Faculty of Technology and Engineering

Chandubhai S. Patel Institute of Technology

Date: / /

Practical Performa

Academic Year	:	2025-26	Semester	:	7 th
Course code	:	OCCSE4001	Course name	:	Reinforcement Learning

Practical- No. 2

Aim: Q-Learning Algorithm (Tabular) Code:

```
import gymnasium as gym
import numpy as np
import random
import time
from IPython.display import clear_output

# -- Environment Setup --
env = gym.make("FrozenLake-v1", is_slippery=True,
render_mode="human")

# -- Q-Learning Parameters --
q_table = np.zeros([env.observation_space.n,
env.action_space.n])
learning_rate = 0.1
discount_factor = 0.99
epsilon = 1.0  # Exploration rate
max_epsilon = 1.0
min_epsilon = 0.01
epsilon_decay_rate = 0.001

# -- Training Loop --
num_episodes = 150
for episode in range(num_episodes):
    state, info = env.reset()
```

```
    done = False
    for step in range(100):  # Max steps per episode
        # Epsilon-greedy action selection
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample()  # Explore
        else:
            action = np.argmax(q_table[state, :])  # Exploit
        new_state, reward, terminated, truncated, info =
env.step(action)
        done = terminated or truncated
        # Q-table update rule
        old_value = q_table[state, action]
        next_max = np.max(q_table[new_state, :])
        new_value = old_value + learning_rate * (reward +
discount_factor * next_max - old_value)
        q_table[state, action] = new_value
        state = new_state
        if done:
            break
    # Decay epsilon
    epsilon = min_epsilon + (max_epsilon - min_epsilon) *
np.exp(-epsilon_decay_rate * episode)

print("--- Training Finished ---")
print("Final Q-Table:")
print(q_table)

# -- Evaluate Agent --
state, info = env.reset()
done = False
env.render()
time.sleep(1)
for step in range(100):
    clear_output(wait=True)
    env.render()
    action = np.argmax(q_table[state, :])
    new_state, reward, terminated, truncated, info =
env.step(action)
    done = terminated or truncated
    state = new_state
    if done:
        clear_output(wait=True)
        env.render()
        print(f"Goal Reached with reward {reward}!")
        break
    time.sleep(0.3)
```

```
env.close()
```

Output:

```
Goal Reached with reward 1.0!
```

Grade/Marks

(_____ / 10)

Sign of Lab Teacher with Date