Project 4

Tanmay Singh
50291086
tanmaypr@buffalo.edu

Abstract

In project 4 we are implementing a deep reinforcement learning algorithm to help the agent navigate in the grid environment.

1) Neural Network

In the neural network three main parts were implemented
- Defining the hidden layers in the neural network along with the activation function and the number of nodes.
-calculation of the epsilon values
-calculation of the Q values

i)hidden layers

```
### START CODE HERE ### (≈ 3 lines of code)
model.add(Dense(self.state_dim))
model.add(Activation('relu'))

model.add(Dense(128))
model.add(Activation('relu'))

model.add(Dense(self.action_dim))
model.add(Activation('linear'))
```

-In this part we have added 3 hidden layers. The first hidden layer takes input of the size of input dimension of the observation space I.e In our case the dimension of the observation space is 4 fx,fy,px,py the coordinates of positions of tom and jerry. The activation function is relu. The relu activation always provide output greater than 0 or equal to 0.
-The second hidden layer consist of 128 nodes and the activation function is relu. Relu makes the output values greater than 0 liner.
- The third hidden layer has no of output of the same dimension as that of the action space. In our case the action space is of size 4 up,down,left,right. The activation function in the output layer is linear because we need an real action value to move from one state to the next.

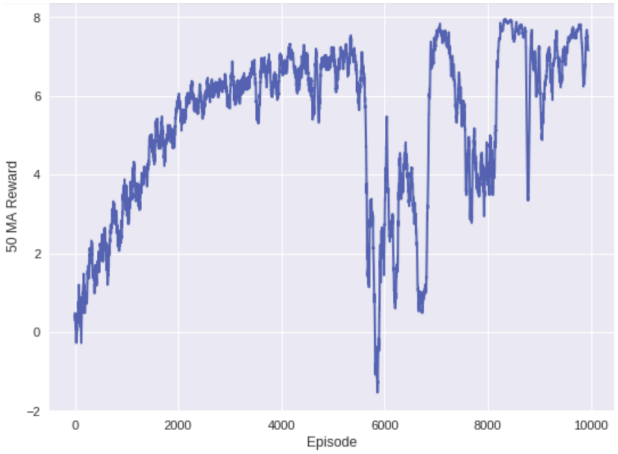
-In the above snipet if we add an additional hidden layer I.e use 4 hidden layers instead of 3 we get the following result.
-

ii) Calculating epsilon

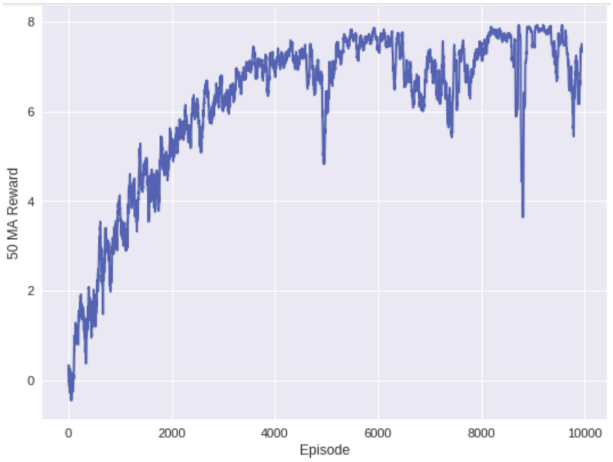
```
### START CODE HERE ### (≈ 1 line of code)
self.epsilon=self.min_epsilon+((self.max_epsilon-self.min_epsilon)*np.exp(-self.lamb*self.steps))#calculating

### END CODE HERE ###
```

we get the following output graph:The hyperparameters are as follows:
Learning Rate=0.01
lambda=0.0005
Number of hidden layers =3
min epsilon=0.05
max epsilon=1



If we change the hyperparameters as follows
min value=0.00
max value=0.00
we get the following result



on decreasing the epsilon value the time required for the model also decreases considerably. The value of reward also stays positive after most of the iterations.

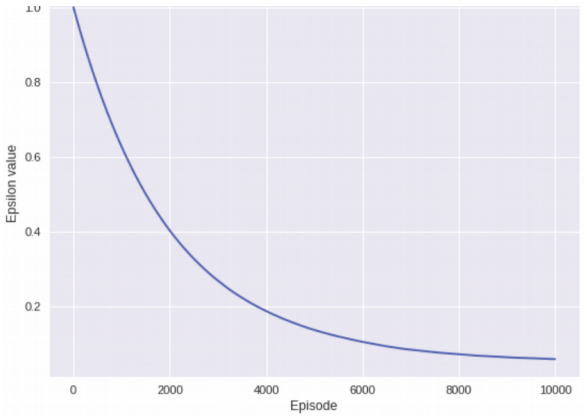
iii) calculating Q values

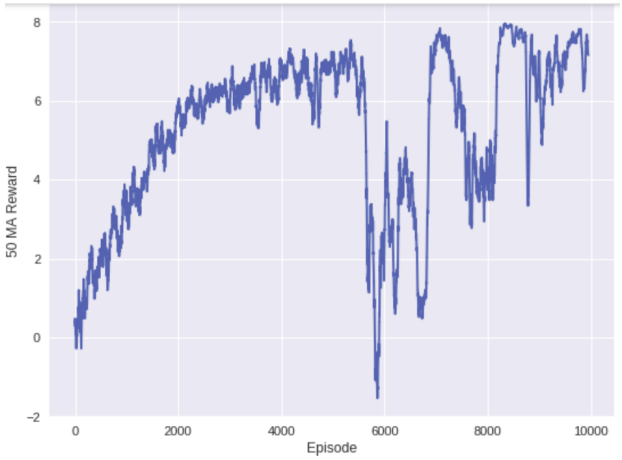
```
### START CODE HERE ### (≈ 4 line of code)
if st_next is not None:
  t[act]=rew + self.gamma* np.amax(q_vals[i])
else:t[act]=rew


### END CODE HERE ###
```

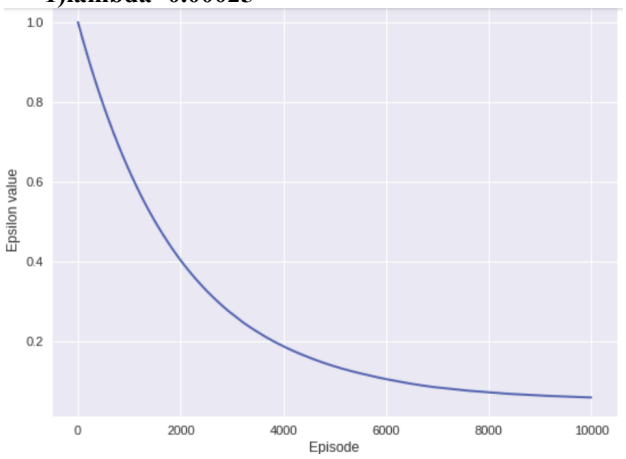
- we make the estimated q value for observation equals to reward if the episode terminates at the next step otherwise we make the q value to be equal to the reward plus the max of the current q value.
-Q value is basically the long term reward for taking a particular action.

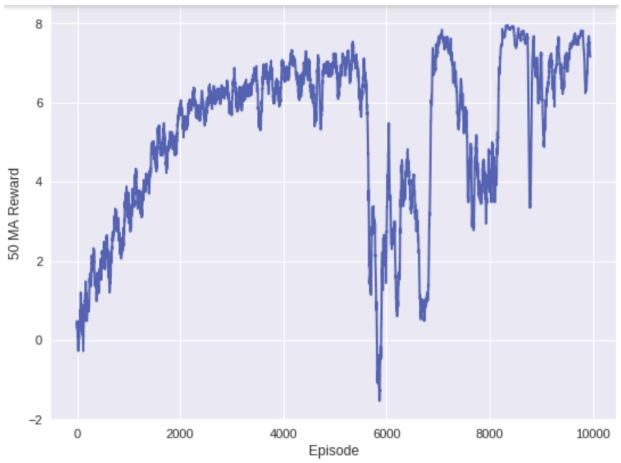
Results on changing the lambda values:



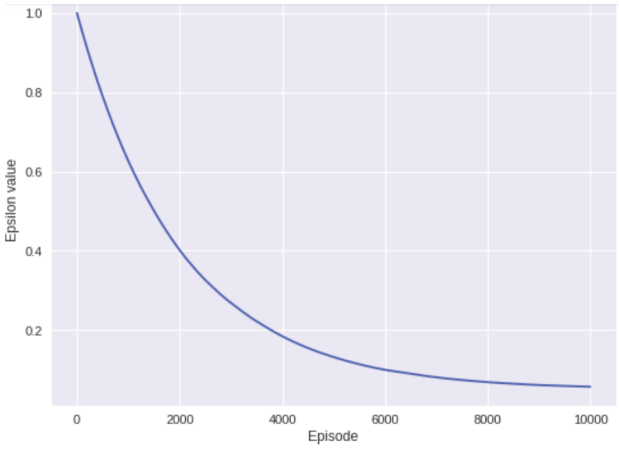


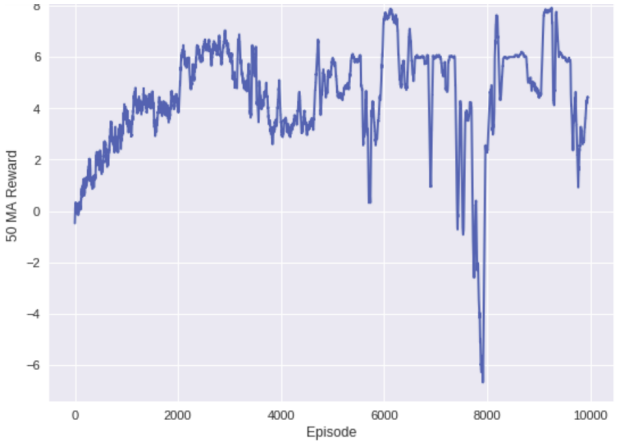
1)lambda=0.00025



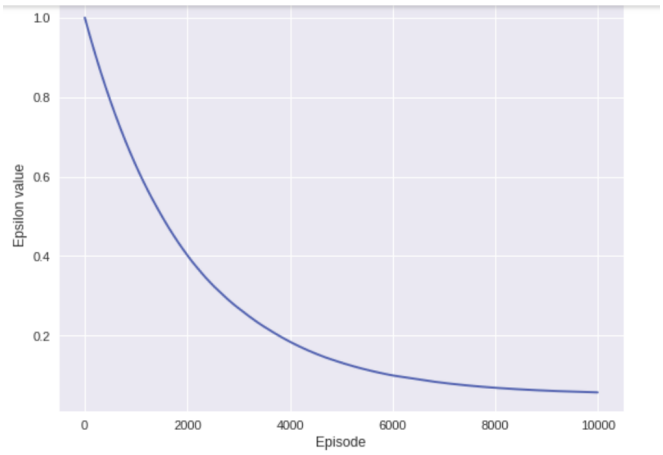


2)lambda=0.0005

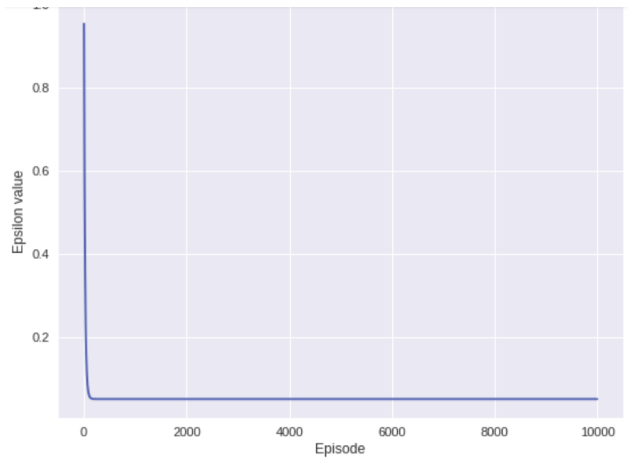


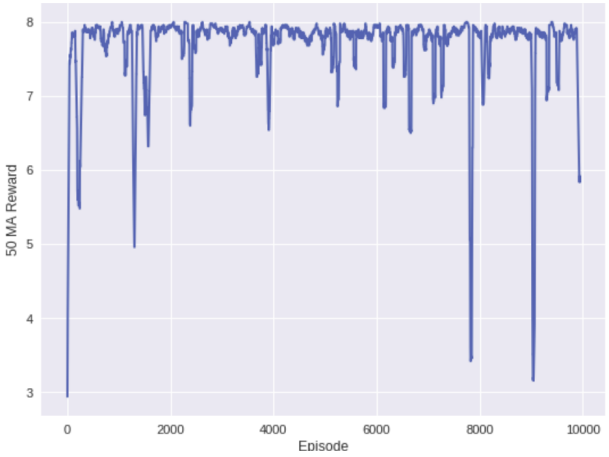


3)lambda=0.005

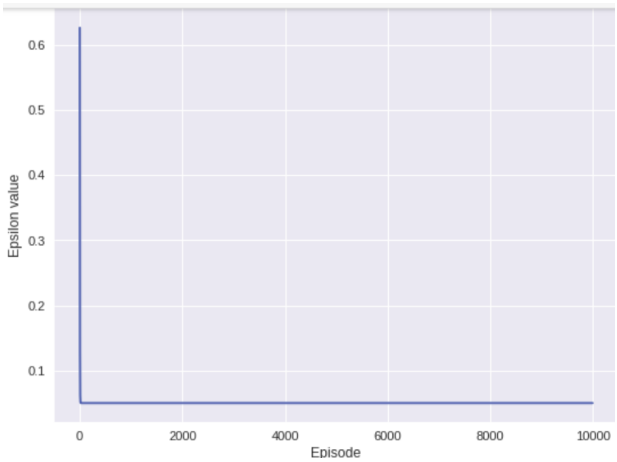


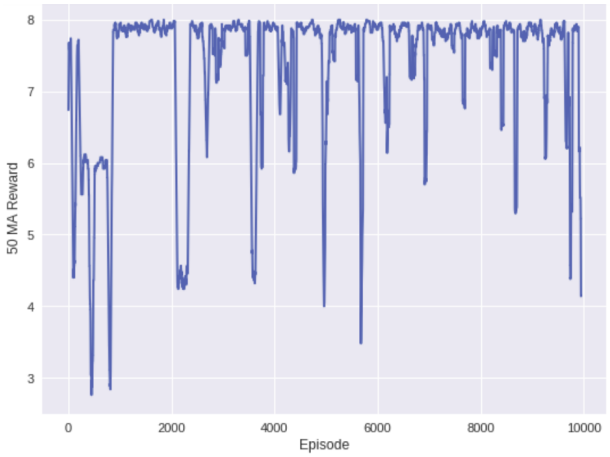
4)lambda 0.5





5)lambda 0.0

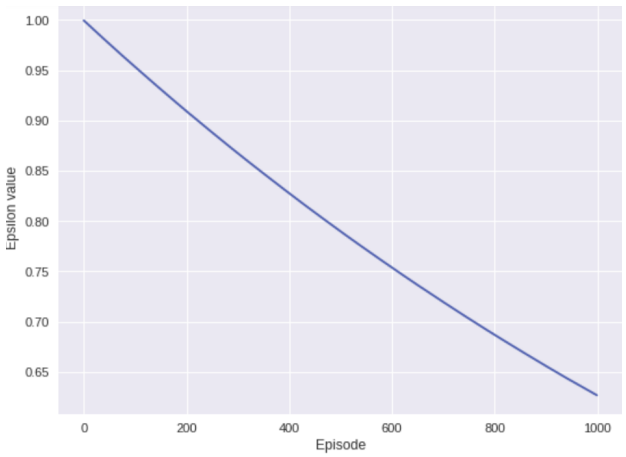


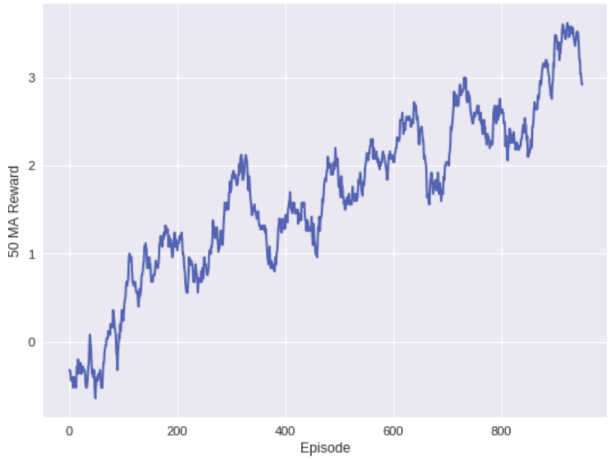


we can see that as the lambda value is decreased the epsilon values become zero much sooner. When the lambda is set at 0.0 and 0.5 the epsilon value is more then 0 just for the initial stage when it is taken randomly and then becomes 0. The mean reward also shows signs of overfitting of the model as we increase the lambda value we constantly get higer reward with a sudden drop in some instances.

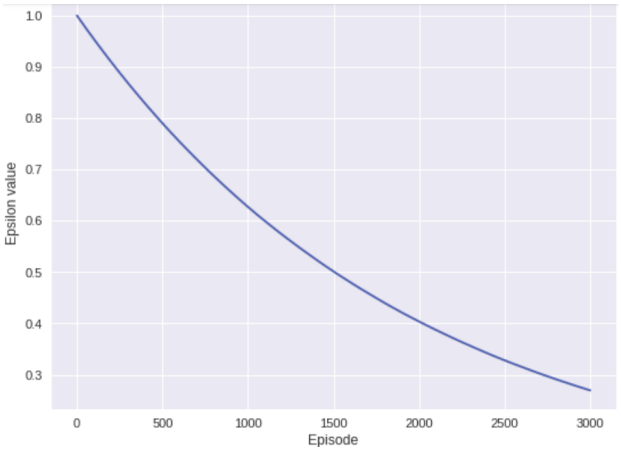
Results on changing number of episodes:

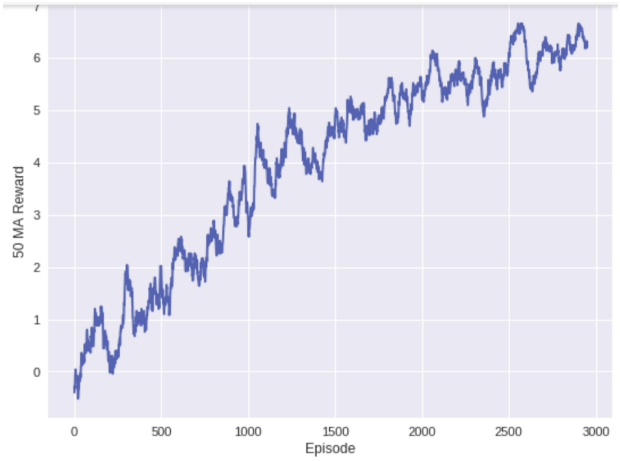
1)1000



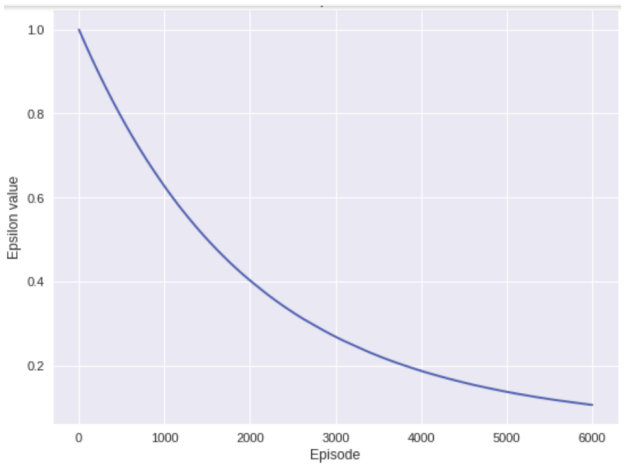


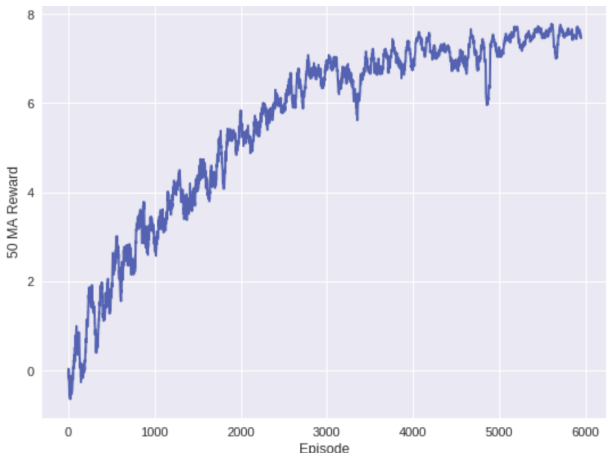
2)3000



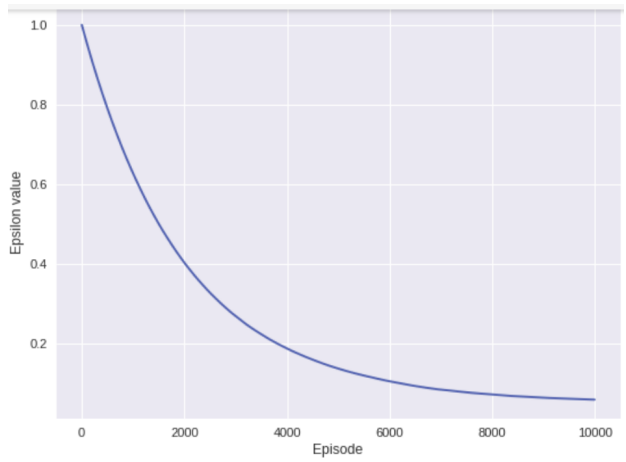


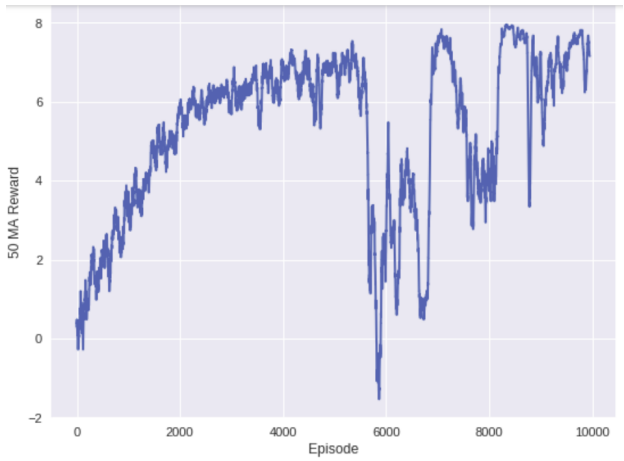
3)6000





4)10000

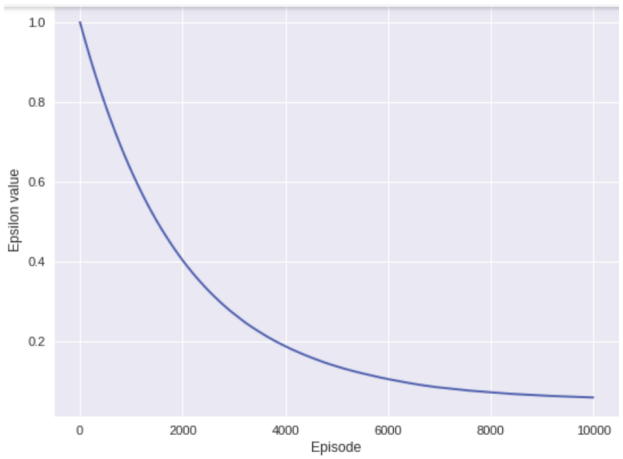


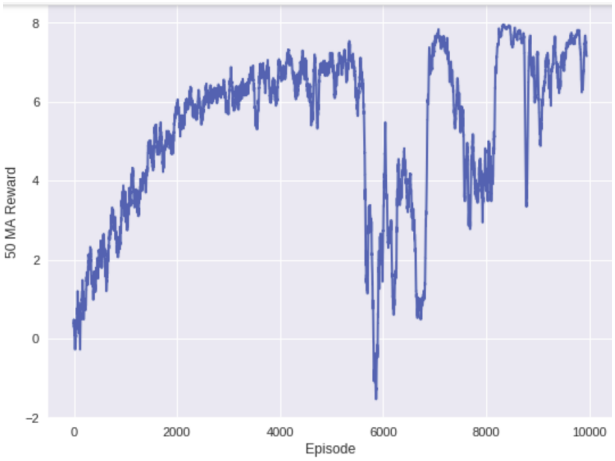


when the number of episodes is 1000 we can see that the maximum mean reward is 3 and then it stops. The value of the mean reward shows that it is under trained. we see the same results when the number of episodes is 3000. when the number of episodes is changed to 6000 we see that the mean reward value increases upto 8 and remains constant even if we further change the number of episodes to 10000 thus 6000 episodes gives us the best results.

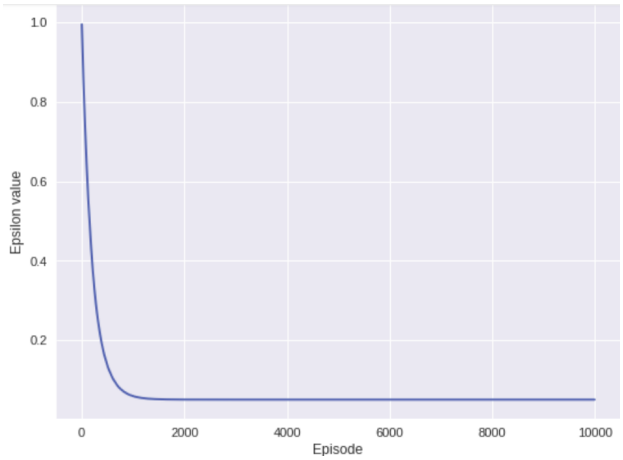
Results on changing the learning rate

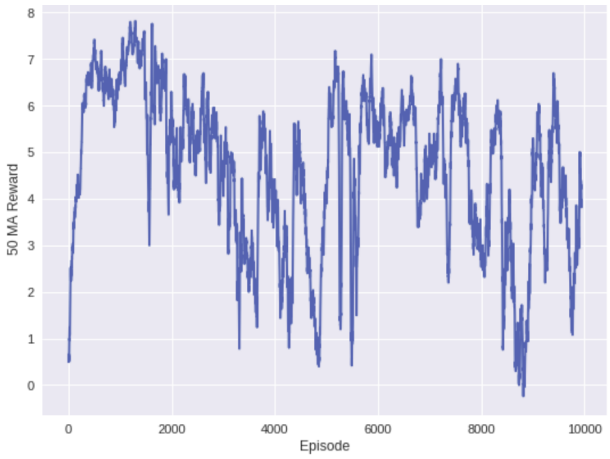
1)lr=0.00025



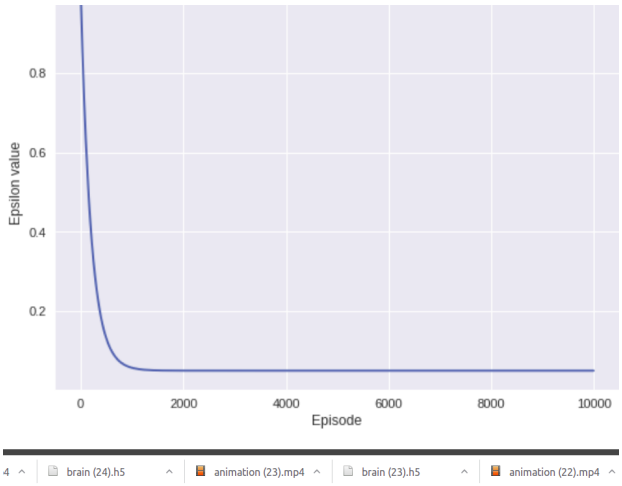


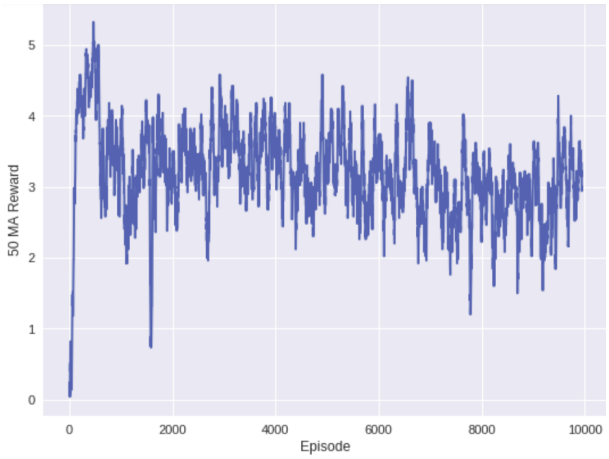
2)lr=0.005



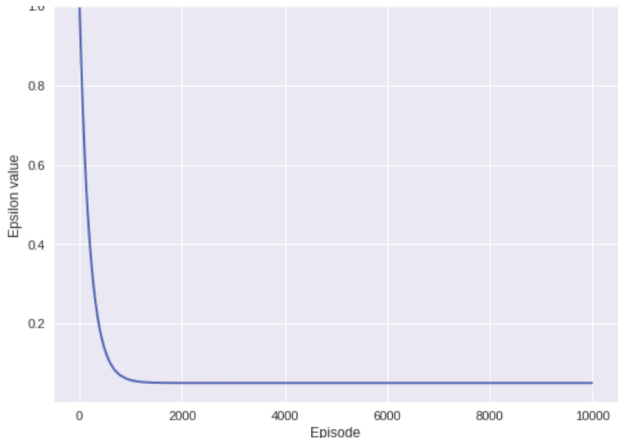


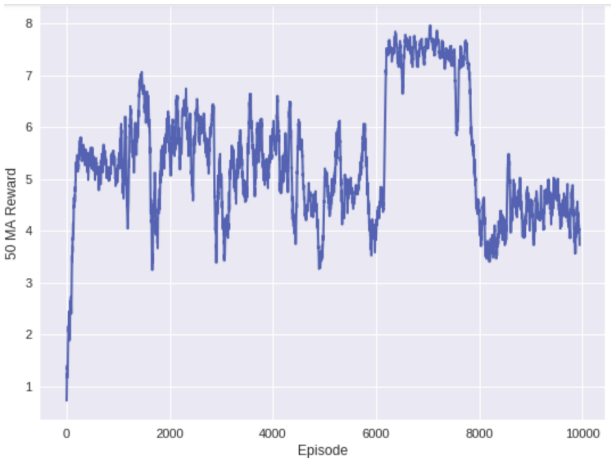
3)lr=0.05





4)lr=0.01





From the above results we can see that as we increase the learning rate the epsilon value becomes zero this shows that the model is taking larger steps in the gradient descent and causing the mean reward to decrease thereby overfitting the model.

1.2 Writing Task

1)
-If the agent always chooses the action that maximizes the Q values then it will not be able to explore at each state to find the best possible action on that state and it will lead to non optimal solutions.
-The agent can be forced to explore by picking random actions occasionally.
-The initial values can be set high so that the unexplored region looks good to the agent and it decides to explore it.

2) given gamma rt =0.99
we know that $Q(st, at) = rt + \gamma * maxaQ(st+1, a)$

Lets start with state s4 going in any direction will result in 0 reward. Therefore we can fill row s4 with zeros.

S0	S1B	P2
S1A	S2	S3B
P1	S3A	GOAL

S1A,S1B and P1,P2 and S3A,S3B are same due to symmetry and therefore will be considered as 1,P,3

Now for state S3
ACTION DOWN
Qt3=1+0.99*max(QtG)
=1

ACTION UP
Qt3=-1+0.99*max(QtP)
= -1+0.99*1.99
=-1+1.97
=0.97

ACTION LEFT
Qt3 =-1+0.99*max(Qt2)
== -1+0.99*1.99
=0.97
ACTION RIGHT
Qt3=0+0.99*max(Qt3)
=0.99

State S2

ACTION UP
Qt2=-1+0.99*max(Qt1)
= -1+0.99*2.97
=1.94

ACTION DOWN
Qt2=1+0.99*max(Qt3)
=1.99

ACTION RIGHT
Qt2=1+0.99*max(Qt3)
=1.99

ACTION LEFT
Qt2=1+0.99*max(Qt1)
=-1+0.99*2.97
=1.94

State S1B

ACTION UP
Qt1=0+0.99*max(Qt1)
= 0.99*2.97
=2.94

ACTION DOWN
Qt1=1+0.99*max(Qt2)
= 1+0.99*1.99
=2.97

ACTION LEFT
Qt1=1+0.99*max(Qt0)
=-1+0.99*3.94
=2.90

ACTION RIGHT

Qt1=1.0.99*max(QtP)=1+0.99*1.99=2.97

State P2

ACTION UP
QtP=0+0.99*max(QtP)=1.97

ACTION DOWN
QtP=1+0.99*max(Qt3)=1.99

ACTION RIGHT
QtP=0+0.99*max(QtP)=1.97

ACTION LEFT
QtP=-1+0.99*max(Qt1)=1.94

State S0

ACTION UP
Qt0=0+0.99*max(Qt0)=3.90

ACTION DOWN
Qt0=1+0.99*max(Qt1)=3.94

ACTION RIGHT
Qt0=1+0.99*max(Qt1)=3.94

ACTION LEFT
Qt0=0+0.99*max(Qt0)=3.90

STATE	UP	DOWN	RIGHT	LEFT
0	3.90	3.94	3.94	3.90
1	2.94	2.97	2.97	2.90
2	1.94	1.99	1.99	1.94
3	0.97	1	0.99	0.97
4	0	0	0	0

References:-

1)https://towardsdatascience.com/welcome-to-deep-reinforcement-learning-part-1-dqn-c3cab4d41b6b