**Deep Learning - Lab 6 – RL**

**Name = Jayathilaka S.A.D.R.P**

**IT Number = IT20631260**

**Answers**

1. Upload the attached Markov\_Decision\_Process (PolicyIteration) notebook and the GridWorld (QLearning) notebook to Google colab.
2. Understand the two codes.
3. Complete the incomplete parts in each notebook (these parts are mentioned by ‘#type your code here’).

**GridWorld.py**

import random

class Q\_Agent():

# Initialize

def \_\_init\_\_(self, environment, epsilon=0.05, alpha=0.1, gamma=1):

self.environment = environment

self.q\_table = dict() # Store all Q-values in a dictionary of dictionaries

for x in range(environment.height): # Loop through all possible grid spaces, create a sub-dictionary for each

for y in range(environment.width):

self.q\_table[(x, y)] = {'UP': 0, 'DOWN': 0, 'LEFT': 0, 'RIGHT': 0} # Populate sub-dictionary with zero values for possible moves

self.epsilon = epsilon

self.alpha = alpha

self.gamma = gamma

def choose\_action(self, available\_actions):

"""Returns the optimal action from the Q-Value table. If multiple optimal actions exist, choose randomly.

Will make an exploratory random action dependent on epsilon."""

if random.uniform(0, 1) < self.epsilon:

# Choose a random action (exploration)

return random.choice(available\_actions)

else:

# Choose the action with the highest Q-value (exploitation)

max\_q\_value = max(self.q\_table[self.environment.current\_state].values())

best\_actions = [action for action in available\_actions if self.q\_table[self.environment.current\_state][action] == max\_q\_value]

return random.choice(best\_actions)

def learn(self, old\_state, reward, new\_state, action):

"""Updates the Q-value table using Q-learning"""

# Get the current Q-value for the old state-action pair

current\_q\_value = self.q\_table[old\_state][action]

# Calculate the maximum Q-value for the new state

max\_next\_q\_value = max(self.q\_table[new\_state].values())

# Update the Q-value using the Q-learning update rule

new\_q\_value = current\_q\_value + self.alpha \* (reward + self.gamma \* max\_next\_q\_value - current\_q\_value)

# Update the Q-value in the Q-table

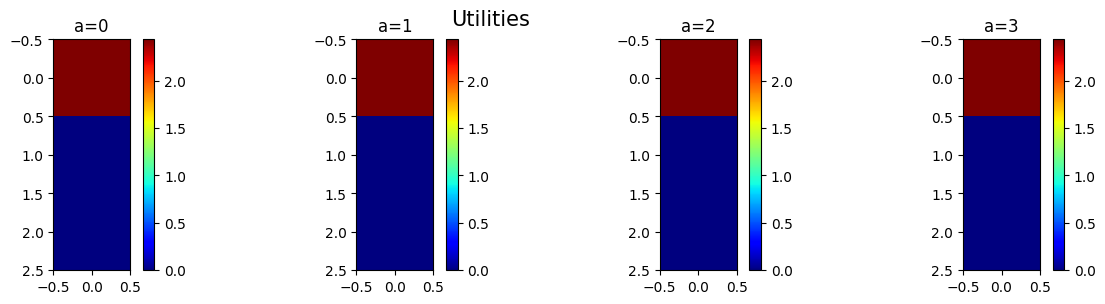
self.q\_table[old\_state][action] = new\_q\_value

A screenshot of a computer

Description automatically generated**Markov\_Decision\_Process .py**

A screenshot of a computer

Description automatically generated



1. Run the notebooks.
2. In the GridWorld notebook, increase the grid size to a large value to see how the execution time and the time to converge changes.

**In the GridWorld notebook, increasing the grid size to a larger value will likely result in longer execution times and potentially longer time to converge for algorithms such as Q-learning or value iteration. The reason is that with a larger grid, there are more states to explore, and the agent needs to take more actions to learn the optimal policy. This can lead to increased computational complexity and longer training times.**

1. Add screenshots of the completed parts of two notebooks (in step 3) in a word file.

**Submission**

Download the final modified notebooks. Add these notebooks and the word file to a new zip file. Upload this zip file to the courseweb submission link. The file name should be your registration number.