Modified Cart Pole

Bertan Imre

MSc Data Science

Introduction

- Who?

  Bertan Imre - Industrial Engineer and Data Scientist

- What?

  Modifying the action and the reward of the Cart Pole environment

- Why?

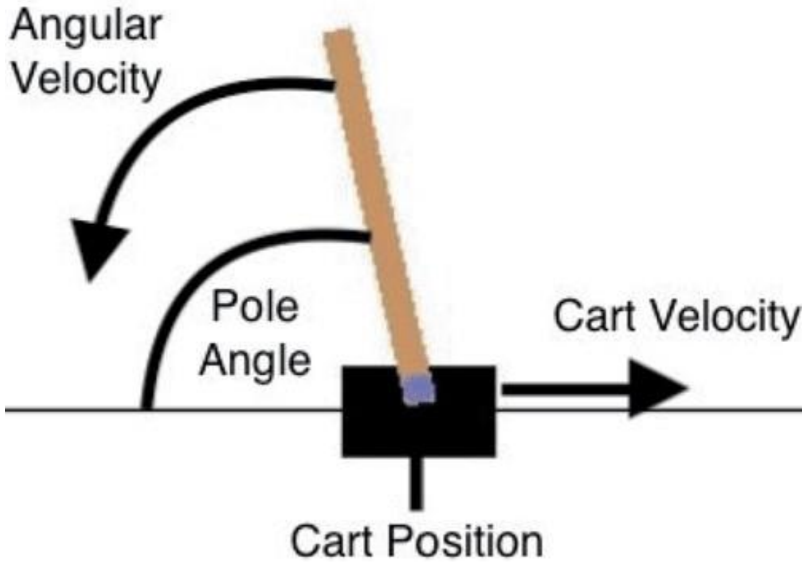
  Learning to modify the classic reinforcement learning environments

- How?

  Modifying the source code, learning the environment, and comparing the actions

Orijinal Environment

- Environment and Goal: A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The goal is to balance the pole by applying forces in the left and right direction on the cart
- Observation Space: Cart position, cart velocity, pole angle, and pole angular velocity
- Action Space: Push the cart to the left and push the cart to the right
- Reward: A reward of +1 for every step taken is allotted



Modified Environment

- Modified Action Space: Push the cart to the left, push the cart to the right, and do not push the cart
- Modified Reward: A reward of +1 for every step taken and a penalty of -0.5 for every push are allotted

```
# self.action_space = spaces.Discrete(2) Changed!!!
##########
self.action_space = spaces.Discrete(3)
##########
```

```
# force = self.force_mag if action == 1
##########
if action == 2:
    force = 0
elif action == 1:
    force = self.force_mag
else:
    force = -self.force_mag
##########
```

```
reward = 1.0
########## Added!!!
if action == 0 or action == 1:
    reward -= 0.5
##########
```

Agent

- Algorithm: Proximal Policy Optimization
- Approach: Model-free policy-based

Algorithm 1 PPO-Clip

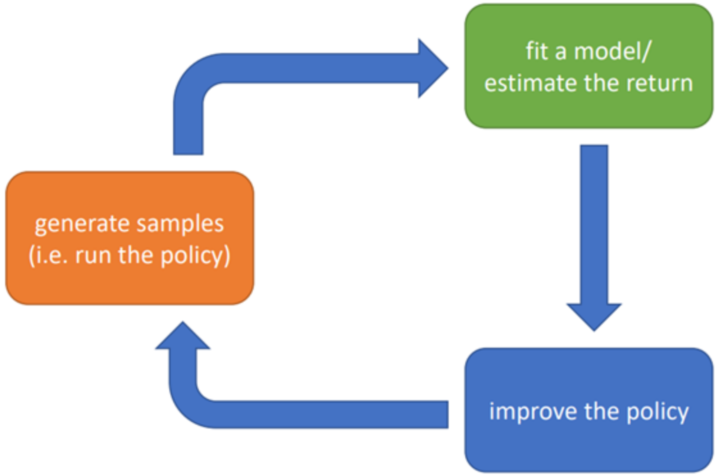
Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

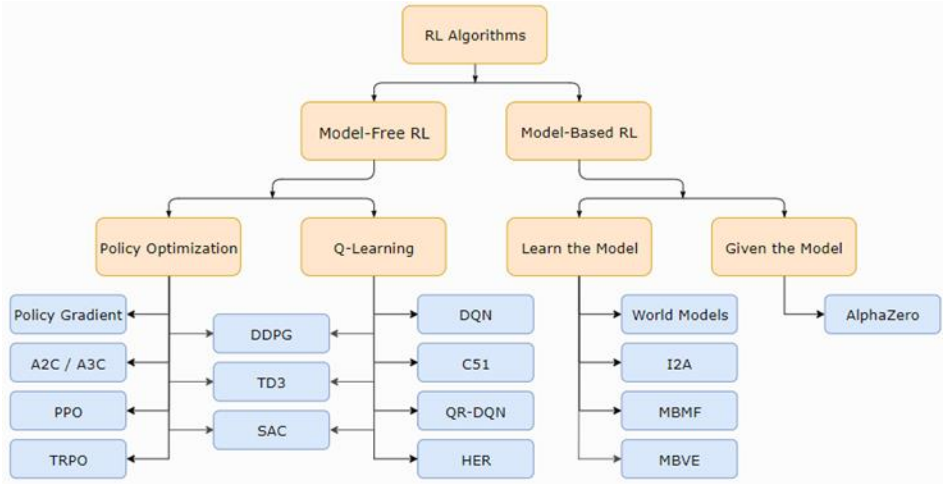
$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

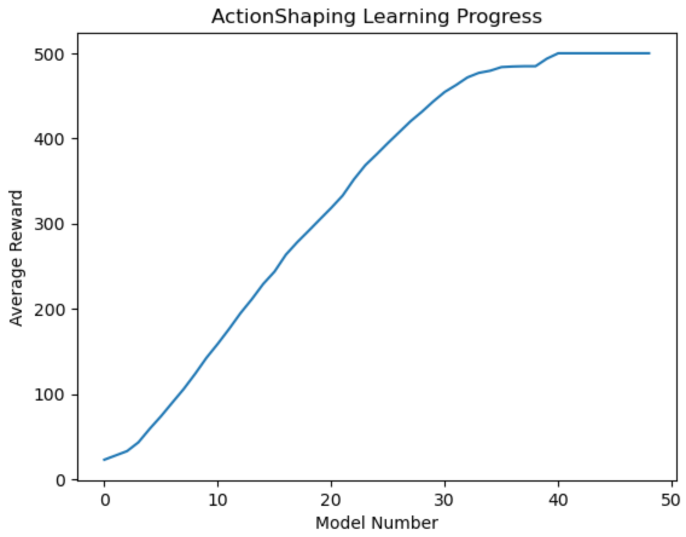




Results

- Action Shaping
  - Reward mean: 500
  - Reward std: 0

  - No pull percentage: 18.54%



- Reward Shaping
  - Reward mean: 492.32 (Step mean: 500)
  - Reward std: 0.92 (Step std: 0)

  - No pull percentage: 96.93%

Future Improvements

- Multi-seed (for the stochastic process)
- Multi-processing (for the computational time)
- Algorithm comparison (for the best algorithm)
- Hyperparameter tuning (for the best hyperparameters)
- Reward function (for a more comprehensive reward representation)
- Initial state (for a harder starting position)

Thank You and Questions