Batch: RL-2	**Roll No.: 16010121221**
Experiment / assignment / tutorial No.: 09	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: 9 To implement Dyna-Q model-based RL agent

AIM: To understand Dyna-Q is an algorithm intended to speed up learning, or policy convergence, for Q-learning.

Expected OUTCOME of Experiment: (Mentions the CO/CO's attained): CO5

Books/ Journals/ Websites referred:

Richard S. Sutton and Andrew G. Barto, "*Reinforcement Learning:An Introduction",* The MIT Press,Second Edition, 2018

Pre Lab/ Prior Concepts:

A Dyna-Q agent combines acting, learning, and planning. The first two components – acting and learning. Q-learning, for example, learns by acting in the world, and therefore combines acting and learning. But a Dyna-Q agent also implements planning, or simulating experiences from a model–and learns from them.
Dyna-Q agent as implementing acting, learning, and planning simultaneously, at all times. But, in practice, one needs to specify the algorithm as a sequence of steps. The most common way in which the Dyna-Q agent is implemented is by adding a planning routine to a Q-learning agent: after the agent acts in the real world and learns from the observed experience, the agent is allowed a series of k- planning steps. At each one of those k-planning steps, the model generates a simulated experience by randomly sampling from the history of all previously experienced state-action pairs. The agent then learns from this simulated experience, again using the same Q-learning rule.

Chosen Problem Statement:

Obstacle avoidance in grid world

The obstacle grid is a set of 16 states of 4 rows and 4 columns, where the goal node is marked in green and obstacles are marked in red. The goal of the learning is to avoid the obstacles and find a path which is the shortest distance from the goal node. The starting state can be any state. The actions can be any from left, right, up and down for all states not on the border, only actions that don't take you out-of-bounds are possible for border states.

A	B	🟥 C	D
🟥 E	F	G	🟥 H
I	J	🟥 K	L
M	N	O	🟩 P

Explain following concepts w.r.t. chosen problem statement:

Policy:

Policy is a mapping of state to action with some probability value. In this problem statement, all the states A to P have a probabilistic value for each action.

Reward function:

```
rewards = {
    'A': 1,
    'B': 2,
    'C': -1,
    'D': 1,
    'E': -2,
    'F': 3,
    'G': 1,
    'H': -1,
    'I': 3,
    'J': 4,
    'K': -1,
    'L': 1,
    'M': 4,
    'N': 5,
    'O': 6,
    'P': 10,
}
```

Value function:

The value function in the problem statement is a state-action value function mapping of state and actions to the corresponding values. Initially each state-action pair is assigned value as 0. The function is updated using Q-Learning.

```
def initialize_policy():
    Q = {}
    for state in states:
        try:
            state_actions = moves[state]['moves']
        except KeyError:
            state_actions = actions
        for action in state_actions:
            Q[state, action] = 0 # np.random.random()

    return Q
```

Model of the environment:

```
moves = {
    'A': {
        'down': 'E',
        'right': 'B',
        'moves': ['down', 'right'],
    },

    'D': {
        'down': 'H',
        'left': 'C',
        'moves': ['down', 'left'],
    },

    'M': {
        'up': 'I',
        'right': 'N',
        'moves': ['up', 'right'],
    },

    'P': {
        'up': 'L',
        'left': 'O',
        'moves': ['up', 'left'],
    },

    'B': {
        'left': 'A',
        'right': 'C',
        'down': 'F',
        'moves': ['down', 'right', 'left'],
    },

    'C': {
        'left': 'B',
        'right': 'D',
        'down': 'G',
        'moves': ['down', 'right', 'left'],
    },

    'E': {
        'up': 'A',
        'right': 'F',
        'down': 'I',
        'moves': ['down', 'right', 'up'],
    },

    'I': {
        'up': 'E',
        'right': 'J',
        'down': 'M',
        'moves': ['down', 'right', 'up'],
    },

    'N': {
        'up': 'J',
        'right': 'O',
        'left': 'M',
        'moves': ['left', 'right', 'up'],
    },

    'O': {
        'up': 'K',
        'right': 'P',
        'left': 'N',
        'moves': ['left', 'right', 'up'],
    },

    'L': {
        'up': 'H',
        'down': 'P',
        'left': 'K',
        'moves': ['left', 'down', 'up'],
    },

    'H': {
        'up': 'D',
        'down': 'L',
        'left': 'G',
        'moves': ['left', 'down', 'up'],
    },

    'F': {
        'up': 'B',
        'down': 'J',
        'left': 'E',
        'right': 'G',
    },

    'G': {
        'up': 'C',
        'down': 'K',
        'left': 'F',
        'right': 'H',
    },

    'J': {
        'up': 'F',
        'down': 'N',
        'left': 'I',
        'right': 'K',
    },
    'K': {
        'up': 'G',
        'down': 'O',
        'left': 'J',
        'right': 'L',
    },
}

from random import randint

class Environment:
    obstacles = ['E', 'K', 'C', 'H']
    actions = ['left', 'right', 'down', 'up']

    states = [
        'A', 'B', 'C', 'D',
        'E', 'F', 'G', 'H',
        'I', 'J', 'K', 'L',
        'M', 'N', 'O', 'P'
    ]

    current_state = ''
    start_state = 'A'
    goal_state = 'P'

    def __init__(self):
        self.current_state = self.start_state
        # return self

    def reset(self, state=None):
        if state:
            self.current_state = state
            return
        self.current_state = self.start_state
        return self.current_state

    def action_sample(self):
        try:
            moves_possible = moves[self.current_state]['moves']
            index = randint(0, len(moves_possible)-1)
            return moves_possible[index]
        except KeyError:
            index = randint(0, 3)
            return self.actions[index]
    def step(self, action):
        done = False
        new_state = moves[self.current_state][action]
        self.current_state = new_state
        new_reward = rewards[new_state]
        if new_state == self.goal_state:
            done = True

        return new_state, new_reward, done

    def print_state(self, state, new_line, endl):
        if state == self.current_state:
            print('\x1b[0;30;43m' + state + '\x1b[0m' + new_line,
end=endl)
        elif state == self.goal_state:
            print('\x1b[6;30;42m' + state + '\x1b[0m' + new_line,
end=endl)
        elif state in self.obstacles:
            print('\x1b[0;30;41m' + state + '\x1b[0m' + new_line,
end=endl)
        else:
            print(state + new_line, end=endl)

    def render(self):
        for i in range(16):
            if i%4 == 3:
                self.print_state(self.states[i], '\n', '')
            else:
                self.print_state(self.states[i], '', ' ')
```

DynaQ Learning –

```
def max_action(Q, state, actions):
    max_value = -99999
    max_action = ''

    for a in actions:
        if Q[state, a] > max_value:
            max_value = Q[state, a]
            max_action = a

    return   max_action

def initialize_model():
    model = {}
    for state in states:
        try:
            state_actions = moves[state]['moves']
        except KeyError:
            state_actions = actions
        for action in state_actions:
            model[state, action] = 0

    return model

import random

n_games = 1000
alpha = 0.1
gamma = 0.99
eps = 0.99
total_rewards = np.zeros(n_games)
Q = initialize_policy()
model = initialize_model()
env = Environment()

observed_state_action_map = {}

state_t = states[random.randint(0, len(states) - 1)]
while state_t == 'P':
    state_t = states[random.randint(0, len(states) - 1)]

env.reset(state_t)
```

```
for _ in range(10):
    while 1:

        try:
            state_actions = moves[state_t]['moves']
        except KeyError:
            state_actions = actions

        action_t = env.action_sample() if np.random.random() < eps else \
                        max_action(Q, state_t, state_actions)

        try:
            observed_state_action_map[state_t].append(action_t)
        except KeyError:
            observed_state_action_map[state_t] = []
            observed_state_action_map[state_t].append(action_t)

        state_t1, reward, done = env.step(action_t)

        try:
            state_actions = moves[state_t1]['moves']
        except KeyError:
            state_actions = actions

        action_t1 = max_action(Q, state_t1, state_actions)

        # Q-Learning
        Q[state_t, action_t] = Q[state_t,action_t] + alpha*(reward +
gamma*Q[state_t1, action_t1] - Q[state_t,action_t])

        # Model Update
        model[state_t, action_t] = (reward, state_t1)

        # Planning Step
        for i in range(0, 10):
            observed_states = list(observed_state_action_map.keys())
            state_ro = observed_states[random.randint(0,
len(observed_states) - 1)]
            action_rt =
observed_state_action_map[state_ro][random.randint(0,
len(observed_state_action_map[state_ro]) - 1)]

            reward, state_ro1 = model[state_ro, action_rt]
            try:
                state_actions = moves[state_ro1]['moves']
            except KeyError:
                state_actions = actions

            action_t1 = max_action(Q, state_ro1, state_actions)

            Q[state_ro, action_rt] = Q[state_ro,action_rt] + 
alpha*(reward + gamma*Q[state_ro1, action_t1] - Q[state_ro,action_rt])

        if done:
            print('Done')
            break
        state_t = state_t1
```

Output:

```
Iteration 0

Policy:
{(('A', 'down')): 0.75(('A', 'right')): 0(('B', 'down')): 3.12(('B', 
'right')): 2.22(('B', 'left')): 0(('C', 'down')): 0(('C', 'right')): 
0(('C', 'left')): 4.41(('D', 'down')): 0(('D', 'left')): 0(('E', 
'down')): 1.46(('E', 'right')): 2.99(('E', 'up')): 1.07(('F', 'left')): 
0.53(('F', 'right')): 0(('F', 'down')): 3.2(('F', 'up')): 0(('G', 
'left')): 0(('G', 'right')): 0(('G', 'down')): 0(('G', 'up')): 0(('H', 
'left')): 0(('H', 'down')): 0(('H', 'up')): 0(('I', 'down')): 0(('I', 
'right')): 3.53(('I', 'up')): 0.62(('J', 'left')): 1.2(('J', 'right')): 
0.31(('J', 'down')): 5.76(('J', 'up')): 1.59(('K', 'left')): 4.57(('K', 
'right')): 0(('K', 'down')): 0(('K', 'up')): 0(('L', 'left')): 0(('L', 
'down')): 0(('L', 'up')): 0(('M', 'up')): 0(('M', 'right')): 0(('N', 
'left')): 0(('N', 'right')): 1.14(('N', 'up')): 5.24(('O', 'left')): 
0(('O', 'right')): 1.9(('O', 'up')): 0(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): 0, ('B', 'down'): (3, 'F'), 
('B', 'right'): (-1, 'C'), ('B', 'left'): 0, ('C', 'down'): 0, ('C', 
'right'): 0, ('C', 'left'): (2, 'B'), ('D', 'down'): 0, ('D', 'left'): 0, 
('E', 'down'): (3, 'I'), ('E', 'right'): (3, 'F'), ('E', 'up'): (1, 'A'), 
('F', 'left'): (-2, 'E'), ('F', 'right'): 0, ('F', 'down'): (4, 'J'), 
('F', 'up'): 0, ('G', 'left'): 0, ('G', 'right'): 0, ('G', 'down'): 0, 
('G', 'up'): 0, ('H', 'left'): 0, ('H', 'down'): 0, ('H', 'up'): 0, ('I', 
'down'): 0, ('I', 'right'): (4, 'J'), ('I', 'up'): (-2, 'E'), ('J',
'left'): (3, 'I'), ('J', 'right'): (-1, 'K'), ('J', 'down'): (5, 'N'),
('J', 'up'): (3, 'F'), ('K', 'left'): (4, 'J'), ('K', 'right'): 0, ('K',
'down'): 0, ('K', 'up'): 0, ('L', 'left'): 0, ('L', 'down'): 0, ('L',
'up'): 0, ('M', 'up'): 0, ('M', 'right'): 0, ('N', 'left'): 0, ('N',
'right'): (6, 'O'), ('N', 'up'): (4, 'J'), ('O', 'left'): 0, ('O',
'right'): (10, 'P'), ('O', 'up'): 0, ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 1

Policy:
{(('A', 'down')): 3.81(('A', 'right')): 9.71(('B', 'down')): 11.13(('B',
'right')): 4.63(('B', 'left')): 4.07(('C', 'down')): 4.09(('C',
'right')): 4.53(('C', 'left')): 7.53(('D', 'down')): 0(('D', 'left')):
6.0(('E', 'down')): 8.28(('E', 'right')): 8.83(('E', 'up')): 6.11(('F',
'left')): 4.47(('F', 'right')): 2.66(('F', 'down')): 12.04(('F', 'up')):
0(('G', 'left')): 5.11(('G', 'right')): 0.78(('G', 'down')): 8.05(('G',
'up')): 3.61(('H', 'left')): 0(('H', 'down')): 0(('H', 'up')): 5.86(('I',
'down')): 0(('I', 'right')): 12.55(('I', 'up')): 5.06(('J', 'left')):
6.91(('J', 'right')): 1.52(('J', 'down')): 15.97(('J', 'up')): 7.63(('K',
'left')): 14.21(('K', 'right')): 0(('K', 'down')): 7.08(('K', 'up')):
3.27(('L', 'left')): 0(('L', 'down')): 0(('L', 'up')): 3.12(('M', 'up')):
12.16(('M', 'right')): 0(('N', 'left')): 2.31(('N', 'right')):
10.82(('N', 'up')): 15.27(('O', 'left')): 8.08(('O', 'right')):
7.18(('O', 'up')): 7.06(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): 0, ('G',
'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1, 'K'),
('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): 0, ('H', 'up'):
(1, 'D'), ('I', 'down'): 0, ('I', 'right'): (4, 'J'), ('I', 'up'): (-2,
'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1, 'K'), ('J', 'down'):
(5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4, 'J'), ('K', 'right'):
0, ('K', 'down'): (6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): 0,
('L', 'down'): 0, ('L', 'up'): (-1, 'H'), ('M', 'up'): (3, 'I'), ('M',
'right'): 0, ('N', 'left'): (4, 'M'), ('N', 'right'): (6, 'O'), ('N',
'up'): (4, 'J'), ('O', 'left'): (5, 'N'), ('O', 'right'): (10, 'P'),
('O', 'up'): (-1, 'K'), ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 2
```

```
Policy:
{(('A', 'down')): 3.81(('A', 'right')): 9.71(('B', 'down')): 11.13(('B',
'right')): 4.81(('B', 'left')): 4.07(('C', 'down')): 4.09(('C',
'right')): 4.53(('C', 'left')): 7.53(('D', 'down')): 0(('D', 'left')):
6.09(('E', 'down')): 8.28(('E', 'right')): 8.83(('E', 'up')): 6.97(('F',
'left')): 4.47(('F', 'right')): 2.66(('F', 'down')): 12.82(('F', 'up')):
0(('G', 'left')): 6.09(('G', 'right')): 0.78(('G', 'down')): 9.01(('G',
'up')): 3.61(('H', 'left')): 0(('H', 'down')): 0(('H', 'up')): 6.07(('I',
'down')): 0(('I', 'right')): 12.55(('I', 'up')): 5.22(('J', 'left')):
6.91(('J', 'right')): 2.67(('J', 'down')): 15.97(('J', 'up')): 7.63(('K',
'left')): 14.21(('K', 'right')): 0(('K', 'down')): 7.08(('K', 'up')):
3.89(('L', 'left')): 0(('L', 'down')): 0(('L', 'up')): 3.59(('M', 'up')):
12.48(('M', 'right')): 0(('N', 'left')): 3.69(('N', 'right')):
10.82(('N', 'up')): 15.27(('O', 'left')): 8.68(('O', 'right')):
7.46(('O', 'up')): 7.66(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): 0, ('G',
'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1, 'K'),
('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): 0, ('H', 'up'):
(1, 'D'), ('I', 'down'): 0, ('I', 'right'): (4, 'J'), ('I', 'up'): (-2,
'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1, 'K'), ('J', 'down'):
(5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4, 'J'), ('K', 'right'):
0, ('K', 'down'): (6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): 0,
('L', 'down'): 0, ('L', 'up'): (-1, 'H'), ('M', 'up'): (3, 'I'), ('M',
'right'): 0, ('N', 'left'): (4, 'M'), ('N', 'right'): (6, 'O'), ('N',
'up'): (4, 'J'), ('O', 'left'): (6, 'O'), ('O', 'right'): (10, 'P'),
('O', 'up'): (-1, 'K'), ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 3

Policy:
{(('A', 'down')): 3.81(('A', 'right')): 10.44(('B', 'down')): 11.65(('B',
'right')): 4.81(('B', 'left')): 4.07(('C', 'down')): 4.09(('C',
'right')): 4.78(('C', 'left')): 7.53(('D', 'down')): 0(('D', 'left')):
6.12(('E', 'down')): 8.28(('E', 'right')): 10.2(('E', 'up')): 6.97(('F',
'left')): 5.03(('F', 'right')): 3.43(('F', 'down')): 13.51(('F', 'up')):
0(('G', 'left')): 6.09(('G', 'right')): 0.78(('G', 'down')): 9.47(('G',
'up')): 3.61(('H', 'left')): 0(('H', 'down')): 0(('H', 'up')): 6.17(('I',
'down')): 0(('I', 'right')): 13.93(('I', 'up')): 5.22(('J', 'left')):
6.91(('J', 'right')): 2.67(('J', 'down')): 15.97(('J', 'up')): 7.63(('K',
'left')): 14.77(('K', 'right')): 0(('K', 'down')): 8.51(('K', 'up')):
3.89(('L', 'left')): 0(('L', 'down')): 1.0(('L', 'up')): 3.59(('M',
'up')): 12.48(('M', 'right')): 0(('N', 'left')): 3.69(('N', 'right')):
10.82(('N', 'up')): 15.27(('O', 'left')): 9.27(('O', 'right')):
7.46(('O', 'up')): 7.07(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): 0, ('G',
'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1, 'K'),
('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): 0, ('H', 'up'):
(1, 'D'), ('I', 'down'): 0, ('I', 'right'): (4, 'J'), ('I', 'up'): (-2,
'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1, 'K'), ('J', 'down'):
(5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4, 'J'), ('K', 'right'):
0, ('K', 'down'): (6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): 0,
('L', 'down'): (10, 'P'), ('L', 'up'): (-1, 'H'), ('M', 'up'): (3, 'I'),
('M', 'right'): 0, ('N', 'left'): (4, 'M'), ('N', 'right'): (6, 'O'),
('N', 'up'): (4, 'J'), ('O', 'left'): (6, 'O'), ('O', 'right'): (10,
'P'), ('O', 'up'): (1, 'L'), ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 4

Policy:
{(('A', 'down')): 4.24(('A', 'right')): 10.75(('B', 'down')): 12.12(('B',
'right')): 4.81(('B', 'left')): 4.07(('C', 'down')): 4.09(('C',
'right')): 5.01(('C', 'left')): 7.53(('D', 'down')): 0(('D', 'left')):
6.16(('E', 'down')): 8.28(('E', 'right')): 10.2(('E', 'up')): 7.4(('F',
'left')): 5.34(('F', 'right')): 3.43(('F', 'down')): 13.51(('F', 'up')):
0(('G', 'left')): 6.09(('G', 'right')): 1.23(('G', 'down')): 9.47(('G',
'up')): 3.9(('H', 'left')): 0(('H', 'down')): 0(('H', 'up')): 6.26(('I',
'down')): 0(('I', 'right')): 14.51(('I', 'up')): 5.51(('J', 'left')):
6.91(('J', 'right')): 2.67(('J', 'down')): 16.43(('J', 'up')): 7.63(('K',
'left')): 15.27(('K', 'right')): 0(('K', 'down')): 8.51(('K', 'up')):
3.89(('L', 'left')): 1.52(('L', 'down')): 1.9(('L', 'up')): 3.59(('M',
'up')): 12.48(('M', 'right')): 0(('N', 'left')): 3.69(('N', 'right')):
10.82(('N', 'up')): 15.72(('O', 'left')): 10.45(('O', 'right')):
7.94(('O', 'up')): 6.82(('P', 'up')): 0(('P', 'left')): 0
Model:
```

```
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): 0, ('G',
'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1, 'K'),
('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): 0, ('H', 'up'):
(1, 'D'), ('I', 'down'): 0, ('I', 'right'): (4, 'J'), ('I', 'up'): (-2,
'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1, 'K'), ('J', 'down'):
(5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4, 'J'), ('K', 'right'):
0, ('K', 'down'): (6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): (6,
'O'), ('L', 'down'): (10, 'P'), ('L', 'up'): (-1, 'H'), ('M', 'up'): (3,
'I'), ('M', 'right'): 0, ('N', 'left'): (4, 'M'), ('N', 'right'): (6,
'O'), ('N', 'up'): (4, 'J'), ('O', 'left'): (6, 'O'), ('O', 'right'):
(10, 'P'), ('O', 'up'): (1, 'L'), ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 5

Policy:
{(('A', 'down')): 10.46(('A', 'right')): 15.21(('B', 'down')):
16.67(('B', 'right')): 7.57(('B', 'left')): 7.02(('C', 'down')):
9.31(('C', 'right')): 5.52(('C', 'left')): 13.84(('D', 'down')): 0(('D',
'left')): 9.89(('E', 'down')): 11.78(('E', 'right')): 15.86(('E', 'up')):
13.35(('F', 'left')): 7.49(('F', 'right')): 4.92(('F', 'down')):
16.27(('F', 'up')): 3.34(('G', 'left')): 10.06(('G', 'right')):
2.86(('G', 'down')): 13.24(('G', 'up')): 5.46(('H', 'left')): 0(('H',
'down')): 1.07(('H', 'up')): 8.55(('I', 'down')): 7.96(('I', 'right')):
16.77(('I', 'up')): 10.24(('J', 'left')): 9.18(('J', 'right')):
2.67(('J', 'down')): 20.79(('J', 'up')): 11.26(('K', 'left')):
18.13(('K', 'right')): 4.39(('K', 'down')): 14.3(('K', 'up')): 9.92(('L',
'left')): 9.81(('L', 'down')): 7.46(('L', 'up')): 5.4(('M', 'up')):
16.4(('M', 'right')): 9.7(('N', 'left')): 12.94(('N', 'right')):
14.0(('N', 'up')): 20.41(('O', 'left')): 14.56(('O', 'right')):
9.28(('O', 'up')): 8.21(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): (2, 'B'),
('G', 'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1,
'K'), ('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): (1, 'L'),
('H', 'up'): (1, 'D'), ('I', 'down'): (4, 'M'), ('I', 'right'): (4, 'J'),
('I', 'up'): (-2, 'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1,
'K'), ('J', 'down'): (5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4,
'J'), ('K', 'right'): (1, 'L'), ('K', 'down'): (6, 'O'), ('K', 'up'): (1,
'G'), ('L', 'left'): (6, 'O'), ('L', 'down'): (10, 'P'), ('L', 'up'): (-
1, 'H'), ('M', 'up'): (3, 'I'), ('M', 'right'): (5, 'N'), ('N', 'left'):
(4, 'M'), ('N', 'right'): (6, 'O'), ('N', 'up'): (4, 'J'), ('O', 'left'):
(6, 'O'), ('O', 'right'): (10, 'P'), ('O', 'up'): (-1, 'K'), ('P', 'up'):
0, ('P', 'left'): 0}

Iteration 6

Policy:
{(('A', 'down')): 10.46(('A', 'right')): 15.54(('B', 'down')):
16.67(('B', 'right')): 7.57(('B', 'left')): 7.02(('C', 'down')):
9.31(('C', 'right')): 5.52(('C', 'left')): 13.84(('D', 'down')): 0(('D',
'left')): 10.65(('E', 'down')): 12.56(('E', 'right')): 16.71(('E',
'up')): 13.35(('F', 'left')): 7.49(('F', 'right')): 4.92(('F', 'down')):
17.85(('F', 'up')): 3.34(('G', 'left')): 10.06(('G', 'right')):
2.86(('G', 'down')): 13.24(('G', 'up')): 5.46(('H', 'left')): 0(('H',
'down')): 1.07(('H', 'up')): 9.03(('I', 'down')): 7.96(('I', 'right')):
16.77(('I', 'up')): 10.59(('J', 'left')): 9.18(('J', 'right')):
2.67(('J', 'down')): 21.27(('J', 'up')): 12.13(('K', 'left')):
18.78(('K', 'right')): 4.39(('K', 'down')): 14.3(('K', 'up')): 9.92(('L',
'left')): 9.81(('L', 'down')): 7.71(('L', 'up')): 5.93(('M', 'up')):
16.72(('M', 'right')): 11.29(('N', 'left')): 12.94(('N', 'right')):
14.64(('N', 'up')): 20.82(('O', 'left')): 14.56(('O', 'right')):
9.28(('O', 'up')): 9.15(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): (2, 'B'),
('G', 'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1,
'K'), ('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): (1, 'L'),
('H', 'up'): (1, 'D'), ('I', 'down'): (4, 'M'), ('I', 'right'): (4, 'J'),
('I', 'up'): (-2, 'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1,
'K'), ('J', 'down'): (5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4,
'J'), ('K', 'right'): (1, 'L'), ('K', 'down'): (6, 'O'), ('K', 'up'): (1,
'G'), ('L', 'left'): (6, 'O'), ('L', 'down'): (10, 'P'), ('L', 'up'): (1,
'L'), ('M', 'up'): (3, 'I'), ('M', 'right'): (5, 'N'), ('N', 'left'): (4,
'M'), ('N', 'right'): (6, 'O'), ('N', 'up'): (4, 'J'), ('O', 'left'): (6,
'O'), ('O', 'right'): (10, 'P'), ('O', 'up'): (-1, 'K'), ('P', 'up'): 0,
('P', 'left'): 0}

Iteration 7

Policy:
{(('A', 'down')): 10.46(('A', 'right')): 16.69(('B', 'down')):
17.07(('B', 'right')): 8.13(('B', 'left')): 7.95(('C', 'down')):
9.31(('C', 'right')): 5.52(('C', 'left')): 14.31(('D', 'down')): 0(('D',
'left')): 11.13(('E', 'down')): 13.35(('E', 'right')): 16.71(('E',
'up')): 14.04(('F', 'left')): 7.49(('F', 'right')): 5.88(('F', 'down')):
18.62(('F', 'up')): 4.9(('G', 'left')): 11.12(('G', 'right')): 2.86(('G',
'down')): 14.13(('G', 'up')): 5.46(('H', 'left')): 0(('H', 'down')):
1.07(('H', 'up')): 9.3(('I', 'down')): 7.96(('I', 'right')): 17.64(('I',
'up')): 11.66(('J', 'left')): 9.18(('J', 'right')): 2.67(('J', 'down')):
21.7(('J', 'up')): 12.98(('K', 'left')): 19.45(('K', 'right')):
4.39(('K', 'down')): 15.96(('K', 'up')): 10.38(('L', 'left')):
10.59(('L', 'down')): 8.5(('L', 'up')): 6.91(('M', 'up')): 16.72(('M',
'right')): 11.29(('N', 'left')): 12.94(('N', 'right')): 15.74(('N',
'up')): 20.82(('O', 'left')): 14.56(('O', 'right')): 9.35(('O', 'up')):
11.56(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): 0, ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'), ('E',
'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2, 'E'),
('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): (2, 'B'),
('G', 'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'): (-1,
'K'), ('G', 'up'): (-1, 'C'), ('H', 'left'): 0, ('H', 'down'): (1, 'L'),
('H', 'up'): (1, 'D'), ('I', 'down'): (4, 'M'), ('I', 'right'): (4, 'J'),
('I', 'up'): (-2, 'E'), ('J', 'left'): (3, 'I'), ('J', 'right'): (-1,
'K'), ('J', 'down'): (5, 'N'), ('J', 'up'): (3, 'F'), ('K', 'left'): (4,
'J'), ('K', 'right'): (1, 'L'), ('K', 'down'): (6, 'O'), ('K', 'up'): (1,
'G'), ('L', 'left'): (-1, 'K'), ('L', 'down'): (10, 'P'), ('L', 'up'):
(1, 'L'), ('M', 'up'): (3, 'I'), ('M', 'right'): (5, 'N'), ('N', 'left'):
(4, 'M'), ('N', 'right'): (6, 'O'), ('N', 'up'): (4, 'J'), ('O', 'left'):
(6, 'O'), ('O', 'right'): (10, 'P'), ('O', 'up'): (-1, 'K'), ('P', 'up'):
0, ('P', 'left'): 0}

Iteration 8

Policy:
```

```
{(('A', 'down')): 11.42(('A', 'right')): 18.3(('B', 'down')): 18.5(('B',
'right')): 9.93(('B', 'left')): 9.88(('C', 'down')): 10.85(('C',
'right')): 8.01(('C', 'left')): 16.23(('D', 'down')): 0.85(('D',
'left')): 12.11(('E', 'down')): 15.02(('E', 'right')): 19.04(('E',
'up')): 14.04(('F', 'left')): 10.93(('F', 'right')): 7.61(('F', 'down')):
19.96(('F', 'up')): 4.9(('G', 'left')): 12.22(('G', 'right')): 2.86(('G',
'down')): 15.38(('G', 'up')): 8.42(('H', 'left')): 3.01(('H', 'down')):
4.12(('H', 'up')): 10.21(('I', 'down')): 11.52(('I', 'right')):
19.3(('I', 'up')): 12.35(('J', 'left')): 11.49(('J', 'right')):
2.67(('J', 'down')): 22.98(('J', 'up')): 12.98(('K', 'left')):
20.14(('K', 'right')): 5.1(('K', 'down')): 17.33(('K', 'up')):
11.38(('L', 'left')): 11.42(('L', 'down')): 8.65(('L', 'up')): 8.24(('M',
'up')): 18.63(('M', 'right')): 15.56(('N', 'left')): 14.49(('N',
'right')): 16.26(('N', 'up')): 23.11(('O', 'left')): 15.15(('O',
'right')): 9.48(('O', 'up')): 13.44(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): (-1, 'H'), ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'),
('E', 'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2,
'E'), ('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): (2,
'B'), ('G', 'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'):
(-1, 'K'), ('G', 'up'): (-1, 'C'), ('H', 'left'): (1, 'G'), ('H',
'down'): (1, 'L'), ('H', 'up'): (1, 'D'), ('I', 'down'): (4, 'M'), ('I',
'right'): (4, 'J'), ('I', 'up'): (-2, 'E'), ('J', 'left'): (3, 'I'),
('J', 'right'): (-1, 'K'), ('J', 'down'): (5, 'N'), ('J', 'up'): (3,
'F'), ('K', 'left'): (4, 'J'), ('K', 'right'): (1, 'L'), ('K', 'down'):
(6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): (-1, 'K'), ('L', 'down'):
(10, 'P'), ('L', 'up'): (1, 'L'), ('M', 'up'): (3, 'I'), ('M', 'right'):
(5, 'N'), ('N', 'left'): (4, 'M'), ('N', 'right'): (6, 'O'), ('N', 'up'):
(4, 'J'), ('O', 'left'): (6, 'O'), ('O', 'right'): (10, 'P'), ('O',
'up'): (-1, 'K'), ('P', 'up'): 0, ('P', 'left'): 0}

Iteration 9

Policy:
{(('A', 'down')): 11.97(('A', 'right')): 18.3(('B', 'down')): 18.5(('B',
'right')): 10.44(('B', 'left')): 9.88(('C', 'down')): 10.85(('C',
'right')): 8.01(('C', 'left')): 16.23(('D', 'down')): 1.71(('D',
'left')): 12.91(('E', 'down')): 15.02(('E', 'right')): 19.42(('E',
'up')): 14.55(('F', 'left')): 10.93(('F', 'right')): 7.61(('F', 'down')):
20.69(('F', 'up')): 4.9(('G', 'left')): 12.22(('G', 'right')): 2.86(('G',
'down')): 15.73(('G', 'up')): 8.42(('H', 'left')): 3.01(('H', 'down')):
```

```
4.12(('H', 'up')): 10.52(('I', 'down')): 11.52(('I', 'right')):
19.3(('I', 'up')): 12.35(('J', 'left')): 11.49(('J', 'right')):
2.67(('J', 'down')): 23.91(('J', 'up')): 13.96(('K', 'left')):
20.14(('K', 'right')): 5.1(('K', 'down')): 17.81(('K', 'up')):
11.86(('L', 'left')): 11.42(('L', 'down')): 8.65(('L', 'up')): 8.64(('M',
'up')): 18.98(('M', 'right')): 15.56(('N', 'left')): 14.49(('N',
'right')): 16.26(('N', 'up')): 23.11(('O', 'left')): 16.32(('O',
'right')): 9.53(('O', 'up')): 13.99(('P', 'up')): 0(('P', 'left')): 0
Model:
{('A', 'down'): (-2, 'E'), ('A', 'right'): (2, 'B'), ('B', 'down'): (3,
'F'), ('B', 'right'): (-1, 'C'), ('B', 'left'): (1, 'A'), ('C', 'down'):
(1, 'G'), ('C', 'right'): (1, 'D'), ('C', 'left'): (2, 'B'), ('D',
'down'): (-1, 'H'), ('D', 'left'): (-1, 'C'), ('E', 'down'): (3, 'I'),
('E', 'right'): (3, 'F'), ('E', 'up'): (1, 'A'), ('F', 'left'): (-2,
'E'), ('F', 'right'): (1, 'G'), ('F', 'down'): (4, 'J'), ('F', 'up'): (2,
'B'), ('G', 'left'): (3, 'F'), ('G', 'right'): (-1, 'H'), ('G', 'down'):
(-1, 'K'), ('G', 'up'): (-1, 'C'), ('H', 'left'): (1, 'G'), ('H',
'down'): (1, 'L'), ('H', 'up'): (1, 'D'), ('I', 'down'): (4, 'M'), ('I',
'right'): (4, 'J'), ('I', 'up'): (-2, 'E'), ('J', 'left'): (3, 'I'),
('J', 'right'): (-1, 'K'), ('J', 'down'): (5, 'N'), ('J', 'up'): (3,
'F'), ('K', 'left'): (4, 'J'), ('K', 'right'): (1, 'L'), ('K', 'down'):
(6, 'O'), ('K', 'up'): (1, 'G'), ('L', 'left'): (-1, 'K'), ('L', 'down'):
(10, 'P'), ('L', 'up'): (1, 'L'), ('M', 'up'): (3, 'I'), ('M', 'right'):
(5, 'N'), ('N', 'left'): (4, 'M'), ('N', 'right'): (6, 'O'), ('N', 'up'):
(4, 'J'), ('O', 'left'): (6, 'O'), ('O', 'right'): (10, 'P'), ('O',
'up'): (-1, 'K'), ('P', 'up'): 0, ('P', 'left'): 0}
```

All Iteration and Planning stages output in policy.txt and model.txt

Conclusion: Thus, we conclude that DynaQ learning technique can be used to optimize the action value function for a grid world obstacle problem. It showed that the agent is able to learn an optimal policy of the expected rewards from each state, action pair. It also created a model simulation of the environment which is nearly same as the real environment.

Post Lab Descriptive Questions

Write notes on models and planning in RL.

A model of the environment refers to anything that an agent can use to predict how the environment will respond to its actions. This includes predicting the next state and reward given a current state and action. There are two types of models - distribution models, which provide a description of all possibilities and their probabilities, and sample models, which produce one possibility sampled according to the probabilities.

Models can be used to simulate or mimic experience by generating possible transitions or entire episodes based on starting states and policies.

Planning refers to any computational process that takes a model as input and produces or improves a policy for interacting with the modelled environment.

There are two approaches to planning in artificial intelligence: state-space planning involves searching through the state space for an optimal policy or path towards a goal, while plan-space planning involves searching through sets of plans with operators transforming one plan into another. Plan-space methods are not efficient for stochastic sequential decision problems focused on in reinforcement learning.

Date: 20-10-2024 **Signature of faculty in-charge**