Applying Proximal Policy Optimization (PPO) to OpenAI Environments

UCLA Samueli School of Engineering

Rahul Malavalli *204-429-252*
Glen Meyerowitz *505-294-306*
Joshua Vendrow *405-102-794*

Background

- Reinforcement Learning (RL) trains an agent to maximize the total rewards of a task; most methods are **value** or **policy**-based
- Policy optimization methods aim to improve a given policy to optimize total reward
- Proximal Policy Optimization (PPO) is a method for policy optimization that improves performance on continuous control tasks and is data efficient and robust
- PPO has become become one of the state-of-the-art RL tools, generalizable to many problem environments

Objectives

- Implement a working PPO algorithm that makes use of the actor-critic method
- Apply to an OpenAI environment, such as CartPole-v1 or CarRacing-v0

Environment Generation

We apply our PPO implementation to OpenAI environments:

- **CartPole-v1:** "Balance a pole on a cart"
  - *Action space:* Discrete, push cart left or right
  - *Observation space:* Contains 4 continuous values; cart position, cart velocity, pole angle, and pole velocity at tip
- **CarRacing-v0:** "Race a car around a track"
  - *Action space:* Discretized to 27 possible actions, each mapped to a (steering angle, gas, brake) action in the original space
  - *Observation space:* Retained original 96x96x3 image observations

Please see https://gym.openai.com/envs/ for more details.

Methods

- The PPO algorithm takes advantage of a clipped learning objective:

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

- The following is the pseudocode for the PPO implementation:

Algorithm 1 PPO

for $k = 0, 1, 2, \ldots$ **do**
  Run policy $\pi_{\theta_{old}}$ in environment for T steps
  Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  Optimize loss L wrt θ with K epochs of SGD (via Adam)
end for

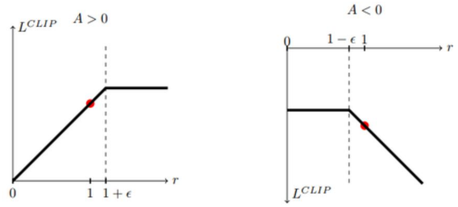
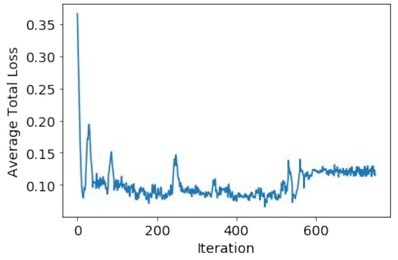


Figure 1: Plot of L^{CLIP} for positive advantages (left) and negative advantages (right) as a function of the probability ratio *r*.

Results

- To the right, we display the average loss and rewards resulting from applying our PPO implementation to the CartPole-V1 environment.
- We cut off our reward function after a reward of 200 because this reward is considered a "solve" for this environment.
- The increase in loss after iteration 500 can be attributed to updates in the critic, which evaluates state values to calculate advantages .

Figure 2: Visualizations of the CarRacing-v0 environment (left) and CartPole-v1 environment (right)



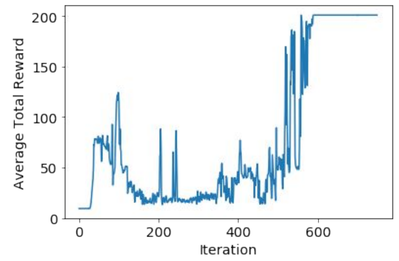


Figure 3: Average reward (top) and average loss (right) for CartPole-V1 environment

Conclusions

- We have successfully implemented a PPO algorithm based on the paper, used with actor-critic models
- Applied PPO to environments with discrete action spaces, such as CartPole-v1, which utilized a fully-connected neural network that converged in <1000 PPO steps.
- In CartPole-v1, over the limited number of training iterations, we were able to see an increase in the reward received by the agent from **X** to **Y**
- In the modified CarRacing-v0 environment, both actor-critic models (new CNN vs. pretrained ResNet-18) suffered from vanishing gradient issues and lack of computational resources. Further model experimentation, hyper-parameter tuning, and prolonged training (via GPUs) may be necessary to obtain better results.

Limitations

- The CarRacing-v0 environment generates a 2D box of pixels that must be interpreted by a CNN. To promote model learning, we may be able to simplify the environment by explicitly extracting features, such as steering angle, speed, and distance to the nearest wall.
- Computational resources have been a major limiting factor in these results. Although we were able to locally train simple fully-connected neural networks for environments like CartPole-v1, we quickly realized that the training larger CNN-based models would require hardware acceleration (GPUs, etc.) to be effective.

Acknowledgements

We learned a tremendous amount about RL and policy optimizations methods while working on this project. We extend our thanks to Dr. Yang, Joris, Qiujing, and Chinmay for their help this quarter.