Project 4 Report

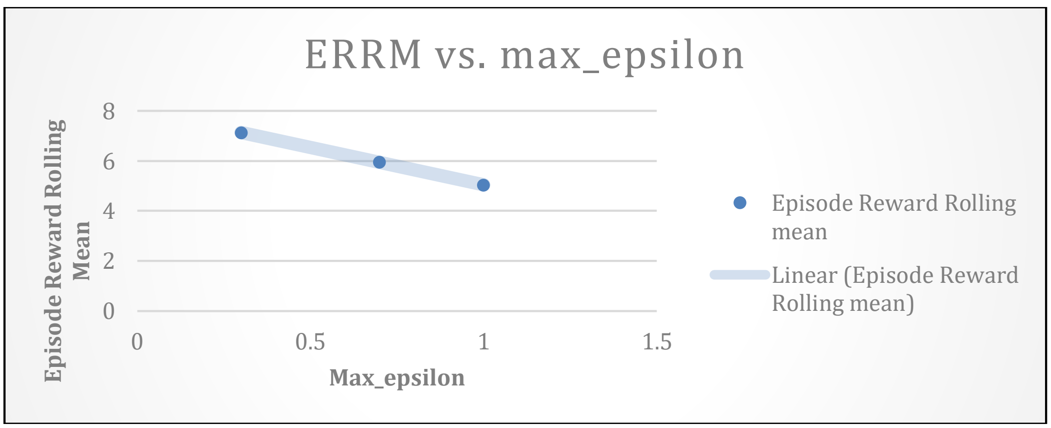
Redwan Ibne Seraj Khan(UBITname: rikhan)
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY-14228
rikhan@buffalo.edu

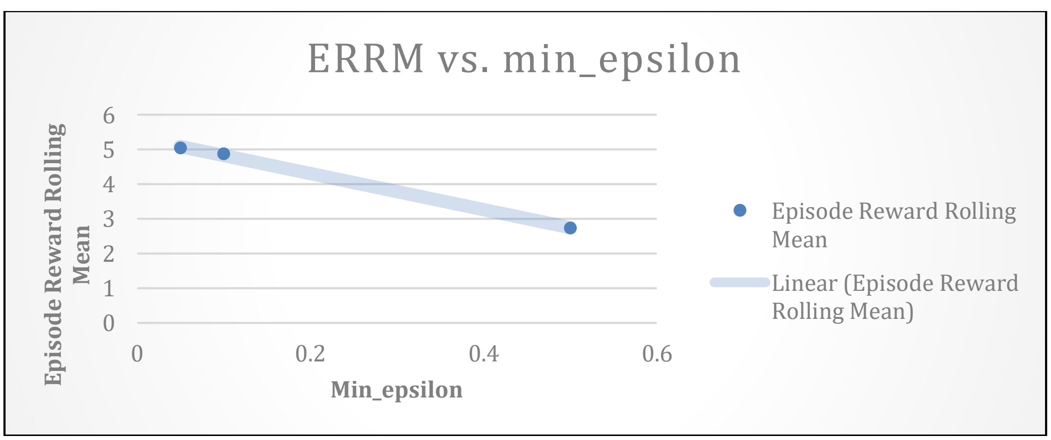
Abstract

In this project it was required to contruct a deep neural network model that is to be used in reinforcement learning. Moreover, an exponential decay function and a Q-function was required to be written. After writing all the required functions different hyperparameters were changed to achieve higher accuracy.

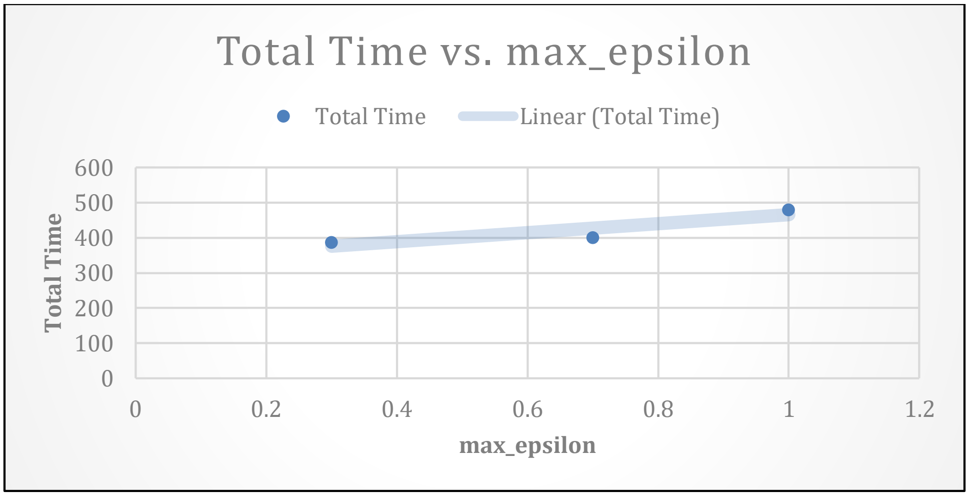
1 Hyperparameter Tuning

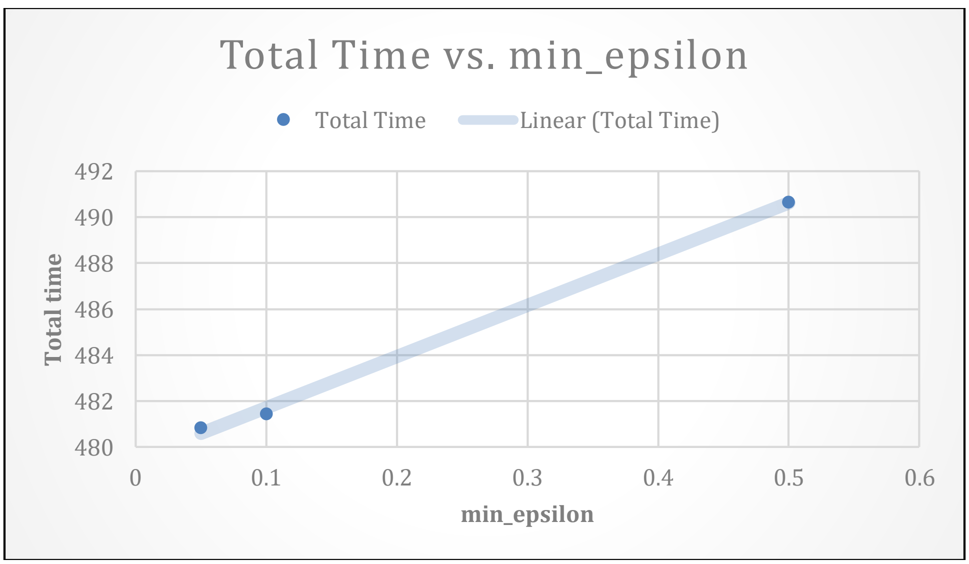
1.1 Epsilon max/min





It was seen that as the value of max_epsilon decreased the episode reward rolling mean increased and as the value of min_epsilon increased the reward mean decreased.

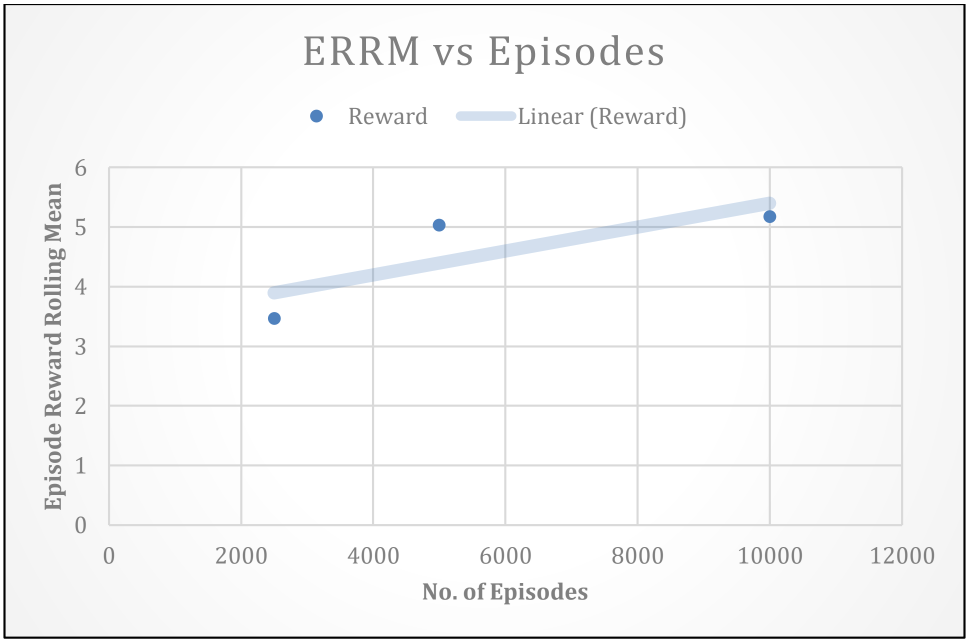


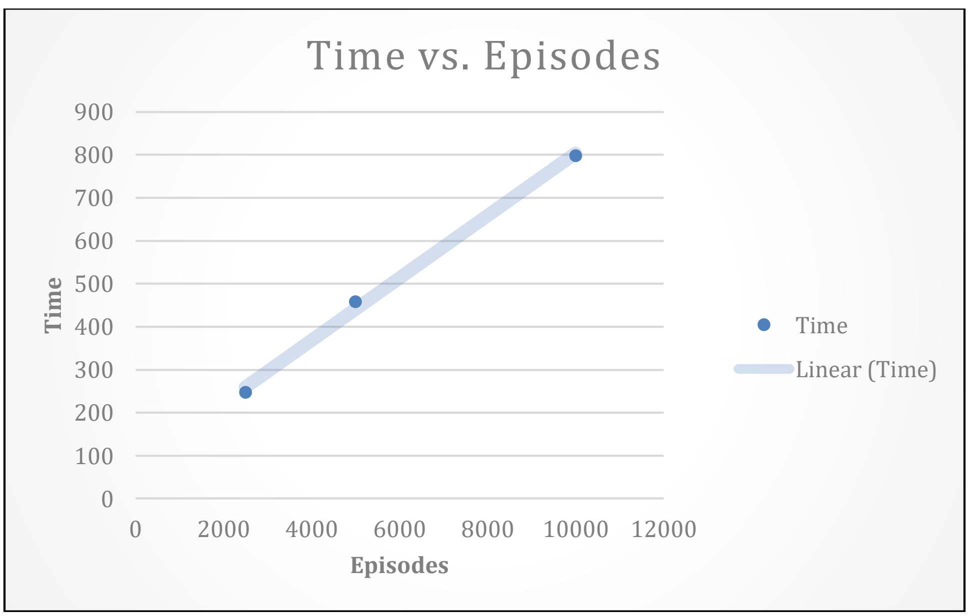


It was seen that the total time to complete the episodes took longer when the value of max_epsilon was increased. It was the same for min_epsilon too.

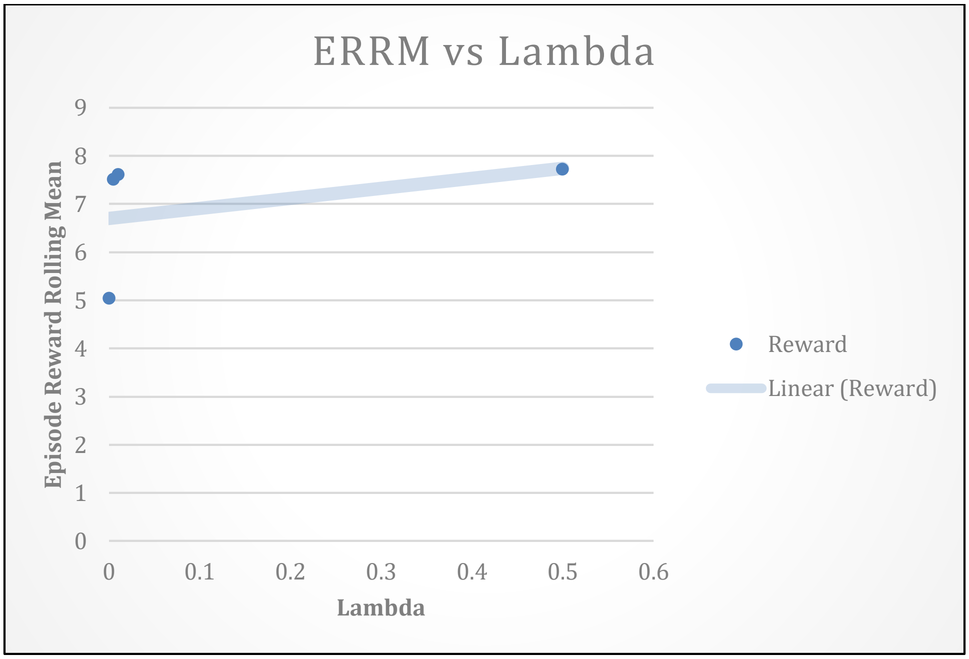
1.2 Number of Episodes

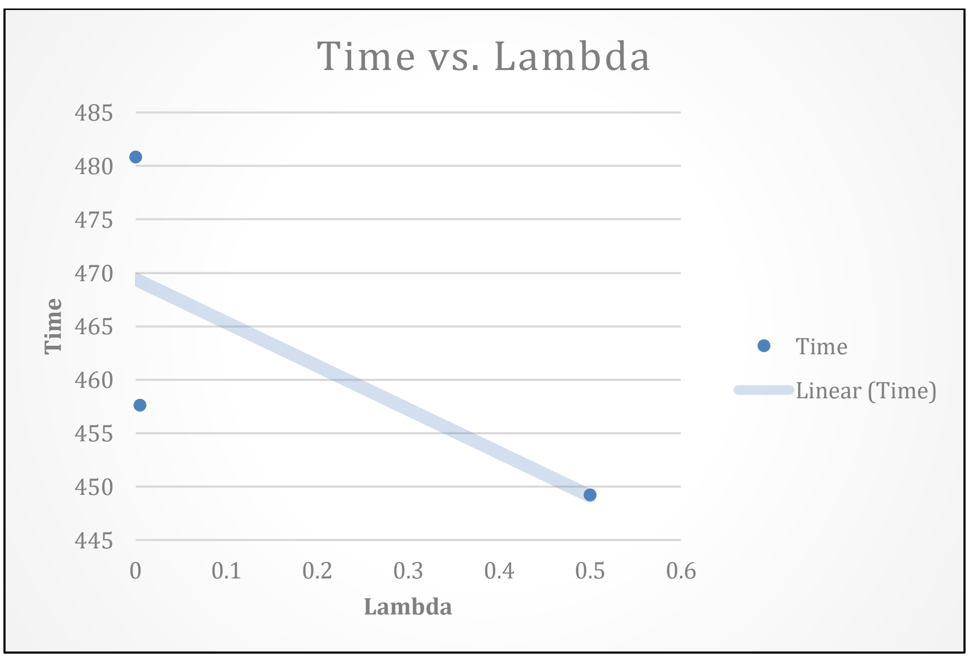
It was seen that as the number of episodes were increased the episode reward rolling mean also increased and the time also increased.





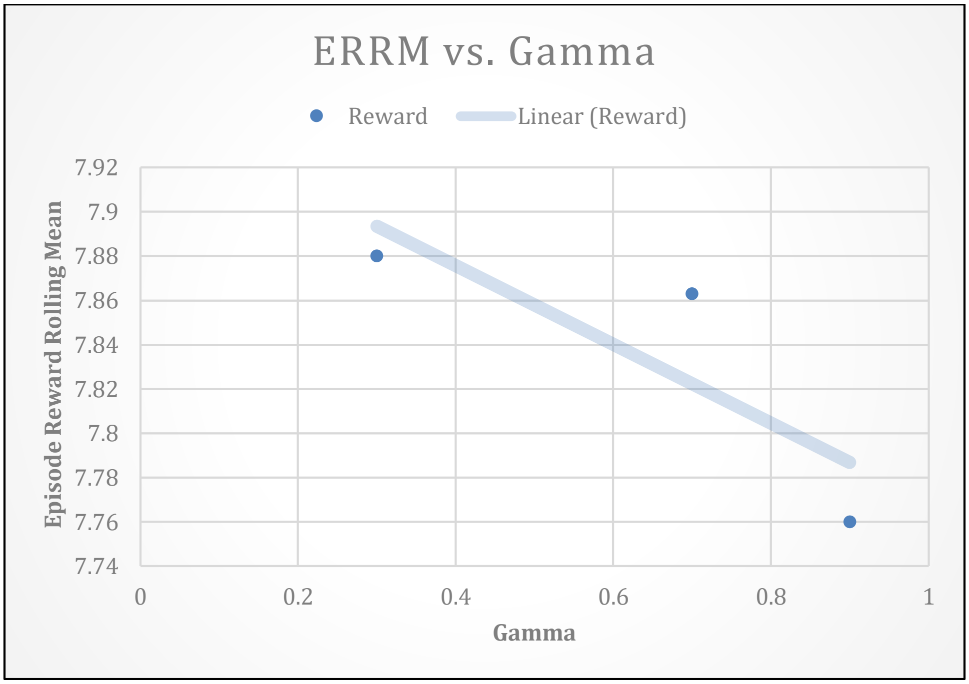
1.3 Lambda value

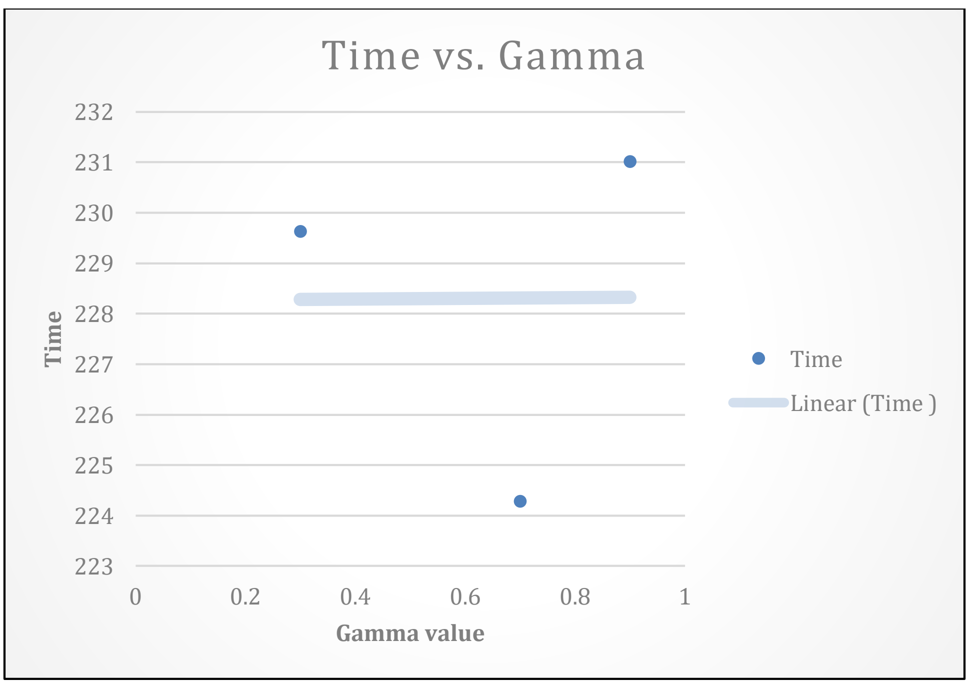




It was seen that the episode reward mean does not change much when value of lambda was changed. If there is some change it increases with the increase in the value of lambda. As lambda is increased the time taken decreases.

1.4 Gamma value





It was seen that as the value of gamma increases the episode reward rolling mean decreases. The time however remains almost constant.

QUESTIONS

Q.1 What parts have you implemented?

- Built a multi-layered neural network using the Keras library
- Coded an exponential-decay formula for epsilon
- Implemented the Q-function

Q.2 What is their role in training the agent?

Role of Multilayered Neural Network

This is the actual brain of the model. Without the neural network layers our agent would not have been able to learn by reducing its mistakes through different propagations. The weights needed to updated for our agent to learn while rewarding and penalizing the agent. The agent needs to learn to chooses such an action that will return the highest Q-value. The neural network model outputs a vector of Q-values for each action possible in the given state.

Role of Exponential Decay Formula

The exponential decay formula is actually the rate at which the agent reduces the number of random actions. At first the agent randomly selects its action by a certain percentage. We want the agent to try all kinds of things before it starts to see patterns. After some time, the agent will predict the reward value based on its current state and pick the action that will give the highest rewards. To decrease the randomness in choosing actions the exponential decay formula is really essential.

Role of Q function

We need a table where we will keep the maximum expected future reward for each action in a particular state. Based on this table, the agent will know which action to take when it comes to a particular state. The Q-function helps us to build that table.

Q.3 Can these snippets be improved and how it will influence the training the agent?

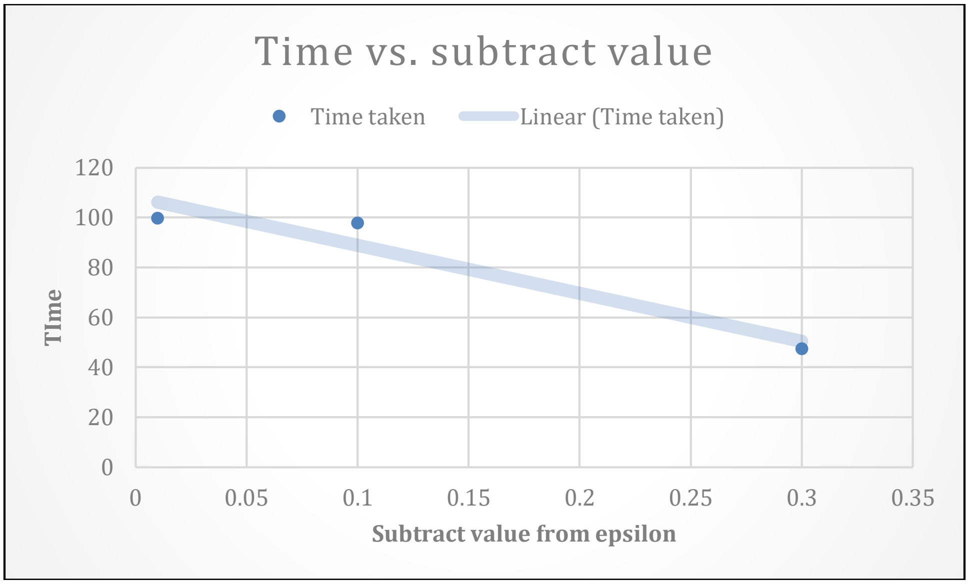
Yes, these snippets could be improved.

Firstly, we could add a few more layers and increase the number of nodes in each layer.

With the model that was required to be built having 2 dense layers with 128 nodes each, the agent could reach the goal only once. The value of max_epsilon was set to be 1, min_epsilon to be 0.05 and lambda to be 0.00005. Although it took less time (546 s) for 5000 episodes, this model is clearly not a good one.

I built another neural network having 3 dense layers. The first layer had 1024 nodes, the second one 512 nodes and the third one had 128 nodes. It was seen that although it took a little more time (623 seconds), the agent was able to reach the goal 3 times in 5000 episodes. The value of max_epsilon was set to be 1, min_epsilon to be 0.05 and lambda to be 0.00005.

Secondly, instead of an exponential decay formula we could decrease the value of epsilon in every step by values like 0.001.



It can be seen that the agent learns much faster if epsilon is decreased by a constant value and the value of subtraction gradually increases.

Q.4 How quickly your agent were able to learn?

My agent was able to learn in approximately 99s. At that time I used max_epsilon = 1, min_epsilon = 0.05 ,lambda = 0.05 and gamma = 0.3. It took about 1100 episodes for my agent to get to the goal.

WRITING TASK

Q.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 the agent would be able to learn how to get to the goal faster. It does not need to learn using an epsilon decay function or by other means. There is no need to construct a Q-table as the agent already knows which action would give it the most reward.

The goal of reinforcement learning is finding the best solution rather than the easiest solution. So it is beneficial to force the agent to explore before finding the best path.

Two ways in which I can force the agent to explore are –

1. Using an epsilon function that would decrease the value by very small steps.

Instead of using $e^{-\lambda|S|}$ we could use $2^{-\lambda/S}$. The value of e = 2.178 decreases the number of explorations. When we used 2 it decreased it. Here λ would be total number of steps passed and S would determine after every how many steps would the model be stored and the agent would become less random. In this way through following this greedy approach would be able to force our agent to explore.

2. Decreasing value of epsilon by a very small constant number

The main idea is to make sure value of epsilon decreases in very small steps. So if we decrease the value of epsilon by a very small number say, 0.000001 it would force the agent to explore more.

Q.2 Calculate Q-value for the given states and provide all the calculation steps.

On next page.

Writing Task 2:

Some points to be noted.

(i) Agent moves from upper-left corner of grid to the goal at the lower right corner. So at a definite state and position, the maximum Q-value would come from moving down or right.

(ii) $Q(S_{x.y.z}, a)$ would mean that at x state in the (y,z) coordinate.

Starting calculating from last state.

$Q(S_{4.3.3}, up) = Q(S_{4.3.3}, down) = Q(S_{4.3.3}, left) = Q(S_{4.3.3}, right) = 0$

$Q(S_{3.2.3}, down) = 1 + 0.99 \times max_a Q(S_{4.3.3}, a) = 1 + 0.99 \times 0 = 1$

$Q(S_{3.2.3}, right) = 0 + 0.99 \times max_a Q(S_{3.2.3}, a)$

$= 0 + 0.99 \times Q(S_{3.2.3}, down)$

$= 0 + 0.99 \times 1 = 0.99$

$\therefore Q_{max}(S_{3.2.3}, a) = 1$

$Q(S_{3.2.3}, up) = -1 + 0.99 \times max_a Q(S_{3.1.3}, a)$

$= -1 + 0.99 \times Q(S_{3.1.3}, down)$

$= -1 + 0.99 \times (1 + 0.99\ max_a Q(S_{3.2.3}, a))$

$= -1 + 0.99 \times (1 + 0.99 \times 1)$

$= -1 + 0.99 \times 1.99$

$= 0.9701$

$Q(S_{3.2.3}, left) = -1 + 0.99 \times max_a Q(S_{2.2.2}, a)$

$Q(S_{2.2.2}, \text{right}) = 1 + 0.99 \times \max_a Q(S_{3.2.3}, a)$

$= 1 + 0.99 \times Q(S_{3.2.3}, \text{down})$

$= 1 + 0.99 \times 1$

$= 1.99$

$Q(S_{2.2.2}, \text{down}) = 1.99$ [$\because$ symmetric steps with $Q(S_{2.2.2}, \text{right})$]

$\therefore Q_{\max_a}(S_{2.2.2}, a) = 1.99$

$\therefore Q(S_{3.2.3}, \text{left}) = -1 + 0.99 \times \max_a Q(S_{2.2.2}, a)$

$= -1 + 0.99 \times 1.99$

$= 0.9701$

$Q(S_{2.2.2}, \text{left}) = -1 + 0.99 \times \max_a Q(S_{2.2.1}, a)$

$= -1 + 0.99 \times Q(S_{2.2.1}, \text{right})$

$= -1 + 0.99 \times (1 + 0.99 \times Q_{\max_a}(S_{2.2.2}, a))$

$= -1 + 0.99 \times (1 + 0.99 \times 1.99)$

$= 1.94$

Alternatively, we could have used,

$Q(S_{2.2.1}, \text{down}) = Q(S_{1.1.2}, \text{right}) = 2.9701$

$Q(S_{2.2.2}, \text{up}) = -1 + 0.99 \times \max_a Q(S_{1.1.2}, a)$

Now,

$Q(S_{1.1.2}, \text{right}) = 1 + 0.99 \times \max_a Q(S_{1.1.3}, a)$

$= 1 + 0.99 \times 1.99$ [$\max_a Q(S_{1.1.3}, a)$ was calculated when calculating $Q(S_{3.2.3}, \text{up})$]

$= 2.9701$

$Q(S_{1.1.2}, \text{down}) = 1 + 0.99 \times \max_a Q(S_{2.1.2}, a)$

$= 1 + 0.99 \times 1.99$

$= 2.9701$

$\therefore Q_{max_a}(S_{1.1.2}, a) = 2.9701$

$\therefore Q(S_{2.2.2}, up) = -1 + 0.99 \times max_a Q(S_{1.1.2}, a)$

$= -1 \times 0.99 \times 2.9701$

$= 1.94$

$Q(S_{1.1.2}, up) = 0 + 0.99 \times max_a Q(S_{1.1.2}; a)$

$= 0 + 0.99 \times 2.9701$

$= 2.9403$

$Q(S_{1.1.2}, left) = -1 + 0.99 \times max_a Q(S_{0.1.1}, a)$

$Q(S_{0.1.1}, right) = 1 + 0.99 \times max_a Q(S_{1.1.2}, a)$

$= 1 + 0.99 \times 2.9701$

$= 3.94$

$Q(S_{0.1.1}, down) = 3.94$ [since symmetric with $Q(S_{0.1.1}, right)$]

$\therefore Q_{max}(S_{0.1.1}, a) = 3.94$

$Q(S_{0.1.1}, up) = 0 + 0.99 \times max_a Q(S_{0.1.1}, a)$

$= 3.90$

$Q(S_{0.1.1}, left) = 0 + 0.99 \times max_a Q(S_{0.1.1}, a)$

$= 3.90$

$\therefore Q(S_{1.1.2}, left) = -1 + 0.99 \times max_a Q(S_{0.1.1}, a) = -1 + 0.99 \times 3.94$

$= 2.9006$

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