Project 3: Policy Gradient & Actor-Critic

Part 1: REINFORCE

```
REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^n$
Initialize policy weights $\boldsymbol{\theta}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    For each step of the episode $t = 0, \ldots, T-1$:
        $G_t \leftarrow$ return from step $t$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t,\boldsymbol{\theta})$
```

REINFORCE is a Monte-Carlo variant of policy gradients. It iteratively updates agent's parameters by computing policy gradient. It updates parameters by stochastic gradient descent. It using policy gradient theorem. It using return Gt as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$. The π(At | St, theta) is the policy which is mapping actions to probability, and Gt is a sample of the value function at time t collected from experience. α is the learning rate.

Part 2: Advantage Actor-critic (A2C)

```
Algorithm  TD Advantage Actor-Critic

Randomly initialize critic network $V_\pi^U(s)$ and actor network $\pi^\theta(s)$ with weights $U$ and $\theta$
Initialize environment E
for episode = 1, M do
    Receive initial observation state $s_0$ from E
    for t=0, T do
        Sample action $a_t \sim \pi(a|\mu,\sigma) = \mathcal{N}(a|\mu,\sigma)$ according to current policy
        Execute action $a_t$ and observe reward $r$ and next state $s_{t+1}$ from E
        Set TD target $y_t = r + \gamma \cdot V_\pi^U(s_{t+1})$
        Update critic by minimizing loss: $\delta_t = \left(y_t - V_\pi^U(s_t)\right)^2$
        Update actor policy by minimizing loss:
            $Loss = -log\left(\mathcal{N}(a \mid \mu(s_t), \sigma(s_t))\right) \cdot \delta_t$
        Update $s_t \leftarrow s_{t+1}$
    end for
end for
```

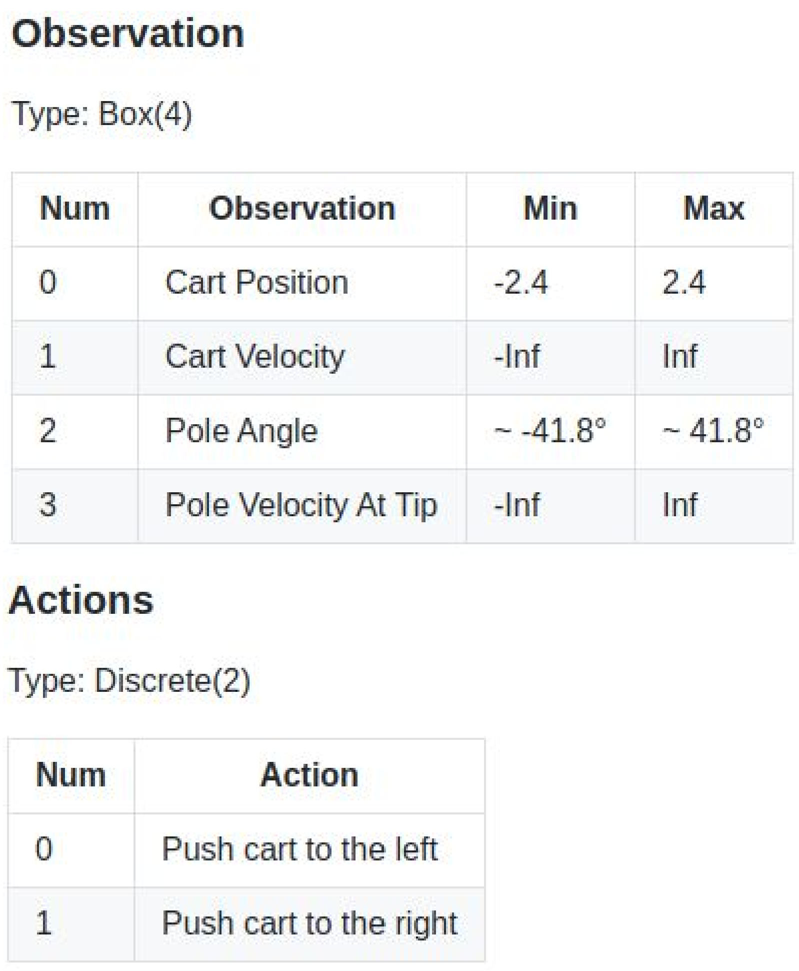
The Actor-critic algorithms maintain two sets of parameters: Critic updates action-value function parameters w. Actor updates policy parameters theta, in direction suggested by critic. It follows an approximate policy gradient :

$$\nabla_\theta J(\theta) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)$$

The actor is the policy $\pi\theta(a|s)$ with parameters θ which conducts actions in an environment. The critic computes value functions to help assist the actor in learning. These are usually the state value, state-action value, or advantage value, denoted as V(s), Q(s, a), and A(s, a), respectively.
A2C is a policy gradient algorithm and it is on-policy. That means that we are learning the value function for one policy while following. We will be using another policy if were using experience replay, because by learning from too old data, we use information generated by a policy slightly different to the current state. The advantage part is that it uses TD error. We can approximate the advantage function by using the TD error. The advantage function can significantly reduce variance of policy gradient, so the critic should really estimate the advantage function.

Gym CartPole Environment



Observation

Type: Box(4)

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

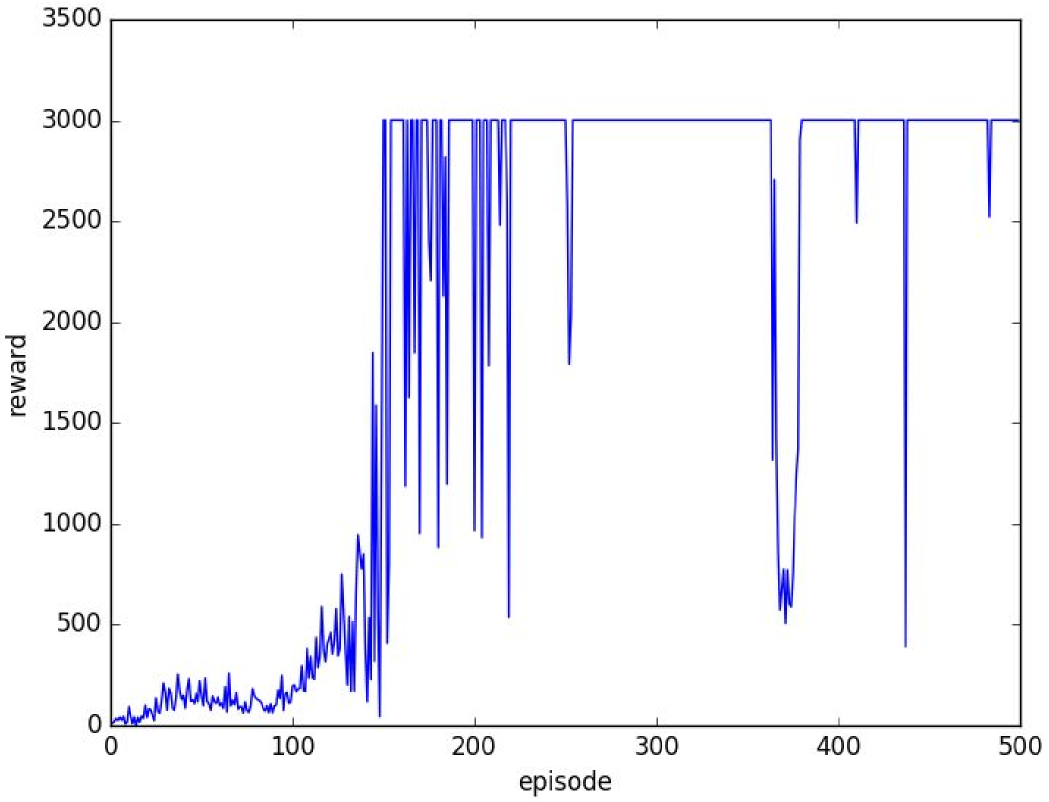
Actions

Type: Discrete(2)

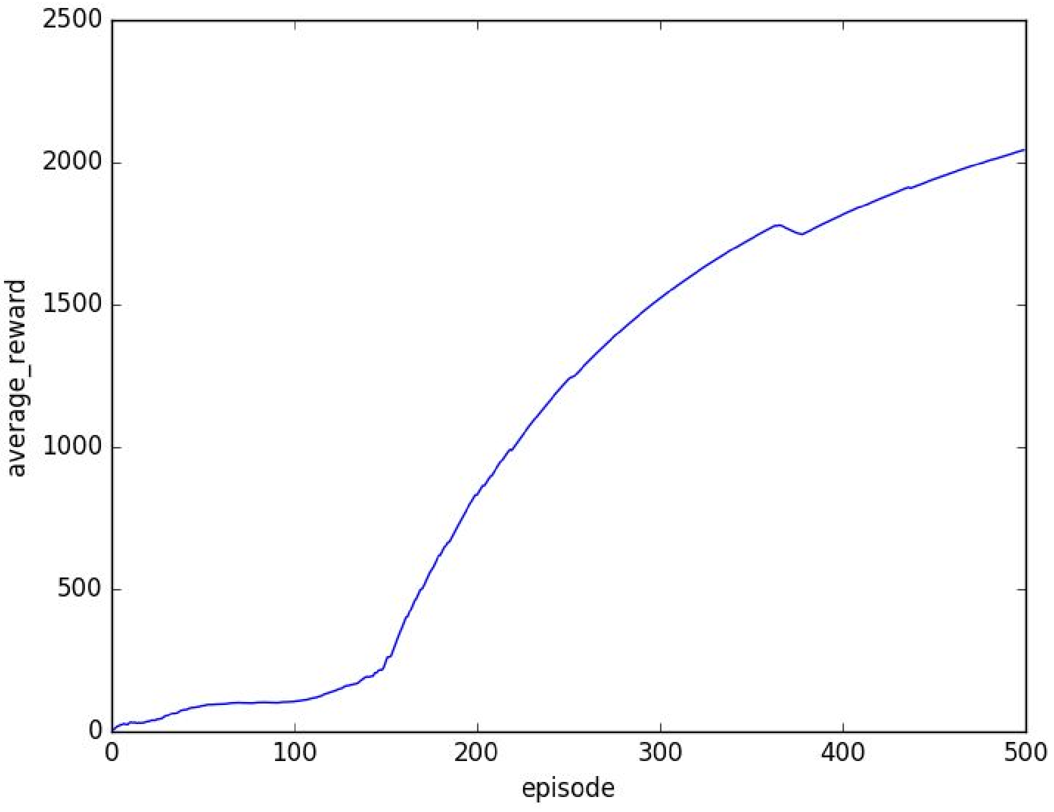
Num	Action
0	Push cart to the left
1	Push cart to the right

The agent is the combination of a pole attached by an un-actuated joint to a cart.
The pendulum starts upright, and the goal is to prevent it from falling over by increasing and reducing the cart's velocity.
Reward is 1 for every step taken, including the termination step.

REINFORCE

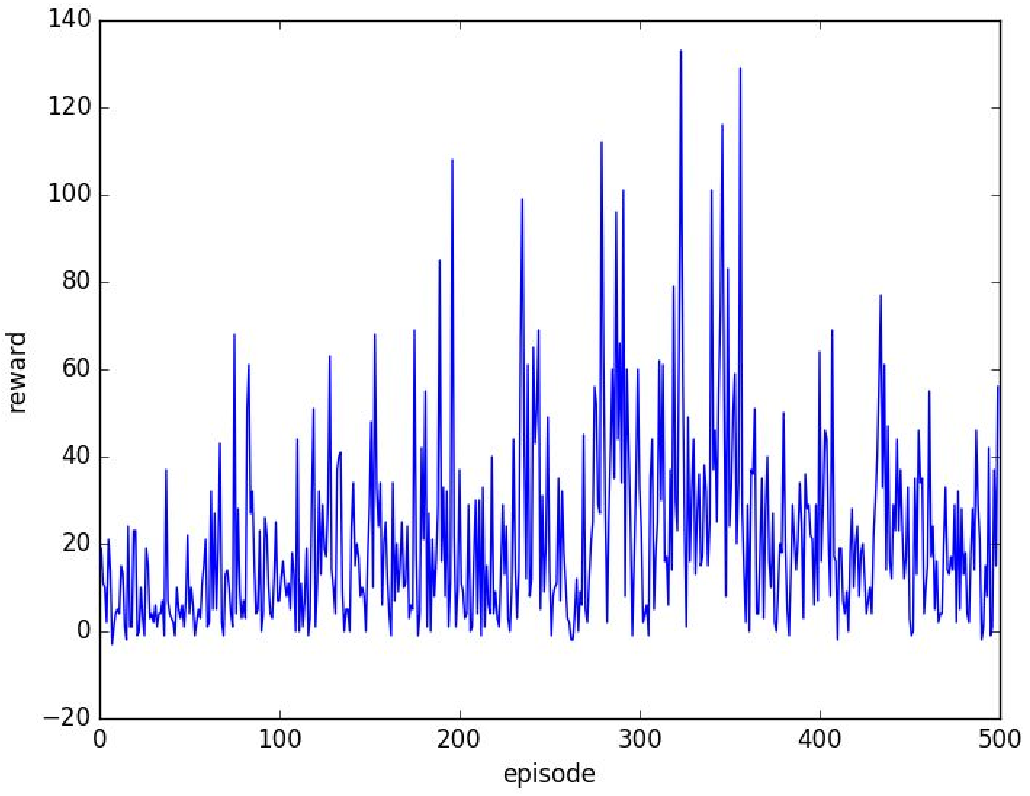


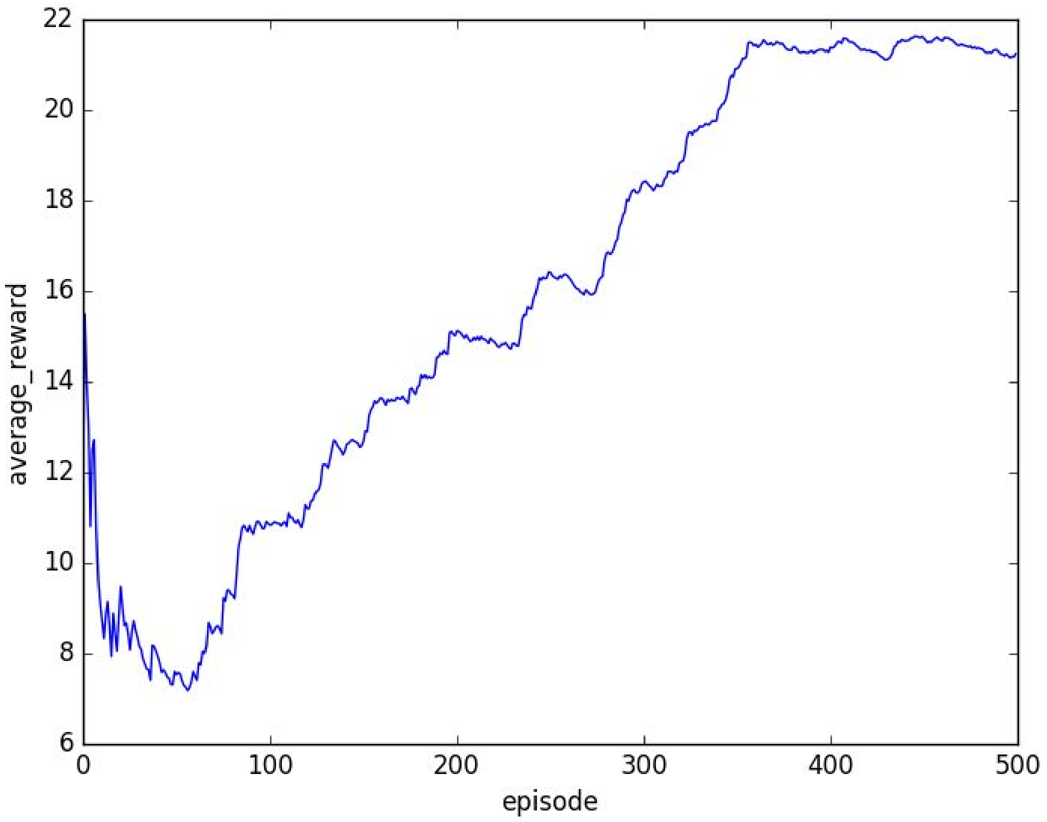
reward
episode
3500
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500



average_reward
episode
2500
2000
1500
1000
500
0
0
100
200
300
400
500

ACTOR-CRITIC





The figures are shown rewards of every episode and average reward during every episode for REINFORCE and Actor-critic.
From the rewards to episode figures, we can see that the REINFORCE algorithm learns very fast and after more than 150 times, it can obtain max scores for the most of the times. And in the figure of Actor-critic, we can see that the reward has a increasing trend with the episode going further. However, it can not hold a high score level after the time passed. The learning is slow and the result is unstable. It looks like a wave.
From the average reward to episode figures, we can see that the REINFORCE algorithm still has very good result. After 150 times, it increasing in a convex curve. And the final result can get to 2000 scores which is very high. In the figure of Actor-critic, it also increased fast but with some small shocks. After around 400 times, it becomes stable at 21 scores. I think it may need more time to learn and will get better, but it will not look like REINFORCE to get such high score.
For this environment, the REINFORCE performanced very well, and Actor-critic can also learn something and get a accepted result. REINFORCE algorithm has very fast learning speed and can get stable and high score. The actor-critic algorithm has slow learning speed and it has unstable and low score. I think actor-critic is not suitable for this enrivonment.