CSE 574: Introduction to Machine Learning

Project 4: Tom and Jerry in Reinforcement Learning

Anunay Rao
anunayra@buffalo.edu

1 Introduction

The project is to apply the deep reinforcement learning algorithm to make rhe agent learn to navigate in the grid-world environment. Here, the Tom is an agent and Jerry is the goal. The main task for the agent is to find the shortest path to the goal.

2 Implementation

2.1 Build a 3-layer neural network using Keras library

```
model = Sequential()

### START CODE HERE ### (≈ 3 lines of code)
model.add(Dense(128, input_shape=(self.state_dim,),activation='relu'))
model.add(Dense(self.action_dim, input_shape=(128,),activation='relu'))
model.add(Dense(self.action_dim))
### END CODE HERE ###
```

3-layer Neural network is created to train the agent as we are using deep reinforcement learning algorithm which implies applying deep learning to Reinforcement learning.

2.2 Implement exponential decay formula for epsilon

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

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

```
### START CODE HERE ### (≈ 1 line of code)
self.epsilon = self.min_epsilon + (self.max_epsilon - self.min_epsilon)*np.exp(-1*self.lamb*np.absolute(self.steps))
### END CODE HERE ###
```

Here ϵ is the exploration rate which makes the agent to take random action. Higher the value of ϵ more the percentage chance that agent will select random action. Initially it is better for the agent to take random action so that it can try all kinds of possibilities before finding any pattern. When the agent does not take the action randomly it selects the action based on the current state and the action that

will maximize the reward. During the process we want to reduce the random actions of the agent as it will tend to find the optimal policy after exploring so we need exponentially decaying epsilon. When epsilon eventually decays to 0 the policy becomes greedy i.e it do not take random actions anymore.
Higher the values of λ smaller is the exploration rate.

2.3 Implementing Q-function

$$\boldsymbol{Q_t} = \begin{cases} \boldsymbol{r_t}, & \boldsymbol{if\ episode\ terminates\ at\ step\ t+1} \\ \boldsymbol{r_t} + \gamma * max_a Q(s_{t+1}, a) \ , & \boldsymbol{otherwise} \end{cases}$$

```
### START CODE HERE ### (≈ 4 line of code)
if st_next is None:
  t[act] = rew
else:
  t[act] = rew + self.gamma*np.amax(t[act])
### END CODE HERE ###
```

Q-Learning is a value-based reinforcement learning algorithm which is used to find the optimal action-selection policy using a Q function. The goal is to maximize the value function Q. Q(state, action) returns the expected future reward of that action at that state. Here, γ is a discount factor which is multiplied by future rewards discovered by the agent to make them of less worth than the immediate rewards.

3 Results:

Configuration:

$\epsilon_{min} = 0.05$

$\epsilon_{max} = 1$

$\gamma = 0.99$

Number of episodes = 10000

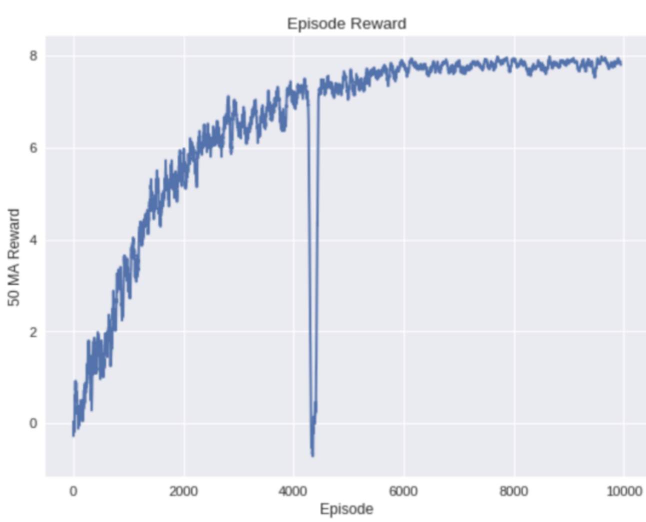


Figure 1: Variation of Rewards with number of Episodes

For the above configuration, it seen quite evident from the graph that the agent converge to the optimal policy in about closer to 6000 episodes. It took around 760 seconds to run for 10000 episodes.

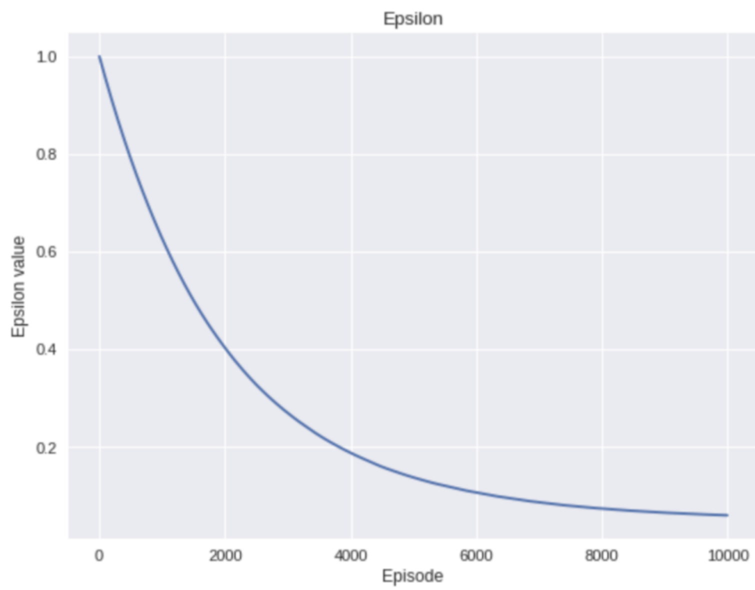


Figure 2: Epsilon is decaying exponentially with number of episodes

4 Hyperparameter Tuning:

4.1 Variaton of total mean-reward with different number of episodes

Configuration:

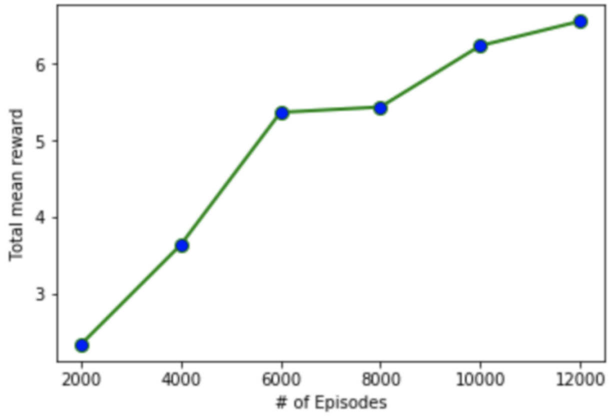
$\epsilon_{min} = 0.05$

$\epsilon_{max} = 1$

$\gamma = 0.99$

Number of episodes = 10000

# of Episodes	**2000**	**4000**	**6000**	**8000**	**10000**	**12000**
Total Mean Reward	**2.34**	**3.64**	**5.37**	**5.44**	**6.24**	**0.21**



4.2 Variation of total mean epsilon value with different number of episodes

Configuration:

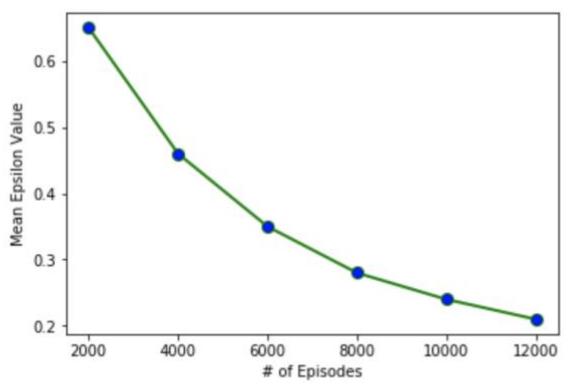
$\epsilon_{min} = 0.05$

$\epsilon_{max} = 1$

$\gamma = 0.99$

Number of episodes = 10000

# of Episodes	**2000**	**4000**	**6000**	**8000**	**10000**	**12000**
Total Mean Epsilon	**0.65**	**0.46**	**0.35**	**0.28**	**0.24**	**0.21**



4.3 Variaton of total mean-reward with ϵ_{max}

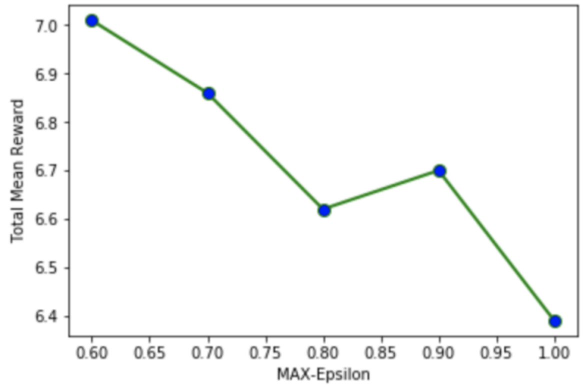
Configuration:

$\epsilon_{min} = 0.05$

$\gamma = 0.99$

Number of episodes = 10000

ϵ_{max}	0.6	0.7	0.8	0.9	1
Total Mean reward	7.01	6.86	6.62	6.70	6.39



4.4 Variaton of total mean-reward with ϵ_{min}

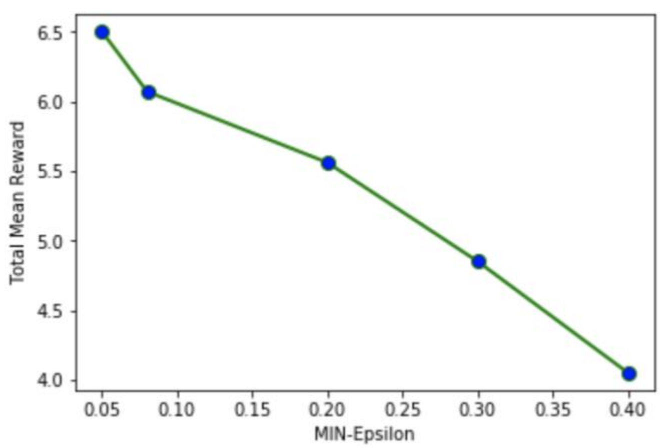
Configuration:

$\epsilon_{max} = 1$

$\gamma = 0.99$

Number of episodes = 10000

ϵ_{min}	0.05	0.08	0.2	0.3	0.4
Total Mean reward	6.50	6.07	5.56	4.85	4.05



5 Writing Tasks

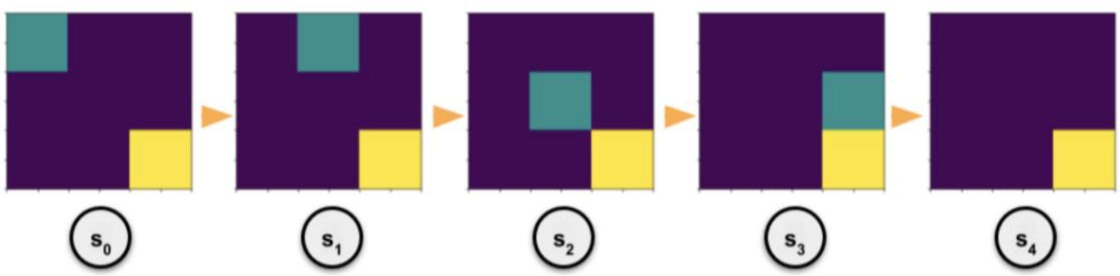
5.1 Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value. Suggest two ways to force the agent to explore?

If the agent always chooses the action that maximizes the Q-value and if it has not found the optimal policy to select which actions to perform then the agent will perform non-optimal actions repeatedly. But if the agent has not found the optimal policy then exploring actions which do not have the highest Q-value might allow it to find a better policy.

The two ways to force the agent to explore are:

- Allow it to pick random actions occasionally.
- Set the initial Q-values high so that the unexplored region good look to explore.

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



For State 4, it is a terminal state which means that agent has reached the goal so Q-value will be zero for any action corresponding to state 4.

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0				
1				
2				
3				
4	0	0	0	0

For State 3,
Q(S3, Down) = 1 + $\gamma * max_a Q(s_{t+1}, a)$ = 1 +
0.99*max_a(Q(S4,UP),Q(S4,DOWN),Q(S4,LEFT),Q(S4,RIGHT))
= 1 + 0.99*0 = 1

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0				
1				
2				
3		1		
4	0	0	0	0

Q(S3,RIGHT) = 0 + 0.99*$max_a Q(S3, a)$ = 0 + 0.99*1 = 0.99

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0				
1				
2				
3		1		0.99
4	0	0	0	0

Q(S3,LEFT) = Q(S3,UP) as these actions are symmetric.
Q(S3,LEFT) = -1 + 0.99* $max_a Q(S2, a)$

For State 2,
Q(S2,RIGHT) = 1 + 0.99*$max_a Q(S3, a)$ = 1 + 0.99*1 = 1.99
Q(S2,DOWN) = Q(S2,RIGHT) as they are symmetric actions.
Q(S2,UP) = -1 + 0.99*$max_a Q(S1, a)$
Q(S2,LEFT) = Q(S2,UP) as these are symmetric actions.

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0				
1				
2		1.99		1.99
3		1		0.99
4	0	0	0	0

For State 1,
Q(S1,DOWN) = 1 + 0.99*$max_a Q(S2, a)$ = 1 + 0.99*1.99 = 2.97
Q(S1,RIGHT) = Q(S1,DOWN) as these are symmetric actions
Q(S1,UP) = 0+ 0.99*2.97 = 2.94
Q(S1,LEFT) = -1 + 0.99*$max_a Q(S1, a)$

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0				
1	2.94	2.97		2.97
2		1.99		1.99
3		1		0.99
4	0	0	0	0

For State 0,
Q(S0,RIGHT) = 1 + 0.99*$max_a Q(S1, a)$ = 1 + 0.99*2.97 = 3.94
Q(S0, DOWN) = Q(S0,RIGHT) as these are symmetric actions
Q(S0,UP) = 0 + 0.99*$max_a Q(S0, a)$ = 0.99*3.94 = 3.90
Q(S0, LEFT) = Q(S0,UP) as these are symmetric actions.

STATE	ACTION			
	UP	DOWN	LEFT	RIGHT
0	3.90	3.94	3.90	3.94
1	2.94	2.97		2.97
2		1.99		1.99
3		1		0.99
4	0	0	0	0

Q(S1,LEFT) = -1 + 0.99*$max_a Q(S1, a)$ = -1 + 0.99*3.94 = 2.90

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

Q(S2,UP) = -1 + 0.99*$max_a Q(S1, a)$ = -1 + 0.99*2.97 = 1.94

STATE	ACTION			
	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		1		0.99
4	0	0	0	0

Q(S3,LEFT) = -1 + 0.99* $max_a Q(S2, a)$ = -1 + 0.99*1.99 = 0.97

	ACTION			
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