

Deep Learning Lab Course: Assignment 04

Yufeng Xiong

January 30, 2017

1 Q-learning

In exercise 1, I wrote the update-rule of Q-learning function and calculated the Q-function values on a simple grid-world after the initial episode:

DL Lab course : Exercise 4

1. Q-Learning Yufeng Xiong

1.1. if j is the goal state :

$$Q_{k+1}(i, u) \leftarrow Q_k(i, u) + \alpha (r(i, u) + \gamma \max_{u'} Q_k(j, u') - Q_k(i, u))$$
$$\leftarrow Q_k(i, u) + \alpha (0 + 0.5 \cdot 0 - Q_k(i, u))$$

else if j is not goal state :

$$Q_{k+1}(i, u) \leftarrow Q_k(i, u) + \alpha (r(i, u) + \gamma \max_{u'} Q_k(g, u') - Q_k(i, u))$$
$$\leftarrow Q_k(i, u) + \alpha (-1 + 0.5 \cdot \max_{u'} Q_k(g, u') - Q_k(i, u))$$

1.2.

Grid-world.

Action

1	2	3	4	5	6	7	8
-1	-1						
2	-1	-1		-1			
3		-1		-1			
4			-1	-1	-1	-1	
5			-1	-1	-1	-1	
6				-1		-1	
7					-1	-1	0
8							0

Reward table

① state 1 → state 2 :

$$Q(1,2) = R(1,2) + 1.0 * \max\{Q(2,1), Q(2,3), Q(2,5)\} = -1$$

② state 2 → state 5 :

$$Q(2,5) = R(2,5) + 1.0 * \max\{Q(5,2), Q(5,4), Q(5,5), Q(5,7)\} = -1$$

③ state 5 → state 5 :

$$Q(5,5) = R(5,5) + 1.0 * \max\{Q(5,2), Q(5,4), Q(5,5), Q(5,7)\} = -1$$

④ state 5 → state 7 :

$$Q(5,7) = R(5,7) + 1.0 * \max\{Q(7,5), Q(7,6), Q(7,8)\} = -1$$

⑤ state 7 → state 8 :

$$Q(7,8) = R(7,8) + 1.0 * \max\{Q(8,8)\} = 0 , \text{ then the Q table:}$$

	1	2	3	4	5	6	7	8	Actions
1	0	-1	0	0	0	0	0	0	
2	0	0	0	0	-1	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	
5	0	0	0	0	-1	0	-1	0	
6	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	

States Q-function table

2 Deep Q-learning

2.1 Network Architecture

The implemented deep Q-learning network is a LeNet5-like neural network, used to predict the action of the deep q-learning agent for a simple maze task, the detailed neural network layers are:

- a. convolutional layer 1
- b. pooling layer 1
- c. convolutional layer 2
- d. pooling layer 2
- e. fully connected layer 1
- f. fully connected layer 2

2.2 Implementation details

2.2.1 Some parameters

training steps: 200,000

testing steps: 30,000

batch_size: 128 (to make the model more stable)

epsilon = 0.2

2.2.2 Memory replay

In the first 3000 steps, the deep Q-learning agent will not learn anything, just takes random actions for memory replay.

After the first 3000 steps' memory replay, the agent will start to learn. In addition, I used epsilon-greedy algorithm for agent exploration, to avoid the agent overestimating the Q-function values.

2.2.3 Saved model

In my example, after 120,000 training steps, the deep Q network can reach some kind of convergence. I saved the trained model in the /model directory after 200,000 steps. At then the deep Q agent becomes a very intelligent agent for the sample maze task, just need very a few steps to reach the terminal state.

2.3 Testing results

In testing phase, I run the saved model for 30,000 steps, the accuracy can reach 99.904%.