learning

As mentioned earlier we can also use Deep Q-Learning algorithm to solve these Discrete gym environments. And there are number of implementations that have achieved good results. However, we are again faced with the fact that Deep Q-Learning is data hungry. Policies learnt with Deep Q-Learning cannot be easily transferred to other tasks for which the agent is not trained (Gary Marcus 2018). Deep Q-Learning also doesn't guarantee convergence in fact there are cases of divergence from the optimal policy.

3.2 Policy Gradient

Policy Gradient methods also can be considered good contenders for solving the stated problem but again they suffer as they don't use the sampled data efficiently especially Vanilla Policy Gradient. Consider the REINFORCE algorithm which has to wait till the end to receive the returns to update the policy. That again serves as an hinderance in the learning process. Another problem with policy gradient methods is they are too sensitive to the step-size we select. Too small a step-size the learning will be slower and too large a step-size may lead to drop in performance due to large step in the wrong direction. Although, PPO is a policy gradient method it makes efficient use of sampled data by running multiple epochs of optimization to get the best out of the same.

4 Project Description

4.1 Actor-Critic Architecture

PPO uses an Actor-Critic architecture as its backbone. The Actor-Critic architecture basically consists of two Artificial Neural Networks. One network is the actor who is responsible for choosing the actions that the agent should take. Another network is the critic which provides a value estimate that represents how good is it to take a certain action in the given state.

This architecture forms the networks used in the PPO algorithm. PPO maintains two policies as we saw earlier. Both the networks are constructed using this A2C architecture. The two main networks represent old and new policies. Each Actor is a Multilayer Perceptron with one input, one output and two hidden layers. And each Critic network is a Multilayer Perceptron with one input, one output and one hidden layer.

4.2 Generalized Advantage Estimation

Raw Policy Gradients methods usually have high variance. To reduce this variance, we use a method called Generalized Advantage Estimation (GAE) to get the advantage estimate. Even though GAE reduces variance, it in turn introduces a bias into the system (Patrick Coady, 2017).

GAE states that instead of using discounted sum of rewards to calculate the advantage, we can use a parameter λ. So, we get a weighted sum of n-step advantages. Each n-step advantage is weighted with a λ^{n-1}. This has the effect of giving less importance to future rewards and more importance to the immediate rewards (Patrick Coady, 2017).

4.3 Algorithm

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1,2,... do
    for actor=1,2,...,$N$ do
        Run policy $\pi_{\theta_{old}}$ in environment for $T$ timesteps
        Compute advantage estimates $\hat{A}_1,\ldots,\hat{A}_T$
    end for
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \le NT$
    $\theta_{old} \leftarrow \theta$
end for
```

As discussed earlier we are using a clipped objective which in turn constraints the degree of our policy updates. The PPO algorithm alternates between sampling the environment for experience, calculating advantages and performing multiple epochs of optimization which also includes updating the old network.

5 Experiments

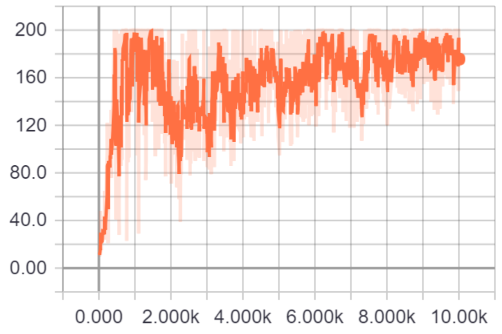

Figure 1: Episode rewards plot for our PPO (over 10K episodes)

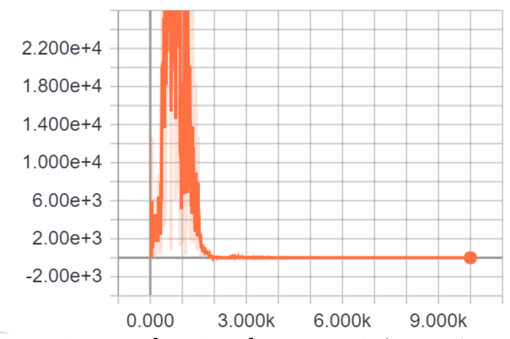

Figure 2: Loss function for our PPO (over 10K episodes)

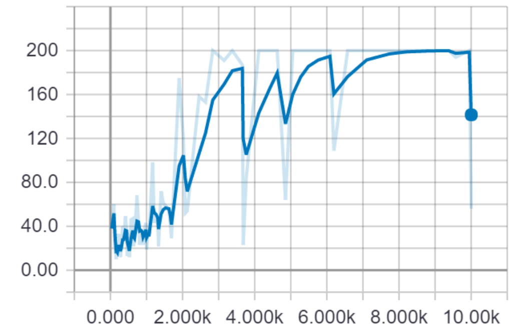

Figure 3: Episode rewards plot baseline PPO1 (over 10K episodes)

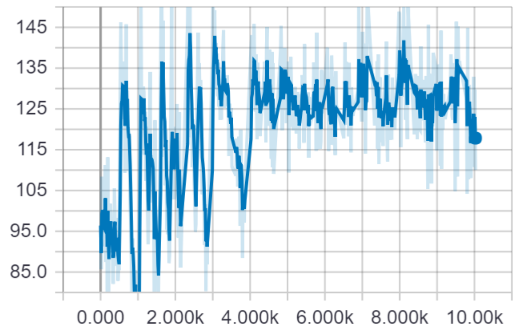

Figure 4: Loss function for baseline PPO (over 10K episodes)

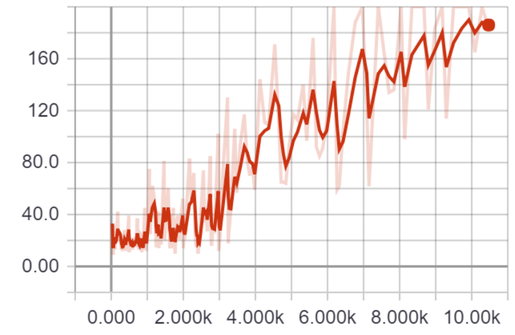

Figure 5: Episode rewards plot baseline TRPO (over 10K episodes)

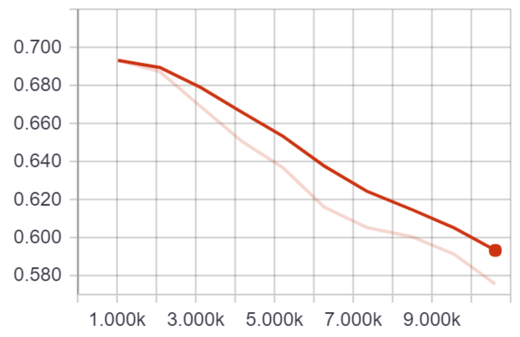

Figure 6: Loss Function for baselines TRPO (over 10K episodes)

We used our PPO implementation on Cart Pole gym environment and observe the episode rewards and loss. We then compare our rewards and loss to the baselines implementation of PPO and TRPO algorithm. It can be seen that with some hyperparameter tuning our implementation of PPO performs slightly better than the baseline implementations of TRPO and almost similar to the baseline PPO. Our implementation can be used with any Discrete gym environment and with some hyperparameter tuning we can achieve good results on the environment. The obvious next step to take would be to test our implementation against all the Discrete gym environments.

6 Conclusion

The experiments section above this shows our implementation's performance on Discrete environment. In the future, we can also adapt our implementation to work with Mujoco environments like Ant-v2 and Humanoid. Again, as PPO is adaptable enough and works directly on policies it would not be that difficult to adapt it to work with other types of environments. This was a good learning experience as I got to try out deep reinforcement learning and tensorflow. As an extension this implementation of PPO can be plugged into a Generative Adversarial Imitation Learning algorithm to achieve imitation learning in different environments.

References

[Gary Marcus 2018] Gary Marcus. *Deep Learning: A Critical Appraisal.* arXiv:1801.00631, Cornell University Library.

[John Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. *Proximal Policy Optimization Algorithms.* arXiv:1707.06347. OpenAI.

[Sergey Levine *et al.*, 2017] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, Pieter Abbeel. *Trust*

Region Policy Optimization. arXiv:1502.05477. University of California, Berkeley, Department of Electrical Engineering and Computer Sciences.

[Patrick Coady 2017] Patrick Coady. *AI Gym Workout*. Learning Artificial Intelligence. learningai.io.