|  |
| --- |
| **Reinforcement Learning-Project 3 Report** |

**Gautam Suryawanshi**

Department of Engineering Science

University at Buffalo

[*gautamja@buffalo.edu*](mailto:gautamja@buffalo.edu)

**Abstract**

The goal of the assignment is to explore reinforcement learning environments and implement actor-critic algorithms. In the first part of the project we will implement REINFORCE, in the second part we will implement actor-critic algorithm. The purpose of this assignment is to understand the basic policy gradient algorithms. We will train our networks on a reinforcement learning environment among OpenAI Gym or other complex environments

* 1. **Reinforce Algorithm**

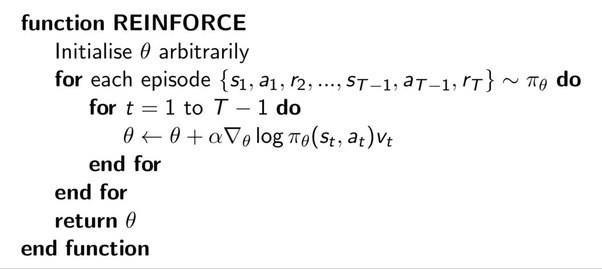
REINFORCE is a family of reinforcement learning methods which **directly update the policy weights** through the following rule:

Δ*θt*=*α*∇*θ*log*πθ*(*at*|*st*)*vt*

Where *α* is the learning rate, *πθ*(*at*|*st*) is the policy (mapping actions to probabilities), and

*vt* is a sample of the value function at time *t* collected from experience.

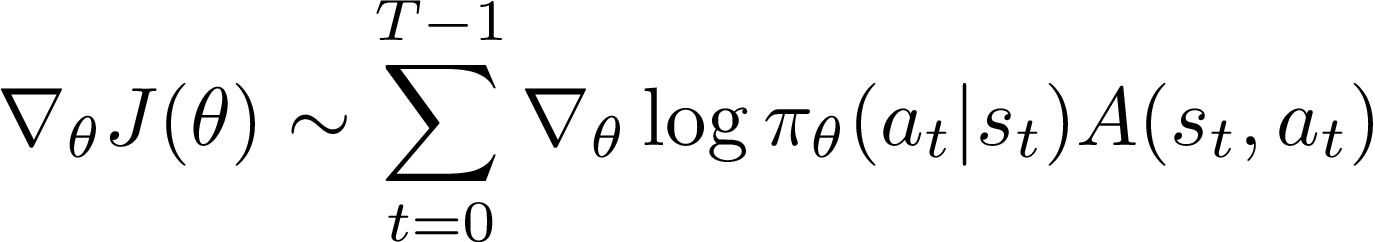
What made REINFORCE truly revolutionary for the time was that previously, one had to learn a value function through function approximation and then *derive* a corresponding policy, but **often** **learning a value function can be intractable** (more unstable / was, at the time, intractable for large state spaces). This provided a way to directly optimize policies to get around this problem.



* 1. **Advantage Actor Critic**

A2C is like A3C but without the asynchronous part; this means a single-worker variant of the A3C. It was empirically found that A2C produces comparable performance to A3C while being more efficient.

The new update equation, replacing the discounted cumulative award from vanilla policy gradients with the Advantage function:

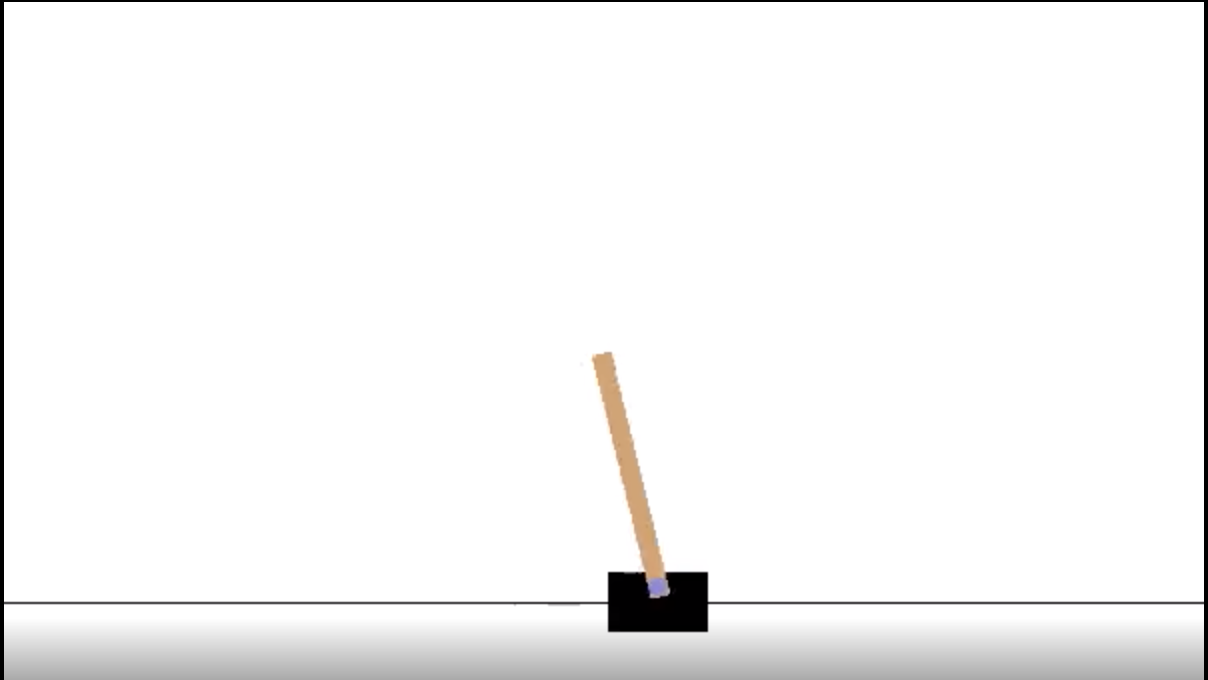


On each learning step, we update both the Actor parameter (with policy gradients and advantage value), and the Critic parameter (with minimizing the mean squared error with the Bellman update equation).

* 1. **Rewards ,States, Actions, Goal in our Environment- Cartpole-V0**

CartPole is a game where a pole is attached by an un actuated joint to a cart, which moves along a frictionless track. The pendulum starts upright, and the goal is to prevent it from falling over by increasing and reducing the cart’s velocity. A single state is composed of 4 elements: cart position, cart velocity, pole angle, and pole velocity at its tip. There are two actions to take in order to move the pole: moving left or right. For every step taken (including the termination step), it gains +1 reward.

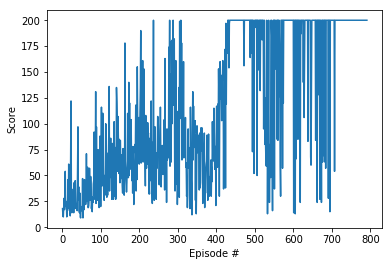
Goal: The game ends when the pole falls, which is when the pole angle is more than ±12°, or the cart position is more than ±2.4 (center of the cart reaches the edge of the display). Newer Gym versions also have a length constraint that terminates the game when episode length is greater than 200.



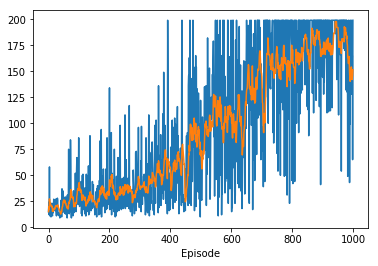
|  |  |  |
| --- | --- | --- |
| **STATES** | **Action** | **Reward** |
| Cart Velocity | Right | +1 every time step |
| Pole Angle | Left |  |
| Angular Velocity |  |  |
| Cart Position |  |  |

* 1. **Results and Comparison of Reinforce and A2C**

Reinforce Algorithm Result



A2C Algorithm Result



As in the REINFORCE algorithm, we update the policy parameter through Monte Carlo updates (i.e. taking random samples). This introduces in inherent high variability in log probabilities (log of the policy distribution) and cumulative reward values, because each trajectories during training can deviate from each other at great degrees. Consequently, the high variability in log probabilities and cumulative reward values will make noisy gradients, and cause unstable learning and/or the policy distribution skewing to a non-optimal direction. Besides high variance of gradients, another problem with policy gradients occurs trajectories have a cumulative reward of 0. The essence of policy gradient is increasing the probabilities for “good” actions and decreasing those of “bad” actions in the policy distribution; both “goods” and “bad” actions with will not be learned if the cumulative reward is 0.Overall, these issues contribute to the instability and slow convergence of vanilla policy gradient methods.

References

[1] <https://ai.stackexchange.com/questions/7390/what-is-the-difference-between-actor-critic-and-advantage-actor-critic>

[2] https://github.com/iocfinc/A2C-CartPole/blob/master/A2C%20-%20Cartpole.py

[3] Reinforcement Learning:An Introduction by Richard S. Sutton and Andrew G. Barto

[4] <https://github.com/jaromiru/AI-blog/blob/master/CartPole-A3C.py>

[5] <https://towardsdatascience.com/deep-reinforcement-learning-build-a-deep-q-network-dqn-to-play-cartpole-with-tensorflow-2-and-gym-8e105744b998>

[6] <https://gym.openai.com>

[7] <https://github.com/thehawkgriffith/CartPole-A2C/blob/master/cartpole_a2c.py>