CSE 546

Reinforcement Learning

Assignment 3

Team 48

Author: Sarveshwar Singhal (50418642)

Submission Date: May 2, 2022

Algorithm:

We implemented an advantage actor critic algorithm. Actor critic algorithm is dependent upon policy gradient plus policy estimation or Q value estimation.

Actor-Critic algorithms maintain two sets of parameters:

a. Critic: Updates action-value function parameters
b. Actor: Updates policy parameters theta, in direction suggested by critic

Actor-Critic algorithms follow an approximate policy gradient

Advantage actor critic can significantly reduce variance of policy gradient, so the critic should really estimate the advantage function.

Actor Critic Vs Value Based Approximation Algorithm

Actor Critic	Value Based
Actor Critic is depended upon policy gradient plus Q value estimation	Value based approximation algorithms are dependent upon State value or Q value.
It's policy evaluation and policy improvement.	Finding optimal value function and then finding policy based on that.
Relatively smoother convergence	Relatively poor convergence

1. The main details of 'CartPole-v1'
    a. Objectives
        i. to achieve the average return greater than or equal to 195.0 over 100 consecutive trials by controlling the cart's velocity
        ii. while preventing a pole which is attached to a cart from falling over an episode whose length is greater than 200
    b. State Space (defined by 4 continuous elements)

	Position (x)	**Velocity** ($\partial x/\partial t$)	**Angle** (θ)	**Angular Velocity** ($\partial\theta/\partial t$)
Domain	$[-4.8,\ 4.8]$	$(-\infty,\ \infty)$	$[-0.418\,rad,\ 0.418\,rad]$	$(-\infty,\ \infty)$

c. Action Space
   i. 0: Push a cart to the *left*.
   ii. 1: Push a cart to the *right*.

d. Rewards
   i. 0: if the game is over or mission is completed
   ii. 1: if the game is not over

e. The termination conditions (game over conditions)
   i. if the pole's angle is more than 12 degrees or
   ii. a cart's position is more than 2.4 (center of the cart reaches the edge of the display)

2. The main details of 'LunarLander-v2'

a. Objectives
   i. to land the lunarlander safely within the specified boundaries.

b. State Space

   Box(-inf, inf, (8,), float32)

c. Action Space

4 discrete values: do nothing, fire left orientation engine, fire main engine, fire right orientation engine

d. Rewards
   a. Reward for moving from the top of the screen to the landing pad and coming to rest is about 100-140 points.
   b. If the lander moves away from the landing pad, it losses reward.
   c. If the lander crashes, it receives an additional -100 points. If it comes to rest, it receives an additional +100 points. Each leg with ground contact is +10 points.
   d. Firing the main engine is -0.3 points each frame. Firing the side engine is -0.03 points each frame. Solved is 200 points.

e. The termination conditions (game over conditions)
   - The game is over if lander reaches the ground either in a single piece or in multiple pieces.

3. The main details of 'Acrobot-v1'
   a. Objective
      i. The system consists of two links connected linearly to form a chain, with one end of the chain fixed. The joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height while starting from the initial state of hanging downwards.
   b. State Space & Action Space

Num	Observation	Min	Max
0	Cosine of theta1	-1	1
1	Sine of Theta1	-1	1
2	Cosine of theta2	-1	1
3	Sine of Theta2	-1	1
4	Angular velocity of theta1	~ -12.56	~12.56
5	Angular velocity of theta2	~ -28.27	~28.27

theta1 is the angle1 of the first joint, where an angle of theta indicates the first link is pointing directly downwards.

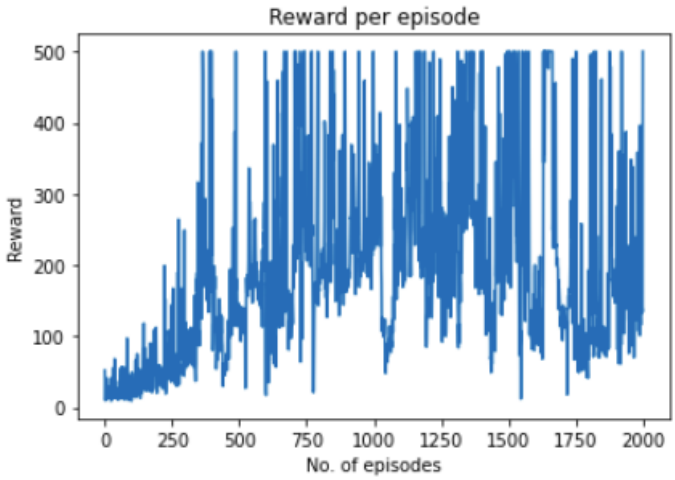
thets2 is relative to the angle of the first link. An angle of theta corresponds to having the same angle between the two links.

c. Reward
    i. The goal is to have the free end reach a designated target height in as few steps as possible, and as such all steps that do not reach the goal incur a reward of -1. Achieving the target height results in termination with a reward of 0. The reward threshold is -100.
d. Termination Condition
    i. The free end reaches the target height, which is constructed as: -cos(theta1) - cos(theta2+theta1) > 1.0
    ii. Episode length is greater than 500 (200 for v0)

Training on 3 different environments

1. CartPole-v0

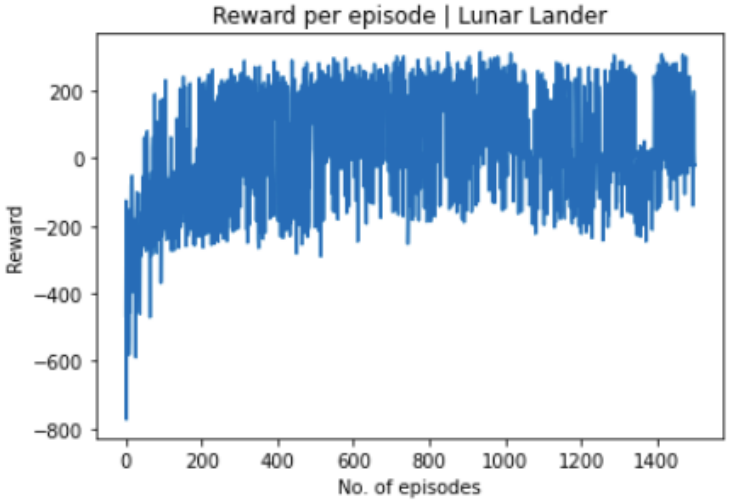
```
In [206]: plt.plot(np.arange(len(episode_rewards)), episode_rewards)
          plt.title("Reward per episode")
          plt.xlabel("No. of episodes")
          plt.ylabel("Reward")
          plt.show()
```



After training our agent on A2C, this was our reward per episode. We can see some fluctuations but for many instances the agent was able to reach the optimal scores i.e. more than 470.

2. LunarLander-v1

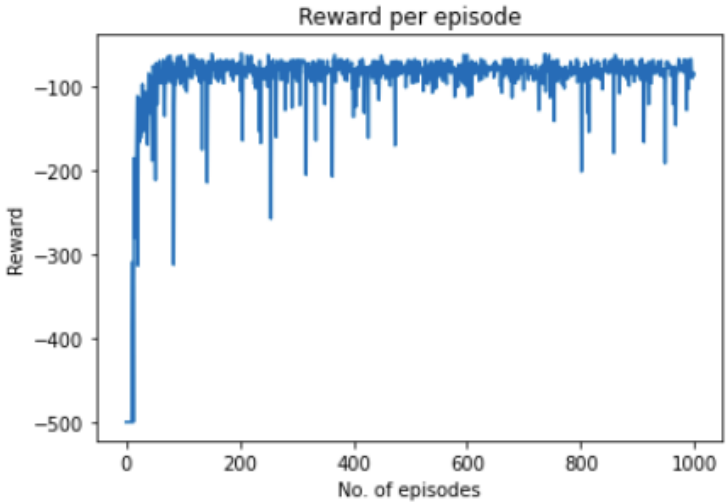
In [216]:
```
plt.plot(np.arange(len(episode_rewards)), episode_rewards)
plt.title("Reward per episode | Lunar Lander")
plt.xlabel("No. of episodes")
plt.ylabel("Reward")
plt.show()
```



After training our agent on the LunarLander environment, this was our results. We can see initially the agent got very high negative rewards, but as the agent learned. It was able to get a reward of 200.

3. Acrobot

In [277]:
```
plt.plot(np.arange(len(episode_rewards)), episode_rewards)
plt.title("Reward per episode")
plt.xlabel("No. of episodes")
plt.ylabel("Reward")
plt.show()
```

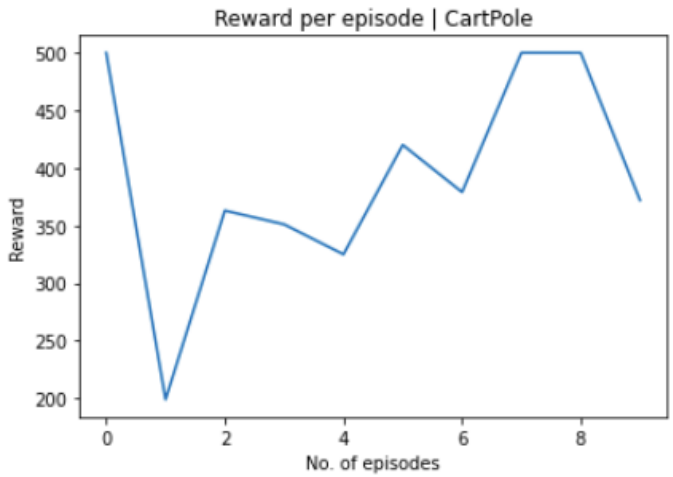


We've trained our agent on the Acrobot environment. We can see there are some negative rewards, mostly this is due to hyper-parameters. With the right set of hyper-parameters this reward can be turned to positive. But we can definitely see one thing, that our agent was learning, initially the reward was very high in negative, as the agent learns it's negative rewards reduced in magnitude.

Evaluation on 3 Different environment

1. CartPole

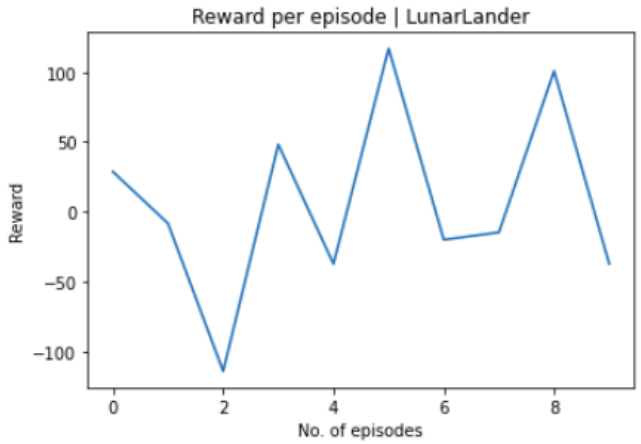
```
In [208]: plt.plot(np.arange(len(eval_episode_rewards)), eval_episode_rewards)
          plt.title("Reward per episode | CartPole")
          plt.xlabel("No. of episodes")
          plt.ylabel("Reward")
          plt.show()
```



We can see in our evaluation there are some fluctuations in reward between (200, 500). Our agent learned good policy, it was able to get and maintain a positive score close to optimal policy.

2. Lunar Lander

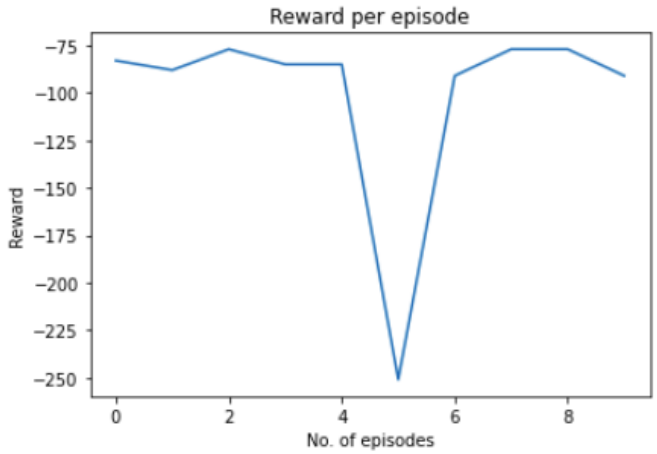
```
In [266]:  plt.plot(np.arange(len(eval_episode_rewards)), eval_episode_rewards)
           plt.title("Reward per episode | LunarLander")
           plt.xlabel("No. of episodes")
           plt.ylabel("Reward")
           plt.show()
```



We can see there are some fluctuations while evaluating in lunar lander learning, this maybe due to excessive training. During training we could see there are many continuous episodes where LunarLander receives a score more than 200. These fluctuations can be reduced further with the help of hyper-parameter tuning.

3. Acrobot

```
In [279]: plt.plot(np.arange(len(eval_episode_rewards)), eval_episode_rewards)
          plt.title("Reward per episode")
          plt.xlabel("No. of episodes")
          plt.ylabel("Reward")
          plt.show()
```



While evaluating for 10 episodes we can see the reward is negative but somewhat constant. This means that our agent was learning and was converging to some results (though not to expected results). On further hyper-parameter tuning we can say that, the agent will get positive reward.

Reference:

1. OpenAI gym Github: https://github.com/openai/gym/*
2. OpenAI gym website: https://gym.openai.com/
3. Class Notes and Lecture slides
4.https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f
5.https://github.com/parvkpr/Simple-A2C-Pytorch-MountainCarv0/blob/master/doodle.py