Project4:
The Learning Task

Manish Reddy Challamala
Department of Computer Science
University at buffalo
UBIT Name: manishre
Person Number : 50289714
manishre@buffalo.edu

December 6, 2018

Abstract

To train the model to find the shortest path between tom and jerry by using reinforcement learning and Deep Q-Network.

1 Introduction

The goal of this project is to implement Reinforcement learning and Deep Q-network to find the shortest path between the tom and jerry.
The code is implemented in by using two learning algorithms:

1. Neural Network [DNN].
2. Reinforcement Learning [RL].

2 Theory

2.1 Reinforcement Learning:

Reinforcement learning is a methods, where an agent learn how to behave in a environment by performing actions and update the rewards for the actions made in the environment.Here Environment is the object and agent is an Reinforcement learning algorithm[RL]. The simple form of reinforcement learning algorithm is given below:

1. Firstly the environment sends the state to the agent and the agent takes the action to that state. The environment again sends the new state and reward for the previous task to the agent where the agent will update its knowledge of reward.
   Here
   Action (A) : possible moves that a agent can take.
   State(S) : Current Situation returned by environment.

Reward(R) : An immediate return send back from the environment to evaluate the last action.
Policy (π) : The strategy that the agent employs to determine ext action based on the current state.
value (V) : expected long term return with discount, as opposed to short term reward (R).
Reinforcement learning need to choose between exploration and exploitation.
Exploring is a process that takes random actions to calculate or obtain more training data.
Exploitation is a process that takes actions from current best version of learned policy.

2.2 Neural Network

The neural network model is a classification model which tries to predict output y for a given input x

1. The neural network model contain two phase:
   1 Learning Phase
   2. Prediction Phase

2. In learning phase, the neural network takes inputs and corresponding outputs.

3. The neural network process the data and calculates the optimal weights to gain the training experience.

4. In prediction process the neural network is presented with the unseen data where it predicts a output by using its past training experience on that given data.

5. A neural network model can be build by using different layers:
   1. Dense Layer : A linear Operation in which every input is connected to every output by weights
   2. Convolution Layer: A linear operation in which input are connected to output by using subsets of weights of dense layer.

3 Coding Task:

Neural Network:

```
 ### START CODE HERE ### ( 3 lines of code)
## Hiddden Layer 1
model.add(Dense(128,input_dim =self.state_dim,activation ='relu'))
## Hiddden Layer 2
model.add(Dense(128, activation ='relu'))
## Output layer
model.add(Dense(self.action_dim, activation ='linear'))
```

1. The Deep Q-Network acts as a brain for our agent.

2. In this project I have implemented both Dense neural network.

3. For Dense neural network ,Creating a model with 3 layers, 1. input layer 2. 2-hidden layer 3. output layer

4. No of nodes for each layer is given below:
   1.No of nodes in input layer = Co-ordinates of Tom and Jerry
   2. No of nodes in hidden layer 128
   3.No of nodes in output layer 4

5. Activation functions used in hidden layer is relu [rectified linear unit] because it introduces the non linearity in the network and linear function is used on the output layer to predict the target class.

6. The input takes 4 values [position of player and fruit] which are then combined with respective to different actions [weights] to produce the 4 output values.The agent always choose such action which gives highest Q-value.

7. In neural network, by tuning the hyper-parameters like no of hidden nodes a, activation functions,optimizer and learning rate we can predict the better action values while exploring the environment which increases the accuracy.

Exponential Decay formula for epsilon:

```
self.epsilon = self.min_epsilon +((self.max_epsilon - self.min_epsilon) *

math.exp(-(self.lamb*abs(self.steps))))
```

1. Epsilon value will determine at what probability a agent will take the random action. This epsilon value will take care off our exploitation and exploration trade-off.

2. we can tune the maximum and minimum epsilon values to get better trade-off between exploration and exploitation and can also change the function of exponential-epsilon decay function to linear or quadratic decay function.

Implement Q-function:

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

1. The above snippet is known as iterative Q-function.

2. In this snippet we are checking weather the next state reaches the goal. If it reaches we update the action with maximum reward value else we are updating the Q-values by discounting the reward with gamma.

3.1 Writing Task 1:

If agent always chooses the action that maximizes the Q-value, it means that we are concentrating more on the Exploitation rather than exploring which leads the agent to follow the same path even if have the some better optimal way to follow. So to overcome this problem we are going for the exploration and exploitation trade off.

ϵ -greedy

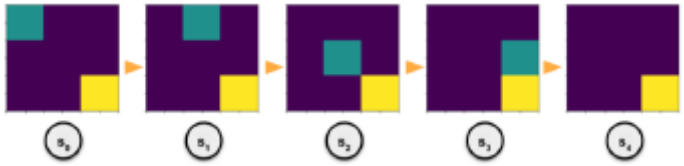
The agent believes to select the optimal path all the time but occasionally acts randomly.The ϵ greedy determines the probability of taking random action, Due to its random action the model concentrates on both exploitation and exploration.

Bayesian Neural Network

Unlike the neural network, The Bayesian neural network acts probabilistically.Where instead of having the single set of fixed weights. The BNN takes the probability distribution over all weights.So by taking the probability distributions over the weights will allow us to obtain the distribution over the actions in reinforcement learning.The variance of of distribution will tell us about the uncertainty of each action.

4 Writing Task 2:

Given:



Condition for sequence of actions: Right $\longrightarrow$ *Down* $\longrightarrow$ *Right* $\longrightarrow$ *Down*

S_0	S_1	S_5
S_6	S_2	S_3
S_7	S_8	S_4

As we are calculating from the final state S_4 will be updating the Q-table values with Zero.

$Q(S_4, U) = 0$
$Q(S_4, D) = 0$
$Q(S_4, L) = 0$
$Q(S_4, R) = 0$

calculating the Q-value of state S_3

$Q(S_3, D) = 1 + 0.99*(\text{MaxQ}(S_4)) = 1 + 0.99*(0) = 1$
$Q(S_3, U) = \text{-1} + 0.99*(\text{MaxQ}(S_5)) = \text{-1} + 0.99*(1.99) = 0.97$
$Q(S_3, L) = \text{-1} + 0.99*(\text{MaxQ}(S_2)) = \text{-1} + 0.99*(1.99) = \text{-1} + 1.97 = 0.97$
$Q(S_3, R) = 0 + 0.99*(\text{MaxQ}(S_3)) = 0 + 0.99*(1) = 0.99$

calculating the Q-value of state S_2

$Q(S_2, R) = 1 + 0.99*(\text{MaxQ}(S_3)) = 1 + 0.99*(1) = 1.99$
$Q(S_2, D) = 1 + 0.99*(\text{MaxQ}(S_8)) = 1 + 0.99*(1) = 1.99$
$Q(S_2, U) = -1 + 0.99*(\text{MaxQ}(S_1)) = -1 + 0.99*(2.97) = 1.94$
$Q(S_2, L) = -1 + 0.99*(\text{MaxQ}(S_6)) = -1 + 0.99*(2.97) = -1 + 2.94 = 1.94$

calculating the Q-value of state S_1

$Q(S_1, R) = 1 + 0.99*(\text{MaxQ}(S_5)) = 1 + 0.99*(1.99) = 2.97$
$Q(S_1, D) = 1 + 0.99*(\text{MaxQ}(S_2)) = 1 + 0.99*(1.99) = 2.97$
$Q(S_1, U) = 0 + 0.99*(\text{MaxQ}(S_1)) = 0.99*(2.97) = 2.94$
$Q(S_1, L) = -1 + 0.99*(\text{MaxQ}(S_0)) = -1 + 0.99*(3.94) = 2.9$

calculating the Q-value of state S_0

$Q(S_0, R) = 1 + 0.99*(\text{MaxQ}(S_1)) = 1 + 0.99*(2.97) = 3.94$
$Q(S_0, D) = 1 + 0.99*(\text{MaxQ}(S_6)) = 1 + 0.99*(2.97) = 3.94$
$Q(S_0, U) = 0 + 0.99*(\text{MaxQ}(S_0)) = 0.99*(3.94) = 3.9$
$Q(S_0, L) = 0 + 0.99*(\text{MaxQ}(S_0)) = 0.99*(3.94) = 3.9$

calculating the Q-value of state S_5

$Q(S_5, D) = 1 + 0.99*(\text{MaxQ}(S_3)) = 1 + 0.99*(1) = 1.99$
$Q(S_5, R) = 0 + 0.99*(\text{MaxQ}(S_5)) = 0.99*(1.99) = 1.97$
$Q(S_5, U) = 0 + 0.99*(\text{MaxQ}(S_5)) = 0.99*(1.99) = 1.97$
$Q(S_5, L) = -1 + 0.99*(\text{MaxQ}(S_1)) = -1 + 0.99*(2.97) = 1.94$

calculating the Q-value of state S_6

$Q(S_6, R) = 1 + 0.99*(\text{MaxQ}(S_2)) = 1+ 0.99*(1.99) = 2.97$
$Q(S_6, D) = 1 + 0.99*(\text{MaxQ}(S_7)) = 1 + 0.99*(1.99) = 2.97$
$Q(S_6, U) = -1 + 0.99*(\text{MaxQ}(S_0)) = -1 + 0.99*(3.94) = 2.9$
$Q(S_6, L) = 0 + 0.99*(\text{MaxQ}(S_6)) = 0 + 0.99*(2.97) = 2.94$

calculating the Q-value of state S_7

$Q(S_7, R) = 1 + 0.99*(\text{MaxQ}(S_8)) = 1 + 0.99*(1) = 1.99$
$Q(S_7, D) = 0 + 0.99*(\text{MaxQ}(S_7)) = 0.99*(1.99) = 1.97$
$Q(S_7, U) = -1 + 0.99*(\text{MaxQ}(S_6)) = -1 + 0.99*(2.97) = 1.94$
$Q(S_7, L) = 0 + 0.99*(\text{MaxQ}(S_7)) = 0.99*(1.99) = 1.97$

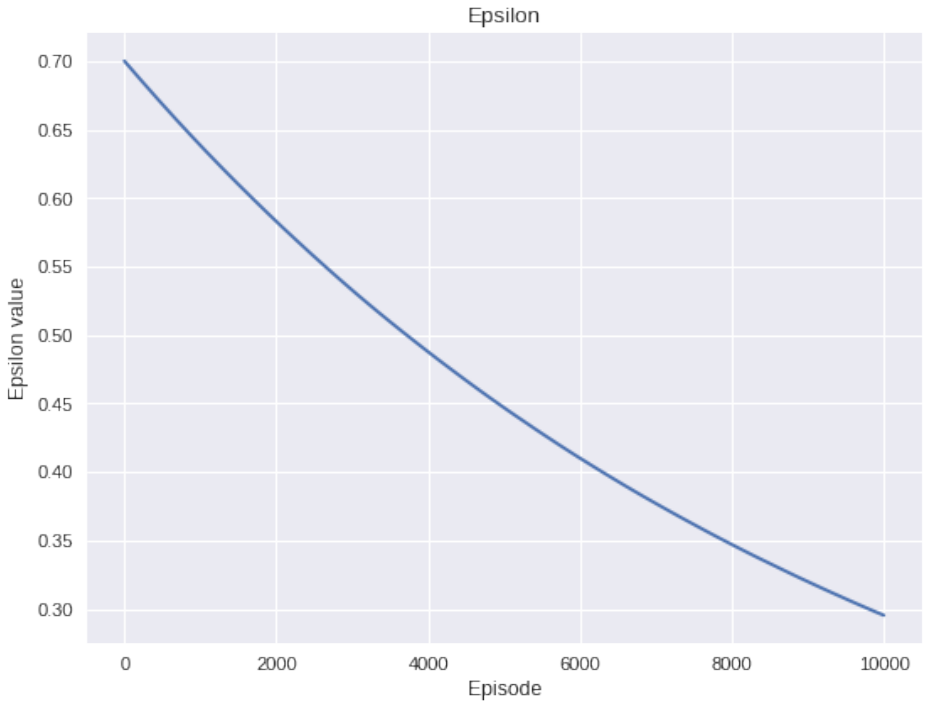
calculating the Q-value of state S_8

$Q(S_8, R) = 1 + 0.99*(\text{MaxQ}(S_4)) = 1 + 0.99*(0) = 1$
$Q(S_8, D) = 0 + 0.99*(\text{MaxQ}(S_8)) = 0.99*(1) = 0.99$
$Q(S_8, U) = -1 + 0.99*(\text{MaxQ}(S_2)) = -1 + 0.99*(1.99) = 0.97$
$Q(S_8, L) = -1 + 0.99*(\text{MaxQ}(S_7)) = -1 + 0.99*(1.99) = 0.97$

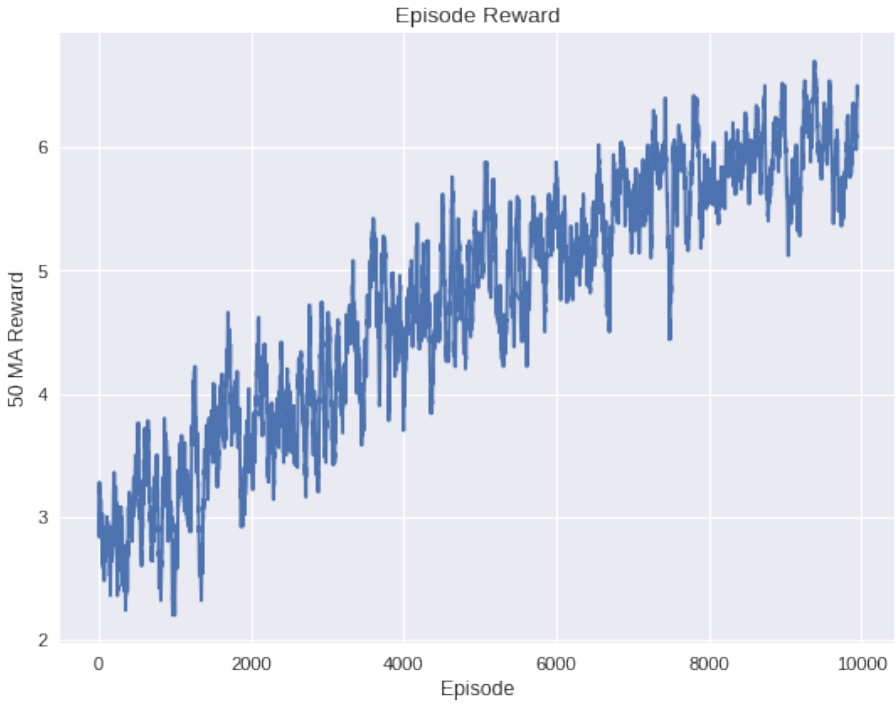
	ACTIONS			
STATE	UP	DOWN	LEFT	RIGHT
S_0	3.9	3.94	3.9	3.94
S_1	2.94	2.97	2.9	2.97
S_2	1.94	1.99	1.94	1.99
S_3	0.97	1	0.97	0.99
S_4	0	0	0	0

5 Hyperparameter tunning:

	Episodes	Gamma	$\max_{e}psilon$	Lamda	Reward Rolling Mean
1.	10000	0.99	1	0.00001	6.29
2.	10000	0.99	0.9	0.0001	7.1191
3.	10000	0.99	0.85	0.0005	7.656
4.	15000	0.80	0.85	0.0005	7.256
5.	20000	0.99	0.70	0.0005	6.458
6.	10000	0.92	1	0.0005	6.352
7.	20000	0.99	1	0.0005	6.874



$fig[1]$ *Epsilon decay function for max episilon* $= 0.7$



fig[2] *Episode reward vs Episode for max episilon* = 0.7

6 References:

1. https://ublearns.buffalo.edu/bbcswebdav/pid-4777083-dt -content-rid-211944651/courses/218924904COMB/15.2- LearningTask
2. https://medium.com/emergent-future/simple-reinforcement- learning-with-tensorflow-part-7-action-selection-strategies-for-exploration-d3a97b7cceaf
3. https://datascience.stackexchange.com/questions/25714/what-is-the-difference-between-expected-return-and-expected-reward-in-the-co
4. https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419
5. https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287