Assignment3 - Actor-Critic

Saeyeon Hwang

May 2, 2022

1 Discuss the algorithm you implemented.

The algorithm implemented in this project is TD Actor-Critic. In Actor-Critic, we use both Value Function and Policy Gradients. Actor refers to policy gradients, and critic refers to value function. Actor updates policy parameters suggested by critic, and it conducts actions in an environment. On the other hand, critic updates action-value function parameter, and computes value functions to help actor learning. The Actor Critic can be different from what policy gradient we use. In this project, TD Actor-Critic is used. So the target is reward + gamma*next value. The weight is updated by compute loss between target and value. Also, the parameter is updated by compute (target - value + log probability of action).

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

The main difference between the actor-critic and value based approximation algorithms is actor-critic algorithm doesn't use replay buffer. Furthermore, actor-critic and value based algorithms both use value function, but value based algorithms doesn't have explicit policy unlike actor-critic algorithms.

3 Briefly describe THREE environments that you used

3.1 CartPole-v1

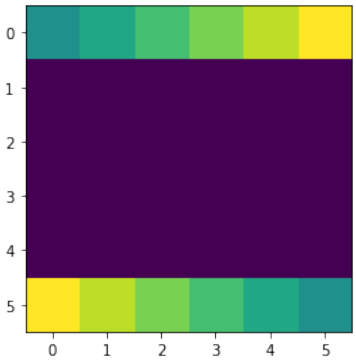
The goal of CartPole problem is to prevent the cartpole from falling over. There are two possible actions in this problem. Possible actions are moving cart to the left or right. Possible states are defined by 4-dimensional states. They are theta, y, theta dot, y dot. Theta means the angles of the pole, y means the position of the cart, and theta dot, y dot means the each velocities. If cartpole isn't falling over at every timestep, +1 reward is added. The episode ends when

the timestep is larger than 500, pole is tilted over 12 degrees from vertical, or the cart moves more than 2.4 from the center.

3.2 LunarLander-v2

The goal of LunarLander problem is to make lander land safely on the landing pad. There are four possible discrete actions available in LunarLander problem. Do nothing(0), fire left orientation engine(1), fire main engine(2), fire right orientation engine(3). Possible states are defined by 8 continuous dimensions, and the range of the space is infinite. If lander moves away from landing pad, it loses reward back. If episode finishes by crushed landing or safe landing, rewards are -100 or +100 points. Firing main engine uses -0.3 points each frame, and the fuel is infinite. Moreover, if lander lands ground with leg, 10 points are added. Last, 200 points of rewards are added if the environment is solved.

3.3 Multi-agent GridWorld

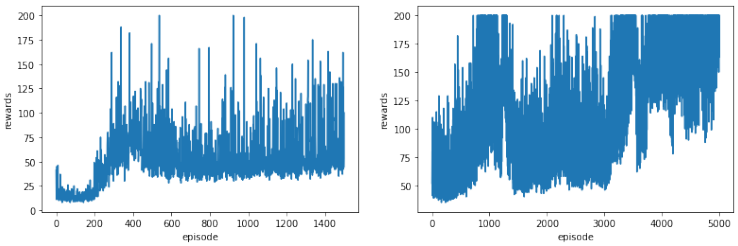


The goal of Multi-agent Gridworld is to find goal position of each agents. The possible states are 6x6 grid world, and the number of agents are 6. The goal positions of each agents are [[0, 5], [0, 4], [0, 3], [0, 2], [0, 1], [0, 0]]. And the agents start with the position [[5, 0], [5, 1], [5, 2], [5, 3], [5, 4], [5, 5]]. The picture above shows the reset environment of Multi-agent Gridworld There are total five discrete actions; Down(0), Up(1), Right(2), Left(3), No move(4). By passing actions to the algorithms, we can compute distance between goal position and agent position. And rewards are added by comparing distance between past distance and new distance. New distance means the distance between goal position and agent position, and past distance means the distance between old position and goal position. If new distance is less than old position, the -0.1 points of reward are added. On the other way, if new distance is bigger, the -0.5 points of reward are added. And if agent arrives at the goal position, +1 point of reward is added.

4 Show and discuss your results after training your Actor-Critic agent on each environment.

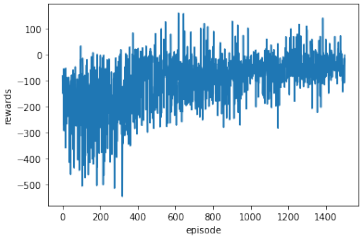
The number of episode was 1500, and the maximum time step was 200 in each environment.

4.1 CartPole-v1



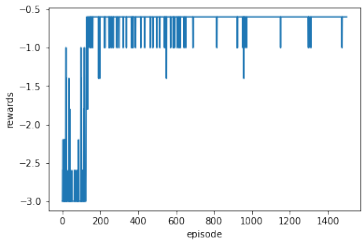
In CartPole, the reward from first episode was about 41, and the reward reaches to 200, which is the maximum rewards. The graph above shows the training process to the environment was quite well-trained. Because the training process doesn't look like obvious, the training process was proceeded with 5000 episodes. The right above graph shows the result, and it indicates the training went well, because the reward goes between 100 and 200 at the end of the episode.

4.2 LunarLander-v2



In LunarLander, the reward from first episode was about -81 and the reward reaches to over 100 as episodes progressed. The graph above shows the training process to the environment was quite well-trained. The reward was negative numbers, but the rewards become bigger as episodes progressed. The

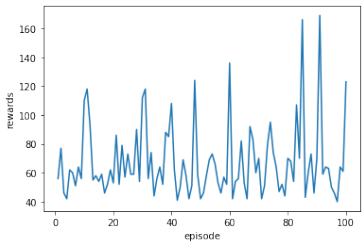
4.3 Multi-agent GridWorld



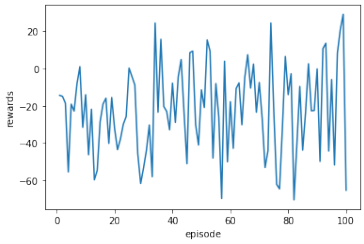
In Multi-agent GridWorld, the maximum reward was -0.6. And the reward from the first episode was -3.0. The above graph shows that the multi-agent environment is trained well because the rewards gradually increases. However, it's not a quite good result, because the agent will get +1 if it arrives at the goal position. The cumulative rewards -0.6 means it never reached to the goal position because the sum of 6 agents if the agents got -0.1 point each is -0.6. The training graph above shows the agents infinitely got close to the goal position, but never reached.

5 Provide the evaluation results for each environments that you used.

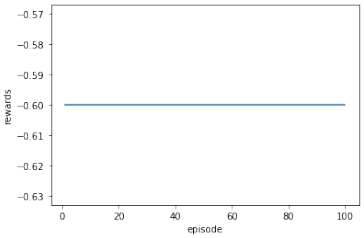
5.1 CartPole-v1



5.2 LunarLander-v2



5.3 Multi-agent GridWorld



The number of episode was 100, and the maximum time step was 200 in each environment. To make the agent chooses only greedy actions, the codes are modified from choosing random action using sample module to choosing action which has maximum probabilities.

In CartPole, although rewards oscillate, the rewards that agent get become bigger as episodes progresses. It means that as the episode progresses, the reward that can be obtained from greedy action increases. The same thing occurs with in LunarLander too. However, In Multi-agent GridWorld, the most of maximum rewards was -0.6. So the evaluation graph looks like a line because the rewards would be -0.6 if the agent only chooses greedy actions.