CSE574 Introduction to Machine Learning

Project 4

Tom and Jerry in Reinforcement Learning

Project Report

Submitted By

Karan Manchandia

Abstract

This project aims to apply the concept of reinforcement learning with deep learning. Reinforcement Learning is a type of machine learning technique that makes an agent learn in an interactive environment by using feedback from its own actions and experiences. Deep Learning is a subset of machine learning that uses a multilayer artificial neural network, in order to deliver great accuracies. Our task in this project is to use reinforcement learning combined with deep learning to teach an agent to navigate in a grid world environment. For our project the agent is Tom, a cat and Tom is catching Jerry, a mouse. In the game, the task for our agent, Tom is to find the shortest path to reach the goal, Jerry. To solve this problem, we will apply deep Q-learning algorithm (DQN).

There are also two writing tasks and the solution to which are included in this report. The writing task 1 presents an exploration / exploitation problem and asks to suggest two ways to force the agent to explore. In the second writing task, we are required to calculate Q-values of all the states in a 3X3 grid.

Introduction:

Reinforcement Learning:

Reinforcement Learning is a machine Learning Technique that is concerned to make a software agent (Tom in our case) learn in an interactive environment, so as to maximize some cumulative reward. Reinforcement Learning uses rewards and punishments as signals for good and bad behaviours. It is very different from supervised learning techniques, where an input is provided to the agent in the form of a correct set of actions to be followed to complete a task.

What is Exploration vs Exploitation trade-off in Reinforcement Learning?

In order to build an optimal policy for the reinforcement learning task, the agent faces the dilemma of exploring new states while maximizing its rewards at the same time i.e. agent faces a dilemma to choose between exploration and exploitation. This is called Exploration vs Exploitation trade-off. Actually, for completing a task in the best possible way, there should a balance between exploration and exploitation.

Figure (Source: Internet)

Markov Decision Processes:

Markov Decision Processes are defined as mathematical frameworks to illustrate an environment in reinforcement learning. An MDP is a five tuple (S, A, P_a, R_a, γ). Here, S is a finite set of states and A is a finite set of actions. P_a is the probability that action *a* in state *s* at time *t* will lead to *s'*at time *t+1*. P_a is given by:

$$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$$

R_a is the award received after moving from state s to s'. Discount factor, γ controls the importance of future rewards. Its value can be greater than or equal to zero and less than one.

What is the relation between discount and discount factor (γ) and what does their relation explains?

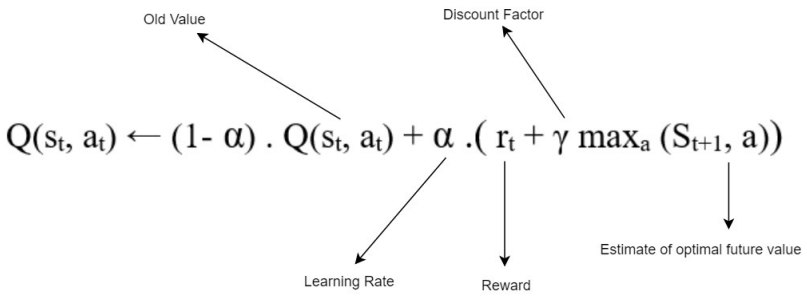
The following are the relation between γ and actual discount:

- The larger the value of the discount factor, the smaller the discount. This means that the learning agent cares more about long term reward. So, the agent will give more preference to exploration then exploitation.

- On the other hand, the smaller the discount factor, the bigger is the discount. This means the agent cares more about the short term reward. So, the agent will give more preference to exploitation then exploration.

Q-Learning:

Q-learning is a model free approach which revolves around updating Q-values, which denote the value on doing action *a* in state *s*.



Why do we need to use Deep Q-network with reinforcement learning for this project?

The q-learning method in reinforcement learning is simple to implement but it lacks generality as it does not have the ability to estimate values for unseen states. This can be overcome by using an advanced algorithm such as Deep Q-Networks, which uses a neural network to estimate q-values.

Coding Tasks:

Explanation of the problem:

- In the coding task, we are given with a 5X5 grid world environment. So, in all 25 states are possible. Each state can be represented by a coordinate (0, 0), (0, 1)…..(4, 4).
- The goal Jerry is set to an initial position (0, 0) and the agent Tom is dynamically changing positions.
- Our goal is to make our agent learn the shortest path to the goal. The agent has, in all 100 steps to receive reward as large as possible.

Coding Task 1 (Build a 3 layer neural network):

- **Code for this task is provided in main.ipynb.**
- **What we have implemented in this task?**

  In this task, I implemented a neural network using the Keras library. The network has 2 hidden layers with 128 neurons each and with RELU activations except for the last layer

which has a linear activation. We have kept activation function for the output layer as linear because it will return real values.

- **What is the role of this task in training the agent?**

  The role of the neural network is to estimate the q-values of states that it is provided as input. The model is then retrained on data generated as the game is played.
- **Can this snippet be improved and how it will influence the training the agent?**

  Yes, this snippet can be improved by adding more neural network layers. Mostly, it is seen that more layers in the neural network model lead to more accuracy.
- The Input dimensions for the first hidden layer is equal to the size of your observation space (state_size) and the output dimension is equal to the size of the action space (action_size).

Coding Task 2 (Implement exponential-decay formula for epsilon):

- **Code for this task is provided in main.ipynb.**
- **What we have implemented in this task?**

  The 2nd coding task is implementing exponential decay formula for epsilon. This is done to prioritize exploration at the beginning of training that is the possibility of taking random actions is high. Epsilon is then reduced as we go and this prioritizes exploitation and choosing the actions with the highest q values.
- **What is the role of this task in training the agent?**

  Implementing the exponential decay formula for epsilon helps us to implement the epsilon greedy technique, which is required to create a balance between exploration and exploitation. Using this formula, the agent will randomly select its actions by a certain percentage called “exploration rate”.
- **Can this snippet be improved and how it will influence the training the agent?**

  Yes, this implementation can be improved by implementing the adaptive-epsilon-greedy method. This method is based on classic epsilon-greedy but allows the value of epsilon to vary in a controlled way throughout the execution. The adaptive-epsilon-greedy is a modified version of what we have implemented. It leads to better balancing between exploration and exploitation in reinforcement learning.

Coding Task 3 (Implement Q-function):

- **Code for this task is provided in main.ipynb.**
- **What we have implemented in this task?**

The 3rd coding task we are implementing the Q-function. The formula for calculating the q-value is called Q-function, and it is shown below. Q-function accepts a state and action as parameters and returns the expected total reward. The same formula has been implemented in coding task 3.

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

- **What is the role of this task in training the agent?**

  Q-function represents the quality of certain action given the state. That is the Q-function tells the agent how good of a choice is taking action *a* when at state *s*.

- **Can this snippet be improved and how it will influence the training the agent?**

  No, Q-function is a formula. Hence, it cannot be improved further.

- **A general question for the three coding task:**

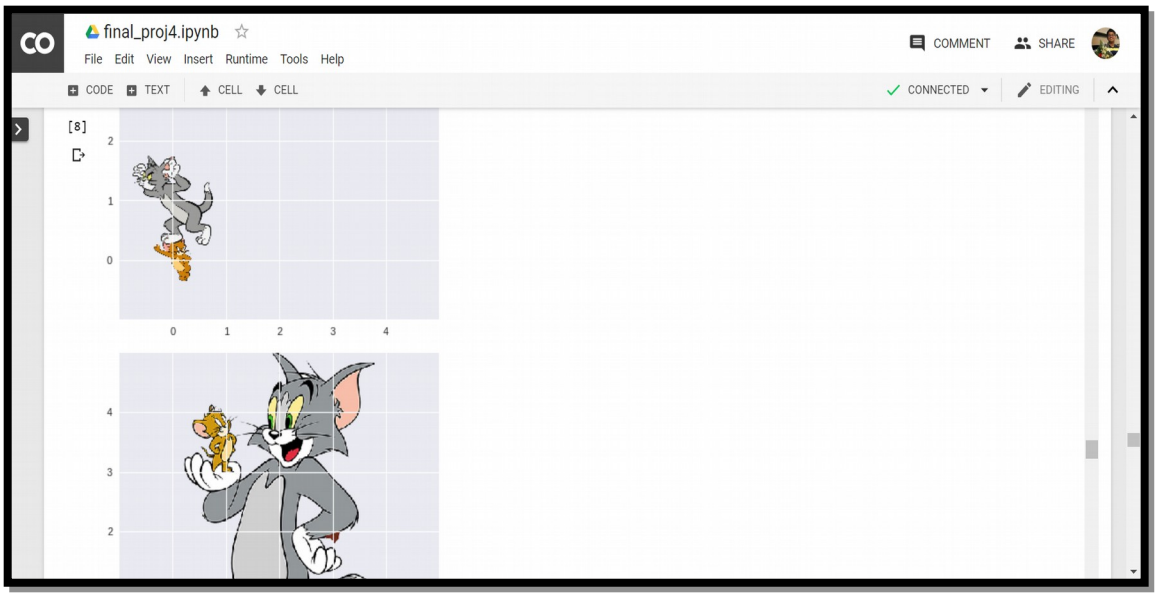
  How quickly your agent was able to learn and how the training time could be affected?

  The agent took 15 minutes to train on a CPU.

  The hyperparameters of the neural network could be tuned such as the number of layers or neurons in each layer and this would affect training time and time taken to reach convergence. Also, another technique for tackling the exploration vs exploitation problem could be employed. This would affect the time to convergence.

Observations, Results, Graphs and Screenshots:

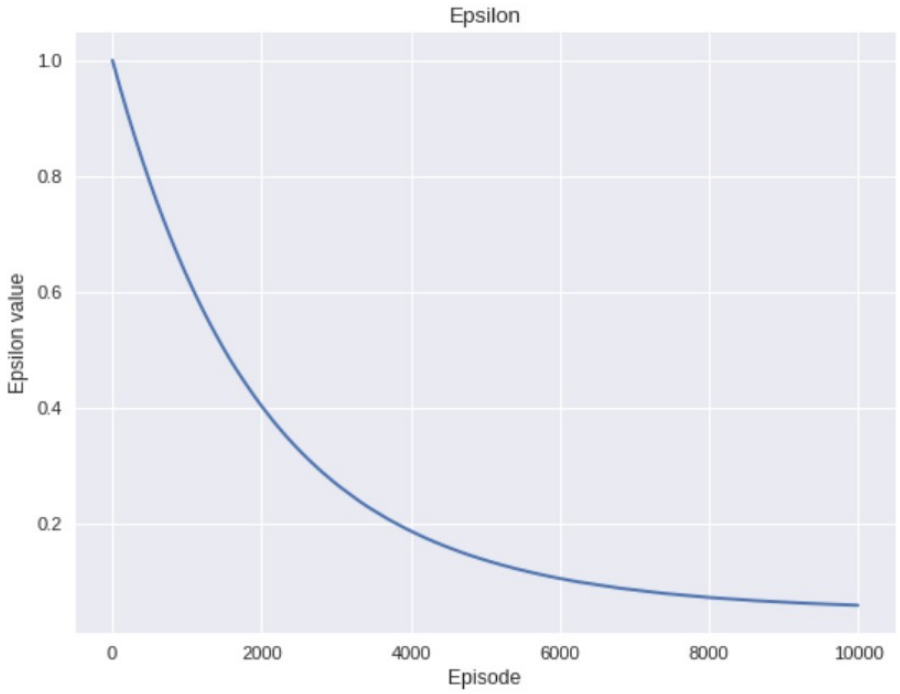
Screenshots:



Results:

Episode	Time Elapsed	Epsilon	Last Episode Reward	Episode Reward Rolling Mean
0	3.55	0.999	-2	none
1000	88.33	0.626	4	1.77
2000	178.03	0.403	7	3.02
3000	268.46	0.249	6	4.04
4000	352.54	0.187	7	4.64
5000	449.66	0.137	8	5.12
6000	531.09	0.106	6	5.48
7000	610.28	0.086	7	5.74
8000	689.71	0.073	8	5.96
9000	769.52	0.065	8	6.14
9900	843.35	0.060	7	6.27

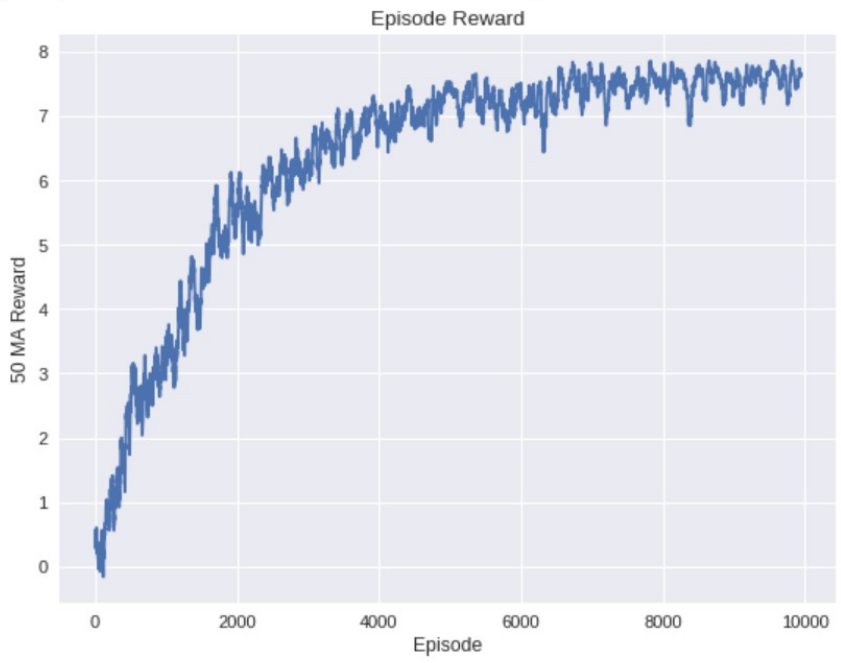
The graph between Epsilon Value and Episode is plotted and is shown below:



Explanation:

It can be clearly seen that there is an inverse relation between the number of Episodes and Epsilon value. This is because, initially as the episodes are low, we want the agent to explore, so the epsilon value is kept high. But as the number of episodes increases we want the agent to exploit, so the epsilon value decreases with increasing episodes. This justifies the inverse relation between epsilon and episodes.

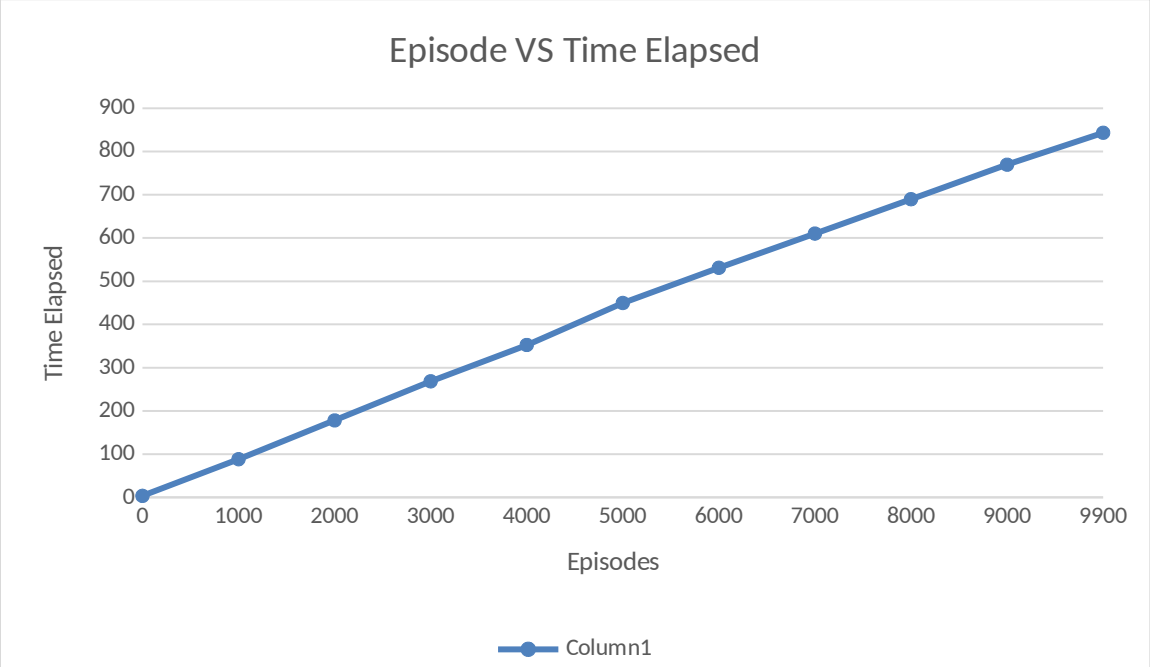
The graph between Mean Reward and number of episodes is shown below:



Explanation:

It can be clearly seen from the graph that, as the episode increases the mean reward also increases. This is because as the number of episodes increases the agent moves from exploration to exploitation. While the agent is exploring, it is taking more random actions in each state, so the reward is low. But as the number of episodes increases the agent learns and uses the q-values to take actions, hence the reward is high.

The graph between number of episodes and time elapsed is shown below:



Explanation:

As the number of episodes increases the agent takes more time and hence the time elapsed increases.

Writing Tasks:

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.

This is the exploration/exploitation problem where if the agent always chooses the action maximizing the q-value of the current policy, it will not explore and hence never find other possibly better actions which could potentially lead to a better policy. In order to build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its rewards at the same time. This is called **Exploration vs Exploitation trade-off**.

Ideally, what should happen is that our agent should, at first, select its actions randomly based on a certain percentage. This percentage is called 'Exploration Rate'. This should be practiced by the agent because at first, the agent should try all kinds of things (Exploration) before it starts to actually follow the policy. After the agent has explored a lot, it can follow the policy and decide the next action based on the current state and the reward value. This process is called Exploitation.

But, if the agent always chooses the action that maximizes the q-value, the agent will be limited to its current knowledge of what the best moves are, but there could be better moves.

The key to solve this problem is to establish a balance between exploration and exploitation. There are a number of techniques for doing this.

Technique 1:

Epsilon Greedy:

Epsilon greedy is one of the techniques used. Epsilon is the chance of taking a random action. When we increase epsilon we call it epsilon greedy, when we decrease it we call it greedy. Increasing epsilon incentivizes the agent to explore, that is, take more random steps.

Key Advantages:

- Very easy to implement.
- Will not get stuck in some local optimal state.
- The best performing arm will be used most of the time.

Technique 2:

Upper Confidence Bound Strategy:

Upper confidence bound strategy is another technique. The algorithm is based on the principle of optimism in the face of uncertainty. We calculate the confidence of our estimation and this leaves us with an interval of possible outcomes. We then pick the action with the highest possible outcome even though uncertainty is high. The thinking behind this is that uncertain events have a lot of potential knowledge to be discovered and be very useful

Key Advantages:

- Very easy to implement.
- Close to optimal

2. Question

Given:

- A 3X3 grid id given where one space of the grid id occupied by the agent (green square) and another is occupied by the goal (yellow square). The agent's action space consists of four actions UP, DOWN, LEFT and RIGHT. Initially, the agent is set to be in the upper left corner and the goal is in the lower right corner.
- Discounting Factor, $\gamma = 0.99$.

Goal:

The goal is to have agent move on to space so that it reaches the goal, in as little moves as possible.

Rewards to the agent:

- 1, when the agent moves closer to the goal.
- -1, when the agent moves away from the goal.
- 0, when the agent does not move at all.

The given states are shown in the figure below:

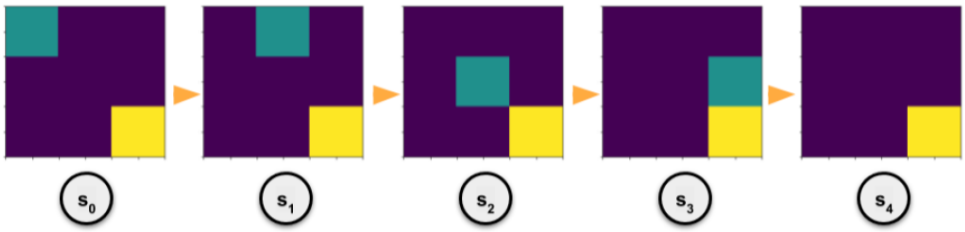


Figure 1

[Note: As the g=iven grid is 3X3. There are in total 9 states possible and only 5 states are given in the question. Hence, we will assume the rest of the states and name them as shown in the figure below.]

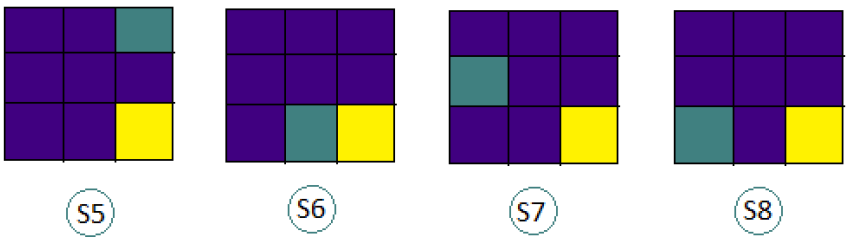


Figure 2

What we need to calculate:

We need to alculate the Q-values for all the Actions for states as shown in Figure 1. The Q-values calculation for states shown in Figure 2 might be used for calculating the Q-values for states shown in figure 1.

Formula Used:

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

Here,

$Q(s_t, a_t)$ is the Q-value for the current state and a current action.

r_t is the current reward.

γ is the discounting factor.

$\max_a Q(s_{t+1}, a)$ is the maximum Q-value for the next state.

Calculation Steps:

[Note: For sake of clarification in the calcualtion of Q-values, each step is numbered. The numbering order is not continuous. The number for each step represents when the calculation has been done (e.g. step 2 is calculated after step 1 and before step 3.)]

(16) $Q(S_3, U) = r_t + \gamma \max_a (S_5, a)$ [Note: This step is numbered (16) because it has been calculated after step (15).]

$= -1 + (0.99) * 1.99$

$= 0.97$

(1) $Q(S_3, D) = r_t + \gamma \max_a (S_4, a)$

$= 1 + 0.99 * 0$

$= 1$

(8) $Q(S_3, L) = r_t + \gamma \max_a (S_2, a)$

$= -1 + 0.99 * 1.99$

$= 0.97$

(2) $Q(S_3, R) = r_t + \gamma \max_a (S_3, a)$

$= 0 + 0.99 * 1$

$= 0.99$

(9) $Q(S_2, U) = r_t + \gamma \max_a (S_1, a)$

$= -1 + 0.99 * 2.97$

= 1.94

(25) $Q(S_2, D) = r_t + \gamma \max_a (S_6, a)$

= 1 + 0.99 * 1

= 1.99

(24) $Q(S_2, L) = r_t + \gamma \max_a (S_7, a)$

= -1 + 0.99 * 2.97

= 1.94

(3) $Q(S_2, R) = r_t + \gamma \max_a (S_3, a)$

= 1 + 0.99 * 1

= 1.99

(10) $Q(S_1, U) = r_t + \gamma \max_a (S_1, a)$

= 0 + 0.99 * 2.97

= 2.94

(4) $Q(S_1, D) = r_t + \gamma \max_a (S_2, a)$

= 1 + 0.99 * 1.99

= 2.97

(11) $Q(S_1, L) = r_t + \gamma \max_a (S_0, a)$

= -1 + 0.99 * 3.94

= 2.9

(17) $Q(S_1, R) = r_t + \gamma \max_a (S_5, a)$

= 1 + 0.99 * 1.99

= 2.97

(6) $Q(S_0, U) = r_t + \gamma \max_a (S_0, a)$

= 0 + 0.99 * 3.94

= 3.9

(26) $Q(S_0, D) = r_t + \gamma \max_a (S_7, a)$

= 1 + 0.99 * 2.97

= 3.94

(7) $Q(S_0, L) = r_t + \gamma \max_a (S_0, a)$

= 0 + 0.99 * 3.94

= 3.9

(5) $Q(S_0, R) = r_t + \gamma \max_a (S_1, a)$

= 1 + 0.99 * 2.97

= 3.94

(14) $Q(S_5, U) = r_t + \gamma \max_a (S_5, a)$

$= 0 + 0.99 * 1.99$

$= 1.97$

(12) $Q(S_5, D) = r_t + \gamma \max_a (S_3, a)$

$= 1 + 0.99 * 1$

$= 1.99$

(13) $Q(S_5, L) = r_t + \gamma \max_a (S_1, a)$

$= -1 + 0.99 * 2.97$

$= 1.94$

(15) $Q(S_5, R) = r_t + \gamma \max_a (S_5, a)$

$= 0 + 0.99 * 1.99$

$= 1.97$

(18) $Q(S_6, U) = r_t + \gamma \max_a (S_2, a)$

$= -1 + 0.99 * 1.99$

$= 0.97$

(20) $Q(S_6, D) = r_t + \gamma \max_a (S_6, a)$

$= 0 + 0.99 * 1$

$= 0.99$

(32) $Q(S_6, L) = r_t + \gamma \max_a (S_8, a)$

$= -1 + 0.99 * 1.99$

$= 0.97$

(19) $Q(S_6, R) = r_t + \gamma \max_a (S_4, a)$

$= 1 + 0.99 * 0$

$= 1$

(21) $Q(S_7, U) = r_t + \gamma \max_a (S_0, a)$

$= -1 + 0.99 * 3.94$

$= 2.9$

(31) $Q(S_7, D) = r_t + \gamma \max_a (S_8, a)$

$= 1 + 0.99 * 1.99$

$= 2.97$

(23) $Q(S_7, L) = r_t + \gamma \max_a (S_7, a)$

$= 0 + 0.99 * 2.97$

$= 2.94$

(22) $Q(S_7, R) = r_t + \gamma \max_a (S_2, a)$

= 1 + 0.99 * 1.99

= 2.97

(27) $Q(S_8, U) = r_t + \gamma \max_a (S_7, a)$

= -1 + 0.99 * 2.97

= 1.94

(30) $Q(S_8, D) = r_t + \gamma \max_a (S_8, a)$

= 0 + 0.99 * 1.99

= 1.97

(28) $Q(S_8, L) = r_t + \gamma \max_a (S_8, a)$

= 0 + 0.99 * 1.99

= 1.97

(29) $Q(S_8, R) = r_t + \gamma \max_a (S_6, a)$

= 1 + 0.99 * 1

= 1.99

(Table 1 – Q-table for States shown in Figure 1 and Figure 2)

STATE	ACTIONS				
	UP	DOWN	LEFT	RIGHT	Maximum Q-value for the state
0	3.9	3.94	3.9	3.94	3.94
1	2.94	2.97	2.9	2.97	2.97
2	1.94	1.99	1.94	1.99	1.99
3	0.97	1	0.97	0.99	1
4	0	0	0	0	0
5	1.97	1.99	1.94	1.97	1.99
6	0.97	0.99	0.97	1	1
7	2.9	2.97	2.94	2.97	2.97
8	1.94	1.97	1.97	1.99	1.99

Final Answer:

(Table 2 – Q-table for States shown in Figure 1)

STATE	ACTIONS			
	UP	DOWN	LEFT	RIGHT
0	3.9	3.94	3.9	3.94
1	2.94	2.97	2.9	2.97
2	1.94	1.99	1.94	1.99
3	0.97	1	0.97	0.99
4	0	0	0	0

References

- https://gym.openai.com/docs/
- https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html
- https://www.slideshare.net/AmazonWebServices/cmp305-deep-learning-on-aws-made-easycmp305
- https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa
- An Adaptive implementation of epsilon greedy in reinforcement learning – science direct – International workshop on adaptive technology – Procedia Computer Science 109C (2017) 1146-1151