Introduction to Machine Learning: Project 4.0

Malvika Sundaram Srinivasan
50290572
msundara@buffalo.edu

1 Introduction

The task of this project is to implement reinforcement learning based on Deep Q-network in order to be able to play the Tom and Jerry game,where the agent i.e. Tom's task is to find a shortest (optimal) path in the fully observable grid-environment and successfully catch Jerry. The reinforcement learning, consists of Markov components like Agent, Environment,Action,State,Reward and Policy.

1. The **agent** here is Tom which moves in the grid world to catch Jerry and the **environment** is the world where the agent takes action to reach the goal and the environment also gets updated based on the action of the agent
2. **Action** in here is Up,Down,Left,Right and the agent can move in any of those four directions which is represented as numbers between [0,3]
3. **State** is the current or present environment where the agent must have taken an action in the environment and a **policy** is a function that maps actions to states and should return a maximum the Q-value.

2 Implementation

- initialize replay memory D
- initialize action-value function Q (with random weights)
- observe initial state s
- repeat
    - select an action a
        - with probability ϵ select a random action
        - otherwise select $a = \backslash\mathbf{argmax}_{a'} Q(s, a')$
    - carry out action a
    - observe reward r and new state s'
    - store experience $< s, a, r, s' >$ in replay memory D
    - sample random transitions $< ss, aa, rr, ss' >$ from replay memory D
    - calculate target for each minibatch transition
        - if ss' is terminal state then $tt = rr$
        - otherwise $tt = rr + \gamma \max_{a'} Q(ss', aa')$
    - train the Q network using $(tt - Q(ss, aa))^2$ as loss
    - $s = s'$

Figure 1: Deep Q-learning Algorithm Pseudocode

This is the overall flow of the deep Q-learning algorithm with experience replay and ϵ-greedy exploration. The portions that were implemented are,

2.1 Brain

1. The role Brain plays in training the agent is very crucial.It implements Neural Network where the model is trained based on the observed data and it predicts the action to be taken based on the trained data.The data in which it gets trained are the Q values that were calculated for the observed data and the respective states.The Q-value for a action is taken as immediate reward and argmax of the predicted Q-value for the future states.The brain helps the agent understand the environment better and helps to take better actions that will help it reach the goal sooner.
2. The agent reacts with the environment and this data is fed into a Neural network model called the brain. Only a set of experiences is stored as memory and the model uses this for training
3. In the brain part a 3 layer Neural network model was defined,with input layer dimensions taken as 4 i.e the state dimension,the output layer has the dimension of action dimension which is also 4 and the hidden layer nodes was 128 with activation function taken as relu and input and output layer had the activation function taken as linear
4. The reason for the linear function being used in output layer is to get real values

The neural network in the brain can be modified to include more layers and the activation function can be changed for better performance.The hyperparameter tuning section has the results of tuning the layers and the activation function.

2.2 Exploration rate

The observed data is the set of actions and states that the agent obtained in the exploration phase that are stored in memory and the memory as it has limited capacity gets updated with new explored data. The exploration rate (epsilon) determines the choice between exploration and exploitation.The epsilon value is calculated using the exponential decay function and based on the value of epsilon the agent can explore randomly or take the maximum Q-Value from prediction.As the epsilon decreases the transition moves from exploration to exploitation. This epsilon is determined using the formula,

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

where, $\epsilon_{min}, \epsilon_{max}$ are the greedy exploration value which [0, 1] , λ is the epsilon parameter , S is the total number of steps

The exponential function can be changes to linear or quadratic,the linear function could be of the form

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

The results are given in the next section.

2.3 Experience Replay

The data is taken from memory and the actions are in terms of Q-values.Experience replay helps in training of the agent from the observations obtained and that were stored in memory.These observations are picked randomly from memory to have the agent learn better.The Q function helps in the calculation of rewards for a particular action with which the model gets trained. The Q function Q(s,a) can be represented as reward for doing action in a state and the discounted reward in the next state.

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

where, $Q_t(s, a)$ is the Q value, γ is the exponential epsilon decay value and r is the reward.

The discount value γ does not allow the model to get infinite rewards, and thus we force the model to determine the goal faster. For very large γ values, we see that the convergence takes considerable amount of time and for low values, the model finds it difficult to learn.

3 Hyperparameter Tuning

The model was run in Google Collab with TPU so the timings have reduced to half of that would be obtained when running with GPU

3.1 MAX EPSILON

MAX EPSILON is the exploration value. It is defined as the rate at which the agent decides the action randomly and it plays influence in the epsilon value. With increase in this value, the agent tends to explore more and the reward mean increases. The time taken reduces, and reward mean increases, with decrease in MAX EPSILON.

MAX EPSILON	Training Time	Max Reward	Mean Reward
1.00	441.23 s	8	6.21
0.60	436.01 s	8	6.82
0.10	438.10 s	8	7.57
0.08	424.24 s	8	7.67

```
----------
Episode 9800
Time Elapsed: 437.08s
Epsilon 0.060594256181614445
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.202350273167714
----------
Episode 9900
Time Elapsed: 441.23s
Epsilon 0.06014075025016973
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.2166105499438835
----------
```

```
----------
Episode 9800
Time Elapsed: 431.81s
Epsilon 0.05664103801098757
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.817544583032677
----------
Episode 9900
Time Elapsed: 436.01s
Epsilon 0.05635739230430514
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.824303642485461
----------
```

```
----------
Episode 9800
Time Elapsed: 433.86s
Epsilon 0.05073207157146738
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.572621379239254
----------
Episode 9900
Time Elapsed: 438.10s
Epsilon 0.0507011194587378
Last Episode Reward: 4
Episode Reward Rolling Mean: 7.572492602795633
----------
```

```
----------
Episode 9800
Time Elapsed: 419.59s
Epsilon 0.05045870287103117
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.681269972167818
----------
Episode 9900
Time Elapsed: 424.24s
Epsilon 0.05043935276734363
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.679726558514437
----------
```

Figure 2: Results for Max epsilon values 1, 0.6, 0.1 and 0.08 (left to right)

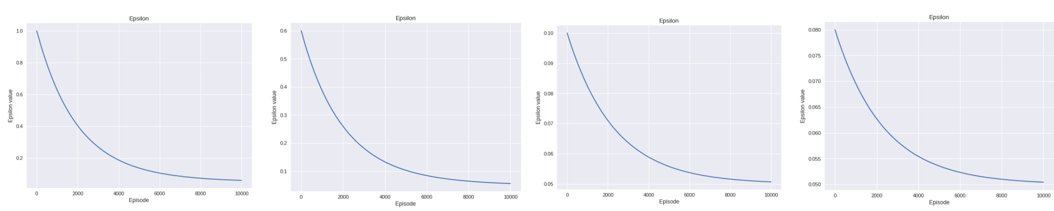


Figure 3: Epsilon for Max epsilon values 1, 0.6, 0.1 and 0.08 (left to right)

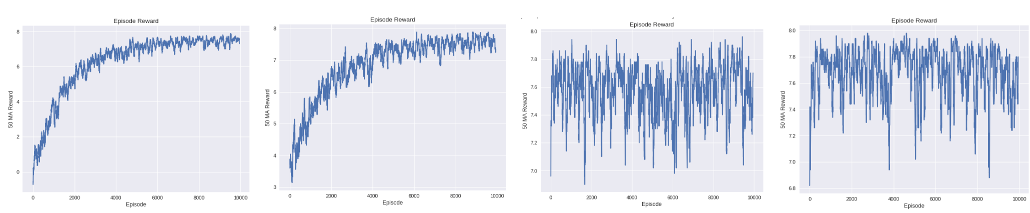
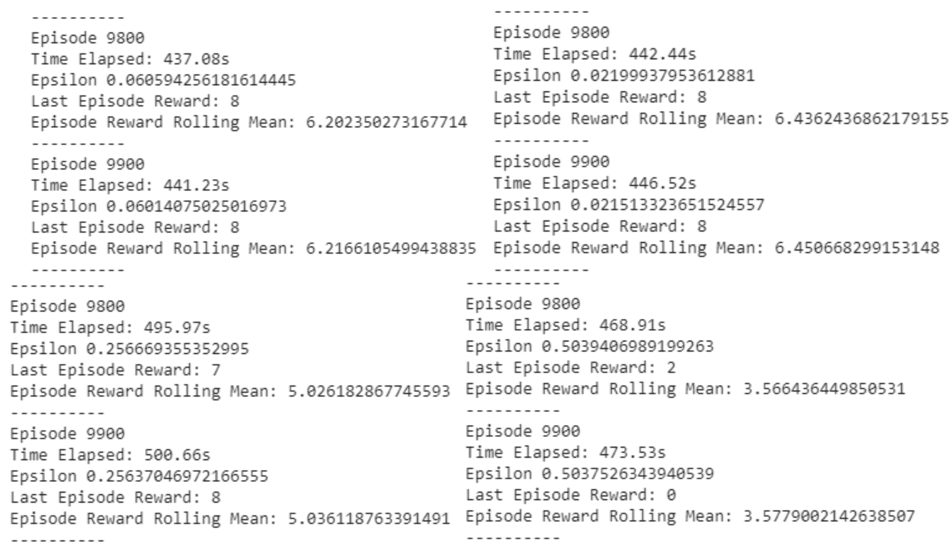


Figure 4: Mean Reward for Max epsilon values 1, 0.6, 0.1 and 0.08 (left to right)

3.2 MIN EPSILON

MIN EPSILON is the exploration prevention value, and is defined as the rate at which the agent decides not to venture out. With increase in this value, the agent tends to exploit more and the reward mean decreases. The time taken increases, and reward mean decreases, with increase in MIN EPSILON.

MIN EPSILON	Training Time	Max Reward	Mean Reward
0.05	441.23 s	8	6.21
0.01	446.52 s	8	6.43
0.25	500.66 s	8	5.03
0.5	473.53 s	5	3.57



```
----------
Episode 9800
Time Elapsed: 437.08s
Epsilon 0.060594256181614445
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.202350273167714
----------
Episode 9900
Time Elapsed: 441.23s
Epsilon 0.06014075025016973
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.2166105499438835
----------
```

```
----------
Episode 9800
Time Elapsed: 442.44s
Epsilon 0.02199937953612881
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.4362436862179155
----------
Episode 9900
Time Elapsed: 446.52s
Epsilon 0.021513323651524557
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.450668299153148
----------
```

```
----------
Episode 9800
Time Elapsed: 495.97s
Epsilon 0.256669355352995
Last Episode Reward: 7
Episode Reward Rolling Mean: 5.026182867745593
----------
Episode 9900
Time Elapsed: 500.66s
Epsilon 0.25637046972166555
Last Episode Reward: 8
Episode Reward Rolling Mean: 5.036118763391491
----------
```

```
----------
Episode 9800
Time Elapsed: 468.91s
Epsilon 0.5039406989199263
Last Episode Reward: 2
Episode Reward Rolling Mean: 3.566436449850531
----------
Episode 9900
Time Elapsed: 473.53s
Epsilon 0.5037526343940539
Last Episode Reward: 0
Episode Reward Rolling Mean: 3.5779002142638507
----------
```

Figure 5: Results for Min epsilon values 0.05, 0.01, 0.25 and 0.5 (left to right)

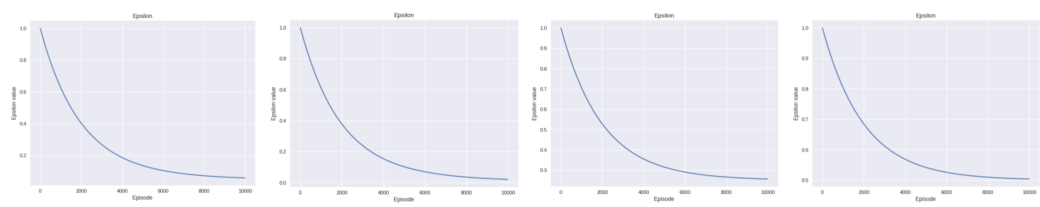


Figure 6: Epsilon for Min epsilon values 0.05, 0.01, 0.25 and 0.5 (left to right)

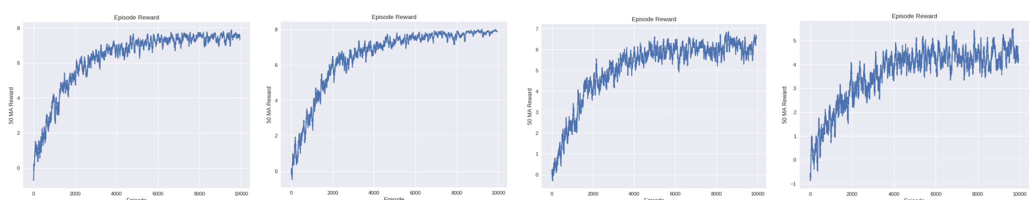


Figure 7: Mean Reward for Min epsilon values 0.05, 0.01, 0.25 and 0.5 (left to right)

3.3 EPISODES

EPISODES determine the number of experiences an agent can undergo. With more number of episodes, the agent has greater opportunity to venture out in the environment, and thus converges faster. The reward mean increases, and the time increases, as number of episodes increase.

EPISODES	Training Time	Max Reward	Mean Reward
10000	441.23 s	8	6.21
5000	244.92 s	8	4.98
18000	841.75 s	8	6.87
25000	1158.14 s	8	6.96

```
----------
Episode 9800
Time Elapsed: 437.08s
Epsilon 0.060594256181614445
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.202350273167714
----------
Episode 9900
Time Elapsed: 441.23s
Epsilon 0.06014075025016973
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.2166105499438835
----------
```

```
----------
Episode 4800
Time Elapsed: 240.20s
Epsilon 0.14515546089630976
Last Episode Reward: 7
Episode Reward Rolling Mean: 4.945968942778133
----------
Episode 4900
Time Elapsed: 244.92s
Epsilon 0.14093654763735364
Last Episode Reward: 8
Episode Reward Rolling Mean: 4.988335763382628
----------
```

```
----------
Episode 17800
Time Elapsed: 836.40s
Epsilon 0.05035259400359892
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.874244392972148
----------
Episode 17900
Time Elapsed: 841.75s
Epsilon 0.05033778758918678
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.8783776192348745
----------
```

```
----------
Episode 24800
Time Elapsed: 1153.74s
Epsilon 0.050017343490758495
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.959313388121938
----------
Episode 24900
Time Elapsed: 1158.14s
Epsilon 0.050016622667294376
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.962057981533003
----------
```

Figure 8: Results for Episodes values 10000, 500, 18000 and 25000 (left to right)

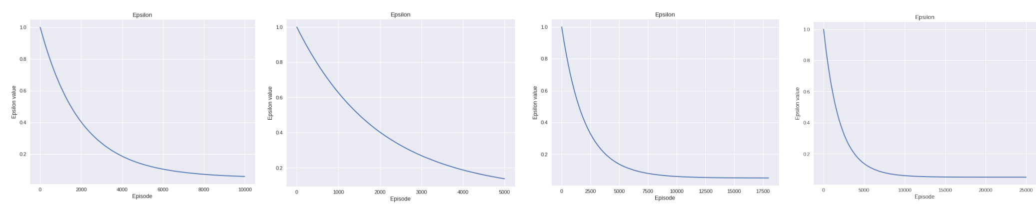


Figure 9: Epsilon for Episodes values 10000, 500, 18000 and 25000 (left to right)

Figure 10: Mean Reward for Episodes values 10000, 500, 18000 and 25000 (left to right)

3.4 GAMMA

GAMMA can used to attain quicker convergence time and obtain finite results. With increase in the GAMMA value, the time and mean reward does not steadily increase nor decrease.

GAMMA	Training Time	Max Reward	Mean Reward
0.02	441.23 s	8	6.21
0.25	416.60 s	8	6.48
0.6	419.87 s	7	6.51
0.99	419.02 s	8	6.47

```
----------
Episode 9800
Time Elapsed: 437.08s
Epsilon 0.060594256181614445
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.202350273167714
----------
Episode 9900
Time Elapsed: 441.23s
Epsilon 0.06014075025016973
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.2166105499438835
----------
```

```
----------
Episode 9800
Time Elapsed: 412.85s
Epsilon 0.06127527172182002
Last Episode Reward: 7
Episode Reward Rolling Mean: 6.475724152149263
----------
Episode 9900
Time Elapsed: 416.60s
Epsilon 0.060812598527588274
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.489031731455974
----------
```

```
----------
Episode 9800
Time Elapsed: 415.92s
Epsilon 0.06128034673588737
Last Episode Reward: 6
Episode Reward Rolling Mean: 6.505824141841047
----------
Episode 9900
Time Elapsed: 419.87s
Epsilon 0.06081097675944479
Last Episode Reward: 7
Episode Reward Rolling Mean: 6.518926640138761
----------
```

```
----------
Episode 9800
Time Elapsed: 414.93s
Epsilon 0.061344828310184896
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.464694361406041
----------
Episode 9900
Time Elapsed: 419.02s
Epsilon 0.06087060094783724
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.478726660544842
----------
```

Figure 11: Results for discounting factor 0.02, 0.25, 0.6 and 0.99 (left to right)

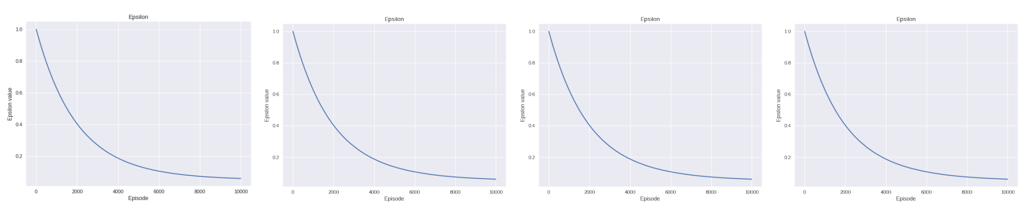


Figure 12: Epsilon for discounting factor 0.02, 0.25, 0.6 and 0.99 (left to right)

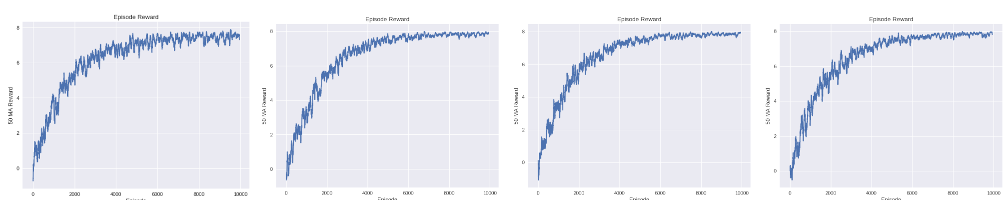


Figure 13: Mean Reward for discounting factor 0.02, 0.25, 0.6 and 0.99 (left to right)

3.5 HIDDEN LAYERS

The hidden layers introduces non-linearity in the model while transforming the input data into output data. With many hidden layers, the reward mean reduces and the time taken increases.

Hidden Layer	Training Time	Max Reward	Mean Reward
None	340.60 s	8	6.45
One	441.23 s	8	6.21
Two	545.70 s	8	6.21
Three	606.70 s	7	5.57

```
----------
Episode 9800
Time Elapsed: 337.45s
Epsilon 0.061155269471355814
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.444387176579734
----------
Episode 9900
Time Elapsed: 340.60s
Epsilon 0.06069538119862734
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.45679012345679
----------
```

```
----------
Episode 9800
Time Elapsed: 437.08s
Epsilon 0.060594256181614445
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.202350273167714
----------
Episode 9900
Time Elapsed: 441.23s
Epsilon 0.06014075025016973
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.2166105499438835
----------
```

```
----------
Episode 9800
Time Elapsed: 540.98s
Epsilon 0.06056516199995019
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.205030409236161
----------
Episode 9900
Time Elapsed: 545.70s
Epsilon 0.060105319663988674
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.219977553310887
----------
```

```
----------
Episode 9800
Time Elapsed: 598.92s
Epsilon 0.05965824528872418
Last Episode Reward: 6
Episode Reward Rolling Mean: 5.557777548706319
----------
Episode 9900
Time Elapsed: 606.70s
Epsilon 0.0592411097808552
Last Episode Reward: 7
Episode Reward Rolling Mean: 5.576267727782879
----------
```

Figure 14: Results for 0, 1 and 2 hidden layers (left to right)

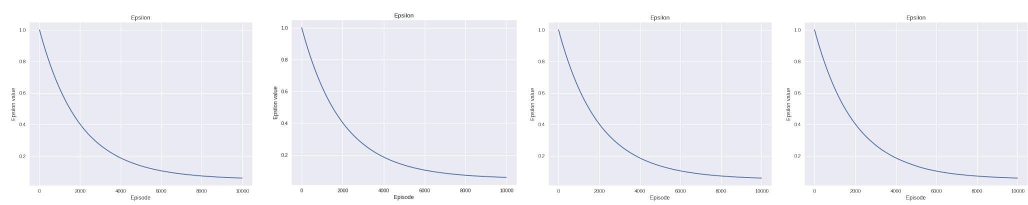


Figure 15: Epsilon for 0, 1 and 2 hidden layers (left to right)

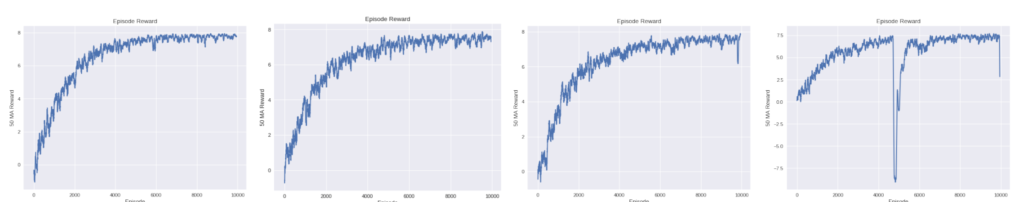


Figure 16: Mean Reward for 0, 1 and 2 hidden layers (left to right)

3.6 ACTIVATION FUNCTION

The activation function ReLu is a non-linear activation function whereas the other activation is Linear activation. It has reduced reward mean and increased time.

Activation function	Training Time	Max Reward	Mean Reward
Linear	407.25 s	8	7.62
ReLu	449.44 s	8	7.20

```
Episode 9800
Time Elapsed: 403.33s
Epsilon 0.04
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.626636429234099
----------
Episode 9900
Time Elapsed: 407.25s
Epsilon 0.04
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.626670747882869
```

```
Episode 9800
Time Elapsed: 445.00s
Epsilon 0.04
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.208638284712916
----------
Episode 9900
Time Elapsed: 449.44s
Epsilon 0.04
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.209876543209877
```

Figure 17: Results for ReLu and Linear Hidden layer activation function (left to right)

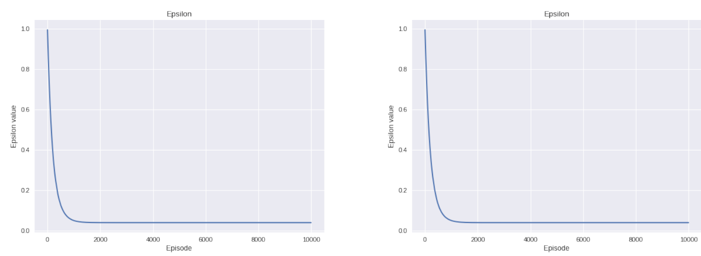


Figure 18: Epsilon for ReLu and Linear Hidden layer activation function (left to right)

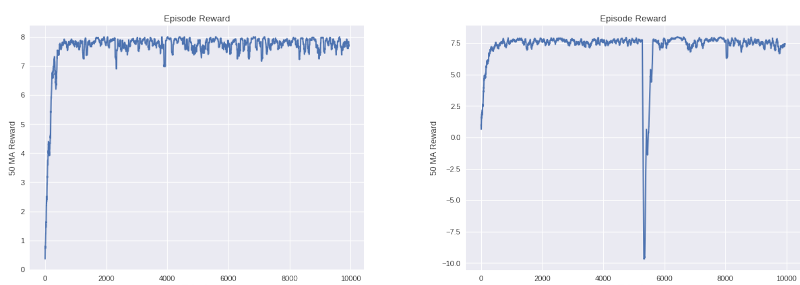


Figure 19: Mean Reward for ReLu and Linear Hidden layer activation function (left to right)

3.7 EPSILON HYPERPARAMETER(λ)

The agent is forced to move towards the reward faster using the epsilon decay speed value. The increase in epsilon results, decrease in reward mean and increased time.

Epsilon Hyperparameter(λ)	Training Time	Max Reward	Mean Reward
0.0005	446.27 s	8	6.28
0.005	433.13 s	8	7.50
0.05	431.03 s	8	7.62

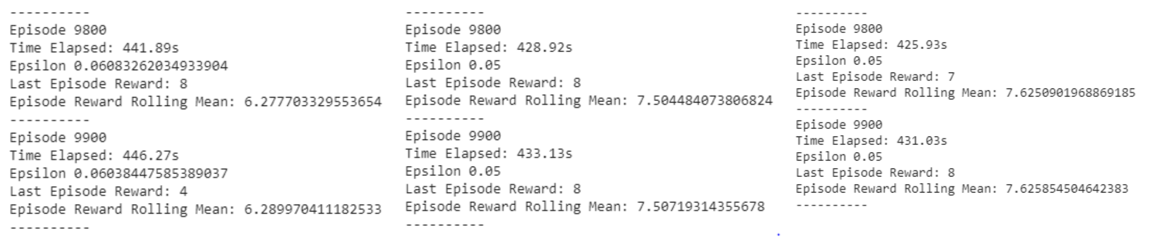

```
----------
Episode 9800
Time Elapsed: 441.89s
Epsilon 0.06083262034933904
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.277703329553654
----------
Episode 9900
Time Elapsed: 446.27s
Epsilon 0.06038447585389037
Last Episode Reward: 4
Episode Reward Rolling Mean: 6.289970411182533
----------
```
```
----------
Episode 9800
Time Elapsed: 428.92s
Epsilon 0.05
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.504484073806824
----------
Episode 9900
Time Elapsed: 433.13s
Epsilon 0.05
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.50719314355678
----------
```
```
----------
Episode 9800
Time Elapsed: 425.93s
Epsilon 0.05
Last Episode Reward: 7
Episode Reward Rolling Mean: 7.6250901968869185
----------
Episode 9900
Time Elapsed: 431.03s
Epsilon 0.05
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.625854504642383
----------
```

Figure 20: Results for 0.0005, 0.005 and 0.05 Epsilon Hyperparameter (left to right)

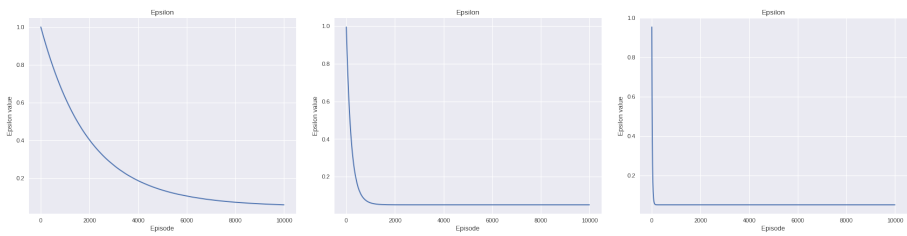


Figure 21: Epsilon for 0.0005, 0.005 and 0.05 Epsilon Hyperparameter (left to right)

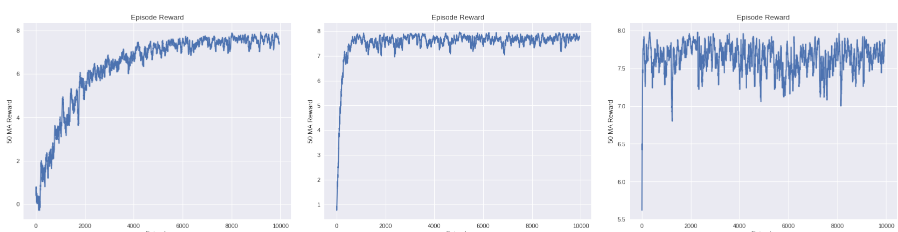


Figure 22: Mean Reward for 0.0005, 0.005 and 0.05 Epsilon Hyperparameter (left to right)

3.8 DECAY FUNCTION

The agent is forced to move towards the reward faster using the epsilon decay speed value. The type of decay function determines how quick the the agent reaches the goal. Using a linear function, we get the opposite of what we require, whereas exponential gives better reward mean.

Decay Function	Training Time	Max Reward	Mean Reward
Exponential	446.27 s	8	6.28
Linear	452.67 s	7	0.739


```
----------
Episode 9800
Time Elapsed: 441.89s
Epsilon 0.06083262034933904
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.277703329553654
----------
Episode 9900
Time Elapsed: 446.27s
Epsilon 0.06038447585389037
Last Episode Reward: 4
Episode Reward Rolling Mean: 6.289970411182533
----------
```
```
----------
Episode 9800
Time Elapsed: 448.28s
Epsilon 4.670325
Last Episode Reward: -6
Episode Reward Rolling Mean: 0.7494072776002474
----------
Episode 9900
Time Elapsed: 452.67s
Epsilon 4.717825
Last Episode Reward: -4
Episode Reward Rolling Mean: 0.7397204366901337
----------
```

Figure 23: Results for Exponential and Linear function (left to right)

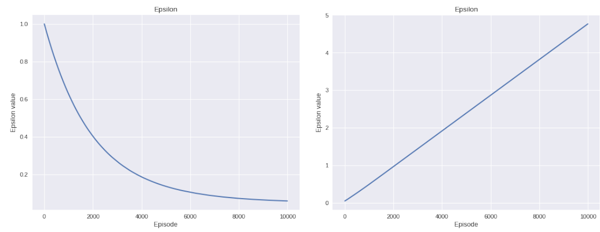


Figure 24: Epsilon for Exponential and Linear function (left to right) (left to right)

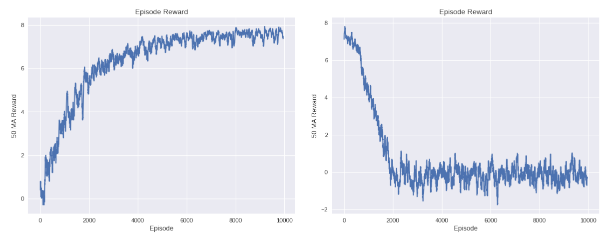


Figure 25: Mean Reward for Exponential and Linear function (left to right) (left to right)

3.9 Observations

Based on the results of tuning the hyperparameters we choose the most optimal values that result in best accuracy, which is listed in the table below.

Hyperparameters	Training Time
MAX EPSILON	1
MIN EPSILON	0.045
EPISODES	10000
GAMMA	0.99
HIDDEN LAYERS	One
ACTIVATION FUNCTION	ReLU
EPSILON HYPERPARAMETER	0.0005
DECAY FUNCTION	Exponential

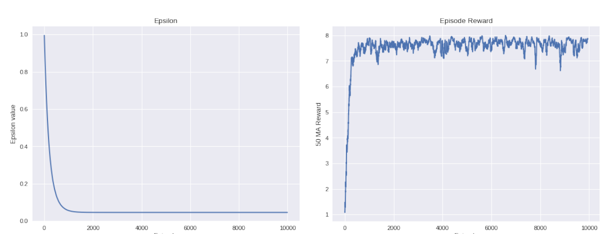


Figure 26: Epsilon and mean reward (left to right) after tweaking the hyperparameters

```
----------
Episode 9800
Time Elapsed: 436.81s
Epsilon 0.045
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.511390578290897
----------
Episode 9900
Time Elapsed: 441.11s
Epsilon 0.045
Last Episode Reward: 8
Episode Reward Rolling Mean: 7.513416998265483
----------
```

Figure 27: Results after tweaking the hyperparameters

4 Writing Task : Answers

4.1 Question 1

If in the reinforcement learning the agent always chooses the action that maximizes the Q-values there is a possibility that it might not reach the goal and still keep traversing the environment based on the maximum Q-value,sometimes it might even get stuck in a infinite loop. The probable options to avoid that scenario is to do exploration which can be in two possible ways

1. The first way is to decide the action on a random basis so it is not biased
2. The second way is to set the Q-value of the initial states high so that the agent reaches the goal as the values might decrease when it moves away from the goal(i.e.)choose the Q-value as the argmax of the policy function used

4.2 Question 2

The states for the calculation of Q-values are mentioned in the diagram for a 3x3 grid and the values for the Q table is calculated for 5 states (s_0, s_1, s_2, s_3, s_4) for the four actions (Up,Down,Left,Right).

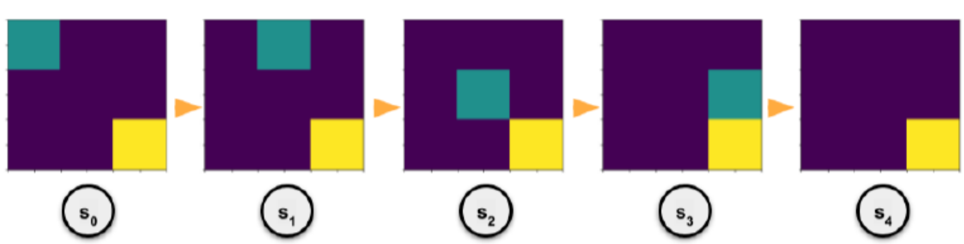


Figure 28: The optimal path chosen to calculate the Q values

The equation used to calculate the Q values and update the table

$$Q(s_t, a_t) = r_t + \gamma * max_a(Q(s_{t+1}, a))$$

where γ is taken as 0.99

Table

ACTIONS				
STATE	UP	DOWN	LEFT	RIGHT
s_0	3.9008	3.9403	3.9008	3.9403
s_1	2.9403	2.9701	2.9008	2.9701
s_2	1.9403	1.99	1.9403	1.99
s_3	0.9701	1	0.9701	0.99
s_4	0	0	0	0

Calculations

1. s_4
   - As the final state s_4 is the goal state, there is no possible action after that needs to be taken after reaching the goal state,therefore all the action reward values are zero. (i.e immediate and future rewards are zero)
   - Q(s_4,down) = 0
   - Q(s_4,up) = 0
   - Q(s_4,left) = 0
   - Q(s_4,right) = 0

2. s_3
   - Q(s_3,down) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_4))$ = 1+0.99 * 0 = 1
     The agent moves towards the goal, so the reward is 1. The next state is max(Q(s4)) = 0.
   - Q(s_3,up) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_2))$ = -1 + 0.99(1.99) = 0.9701
     The agent does not move towards the goal, so the reward is -1. The next state is max(Q(s2)) = 1.99, which would be calculated in the next step.
   - Q(s_3,left) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_2))$ = -1 + 0.99(1.99) = 0.9701
     The agent does not move towards the goal, so the reward is -1. The next state is max(Q(s2)) = 1.99, which would be calculated in the next step. This state is symmetric to the state $Q(s_3, up)$
   - Q(s_3,right) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_3))$ = 0 + 0.99(1) = 0.99
     The agent does not move, so the reward is 0. The next state is max(Q(s3)) = 1.

3. s_2
   - Q(s_2,down) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_3))$ = 1+0.99 * 1 = 1.99
     The agent moves towards the goal, so the reward is 1. The next state is max(Q(s3)) = 1.
   - Q(s_2,up) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_1))$ = -1 + 0.99(2.9701) = 1.9403
     The agent does not move towards the goal, so the reward is -1. The next state is max(Q(s1)) = 2.97, which would be calculated in the next step.
   - Q(s_2,left) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_1))$ = -1 + 0.99(2.9701) = 1.9403
     The agent does not move towards the goal, so the reward is -1. The next state is max(Q(s1)) = 2.9701, which would be calculated in the next step. This state is an symmetric of the state $Q(s_2, up)$
   - Q(s_2,right) = $\mathrm{r}_t + 0.99 * max(Q(\mathrm{s}_3))$ = 1 + 0.99(1) = 1.99
     The agent moves towards the goal, so the reward is 1. The next state is max(Q(s3)) = 1.This state is an symmetric of the state $Q(s_2, down)$

4. s_1

- Q(s_1,down) = $r_t + 0.99 * max(Q(s_2))$ = 1+0.99 * 1.99 = 2.9701
The agent moves towards the goal, so the reward is 1. The next state s max(Q(s2)) = 1.99.
- Q(s_1,up) = $r_t + 0.99 * max(Q(s_1))$ = 0 + 0.99(2.9701) = 2.9403
The agent does not move, so the reward is 0. The next state is max(Q(s1)) = 2.9701.
- Q(s_1,left) = $r_t + 0.99 * max(Q(s_0))$ = -1 + 0.99(3.9403) = 2.9008
The agent does not move towards the goal, so the reward is -1. The next state is max(Q(s0)) = 3.9403, which would be calculated in the next step.
- Q(s_1,right) = $r_t + 0.99 * max(Q(s_2))$ = 1 + 0.99(1.99) = 2.9701
The agent moves towards the goal, so the reward is 1. The next state is max(Q(s2)) = 1.99 .This state is an symmetric of the state $Q(s_1, down)$

5. $\underline{s_0}$

- Q(s_0,down) = $r_t + 0.99 * max(Q(s_1))$ = 1+0.99 * 2.9701 = 3.9403
The agent moves towards the goal, so the reward is 1. The next state s max(Q(s1)) = 2.9701.
- Q(s_0,up) = $r_t + 0.99 * max(Q(s_0))$ = 0 + 0.99(3.9403) = 3.9008
The agent does not move, so the reward is 0. The next state is max(Q(s0)) = 3.9403
- Q(s_0,left) = $r_t + 0.99 * max(Q(s_0))$ = 0 + 0.99(3.9403) = 3.9008
The agent does not move, so the reward is 0. The next state is max(Q(s0)) = 3.9403. This state is an symmetric of the state $Q(s_1, up)$
- Q(s_0,right) = $r_t + 0.99 * max(Q(s_1))$ = 1 + 0.99(2.9701) = 3.9403
The agent moves towards the goal, so the reward is 1. The next state is max(Q(s1)) = 2.9701 .This state is an symmetric of the state $Q(s_0, down)$

5 References

1. https://frnsys.com/ai_notes/artificial_intelligence/reinforcement_learning.html
2. https://ai.intel.com/demystifying-deep-reinforcement-learning/
3. https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df