CSE 574
Project 4
Report

Name- Ankit Kumar Sinha
UB Person#-50286874

Code Task:

1. Build Neural Network using Keras for Deep Learning model.
2. Implement Reinforcement Learning of Q learning.

Reinforcement Learning:

It is an aspect of machine learning where an agent learns to interact with the environment by performing the action using the rewards for that particular action of the individual state. Here agent works to maximize the rewards by performing optimal action. It tries to manage a balance between exploration and exploitation. Exploration is finding new things about the environment whereas exploitation is exploiting the already known path.

Q Learning:

Q Learning is the value based reinforcement learning. It has a Q table which acts like a lookup table which helps the agent to decide the optimal action to take when at a particular state.

Question from the code:

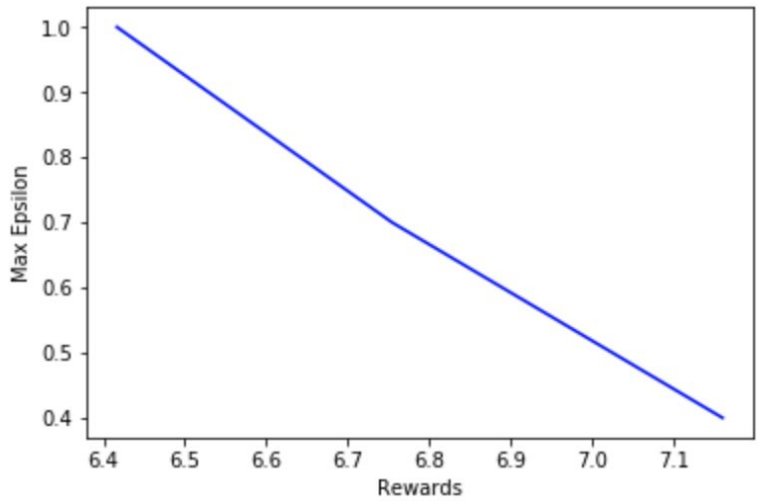
1. What part have you implemented?
   ->In this project, I have implemented a 3 layer of Neural Network, epsilon exponential decay formula and Q function.
2. What is their role in training the agent?
   -> For training of agent, Neural network acts as the brain of the model. It provides the output for the Q value for the action of a given state.
   Epsilon is the exploration rate, it decides the rate of steps to be done randomly i.e exploration by randomly choosing the actions. It also helps the model to converge and stop when the agent is confident about the Q values.
   Q Function has Q table updating implementation. This table helps the agent to reach the goal in an optimal way. This consist of state and the values with respect to the actions of each individual state. So this helps the agent decide which action to take on the particular state.
3. Can these snippets be improved and how will it influence the training of the agent?
   -> Yes these can be improved with the implementation of the better model to train on like more number of hidden layers this will cause more time taken by the agent to train, changing the hyperparameters will also improve the result but this varies from environment to environment.
4. How quickly your agent was able to learn?
   -> Agents starts learning when it starts exploring the environment. Initially, as the environment is new it starts to explore more to understand the new ways and then slowly

starts decreasing the exploration rate as it becomes confident about the environment. Then it starts exploiting the known ways to get the best way.
My agent was able to complete the learning in 766.89s.

Hyper Parameters Tuning:

1. Max Epsilon
   It is the maximum rate at which agent decides its action.
   a. Max Epsilon =1 :
      Time Elapsed: 782.69s
      Epsilon 0.061825467769376214
      Episode Reward Rolling Mean: 6.416573975533517
   b. Max Epsilon =0.7 :
      Time Elapsed: 801.34s
      Epsilon 0.05739566289723348M
      Episode Reward Rolling Mean: 6.754104682695592

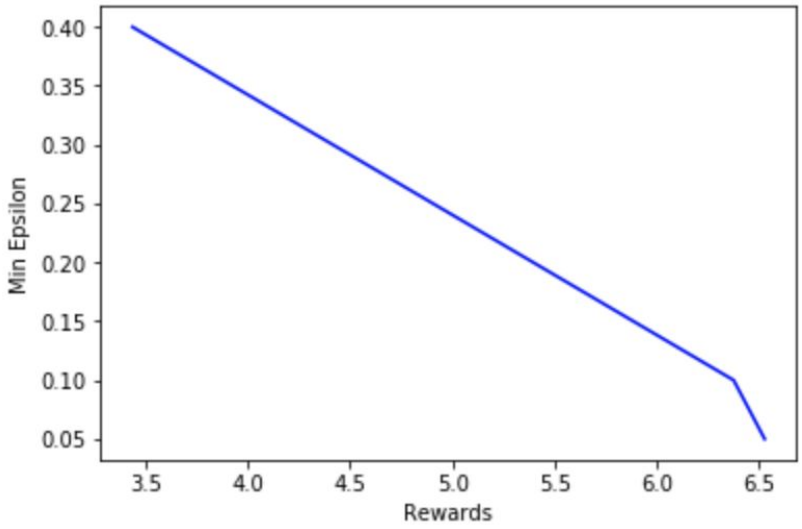
   c. Max Epsilon =0.4 :
      Time Elapsed: 807.92s
      Epsilon 0.055346799250190035
      Episode Reward Rolling Mean: 7.159475432501786



So as we can observe from the above results that as the max epsilon value decreases, the decrease in epsilon also stops late, so exploration stops late and mean reward increases. In this, as max epsilon decreases more time is taken.

2. Min Epsilon
   a. Min Epsilon= 0.4:
      Time Elapsed: 915.14s
      Epsilon 0.5075689907855368
      Episode Reward Rolling Mean: 3.436292868723201

b. Min Epsilon= 0.1:
   Time Elapsed: 865.32s
   Epsilon 0.1065258127352395
   Episode Reward Rolling Mean: 6.374586993522072
c. Min Epsilon= 0.05:
   Time Elapsed: 782.87s
   Epsilon 0.0599756756297529
   Episode Reward Rolling Mean: 6.526367786393517

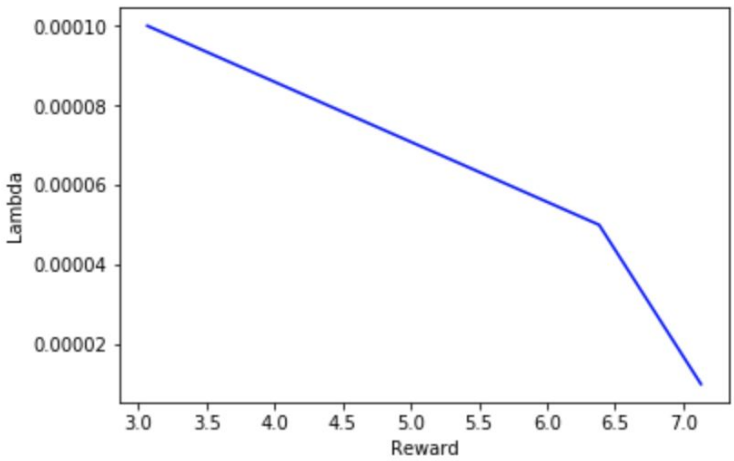


So as we can observe from the above results that as the min epsilon value decreases, the decrease in epsilon stops late, so exploration stops late and mean reward increases. In this, as min epsilon decreases time also decreases.

3. Lambda
   a. LAMBDA=0.0001
      Time Elapsed: 802.74s
      Epsilon 0.40688085429129356
      Episode Reward Rolling Mean: 3.064993368023671

   b. LAMBDA = 0.00005
      Time Elapsed: 772.69s
      Epsilon 0.06075436785984266
      Episode Reward Rolling Mean: 6.382205897357412

   c. LAMBDA = 0.00001
      Time Elapsed: 758.18s
      Epsilon 0.050167321525463854
      Episode Reward Rolling Mean: 7.129884705642281

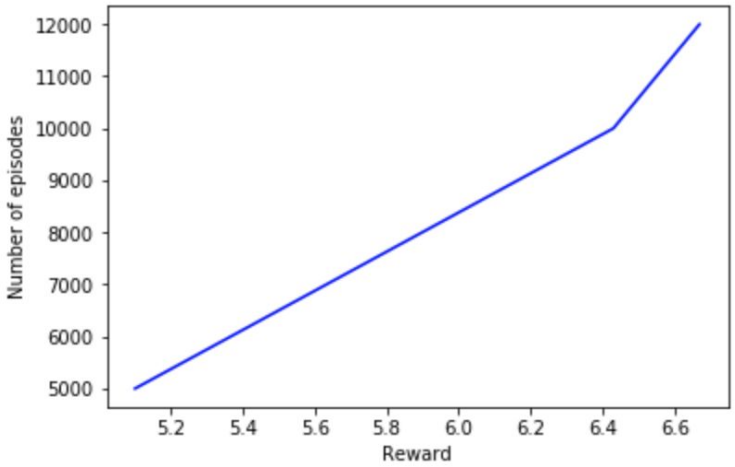


So as we can observe from the above results that as lambda values increases, epsilon value increases. So exploration stops late and rewards increases.

4. Number of Episodes
    a. episode-5000
    Time Elapsed: 402.37s
    Epsilon 0.14242633830260615
    Episode Reward Rolling Mean: 5.101437200583212

    b. Episode 10000
    Time Elapsed: 789.03s
    Epsilon 0.060776436927020436
    Episode Reward Rolling Mean: 6.430058157330885

    c. Episode 12000
    Time Elapsed: 947.16s
    Epsilon 0.054658586074438055
    Episode Reward Rolling Mean: 6.669180577917126

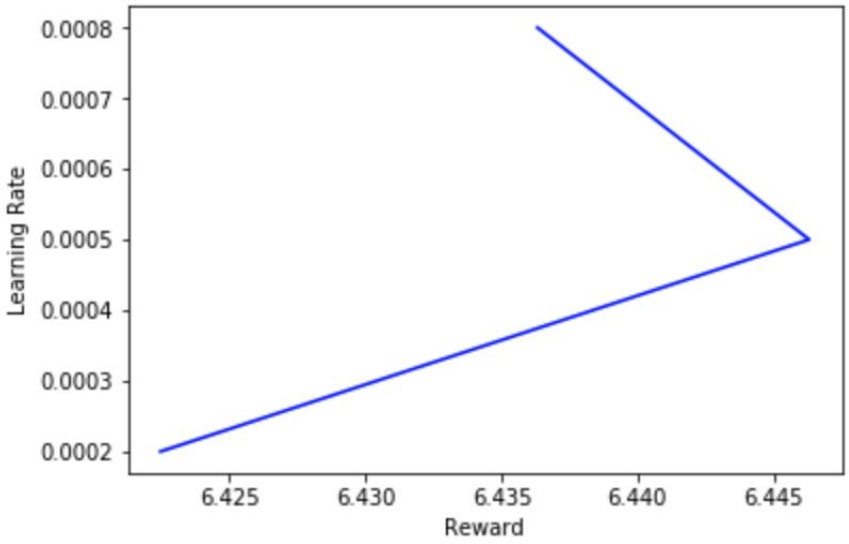


So as we can observe from the above result that as episodes values increases, the agent gets trained more which leads to more rewards. As the agent trains more, it takes more time to complete.

5. Learning Rate:
    a. Learning rate=0.0002
       Time Elapsed: 784.27s
       Epsilon 0.06085683265297832
       Episode Reward Rolling Mean: 6.422463956487326

    b. Learning rate=0.0005
       Time Elapsed: 891.94s
       Epsilon 0.06085632954743872
       Episode Reward Rolling Mean: 6.446312458762146

    c. Learning rate=0.0008
       Time Elapsed: 897.97s
       Epsilon 0.06076513651783674
       Last Episode Reward: 7
       Episode Reward Rolling Mean: 6.436324178642623

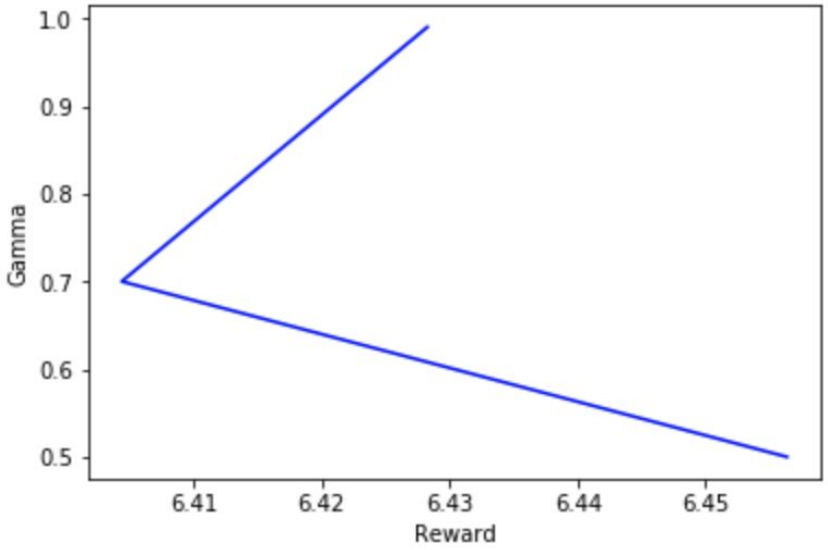


So as we can observe from the above result that as the learning rate increases, the mean reward increases but if it goes big the result start converging and starts missing data which makes method go wrong.

6. Gamma:
    a. Gamma=0.5
       Time Elapsed: 926.35s
       Epsilon 0.06717865318451348
       Episode Reward Rolling Mean: 6.456346127659447

b. Gamma=0.7
   Time Elapsed: 916.31s
   Epsilon 0.06063264567587345
   Episode Reward Rolling Mean: 6.404368323479238

c. Gamma=0.99
   Time Elapsed: 782.98s
   Epsilon 0.0608642356897237
   Episode Reward Rolling Mean: 6.428237529758723



Gamma is the discount rate, normally the agent tries to find the nearby max Q values so it won't try to explore if it gets values nearby. So gamma drops the Q values of the same element so that agents don't go to the same state always and tries to explore more. Hence as we increase the lambda values and the mean reward increases. But this is not the case every time, relationship on rewards also depends on the environment as the graph above showcase the effect going inverse also.

Result:
The original result after completing the snippet is:
Episode 9900
Time Elapsed: 766.89s
Epsilon 0.060802331436627444
Last Episode Reward: 8
Episode Reward Rolling Mean: 6.426793184368942

Note: In task 3 of Q function implementation, as in the explanation part Q(st,at;Θ) .ie it's for the current state, so i have written np.amax(t) but according to notes it should have been next state i.e t1=q_vals_next and np.amax(t1).

Writing Task:

Q1. 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.
Ans If the agent chooses the action that maximizes the Q-value then the agent always search for maximum Q value and if it gets this in the nearby states then it won't be exploring more and will be more into exploitation. The agent might get more short-term rewards but the long final reward might not be optimal as it won't be exploring more, also it won't converge optimally and the time taken by the agent to maximize the Qvalue might also be more.
These are the few following ways to force the agent to explore:

a. Decay Epsilon Greedy strategy:
   This is the exploration rate of the agent, it is set to a high value near 1 initially so that the agent is forced to explore more as it is new to the environment and slowly by decaying the exploration rate so that it balances between exploration and exploitation.
b. Parameters Space Noise:
   In this, the parameters of the neural network policy are added with adaptive noise. This adds noise directly to the agent parameters altering the type the decision agent makes every time so agent tries to explore more. It makes the agent exploration more consistent across different timesteps
c. Action space Noise:
   In this, after the model is trained a noise is added to change the likelihood of each action. This changes the decision everytime the agents make, so it tries to explore more instead of taking the same action. It makes the exploration unpredictable as the action itself is changed randomly.
d. Gamma:
   It is also known as the discount rate. In Q learning if the agent finds the more Q values around it then agent won't go beyond for exploration and will run infinitely around. Gamma decreases the Q value of the explored states so that the agent tries to explore far away states.

Q2. Provide Q table for the given steps and calculation.
Ans Steps for calculating the q values of the states:

a. Initialize the Q table with zeros. Start from the last state i.e the goal and proceed in a bottom-up style. Use the Equation to update the Q table values.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	0	0	0
1(S_1)	0	0	0	0
2	0	0	0	0

3	0	0	0	0
4(S_2)	0	0	0	0
5(S_3)	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

The equation for updating of Q Table:

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

In this equation, r =reward for that action

γ =gamma

$max_a Q(s_{t+1}, a_t)$=when the agent takes an action on the current state what is the max value from that action the agent can get.

b. Choose an action for the state.
   In this, we start from the goal state and find the Q values for each action of it using the above equation and update in Q table.
c. Repeat this process until to find the optimal path and till the agent becomes confident of it.

Calculation Step:

a. Calculate the Q values for every action of S4:
   i. Q(8, Up)=0+0.99(0)=0 since now Q table has all value as 0 for all action.
   ii. Q(8, Down)=0+0.99(0)=0 since now Q table has all value as 0 for all action.
   iii. Q(8, Left)=0+0.99(0)=0 since now Q table has all value as 0 for all action.
   iv. Q(8, Right)=0+0.99(0)=0 since now Q table has all value as 0 for all action.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	0	0	0
1(S_1)	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4(S_2)	0	0	0	0
5(S_3)	0	0	0	0

6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

b. Calculate the Q values for every action of S3:
   i. Q(5, Up)=-1+0.99(0)=-1 since now Q table has all value as 0 for all action.
   ii. Q(5, Down)=1+0.99(0)=1 since now Q table has all value as 0 for all action.
   iii. Q(5, Left)=-1+0.99(0)=-1 since now Q table has all value as 0 for all action.
   iv. Q(5, Right)=0+0.99(0)=0 since this action cannot be done.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	0	0	0
1(S_1)	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4(S_2)	0	0	0	0
5(S_3)	-1	1	-1	0
6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

c. Calculate the Q values for every action of S2:
   i. Q(4, Up)=-1+0.99(0)=-1 since now Q table has all value as 0 for all action.
   ii. Q(4, Down)=1+0.99(0)=1 since now Q table has all value as 0 for all action.
   iii. Q(4, Left)=-1+0.99(0)=-1 since now Q table has all value as 0 for all action.
   iv. Q(4, Right)=1+0.99(1)=1.99 since S3 has the max value of 1.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	0	0	0
1(S_1)	0	0	0	0

2	0	0	0	0
3	0	0	0	0
4(S_2)	-1	1	-1	1.99
5(S_3)	-1	1	-1	0
6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

d. Calculate the Q values for every action of S1:
   i. Q(1, Up)=0+0.99(0)=0 since this action cannot be done.
   ii. Q(1, Down)=1+0.99(1.99)=2.97 since S2 has the max value of 1.99.
   iii. Q(1, Left)=-1+0.99(0)=-1 since now Q table has all value as 0 for all action.
   iv. Q(1, Right)=1+0.99(0)=1 since now Q table has all value as 0 for all action.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	0	0	0
1(S_1)	0	2.97	-1	1
2	0	0	0	0
3	0	0	0	0
4(S_2)	-1	1	-1	1.99
5(S_3)	-1	1	-1	0
6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

e. Calculate the Q values for every action of S0:
   i. Q(0, Up)=0+0.99(0)=0 since this action cannot be done.
   ii. Q(0, Down)=1+0.99(0)=0 since now Q table has all value as 0 for all action.
   iii. Q(0, Left)=0+0.99(0)=0 since this action cannot be done.
   iv. Q(0, Right)=1+0.99(2.97)=3.94 since S1 has the max value of 2.97.

State/ Action	UP	DOWN	LEFT	RIGHT
0 (S_0)	0	1	0	3.94
1(S_1)	0	2.97	-1	1
2	0	0	0	0
3	0	0	0	0
4(S_2)	-1	1	-1	1.99
5(S_3)	-1	1	-1	0
6	0	0	0	0
7	0	0	0	0
8(S_4)	0	0	0	0

So finally the update Q table is:

State/ Action	UP	DOWN	LEFT	RIGHT
S_0	0	1	0	3.94
S_1	0	2.97	-1	1
S_2	-1	1	-1	1.99
S_3	-1	1	-1	0
S_4	0	0	0	0

References:

1. https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe
2. https://medium.freecodecamp.org/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc