CSE 474/574: Introduction to Machine Learning

University at Buffalo

Keming Kuang (kemingku)

personal #: 50161776

December 5, 2018

Contents

1 Description

1.1 Objective

In this project, we will train the agent to navigate in the 5*5 grid-world environment. The goal is to teach the agent learn the shortest path the the target.

1.2 Task

With given agent "Tom" and target "Jerry", our task is to apply deep reinforcement learning algorithm. During the project, we will answer the following question.

Q1: Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value. Suggest two ways to force the agent to explore.

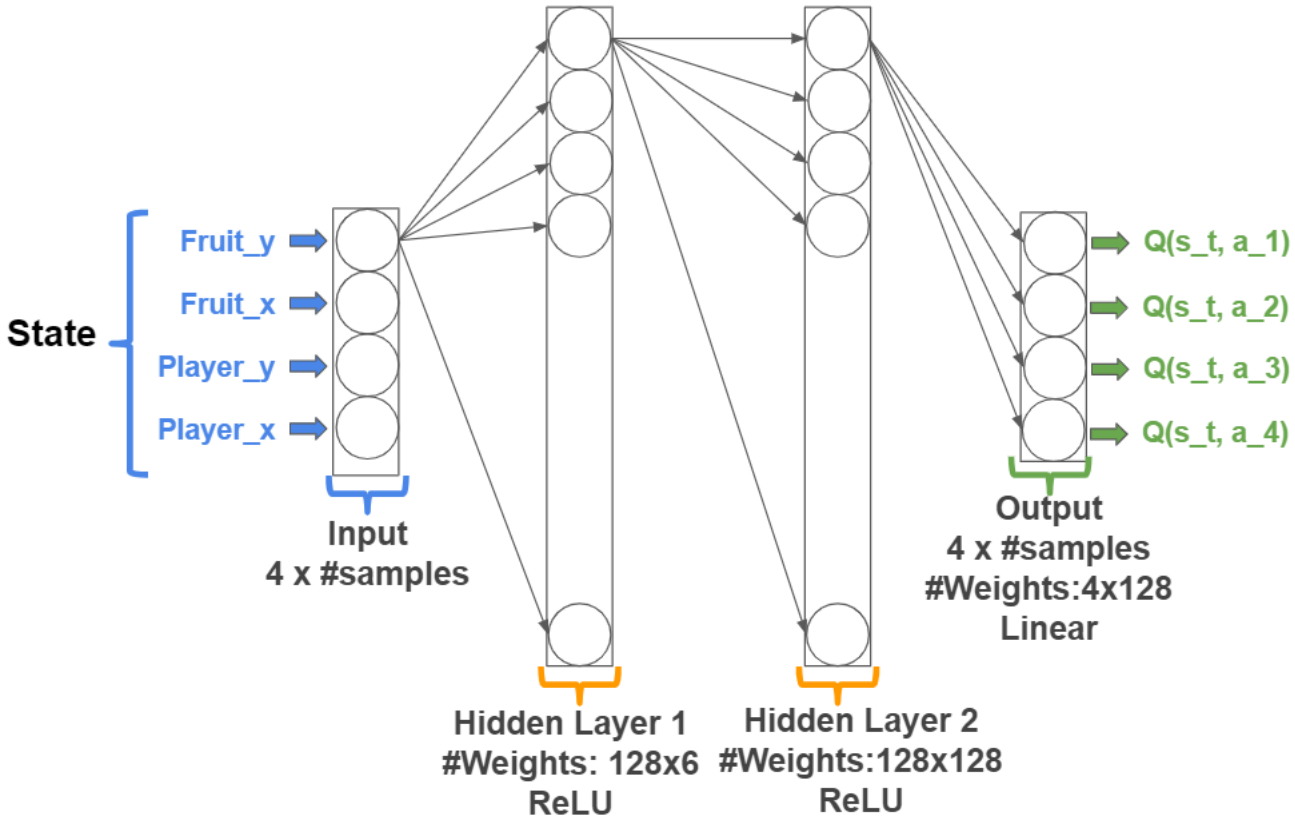
Q2: Calculate Q-value for the given states and provide all the calculation steps.

1.3 Plan of Work

While the major codes of deep reinforcement learning algorithm is given, we will need to complete building:
3 layer neural network
Exponential-decay formula for epsilon
Q-function.

2 Coding Tasks

2.1 3 Layer Neural Network



Since we are training the agent the shortest path to catch the target, we use the neural network to map state-action pairs to rewards. In reinforcement leaning, 3 layers neural network are used to approximate the function relating inputs to outputs.

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

2.2 Exponential-decay Formula for Epsilon

```
self.epsilon = self.min_epsilon+(self.max_epsilon-self.min_epsilon)*math.exp(-self.lamb*self.steps)
```

Before Running Q-function, we need to balance between exploration and exploitation in our reinforcement learning algorithm. This usually depends on how many iteration running, how many steps taken each episode, and what we want ti to decay to and when.
In our project, we use the "Exponential-decay formula for epsilon":

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) * e^{-\lambda|S|}, \tag{1}$$

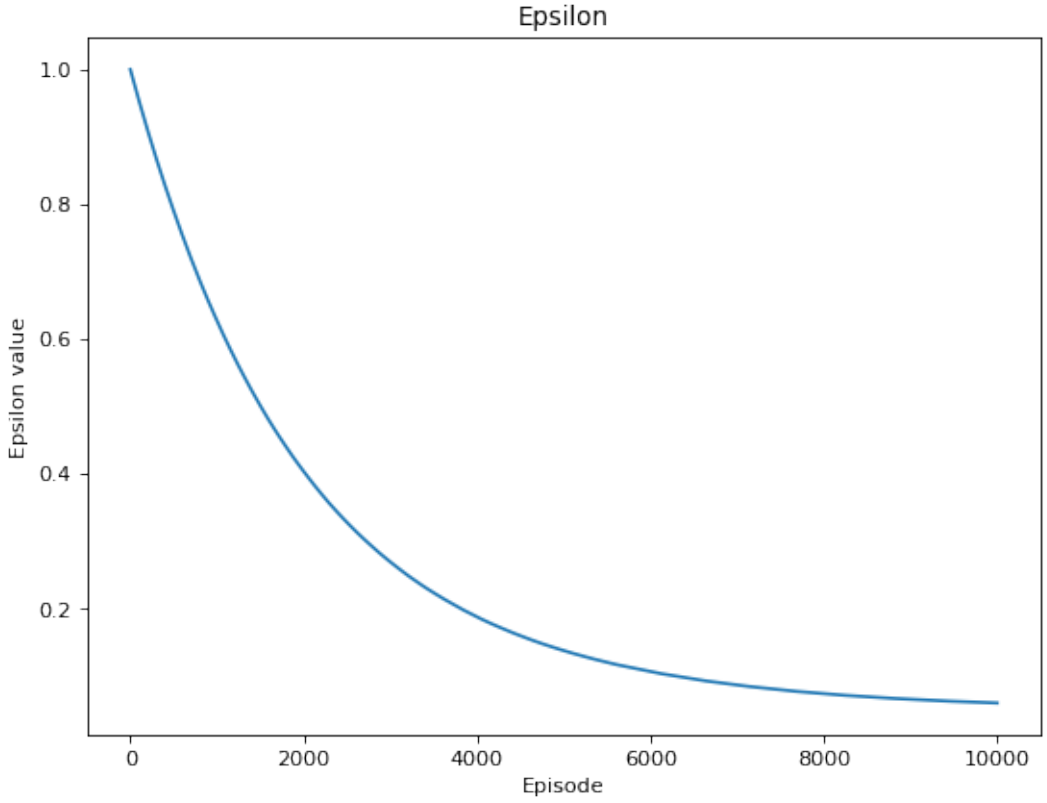
where $\epsilon_{min}, \epsilon_{max} \in [0, 1]$
λ - hyperparameter for epsilon
$|S|$ - total number of steps

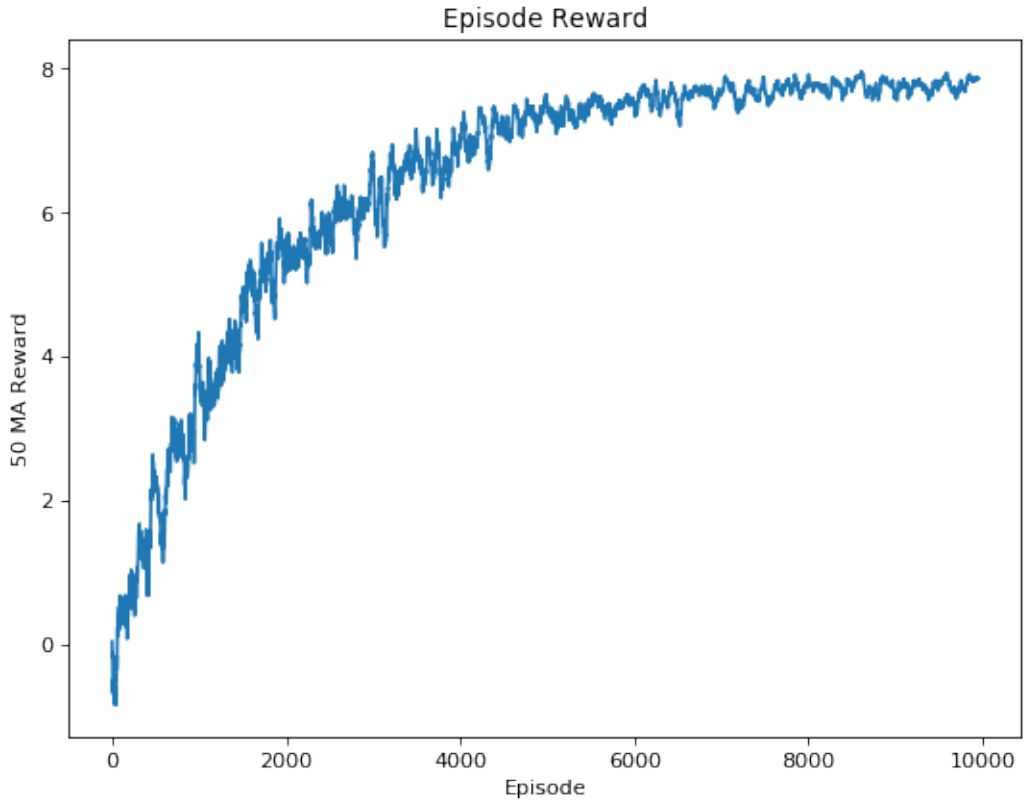
2.3 Q-function

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

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

Q learning calculates the long term value of an action through observations. By setting Q-value to o for every state-action pair, we calculate every Q(s,a) base on our learning rate and discount rate. As Q takes all past Q-values, we have the maximized Q-value for every new state. Thus we can get the optimized action.





3 Writing Tasks

3.1 Q1

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

Answer:

Before we start the Q learning, we should always balance the exploration and exploitation our algorithm. In our question assumption, the agent always has the maxQ-values action.

Solution 1: Increase the epsilon value. A possible explanation is the game set the epsilon too low that the agent have low interest in exploration.

Solution 2: Creating obstacle in the environment. While no obstacle involved, the agent will aways have 2 positive initial rewards for 2 direction and 2 negative initial rewards for 2 other direction except when they are next to the boundary. And there will be a fair chance the agent can always get the maxQ-value direction with 50 percent possibility except the edge cases. By introducing obstacle in the environment, it decreases the possibility that the agent chooses the "right direction" and creates more situation that the agent have high chance to explore and get punishment.

3.2 Q2

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

Answer:

	Action			
State	UP	DOWN	LEFT	RIGHT
0	3.90099501	3.940399	3.90099501	3.940399
1	2.940399	2.9701	2.90099501	2.9701
2	1.940399	1.99	1.940399	1.99
3	0.9701	1	0.9701	0.99
4	0	0	0	0

Calculation Steps:

Assume the 3*3 grid-world environment as

S11	S12	S13
S21	S22	S23
S31	S32	S33

then we have

State 0 = S11 State 1 = S12 State 2 = S22 State 3 = S23 State 4 = S33

Q(S33, a) = maxQ(S33,a) = 0

Q(S23, UP) = -1 + 0.99(maxQ(S13,a)) = 0.9701
Q(S23, DOWN) = 1 + 0.99(maxQ(S33,a)) = 1
Q(S23, LEFT) = -1 + 0.99(maxQ(S22,a)) = 0.9071
Q(S23, RIGHT) = 0 + 0.99(maxQ(S23,a)) = 0.99

Q(S22, UP) = -1 + 0.99(maxQ(S12,a)) = 1.940399
Q(S22, DOWN) = 1 + 0.99(maxQ(S32,a)) = 1.99
Q(S22, LEFT) = -1 + 0.99(maxQ(S21,a)) = 1.940399
Q(S22, RIGHT) = 1 + 0.99(maxQ(S23,a)) = 1.99

Q(S12, UP) = 0 + 0.99(maxQ(S12,a)) = 2.940399
Q(S12, DOWN) = 1 + 0.99(maxQ(S22,a)) = 2.9701
Q(S12, LEFT) = -1 + 0.99(maxQ(S11,a)) = 2.90099501
Q(S12, RIGHT) = 1 + 0.99(maxQ(S13,a)) = 2.9701

Q(S11, UP) = 0 + 0.99(maxQ(S11,a)) = 3.90099501
Q(S11, DOWN) = 1 + 0.99(maxQ(S21,a)) = 3.94039
Q(S11, LEFT) = 0 + 0.99(maxQ(S11,a)) = 3.90099501
Q(S11, RIGHT) = 1 + 0.99(maxQ(S12,a)) = 3.94039

4 Reference Website

https://skymind.ai/wiki/neural-network
https://skymind.ai/wiki/deep-reinforcement-learning
https://stackoverflow.com/questions/42589683/epsilon-decay-in-q-learning
https://www.oreilly.com/ideas/reinforcement-learning-explained?fbclid=IwAR1WzGcWjZb1NtZ1eYP5Ix6_alMduVDGTNIr
https://drive.google.com/file/d/1opPSz5AZ_kVa1uWOdOiveNiBFiEOHjkG/view