Reinforcement Learning Assignment 2 - Group 05

Vinay Kudari
University at Buffalo
Buffalo, NY, 14215
vkudari@buffalo.edu

Ujwal Sri Harsha Orchu
University at Buffalo
Buffalo, NY, 14215
ujwalsri@buffalo.edu

Abstract

Evaluate Advantage Actor Critic and Soft Actor Critic Algorithms on Maze, Cart-Pole and LunarLander environments

1 Algorithms Implemented

1.1 Advantage Actor Critic (A2C)

Actor Critic algorithm is a combination of Policy Based and Value Based Algorithms. It has two function approximators (here, Neural Networks), namely actor network and critic network. The policy is estimated by actor network and the value by the critic network. Both the actor and critic help each other, actor adjusts its predictions based on critics judgement. Advantage Actor-Critic specifically uses estimates of the advantage function for its bootstrapping. Advantage function indicates how good an action is compared to other actions possible at that particular state.

1.2 Soft Actor Critic (SAC)

Soft Actor Critic, or SAC, is an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework. In this framework, the actor aims to maximize expected reward while also maximizing entropy. That is, to succeed at the task while acting as randomly as possible. Prior deep RL methods based on this framework have been formulated as Q-learning methods. SAC combines off-policy updates with a stable stochastic actor-critic formulation.

2 Difference between Value based and Actor critic methods

In Value based methods, we try to learn how good a state is or an action is based on the estimated action value, and select the best action. In policy based methods, we directly estimate the action distribution by parameterizing the policy. Actor critic methods are a combination of both, we make use of two function approximators, like neural networks, to represent policy which estimates the action distribution and the critic which estimates the value of the state. Policy gets updated according to the direction suggested by critic.

3 Environment

3.1 CartPole-v1

The cartpole environment consists of a pole attached to a cart that drives over a friction-less track by an un-actuated joint. A force of +1 or -1 is applied to the cart to control the system. The pendulum starts upright, and the goal is to prevent it from falling over. Every time-step that the pole remains upright the environment awards a +1 reward. When the pole is more than 15 degrees from vertical or the cart goes more than 2.4 units away from the center, the episode terminates.

CartPole-v1 was derived from the original Cartpole-v0 environment, below are the key differences between environments:

1. The maximum timesteps have been extended from 200 to 500.
2. The reward limit has been raised from 195.0 to 475.0.

Other than that the definition of the Cartpole-v1 as defined by OpenAI Gym (`https://gym.openai.com/envs/CartPole-v1/`) is:

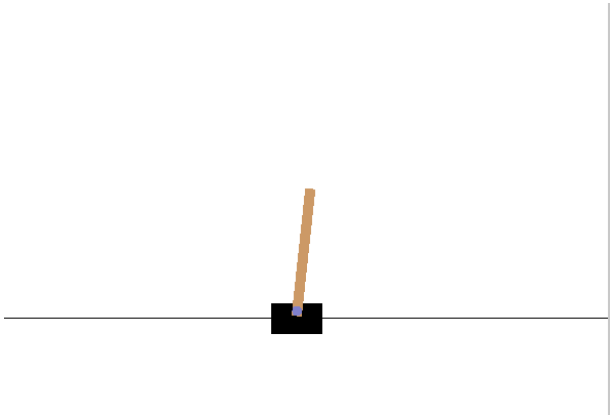


Figure 1: Cartpole-v1 Environment

3.2 Lunar LanderContinuous-v2 / Discrete-v2

The LunarLander environment consists of a moon lander that is supposed to land at a given target(Launch Pad). The Launch pad is always at (0,0). The goal is to land the lander at (0,0) and have a speed of 0 at point of impact.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. Landing outside landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt. Four discrete actions available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine.

Figure 2: LunarLanderContinuous-v2 Environment

3.3 Maze Environment

The goal of the environment is to Solve maze by reaching goal state optimally avoiding forbidden states. They include out-of-bounds positions, walls and already visited states.

Actions are Up, Down, Left and Right

Agent position tuple (x, y) determines the current state. Environment with 16 states has been chosen for convenience but can be easily modified by passing a larger maze to GridEnv class. States with value of 1.0 are legal positions in which our agent can move, whilst others with a value of 0.0 are walls that our agent should avoid

Initially rewards were given according to the euclidean distance between agent and the goal, but policy was not able to converge in a limited time. Later, new dynamics were chosen and they seem to work well for the current task. The given pairs represent the appropriate reward dynamics, with key representing the current state type and value being the reward

1. Goal: +20.0
2. Wall: -5.0
3. Out of bounds: -0.75
4. Visited: -0.85
5. Legal Move: -0.05

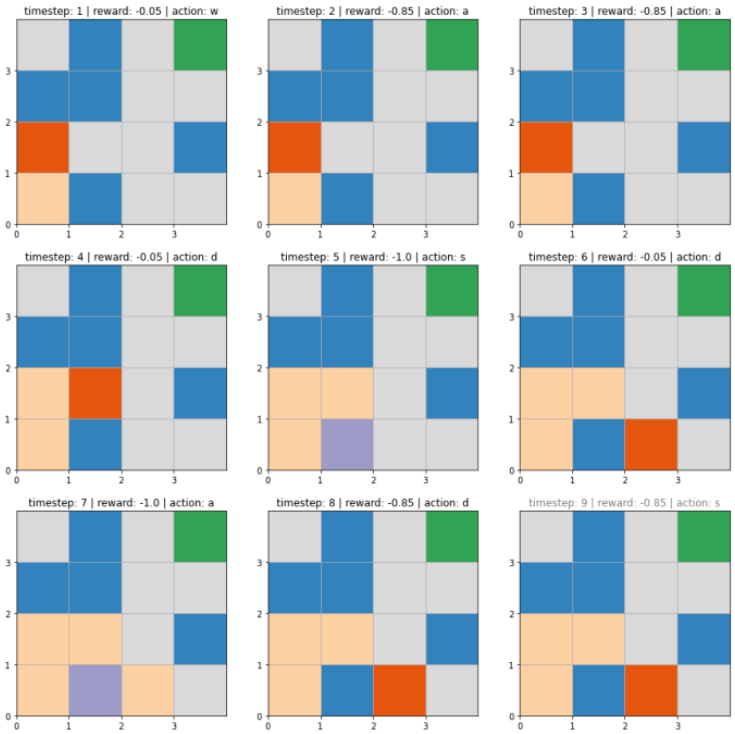


Figure 3: Maze Solver Environment

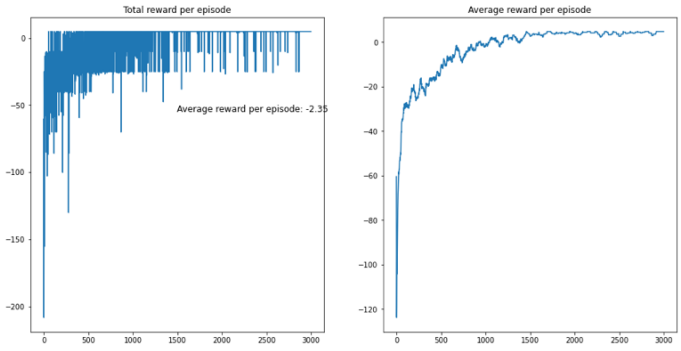
4 Training Results

4.1 Maze

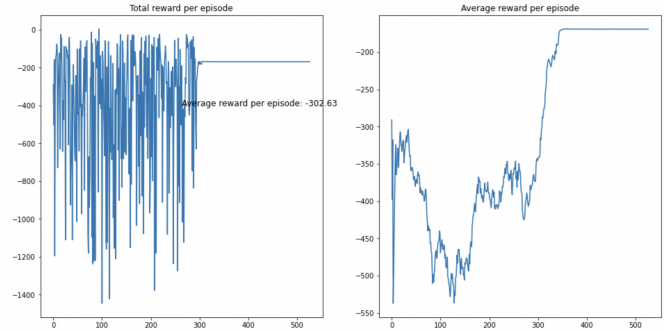
Following are the results for Maze environment, after applying Advantage actor critic and Soft actor critic algorithms. As we can see from the graphs, a2c agent converges around 1500 episodes.

Whereas, soft actor critic agent doesn't seem to converge within 500 episodes. Due to time constraints we couldn't train it for a longer period. We also noticed that SAC algorithm works best for continuous action spaces.

4.1.1 Advantage Actor Critic



4.1.2 Soft Actor Critic

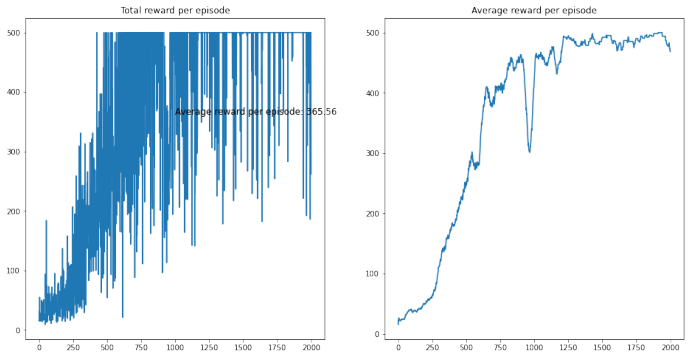


4.2 Cartpole-V1

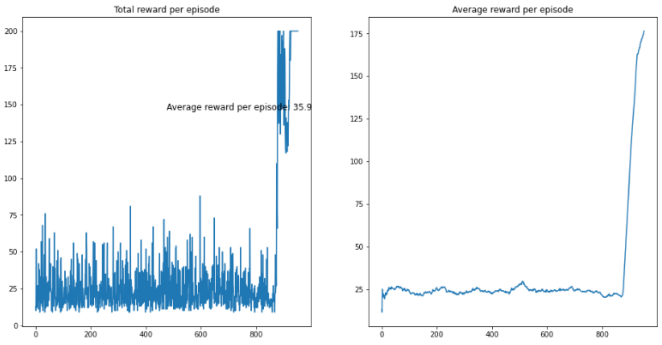
The environment converges at around 1500 episodes when A2C algorithm is implemented. Whereas, doesn't seem to converge for 1000 episodes using SAC. The maximum reward obtained using SAC is 200.

The following plots are the results of solving the OpenAI Gym Environment: Cartpole-v1

4.2.1 Advantage Actor Critic



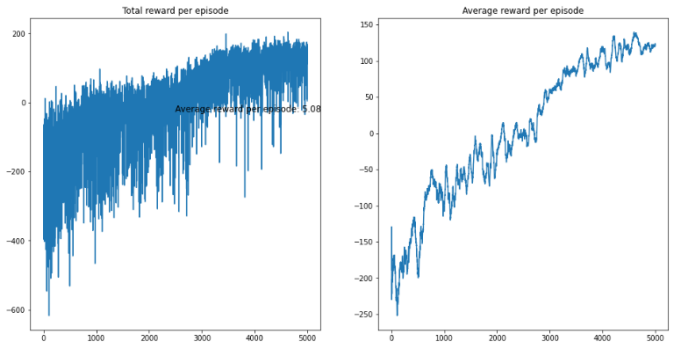
4.2.2 Soft Actor Critic



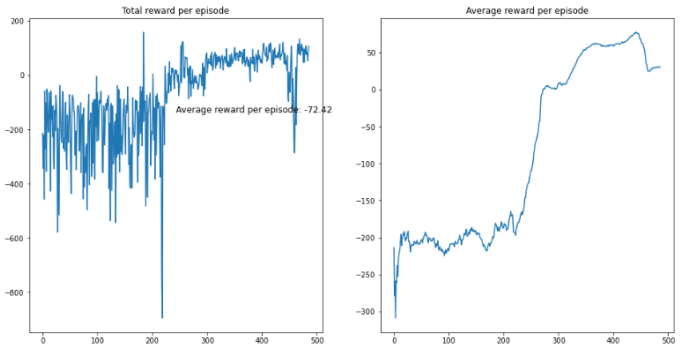
4.3 LunarLander-V2

The following plots are the results of solving the OpenAI Gym Environment: LunarLander-v2
A2C solves the environment at around 4000 episodes. SAC agent doesn't converge unlike A2C within the same time period.
To show how SAC performs on a continuous action space, we've implemented SAC for continuous action spaces on LunarLanderContinuous-v2 gym environment. This environment is an extension of the original LunarLander-v2 environment. The primary difference is that this environment has a continuous action space instead of a discrete action space.
The Agent has been trained for 500 episodes and the loss has been noticed to reduce drastically. As we can see from the plots, the algorithm is close to convergence. Due to time constraints, we were unable to run this algorithm for much longer. However, SAC is definitely a major improvement over a2c.

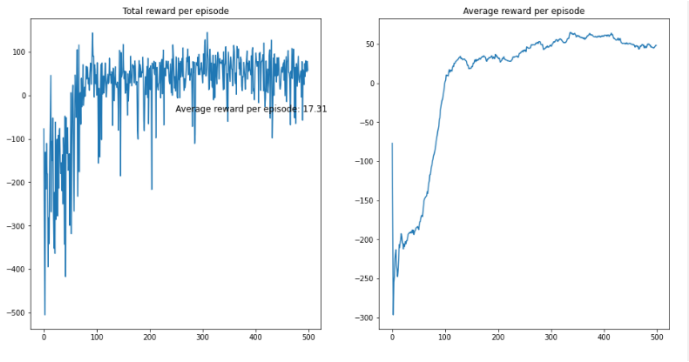
4.3.1 Advantage Actor Critic



4.3.2 Soft Actor Critic



4.3.3 Soft Actor Critic on Continuous Action Space



5 Evaluation Results

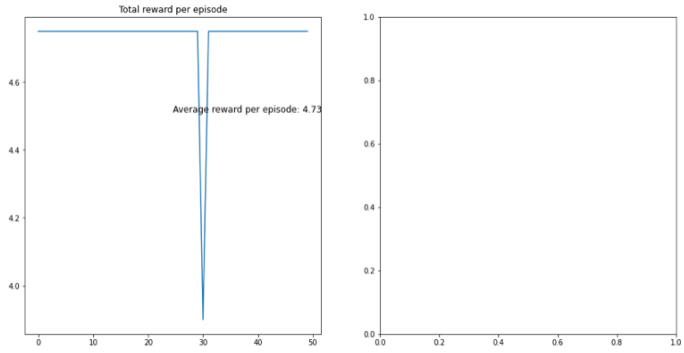
Following are the results of evaluating the environments over 50 episodes using only greedy policy

5.1 MazeSolver

Evaluation Results of MazeSolver environment over 50 test runs

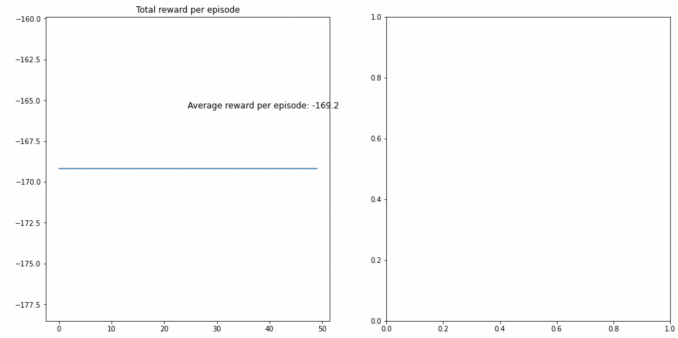
5.1.1 MazeSolver using A2C

Evaluation Results of MazeSolver environment using vector state representation over 50 test runs



5.1.2 MazeSolver using SAC

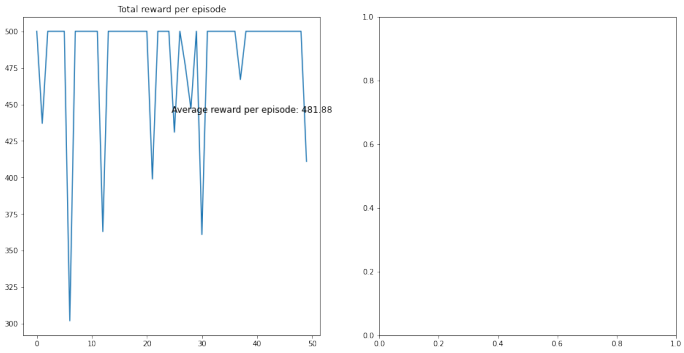
Evaluation Results of MazeSolver environment using vector state representation over 50 test runs



5.2 CartPole-v1

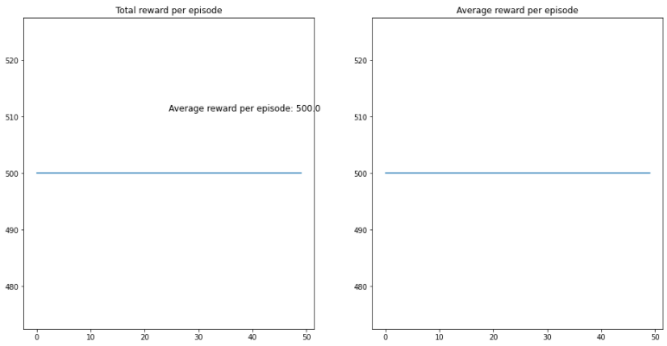
5.2.1 CartPole-v1 using A2C

Evaluation Results of CartPole environment using A2C Agent over 50 test runs



5.2.2 CartPole-v1 using SAC

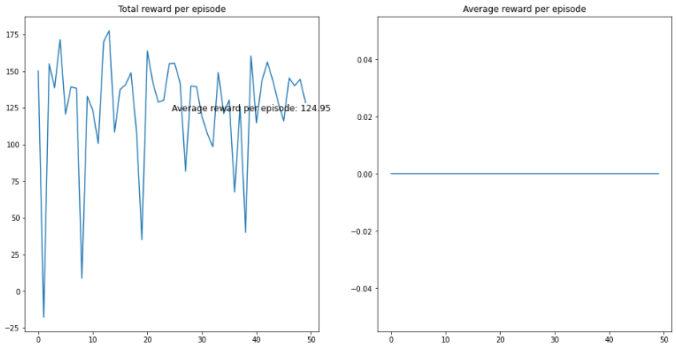
Evaluation Results of MazeSolver environment using vector state representation over 50 test runs



5.3 LunarLander-v2

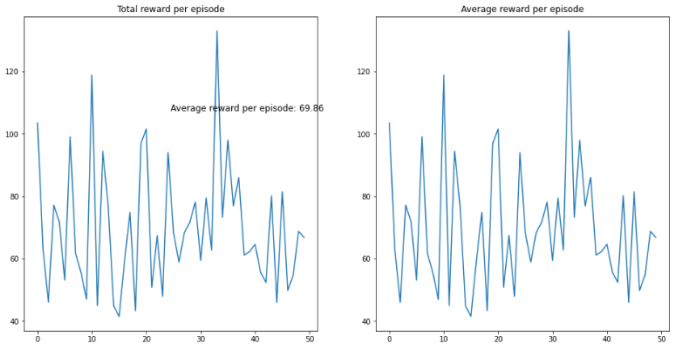
5.3.1 LunarLander-v2 using A2C

Evaluation Results of LunarLander-v2 environment using A2C Agent over 50 test runs



5.3.2 LunarLander-v2 using SAC

Evaluation Results of LunarLander-v2 environment using soft actor critic algorithm over 50 test runs



6 Contribution Table

Team Member	Assignment Part	Contribution
Ujwal Sri Harsha Orchu	Part-1 and Part-2	50
Vinay Kudari	Part-1 and Part-2	50

7 References

https://gym.openai.com/envs/classic_control for different environments