Exercise Session 1

Reinforcement Learning

Sebastian Griesbach / Prof. Carlo D'Eramo

Summer semester 2024 / June 18th

In this exercise, you will implement the Q-Learning and SARSA algorithms to understand the difference between off-policy and on-policy learning. **Please only upload Your agent.py file as your solution. Write your answers to exercise 3 and 4 as comments at the bottom of the file.**

Exercise 1

We are using a custom grid world setting. You can ignore most of the files. Most of them handle the environment mechanics as well as the rendering. The relevant files for you are **agent.py** and **run_exercise.py**. Open the **agent.py** and implement the **getValue**, **getAction**, and **update** function in the **QLearningAgent** class.

The pseudo-code of Q-learning might be helpful:

```
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily and $Q(\text{terminal-state}, \cdot) = 0$
Loop (for each episode):
    Initialize $s$
    Loop (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $a$, observe $r$, $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$
        $s \leftarrow s'$
    until s is terminal
```

When you think you are done, execute **run_exercise.py** to watch your agents learning. To quickly check your result set **episodes** to **1000** and **quiet** to **True**. The resulting policy should take the shortest path to the "winning" state if you don't change the other settings (this is not always the case, but most of the time).

Exercise 2

Now implement the rest of the **update** function in the **SARSA** class in **agent.py**. This is the pseudo-code for the SARSA algorithm:

```
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily and $Q(\text{terminal-state}, \cdot) = 0$
Loop (for each episode):
    Initialize $s$
    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Loop (for each step of episode):
        Take action $a$, observe $r$, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
        $s \leftarrow s', a \leftarrow a'$
    until s is terminal
```

When you are done check your results as you did with Q-learning. Put **"sarsa"** for the **agent** argument. The resulting policy should reach the goal but avoids the edge of the cliff (again most of the time).

Exercise 3

Think about the difference between these two results. SARSA and Q-Learning are very similar yet they result in different policies. Why is that? Come up with an answer. Maybe compare the pseudo-code of both algorithms.

Exercise 4

Watch the agent learn with the **quiet** setting on **False**. How does the value information propagate through the grid? Do you see a problem with this? What would be a better way of doing it?