Deep Reinforcement Nano Degree Project 1: Navigation

Peng Zhao

December 19, 2019

Abstract

This report briefly summarizes the deep reinforcement nano degree project 1, which is training the agent to collect yellow banana using deep reinforcement learning method. The problem is solved after 191 episodes in this experiment.

1 Deep Q-Network Reinforcement Learning Algorithm

In deep Q-learning, a neural network is used to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as output. The steps of deep Q-Network reinforcement learning algorithm are:

- Step 1: Store experience in replay memory
- Step 2: Obtain minipacth of tuples (s_j, a_j, r_j, s_{j+1}) from memory
- Step 3: Set target $y_j = r_j + \gamma \max_a \hat{q}(s_{j+1}, a, w^-)$
- Step 4: Update: $\Delta w = \alpha(y_j - \hat{q}(s_j, a_j, w))\nabla_w \hat{q}(s_j, a_j, w)$
- Step 5: Go back to Step 2 until beyond the replay times
- Step 6: Go back to Step 1 until reach the required score

where s_j, a_j, r_j are the state, action, and reward respectively in episode j, $\hat{q}$ is the action-value function, w represents the weights, w^- represents the target action-value weights, α is the learning rate. The Step 4 is using the gradient descending algorithm.

2 DQN Architecture

The Deep Q-Network consists of 3 layers with Relu activation. The input data size of the first layer is the state size, which is 37. The output size of the first layer is 64. The output size of the second layer is also set to 64 and the out put of the third layer is the action size 4.

3 Parameter

The choice of ε-greedy parameter ε can significant effect the speed of learning. In this experiment, I use the following equation to determine this parameter:

$$\varepsilon = \max(\varepsilon_0 \times d, \varepsilon_{min}) \tag{1}$$

where ε_0 is the initial value of ε, d is decay rate for each step in episode, ε_{min} is the minimum value that ε can use. In this experiment, we use $\varepsilon_0 = 0.5$, $d = 0.98$, $\varepsilon_{min} = 0.01$.

The other hyper parameters used in this experiment are shown in Table 1.

Parameter term	Value
replay buffer size	1e6
minibatch size	128
discount factor	0.99
learning rate	1e-4
target parameter for soft update	1e-3

Table 1: Hyper parameters

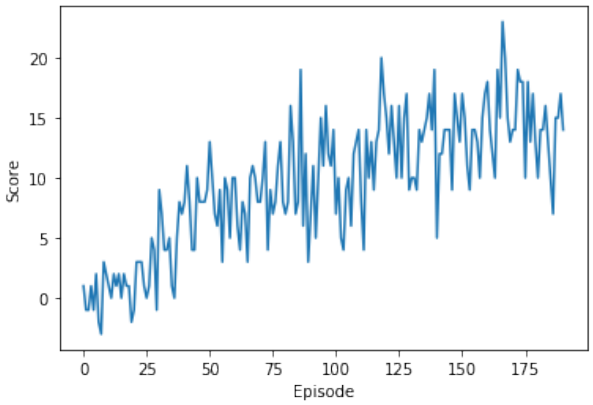

Figure 1: Result

4 Result

The environment has been solved in this experiment, which is shown in the following figure. As shown, after an initial training, the agent began to obtain higher scores after about 25 episodes. The required score (13.0) is finally reached after 191 episodes.

5 Future Work

I observed the required episodes to solve the environment varies with the training run. The initial training episodes where the agent obtains low scores significantly effect the total required episodes. To reduce the total episodes, a potential method might be reducing the replay frequency in the initial training. This is because, in the initial training episodes, the action tends to be random and little can be learned. Thus, the next I will use unfixed replay frequency that is relative small in the beginning phase to see whether the required episodes can be reduced.