Project 4 Report **Koustubh Vijay Kulkarni**
CSE 574
Prof. Sargur N. Srihari UBID: Kkulkarn
TA: Mihir Chauhan, Tiehang Duan, Alina Vereshchaka and others Person Number: 50288207

Objective

The objective of this project is to implement a model which combines reinforcement learning and deep learning. Task is to teach the agent to navigate in the grid-world environment. We have modeled Tom and Jerry cartoon, where Tom, a cat, is chasing Jerry, a mouse. In our modeled game the task for Tom (an agent) is to find the shortest path to Jerry (a goal), given that the initial positions of Tom and Jerry are deterministic.

Task

Our task is to implement a 3-layer neural network using Keras library, implement exponential-decay formula for epsilon and implement Q-Function.

What is Reinforcement Learning?

Reinforcement learning is an important type of Machine Learning where an agent learn how to behave in a environment by performing actions and seeing the results. Unlike the supervised learning, we do not have any training data available with us. The idea behind Reinforcement Learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions.

DQN

We achieve reinforcement learning with the help of Q learning. In Q-learning we compute the Q-table for each state against each action and based on that agent takes action to maximize the expected reward. The main idea in this case is to create a deep neural network that will provide us the different Q-values for each action , given a state. As, in complex environment in might be computationally heavy task to create a big Q-table. With the help of DNN this need of creation of Q-table is not required.

0.1 Neural network

As an input to the model we pass a tuple of 4 values: x, y coordinates of Tom and Jerry. These pass through its network, and output a vector of Q-values for each action possible in the given state. We need to take the maximum Q-value of this vector to find our best action. Our memory buffer stores experience tuples while interacting with the environment, and then we sample a small batch of tuple to feed our neural network. When our network predicts the Q values for the state and action for records in mini-batch we apply Q function to it and update target and feed those evalutes to DNN to train on. If we train our network in sequential order, we risk our agent being influenced by the effect of this correlation. Sampling breaks this correlation.

NN implementation :

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

We implement a Neural network with 128 input nodes starting nodes , The activation function that we have used for the first 2 layers is 'relu. The last node is for 4 nodes each representing an action and the activation function that we give it is 'linear' as we want real values for Q-value for each action.

0.2 Implementation of exponential-decay formula for epsilon

The goal of our agent is to maximize the expected cumulative reward. For this to take place our agent should learn the value of each of the actions in each of those states. Thus, In order for an agent to learn how to deal optimally with all possible states in an environment, it must be exposed to as many of those states as possible. Hence we can say that **Exploration** is need to learn right experiences to learn a good policy. **Exploitation** is acting on what we have already learned.
It is very important to develop techniques for meaningfully balancing the exploration and exploitation tradeoff.

Epsilon decay function - Our agent will randomly select its action at first by a certain percentage , The agent at first try all kinds of things before it starts to see the patterns. After enough exploring the agent will agent will predict the reward value based on the current state and pick the action that will give the highest reward.
The value of epsilon decreases as number of steps increases.Below is the epsilon formula that we have implemented in python , the value of epsilon will decay as the number of states increases. Forcing our agent to chose the action that gives the the highest reward.

```
self.epsilon =
↪  self.min_epsilon+(np.subtract(self.max_epsilon,self.min_epsilon) *
↪  np.exp(-self.lamb*self.steps))
```

0.3 Implementing Q function

Our last task was to implement the Q function , which basically calculates the maximum expected future reward, for each action at each state.
we implement the Q function with the below formula

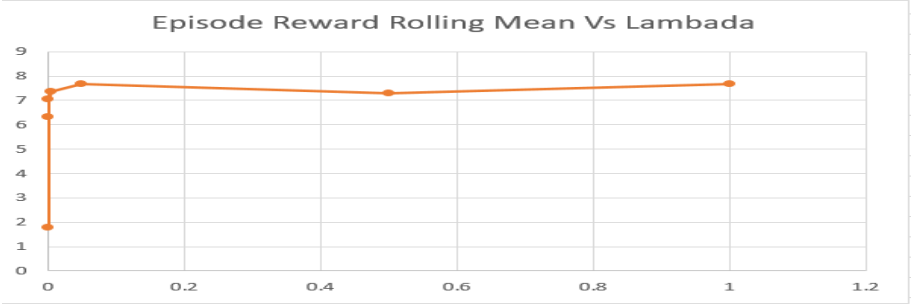
```
if states_next is None :
      t[act]= rew
    else :
      t[act] = rew + self.gamma * np.argmax(q_vals_next[i])
```

where , if the next stage is the final stage then the reward of the action in the current stage is equal to the immediate reward.
Else , the return for the action is calculated as the immediate reward added with product of gamma and the maximum Q value of the next stage.

Hyper-parameter Tuning

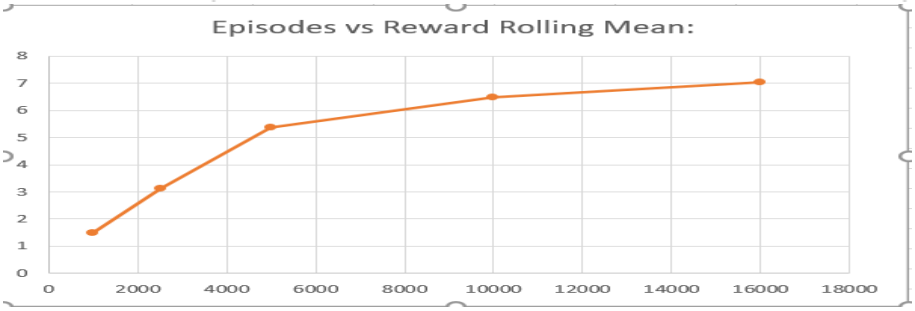
- **Changing the value of lambada**:- Lambada indicates speed of decay for epsilon. I have changed value of lambada ranging from 0.000005 to 1 and here are my observations:-

Gamma	Episode Reward Rolling Mean
0.000005	1.767370676
0.00005	6.319355168
0.0005	7.043566983
0.005	7.368533823
0.05	7.676563616
0.5	7.297928783
1	7.659626569



- **Changing the value of No. Of Episodes**:- I have changed value of no of episodes a ranging from 1K to 16K and here are my observations:-

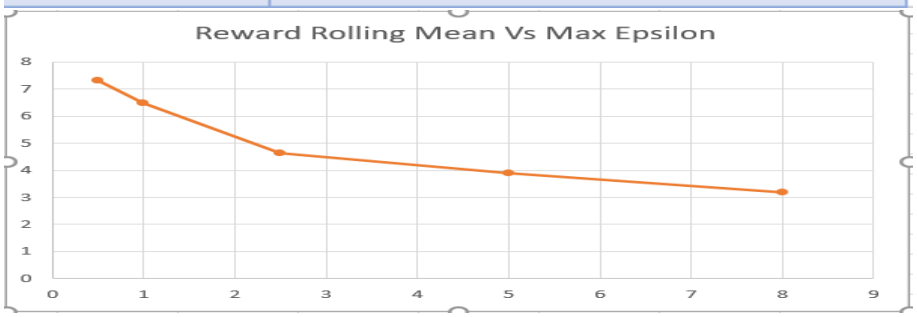
Episodes	Episode Reward Rolling Mean:
16000	7.035132761
10000	6.475155596
5000	5.383461779
2500	3.112159911
1000	1.493133583



It is clear that as episode increases, reward rolling mean is also increasing as agent gets lot of time to explore the environment and exploit best decisions from it.

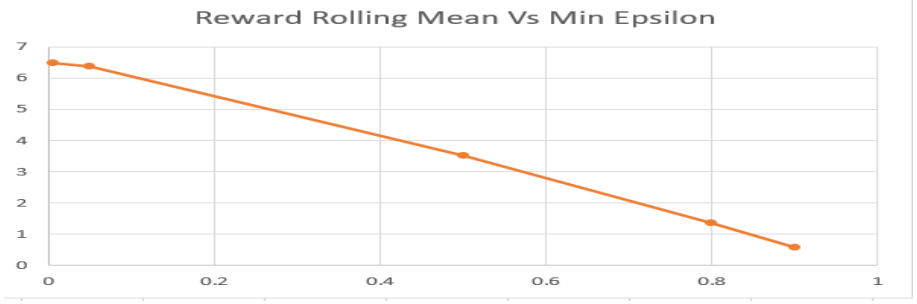
- **Changing the value of Max-Epsilon**:- I have changed value of Max-Epsilon and here are my observations:-

Epsilon max	Episode Reward Rolling Mean:
0.5	7.315273952
1	6.475155596
2.5	4.62
5	3.892561983
8	3.184981124



- **Changing the value of Min-Epsilon**:- I have changed value of Min-Epsilon and here are my observations:-

Epsilon min	Episode Reward Rolling Mean:
0.9	0.570962147
0.8	1.35
0.5	3.512702785
0.05	6.37781859
0.005	6.475155596



Writing Task

0.3.1 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.

In simple terms, in reinforcement learning agent has to achieve a target in an Environment. Agent always tries to maximize it's expected return. Now, if agent took some actions in some state and as a result he got some reward and changed it's state. Now after certain number of steps, either it will reach to the target or it will terminate as he is out of number of steps for

that episode. Now, next time if he chooses the action which maximizes the Q-value, he will take same path again and again because he has not explored other states.

If the agent always chooses the action that maximizes the Q-value , then the agent will not be able to explore enough to get the most desirable action from each state. Which will result in the formation of a **non - optimal policy**. As always choosing the action that maximizes the q- value will always force the agent to exploit.

This causes the agent to not learn all of his environment properly. There might be bigger rewards which the agent has failed to discover as he always chooses the greedy approach. There might be another optimal policy which is better than the policy an agent has taken before. An agent needs the right experiences to learn a good policy.

Exploring early on is important for the agent as it explores the environment, the Q-table will give a better and better approximation by iteratively updating the Q values , which will result in choosing an optimal policy. There has to be balance between exploration and exploitation. Ideally, such an approach should encourage exploring an environment until the point that it has learned enough about it to make informed decisions about optimal actions.

Consider the example of Tom and Jerry in our Project:- If we consider the example of Tom and Jerry in our project, if Tom finds a path by always taking action maximizing Q-value, he might find a path to catch Jerry but that path might not be the shortest one. If Tom explores the Grid environment completely then he can identify optimal policy and he will be able to find the shortest path to reach Jerry. In some cases it might happen that, Tom will repeat same actions again and again in order to maximize Q value, and this is clearly not a desirable behavior.

Therefore, if the agent always chooses the action that maximizes the Q-value it will lead to a sub-optimal solution.
Ways that force the agent to explore :

- **Epsilon function** - Epsilon is a decay function. Our agent will randomly select its action at first by a certain percentage. In the beginning, value of Epsilon must be at its highest value. This forces the agent to do a lot of exploration, by randomly choosing actions. As the number of stages increases the epsilon value is decremented which forces the agent to exploit more , hence creating a balance between exploration and exploitation.
- **Boltzmann Approach**- Boltzmann exploration instead of always taking the optimal action, or taking a random action,we choose an action with weighted probabilities.It uses softmax over the networks estimates of value for each action.
- **Random Approach**- This is fairly simple approach in which you force agent to always take random actions. This is exactly opposite of taking the action which maximizes the Q-value.

0.3.2 Calculate Q-value for the given states.

At initial stage, Q-table will have 0 values for all states in it.
For first iteration
For State S4, as it is terminating condition, no reward (nor penalties) will be given for addi-

tional actions. Hence Q-table for State S4 will be:-

UP	DOWN	LEFT	RIGHT
0	0	0	0

For **State S3**, we will calculate Q-value for each action:-

- Q(S3,DOWN) :- When we take action DOWN on S3 we reach to Goal state, hence as per the formula of Q function, we assign complete reward of that transition. Hence Q-value will be **1**.
- Q(S3,RIGHT) :- As it tries to move into an edge. Q value will be **0**.
- Q(S3,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state. As, Q-value of that state is 0. We have Q(S3, LEFT)= -1 + 0.99 (0)= **-1**
- Q(S3,UP) :- Similarly,Q(S3, UP)= -1 + 0.99 (0)= **-1**

UP	DOWN	LEFT	RIGHT
-1	1	-1	0

Q-value of S3 is **1**.
For **State S2**, we will calculate Q-value for each action:-

- Q(S2,RIGHT) :- As going to RIGHT will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S3. As, Q-value of that state is 1. We have Q(S2, RIGHT)= 1 + 0.99 (1)= **1.99**
- Q(S2,DOWN) :- When we take action DOWN on S2 we reach near to Goal state, reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state which is 0. We have Q(S2, DOWN)= 1 + 0.99 (0)= **1**
- Q(S2,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state. As, Q-value of that state is 0. We have Q(S2, LEFT)= -1 + 0.99 (0)= **-1**
- Q(S2,UP) :- Similarly,Q(S2, UP)= -1 + 0.99 (0)= **-1**

UP	DOWN	LEFT	RIGHT
-1	1	-1	1.99

Q-value of S2 is **1.99**.
For **State S1**, we will calculate Q-value for each action:-

- Q(S1,DOWN) :- As going to DOWN will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S2. As, Q-value of that state is 1.99. We have Q(S1, DOWN)= 1 + 0.99 (1.99)= **2.97**
- Q(S1,RIGHT) :- When we take action RIGHT on S1 we reach near to Goal state, reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state which is 0. We have Q(S1, RIGHT)= 1 + 0.99 (0)= **1**

- Q(S1,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state. As, Q-value of that state is 0. We have Q(S1, LEFT)= -1 + 0.99 (0)= **-1**
- Q(S1,UP) :- As it tries to move into an edge. Q value will be **0**.

UP	DOWN	LEFT	RIGHT
0	2.97	-1	1

Q-value of S1 is **2.97**.
For **State S0**, we will calculate Q-value for each action:-

- Q(S0,RIGHT) :- As going to RIGHT will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S1. As, Q-value of that state is 2.97. We have Q(S1, RIGHT)= 1 + 0.99 (2.97)= **3.94**
- Q(S0,DOWN) :- When we take action DOWN on S0 we reach near to Goal state, reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state which is 0. We have Q(S0, DOWN)= 1 + 0.99 (0)= **1**
- Q(S0,LEFT) :- As it tries to move into an edge. Q value will be **0**.
- Q(S0,UP) :- As it tries to move into an edge. Q value will be **0**.

UP	DOWN	LEFT	RIGHT
0	1	0	3.94

Q-value of S0 is **3.94**.
Updated value of State S3 after multiple such iterations:-

- Q(S3,DOWN) :- Q-value will be **1**. (No change).
- Q(S3,RIGHT) :- As it tries to move into an edge. Reward will be 0 and as it will remain in same state. So, Q(S3,RIGHT)=0+0.99(Q-value(S3))=0+0.99(1)=**0.99**
- Q(S3,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state. Now that we have the Q-value of S2 i.e. 1.99. We have Q(S3, LEFT)= -1 + 0.99 (1.99)= **0.97**
- Q(S3,UP) :- Let's assume that this action will lead to state S5.We can say that Q-value of this state will be maximum if it takes action S5-DOWN as this will move agent closer to Goal. Hence, Q(S5,DOWN)=1+0.99(1)=1.99. Hence now we can calculate Q(S3,UP)= -1+0.99(1.99)=**0.97**.

Updated value of State S2 after multiple such iterations:-

- Q(S2,RIGHT) :- As going to RIGHT will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S3. As, Q-value of that state is 1. We have Q(S2, RIGHT)= 1 + 0.99 (1)= **1.99**

- Q(S2,DOWN) :- When we take action DOWN on S2 we reach near to Goal state, reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state. Now, if we assume Next state as S6. We now that as Q(S6,RIGHT) will lead to goal, hence Q-value of S6 will be 1. Therefore, we can calculate Q(S2, DOWN)= 1 + 0.99 (1)= **1.99**

- Q(S2,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state. Now assume this state be S7. We know that if S7 takes action as moving RIGHT then it will be on optimal path hence, Q(S7,RIGHT)=1+0.99(1.99)=2.97. Therefore, we can now calculate Q(S2, LEFT)= -1 + 0.99 (2.97)= **1.94**

- Q(S2,UP) :- Q(S2, UP) will be S1. We know now that, Q(S1) is 2.97. Hence, Q(S2, UP)= -1 + 0.99 (2.97)= **1.94**

Updated value of State S1 after multiple such iterations:-

- Q(S1,DOWN) :- As going to DOWN will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S2. As, Q-value of that state is 1.99. We have Q(S1, DOWN)= 1 + 0.99 (1.99)= **2.97**

- Q(S1,RIGHT) :- When we take action RIGHT on S1 we reach to S5. Reward will be 1 and we need to add discounted Q-value of Next state which is 1.99. We have Q(S1, RIGHT)= 1 + 0.99 (1.99)= **2.97**

- Q(S1,LEFT) :- As this indicates moving away from Goal state, reward will be -1 and we need to add discounted Q-value of Next state which is S0. As, Q-value of that state is 3.94. We have Q(S1, LEFT)= -1 + 0.99 (3.94)= **2.90**

- Q(S1,UP) :- As it tries to move into an edge. Reward will be 0 and as it will remain in same state. So, Q(S1,UP)=0+0.99(Q-value(S1))=0+0.99(2.97)=**2.94**

Updated value of State S0 after multiple such iterations:-

- Q(S0,RIGHT) :- As going to RIGHT will take agent close to 'Goal' reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e S1. As, Q-value of that state is 2.97. We have Q(S0, RIGHT)= 1 + 0.99 (2.97)= **3.94**

- Q(S0,DOWN) :- When we take action DOWN on S0 we reach near to Goal state, reward will be 1. Reward will be 1 and we need to add discounted Q-value of Next state i.e. S7 which is 2.97. We have Q(S0, DOWN)= 1 + 0.99 (2.97)= **3.94**

- Q(S0,LEFT) :- As it tries to move into an edge. Reward will be 0 and as it will remain in same state. So, Q(S0,LEFT)=0+0.99(Q-value(S0))=0+0.99(3.94)=**3.90**

- Q(S0,UP) :- As it tries to move into an edge. Reward will be 0 and as it will remain in same state. So, Q(S0,UP)=0+0.99(Q-value(S0))=0+0.99(3.94)=**3.90**

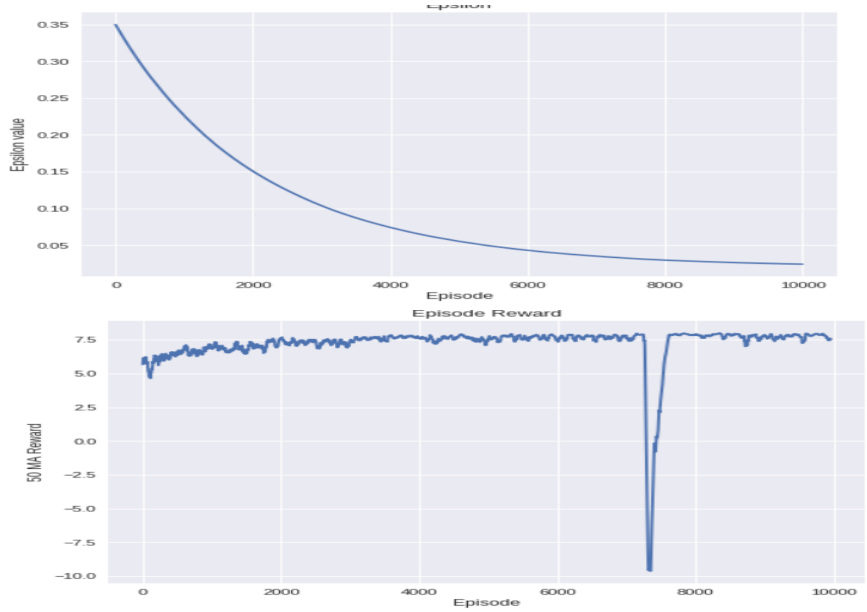
Final Q-Value Table:-

STATE	UP	DOWN	LEFT	RIGHT
S0	3.90	3.94	3.90	3.94
S1	2.94	2.97	2.90	2.97
S2	1.94	1.99	1.94	1.99
S3	0.97	1	0.97	0.99
S4	0	0	0	0

Final Evaluation

Here are my results when I tested my model with following hyper parameter settings:-MAX_EPSILON = 0.35,MIN_EPSILON = 0.02,LAMBDA = 0.00005, num_episodes = 10000

```
----------
Episode 9900
Time Elapsed: 969.19s
Epsilon 0.02439511041839331
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.170696867666565
----------
```



How quickly your agent were able to learn?
When last episode reward is 8 that indicates that Tom has caught Jerry in that episode. In my implementation, I found out that after 200 episodes my last episode reward was 8 but my Mean rolling reward was 5.77. **Why this happened?** Well a simple explanation is as agent has not explored enough environment, it is still making lot of misjudgments and hence getting heavily penalized.

```
----------
Episode 200
Time Elapsed: 24.00s
Epsilon 0.31967323203280446
Last Episode Reward: 8
Episode Reward Rolling Mean: 5.772277227722772
----------
```

In order to determine, when agent started to learn we can say that if a rolling mean reward is constant for some of the episodes then it is safe to say that agent has learned. In my case, agent was showing steady changes in rolling mean after 5000 episodes.

References

[1] Class and Recitation notes.

[2] https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-7-action-selection-strategies-for-exploration-d3a97b7cceaf

[3] https://medium.com/@dennybritz/exploration-vs-exploitation-f46af4cf62fe

[4] https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419

[5] https://stable-baselines.readthedocs.io/en/master/modules/dqn.html

[6] https://blog.openai.com/openai-baselines-dqn/