林柏均 107062240

April 12, 2021

Deep Reinforcement Learning

Homework 2

What have I done in this assignment?

Instead of using traditional Deep Q-Learning, this time I used double DQN to train my agent. The reason why I abandoned traditional way to train my agent because of the performance. After 10000 episodes, the total reward from the simulation was still negative. Therefore, I started to search whether there was a better way to train my agent, and I found that Rainbow was a very good training model, however, I still not quite understand the mechanism of the model, thus, I choose double DQN as my final decision.

Double Deep Q-Learning

Traditional Deep Q Learning tends to overestimate the reward, which leads to unstable training and lower quality policy. Let's consider the equation for the Q value:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{a'} Q^*(s',a') |\ s,a]$$

The last part of the equation takes the estimate of the maximum value. This procedure results in systematic overestimation, which introduces a maximization bias.

Since Q-learning involves learning estimates from estimates, such overestimation is especially worrying.

To avoid such a situation, I will define a new target network. The Q values will be taken from this new network, which is meant to reflect the state of the main DQN. However, it doesn't have identical weights because it's only updated after a certain number of episodes. This idea has been first introduced in Hasselt et al., 2015. The

addition of the target network might slow down the training since the target network I is not continuously updated. However, it should have a more robust performance over time.

Description of my model

Deep Q Network:

```
model = torch.nn.Sequential (
        torch.nn.Linear(state_dim, hidden_dim),
        torch.nn.LeakyReLU(),
        torch.nn.Linear(hidden_dim, hidden_dim * 2),
        torch.nn.LeakyReLU(),
        torch.nn.Linear(hidden_dim * 2, action_dim)
)
```

I have three Linear layers and two LeakyReLU layers in the model. First Linear layer input size is state dimension, which is 84, and output to next layer with 50 nodes, so other layer you can follow the variables I shown above to conclude.

Q-Learning:

Below I will show the procedure of the Q - Leaning algorithm:

(1) Initial final_reward, memory, current episode and replay time.

(2) For every episode, we will update the target network every 10 episodes and reset the environment.

(3) In while loop, if the game is not over :

a. Random search if the random variable is smaller than epsilon, or we choose best q_value in the current state as our next action.

b. Take action and add reward to total

c. Update total and memory

d. If the game is over, then break the loop, otherwise, update network weights using replay memory.

(4) Update epsilon, final reward and show episode running time.

Conclusion

I have heard Q - Learning this learning model for so long, however, I haven't had chance to implement it. Honestly, it is a very interesting training method, but I have to spend lots of time to understand the mechanism of the model. In this assignment, I choose Double Deep Q - Learning to improve my agent's performance, after 10000 episodes of training, it seems that this improved version of the Deep Q - Learning has indeed better learning performance.