Policy Gradient & Actor-Critic

Rugved Hattekar

April 2020

1 Introduction

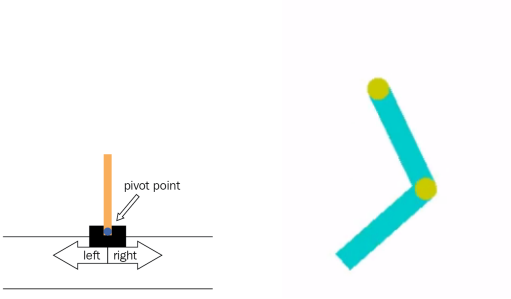
The goal of the assignment is to explore reinforcement learning environments and implement actor-critic algorithms. In the first part of the project will implement REINFORCE or (Monte Carlo PG) on Gym Acrobot Environment, and in the second part we will implement actor-critic(a2c) algorithm on Cartpole Environment. 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.

2 Environments

In this project, I implemented two gym environments. Gym is a toolkit for developing and comparing reinforcement learning algorithms.

2.1 1. Acrobot Environment

The acrobot system includes two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height. Each time-step has a reward of -1.



(a) Cartole (b) Acrobot

Figure 1: Environments

2.2 2. Cartpole Environment

Cartpole - known also as an Inverted Pendulum is a pendulum with a center of gravity above its pivot point. It's unstable, but can be controlled by moving the pivot point under the center of mass. The goal is to keep the cartpole balanced by applying appropriate forces to a pivot point. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

3 What is Reinforce Algorithm?

REINFORCE is a Monte-Carlo variant of policy gradients (Monte-Carlo: taking random samples). The agent collects a trajectory of one episode using its current policy, and uses it to update the policy parameter. Since one full trajectory must be completed to construct a sample space, REINFORCE is updated in an off-policy way.

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function

Pseudo code from UToronto lecture slides

(a) Reinforce Psuedo Code

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} r_{t+1} | \pi_\theta]$$

(b) Objective Reward Function

Figure 2: Reinforce Algorithm

1. Perform a trajectory roll-out using the current policy
2. Perform a trajectory roll-out using the current policy
3. Store log probabilities (of policy) and reward values at each step
4. Calculate discounted cumulative future reward at each step
5. Compute policy gradient and update policy parameter
6. Repeat 1–4

4 Actor-Critic Algorithm

Reinforce (MCPG) has a high variance and noise as in the REINFORCE algorithm, we update the policy parameter through Monte Carlo updates (i.e. taking random samples). This happens because the 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. The policy gradient is not capable of differentiate between the good reward and a bad reward. The differentiating parameter being the cumulative reward, because of which the vanilla objective function gives slow convergence.
In actor critic methods,

1. The "Critic" estimates the value function. This could be the action-value (the Q value) or state-value (the V value).

2. The "Actor" updates the policy distribution in the direction suggested by the Critic (such as with policy gradients).

and both the Critic and Actor functions are parameterized with neural networks, using the V function as the baseline function, we subtract the Q value term with the V value. Intuitively, this means how much better it is to take a specific action compared to the average, general action at the given state. We will call this value the advantage value:

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$$

(a) A2C Advantage Function

$$\begin{aligned}\nabla_\theta J(\theta) &\sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)\end{aligned}$$

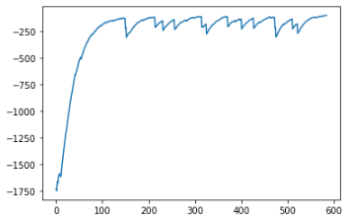
(b) Actor Critic Objective

Figure 3: Actor Critic Algorithm

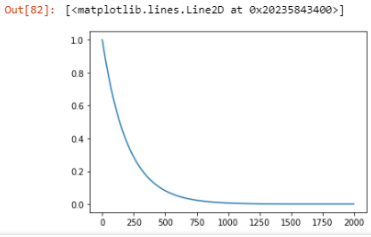
5 Implementation

In this project both the monte carlo policy gradient and the actor critic with the advantage function (A2C) has been implemented. The Implemenataion of the bpth the algorithm is done in python using above mentioned gym environments. Implementation steps of the policy gradient algorithm is mentioned in the above section. The Advantage actor critic is implemented with two neural networks 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).

5.1 Policy Gradient Results



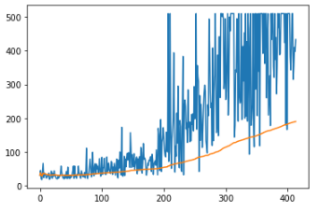
(a) Reward Function



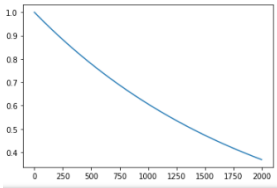
(b) Exploration Decay

Figure 4: Reinforce Results

5.2 Actor Critic Advantage Function Results



(a) Actor Critic on Cartpole



(b) Epsilon Decay

Figure 5: Actor Critic a2c Results

5.2.1 Discussion on Results

Actor Critic Results are pretty fast and smooth as compared to the Reinforce Algorithm. This happens because the 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 and cumulative reward values, because each trajectories during training can deviate from each other at great degrees. Here the advantage function basically helps by explicitly subtracting the average reward from the current step and outlining how good the only current action is compared to the other actions. Higher the Advantage function value more "Good" is the action, it resembles the role of the critic to the actor, which judges the actions of the agent. This is why the results are bettr with the actor critic than the monte carlo policy gradient(Reinforce) algorithm.

6 Conclusion

In this project we studied Reinforce Algorithm (Policy Gradient) and the Actor Critic Algorithm using pytorch and gym environments. The hyper-parameters used in this assignment are as follow.
For Reinforce Algorithm:
1. **Reinforce Acrobot Hyper-Paramaters**
$DiscountFactor = 0.995$
$LearningRate = 0.001$
$Epsilon = 1$
$EpsilonDecay = 0.995$

1. **A2C Cartpole Hyper-Paramaters**
$DiscountFactor = 0.995$
$LearningRate = 0.01$
$Epsilon = 1$
$EpsilonDecay = 0.995$

Hence the Reinforce and Actor critic algorithm with advantage function has been studied thoroughly and implemented in this assignment.

7 References

1. Introduction to Reinforcement Learning by Sutton and Burto.
2. https://gym.openai.com/envs/CartPole-v0/
3. https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63
4. https://github.com/pytorch/examples/blob/master/reinforcement_learning/reinforce.py