ECE 5984 – Deep Reinforcement Learning

Project Assignment #2: SARSA & Q-learning on FrozenLake

SAR and Q-learning:

Both SARA and Q-learning algorithms are value-based temporal difference methods, which solve the problem with iteration over the full set of the state-action pairs. We use states and actions obtained from the environment to update the values of state-action pairs, i.e., q values. In this assignment, you are asked to implement the SARSA & Q-learning algorithms to solve the frozen lake problem.

I. SARSA:

A. Set up/import Python libraries for this project

```
import gym
import collections

ENV_NAME = "FrozenLake-v1"
GAMMA = 0.9
ALPHA = 0.2
TEST_EPSOIDS = 200
```

B. Implement an `Agent_SARSA` class outlined below:

```
class Agent_SARSA:
    def __init__(self):
      self.env = gym.make(EVE_NAME)
      self.state = self.env.reset()
      self.values = collections.defaultdict(float)

    def sample_env(self):
      """
      Inputs:
            - self: an agent

      Returns:
```

```
        - a tuple: (old_state, action, reward, new_reward)
    """


def choose_action(self, state):
    """
    Inputs:
        - self: an agent
        - state: current state

    Returns:
        - next_a: the next action taken.
    """


def value_update(self, s, a, r, next_s, next_a):
    """
    Inputs:
        - self: an agent
        - s: state
        - a: action
        - r: reward
        - next_s: next state


    Returns:
        - self.values[(s, a)]: the updated value of (s, a).
    """




def play_episode(self, env):
    """
    Inputs:
        - self: an agent
        - env: the environment

    Returns:
        - total_reward: the total reward after playing an
        episode
    """
```

C. Implement the train loop: we first create a test environment and agent, then in the loop, we do one step in the environment and perform a value update using the obtained data. We test the current policy by playing several test episodes. If a good reward is obtained, then we stop the training.

II. Q-learning:

A. Set up/import Python libraries for this project

```
import gym
import collections

ENV_NAME = "FrozenLake-v0"
GAMMA = 0.9
ALPHA = 0.2
TEST_EPSOIDS = 200
```

B. Implement an `Agent_QLearning` class outlined below:

```
class Agent_QLearning:
    def __init__(self):
      self.env = gym.make(EVE_NAME)
      self.state = self.env.reset()
      self.values = collections.defaultdict(float)

    def sample_env(self):
      """
      Inputs:
            - self: an agent

      Returns:
            - a tuple: (old_state, action, reward, new_reward)
      """


    def best_value_and_action(self, state):
      """
      Inputs:
            - self: an agent
            - state: current state

      Returns:
            - best_value: the best value updated
```

```
            - best_action: the best action taken.
        """


    def value_update(self, s, a, r, next_s):
        """
        Inputs:
            - self: an agent
            - s: state
            - a: action
            - r: reward
            - next_s: next state

        Returns:
            - self.values[(s, a)]: the updated value of (s, a).
        """


    def play_episode(self, env):
        """
        Inputs:
            - self: an agent
            - env: the environment

        Returns:
            - total_reward: the total reward after playing an
            episode
        """
```

C. Implement the train loop: we first create a test environment and agent, then in the loop, we do one step in the environment and perform a value update using the obtained data. We test the current policy by playing several test episodes. If a good reward is obtained, then we stop the training.

You need to prepare **a written report** (in the pdf format) including the following sections: (1) Approaches, (2) Experimental Results, and (3) Discussion.

In addition, you need to attach your **implementation codes/notebooks** as separate files to the report.