Reinforcement Learning

Assignment 3

Sumeet Aher
sumeetmi
sumeetmi@buffalo.edu

Shengfeng Xue
shengfen
shengfen@buffalo.edu

"I certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I did not receive any external help, coaching or contributions during the production of this work."

Abstract

- Develop A2C algorithm and implement on the below three environments:
    - CartPole-V1
    - LunarLander-V1
    - BipedalWalkerV3

1 Discuss the algorithm you implemented.

1. Regularly, the actor-critic algorithm is done in these parts:
    a) Setting up actor-critic NN, hyperparameters.
    b) Doing the training loop for a certain number of max episodes until the recent 100 episodes yield 75% of almost-full-score rewards.
        i) Running the agent in some timesteps and keeping record of actor values, critic values and rewards.
        ii) Getting expected returns.
        iii) Computing the loss.
        iv) Updating the network using gradients derived from the loss.
    c) Testing the agent.
    d) Plotting the episode rewards.

In this case, A2C is used. It replaces the original rewards in the critic NN with the advantage function as a measurement of the comparison of values between the selected action and the average of all actions.

1.1 What is the main difference between the actor-critic and value based approximation algorithms?

AC is different from value-based algorithms in that the former uses policy gradient while the latter samples a large number of values.

1.2 Briefly describe THREE environments that you used (e.g. possible actions, states, agent, goal, rewards, etc). You can reuse related parts from your Assignment 2 report.

1. CartPole-V1

- Action space = 2
- Observation space = 4
- Goal: To hold the cartpole upright for as much time as possible. The env is solved if for 10 episodes has an average reward of more than 470.
- Reward: A reward of 1 is provided for every timestep the cartpole is upright

2. Lunar Lander V2

- Action space = 4
- Observation space = 8
- Goal: To land the lunar lander between the flag on coordinates (0,0).
- Reward: Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points

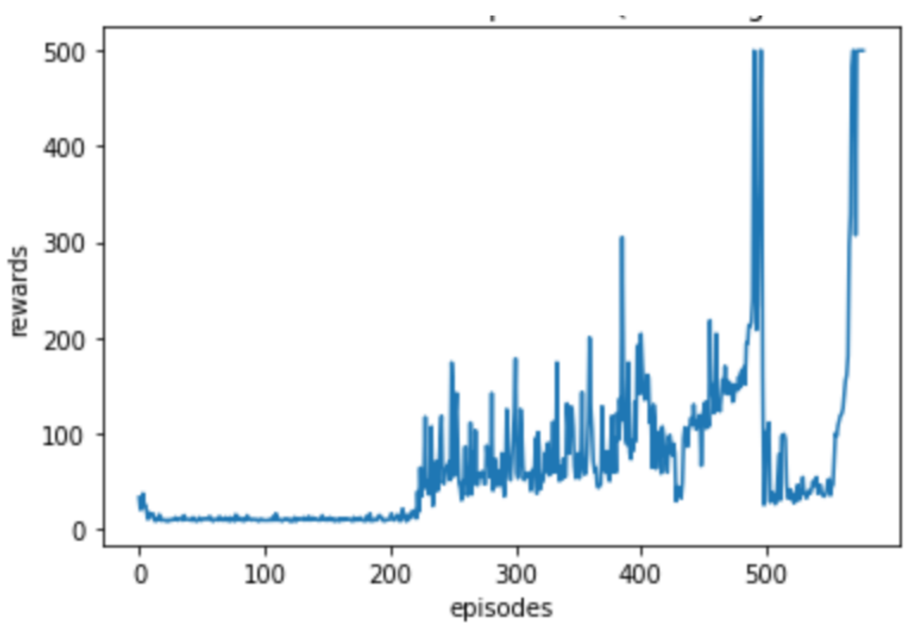
3. Bipedal Walker V3

- Action space = 4
- Observation space = 24
- State: State consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements.
- Goal: Bipedal walker has to keep walking upright. On achieving a average reward of more than 300 for consecutive 10 episodes, the env is considered as sovled.
- Reward: Reward is given for moving forward, total 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small amount of points, more optimal agent will get better score.

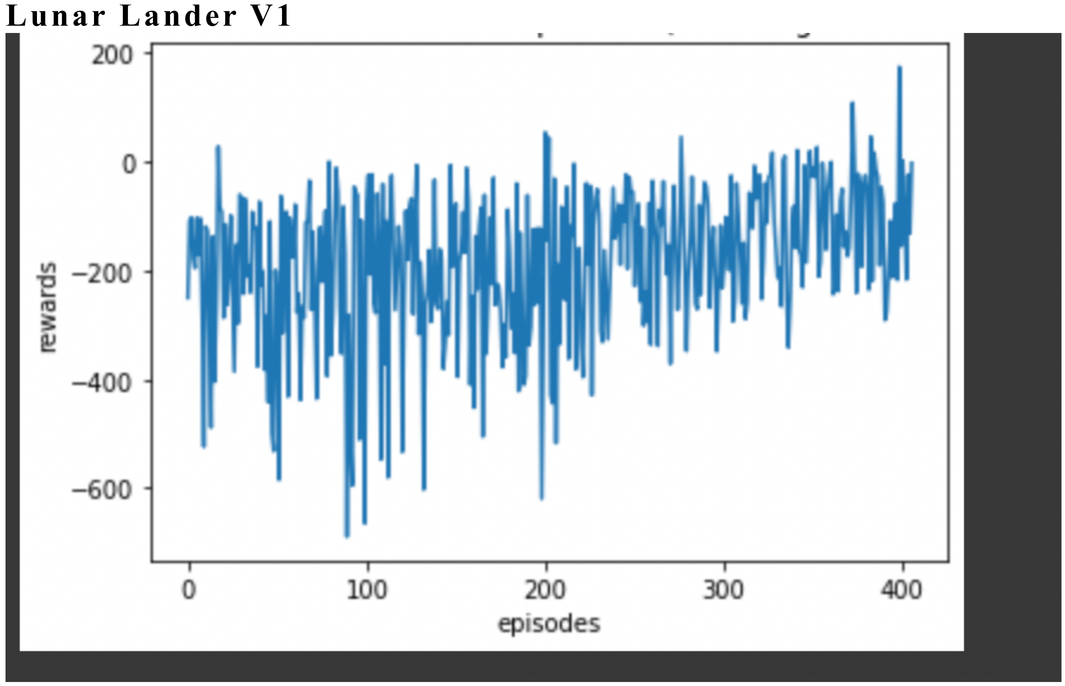
1.3 Show and discuss your results after training your Actor-Critic agent on each environment. Plots should include the

reward per episode for THREE environments. Compare how the same algorithm behaves on different environments while training.

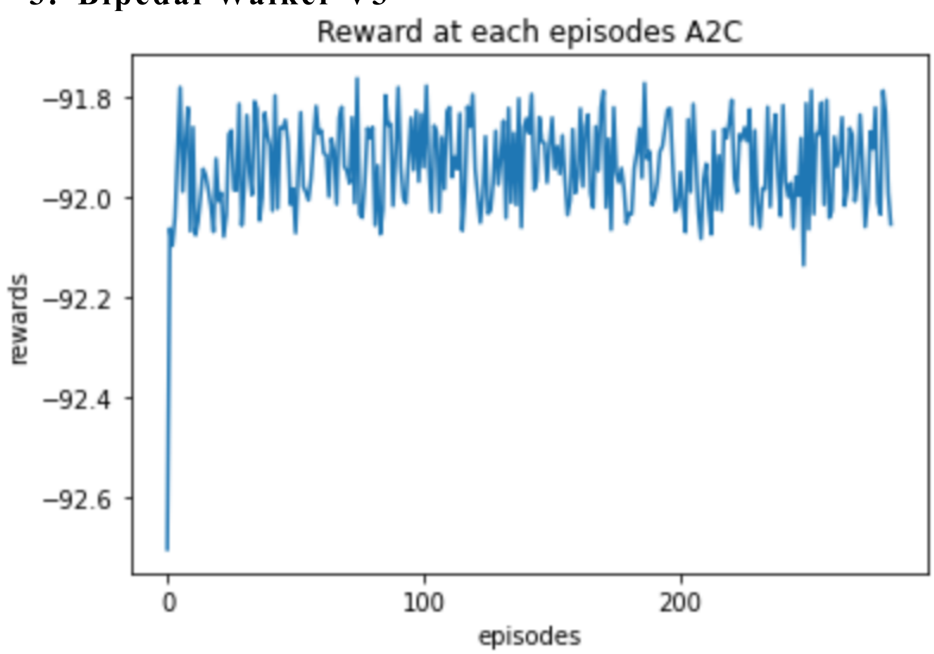
1. **CartPole V1**



2. **Lunar Lander V1**



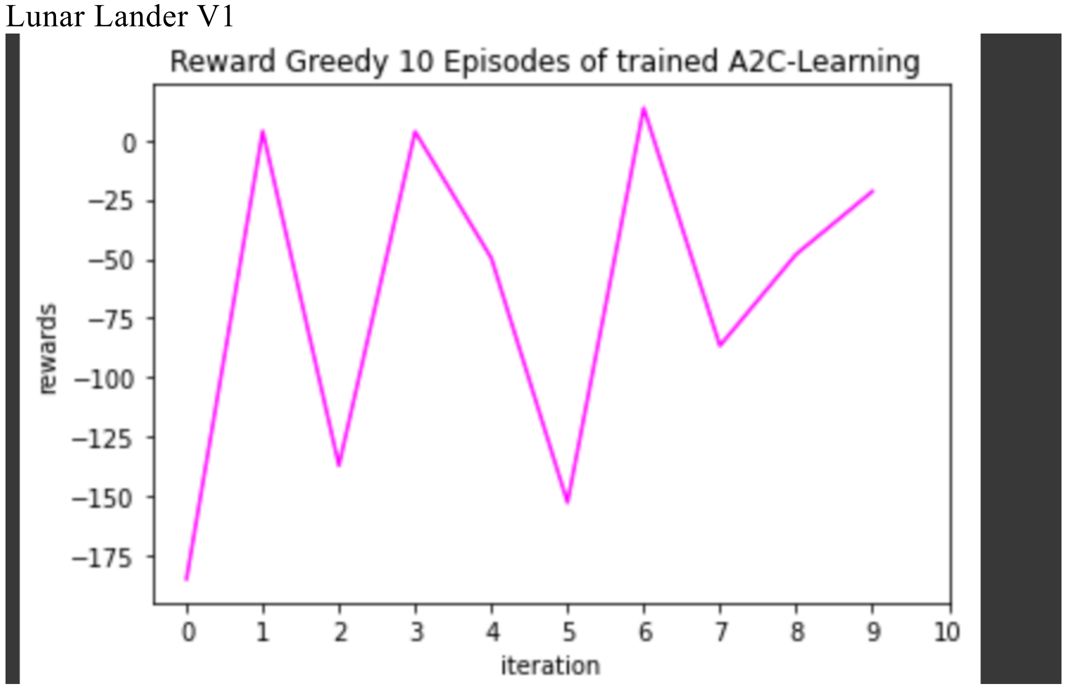
3. **Bipedal Walker V3**



2 Provide the evaluation results for each environments that you used. Run your environments for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

1. CartPole V1

2. Lunar Lander V1



3. Bipedal Walker V3

2.1 Contribution:

Name	Contribution
Sumeet Aher	**Equal**
Shengfeng Xue	**Equal**

References

[1] https://gym.openai.com/docs/

[2] https://gym.openai.com/envs/LunarLander-v2/

[3] https://github.com/Kyziridis/BipedalWalker-v2/blob/master/actor_critic/actor_Lstm.py

[4] https://towardsdatascience.com/how-to-create-a-custom-loss-function-keras-3a89156ec69b

[5] https://github.com/hermesdt/reinforcement-learning/blob/master/a2c/cartpole_a2c_online.ipynb

[6] https://github.com/marload/DeepRL-TensorFlow2/blob/master/A2C/A2C_Discrete.py

[7]https://gym.openai.com/envs/CartPole-v1/

[8] https://gym.openai.com/envs/BipedalWalker-v2/