CSE 474/574: Introduction to Machine Learning

Project 4 Report: Tom and Jerry in Reinforcement Learning

Xinyi Song
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14228
xinyison@buffalo.edu

1 Introduction

In this project, we are assigned to teach the agent to navigate in the grid-world environment by combining reinforcement learning and deep learning. The game is that Tom, a cat, is chasing Jerry who is a mouse. Our task is to find the shortest path to the goal (Jerry), given that the initial positions (Tom and Jerry). We would apply deep reinforcement learning algorithm – DQN (Deep Q-Network) to solve the problem.

2 Tasks Implementation

The coding consists of four classes, Environment, Memory, Brain and Agent. Class Environment defines grid-world environment, behaves actions from agents, makes changes to new state and gives reward to agent. Class Memory records post experiences. Brain is where the neural network model is created and held. Agent uses Memory and Brain to observe current state and make an action, stores the record into memory and trains the brain based on updated memory.
There are two tasks of the project, coding tasks and writing tasks. Coding part includes building a 3-layer neural network using Keras library for the deep learning part and writing code in Python for the reinforcement learning part. As for the writing part, we have to answer two questions mentioned in project4_task.pdf.

2.1 Coding tasks

There are three coding tasks and the implementations are shown as following.

1. Neural Network

A 3-layer neural network acts as the Deep Q-Network (DQN). The coding is shown in figure 2-1.

```
model.add(Dense(units=128, activation='relu', input_dim=state_dim))
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=action_dim, activation='linear'))
```

Figure 2-1 Coding for Neural Network

The inputs of neural network are the current positions of Tom and Jerry. The neural network has two hidden layers and one output layer, which outputs a vector of Q-values for each action possible in the given state. Later, we will choose such an action that will return the highest Q-value.

2. Implement Exponential-Decay Formula

The agent will randomly select its action at first by a certain percentage in order to make the agent try all kinds of things before it starts to see the patterns, called this process 'epsilon'. But we also want our agent to decrease the number of random action, so exponential-decay formula for epsilon is introduced. The formula is $\epsilon=\epsilon min+(\epsilon max-\epsilon min)*e-\lambda|S|$, and the coding part is shown in figure 2-2.

```
self.epsilon = self.min_epsilon + (self.max_epsilon - self.min_epsilon) * np.exp((-1) * self.lamb * np.abs(self.steps))
```

Figure 2-2 Coding for epsilon

When using default parameters, epsilon values decay along with episode is shown in figure 2-3.

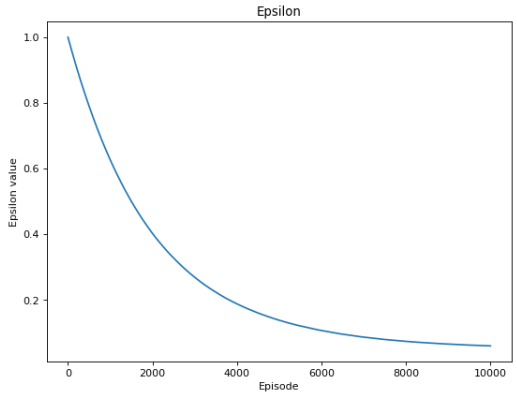

Figure 2-3 Epsilon decays along episode

3. Implement Q-function

First, the formula of Q-function is

$$Q_t = \begin{cases} r_t, & if\ episode\ terminates\ at\ step\ t+1 \\ r_{t+\gamma max_a Q(s_t,a_t;\theta)}, & otherwise \end{cases}$$

In addition, the coding part is shown in figure 2-4.

```
if st_next is None:
  t[act] = rew
else:
  t[act] = rew + self.gamma * np.max(q_vals_next[i])
```

Figure 2-4 Coding for Q-function

Because we have no training data fed into the reinforcement learning model, model needs to generate data in the process of learning. Variable t is the target value passed through the neural network, which attempts to maximize discounted accumulative reward. So after training neural network model based on these data, the agent should be able to predict the reward value based on the current state and pick the action that will give the highest reward.

4. Tuning Parameters

We could change the activation function of neural network or change the number of nodes of each layer to improve the performance of neural network. As for the exponential-decay function and Q-function, we could tuning parameters like gamma, lambda and epsilon max/min, etc. I mainly tune three parameters, epsilon min, number of episodes and gamma.

When I tune parameter epsilon min, other parameters are set as the same as default values. Episode is 10000 and gamma is 0.99. From table 2-1, we could find that when the value of epsilon min increases, value of mean reward decreases and time becomes longer. Thus 0.05 is a good value for parameter epsilon min.

Table 2-1 Tuning MIN_EPSILON

MIN_EPSILON	0.05	0.1	0.3	0.5
MEAN REWARD	6.25	6.04	4.78	3.65
TIME/s	502.95	502.95	581.31	618.22

Table 2-2 is the process of tuning parameter episode. We could find that reward increase along with episode. But when episode is more than 20000, the increasing rate slows down. So episode is set as 20000.

Table 2-2 Tuning episode

EPISODE	10000	15000	20000	25000
MEAN REWARD	6.25	6.37	6.71	6.78
TIME/s	502.95	776.12	1024.14	1275.99

In addition, I also tune parameter gamma, which is shown in table 2-3.

Table 2-3 Tuning gamma

GAMMA	0.99	0.8	0.7	0.6	0.5
MEAN REWARD	6.25	6.38	6.47	6.48	6.47
TIME/s	502.95	548.20	615.51	557.87	537.6

From the table above, we could know that 0.5 is a good value for gamma.

5. Best Performance of Model

After tuning parameters, the best performance of model is shown in figure 2-5.

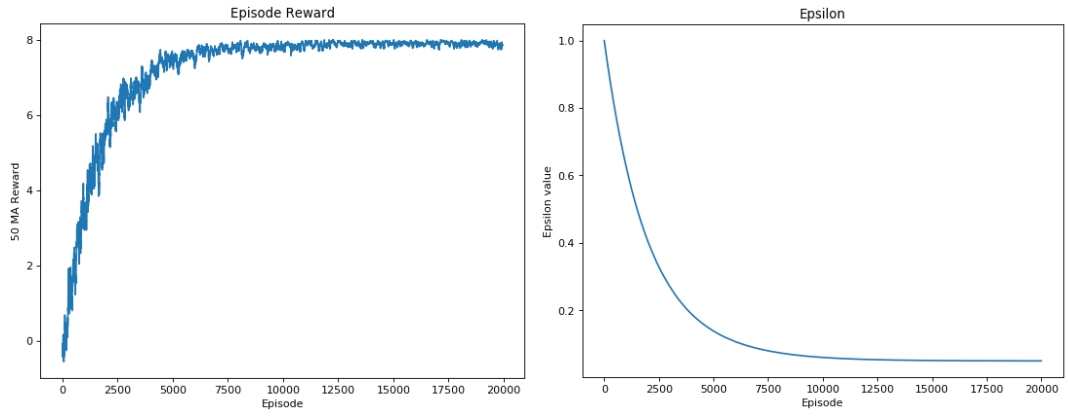

Figure 2-5 Best performance of the model (a) mean reward and episode (b) epsilon

The mean reward is 7.18 and the running time is 1070.00 second on my local machine.

2.2 Writing tasks

1. Explain what happens in reinforcement learning if the agent always choose the action that maximizes the Q-value. Suggest two ways to force the agent to explore.

If the agent always choose the action that maximizes the Q-value, it will stuck in non-optimal policies because the samples maybe insufficient and some states are not included. In other word, the model needs to explore new actions in order to gain a better reward.
To force the agent to explore, model can pick random actions. This is the strategy used in our model. We also can use model-based approaches that compute a choice of action based on its expected reward and the model's uncertainty about that reward. For example, we could use neural network to implement this by using error function as model's uncertainty.

2. Calculate Q-value for the given states and provide all the calculation steps.

Following is Q-Table.

Table 2-4: Q-Table (for the states in Figure 1 in project4_task.pdf)

	ACTIONS			
STATE	UP	DOWN	LEFT	RIGHT
0	3.90	3.94	3.90	3.94
1	2.94	2.97	2.90	2.97
2	1.94	1.99	1.94	1.99
3	0.97	1	0.97	0.99
4	0	0	0	0

The calculation steps are as following.
We assume the axis for S_4 is (3, 3) and S_0 is (1, 1).
First, because we assume that the agent can move after reaching the goal, so the Q-value for every action in S_4 is 0. So, we have

Q (S_4, down) = Q (S_4, left) = Q (S_4, up) = Q (S_4, right) =0.

Then here are the calculations for S_3.

Q (S_3, down) = $1 + \gamma Q((3,3),\ \text{down}) = 1 + 0.99 \times 0 = 1$

Because Q ((2, 2), right) = $1 + \gamma Q((2,3),\ \text{down}) = 1 + 0.99 \times 1 = 1.99$, we have

Q (S_3, left) = $-1 + \gamma Q((2,2),\ \text{right}) = -1 + 0.99 \times 1.99 = 0.97$

Because Q ((1, 3), down) = $1 + \gamma Q((2,3),\ \text{down}) = 1 + 0.99 \times 1 = 1.99$, we have

Q (S_3, up) = $-1 + \gamma Q((1,3),\ \text{down}) = -1 + 0.99 \times 1.99 = 0.97$

Q (S_3, right) = $0 + \gamma Q((2,3),\ \text{down}) = 0 + 0.99 \times 1 = 0.99$

Next are calculation steps for S_2.

Q (S_2, down) = $1 + \gamma Q((3,2),\ \text{right}) = 1 + 0.99 \times 1 = 1.99$

Because Q ((2, 1), right) = $1 + \gamma Q((2,2),\ \text{right}) = 1 + 0.99 \times 1.99 = 2.97$, we have

Q (S_2, left) = $-1 + \gamma Q((2,1),\ \text{right}) = -1 + 0.99 \times 2.97 = 1.94$

Because Q ((1, 2), down) = $1 + \gamma Q((2,2), \text{ right}) = 1 + 0.99 \times 1.99 = 2.97$, we have

Q (S_2, up) = $-1 + \gamma Q((1,2), \text{ down}) = -1 + 0.99 \times 2.97 = 1.94$

Q (S_2, right) = $Q((2,2), \text{right}) = 1.99$

Then, next are calculations for S_1.

Because Q ((1, 1), right) = $1 + \gamma Q((1,2), \text{ down}) = 1 + 0.99 \times 2.97 = 3.94$, we have

Q (S_1, left) = $-1 + \gamma Q((1,1), \text{ right}) = -1 + 0.99 \times 3.94 = 2.90$

Q (S_1, up) = $0 + \gamma Q((1,2), \text{ down}) = 0 + 0.99 \times 2.97 = 2.94$

Q (S_1, right) = $1 + \gamma Q((1,3), \text{ down}) = 1 + 0.99 \times 1.99 = 2.97$

Q (S_1, down) = $Q((1,2), \text{down}) = 2.97$

Finally, these are calculation steps for S_0.

Q (S_0, down) = $1 + \gamma Q((2,1), \text{ right}) = 1 + 0.99 \times 2.97 = 3.94$

Q (S_0, left) = $0 + \gamma Q((1,1), \text{ right}) = 0 + 0.99 \times 3.94 = 3.90$

Q (S_0, up) = $0 + \gamma Q((1,1), \text{ right}) = 0 + 0.99 \times 3.94 = 3.90$

Q (S_0, right) = $Q((1,1), \text{ right}) = 3.94$

3 Conclusion

In this project, I implement a model that makes Tom cat find the shortest path to Jerry after giving the initial position of them. We solve this problem by applying deep reinforcement learning algorithm – DQN. In addition, I also learn the role that neural network plays in this machine learning model, and I learn how to implement exponential-decay function and Q-function. What's most important is that this task makes me have a general idea of how a reinforcement learning model works, which could be applied into my study in the future.