Artificial Intelligence: Reinforcement Learning

In this project, we aim to implement both `SARSA` and `Q-Learning` algorithms.

1 Instructions

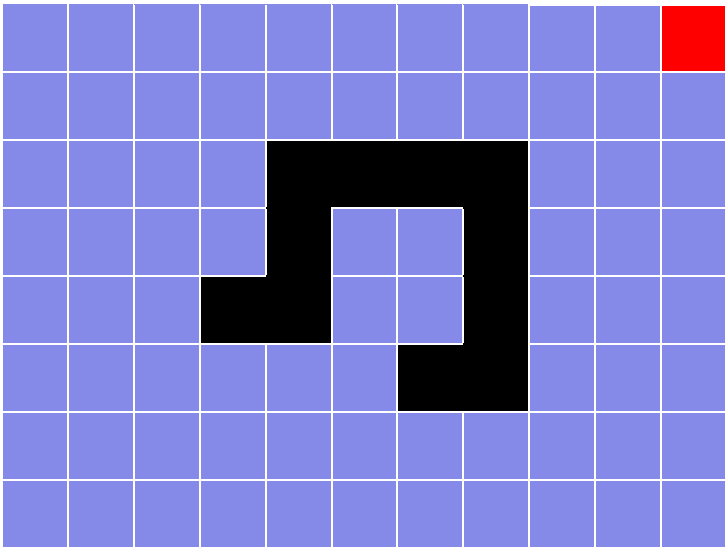

Figure 1: Maze

We extend the windy maze defined in P1 with probabilistic outcome after an action. It becomes a MDP problem. However, we assume that the agent doesn't know either the reward function or the transition model. The agent aims to run many trials in order to obtain Q-value for each (state, action) pair and the optimal action at each state.

Environment In your implementation, you need to simulate the windy maze environment: the wind comes from the south and the cost of one step is defined as follows: 1 for moving northward; 2 for moving eastward or westward; 3 for moving southward. The agent can drift to the left or the right from the perspective of moving direction with probability 0.1. If the drifting direction is an obstacle, it will be bounced back to the original position. The maze map is shown in Figure 1 for your convenience.

Active Reinforcement Learning In your implementation, you will generate many trials, each of which will result in a trajectory of (state, action, reward) tuple. The agent will use the ε-Greedy algorithm to choose an action at each state along each trajectory, where $\varepsilon = 0.05$: the agent chooses a latest optimal action at each state with 95% and a random action with 5%. The initial state for each trial is chosen randomly and each trial will end at the goal state. Along each trajectory, the agent can use either of these two learning algorithms to update the Q-values: SARSA and Q-Learning. Since the reward function $R(s, a)$ here depends on both the state and the action taken at this state, the Q-value update equations should be revised accordingly.

- For SARSA, it should be revised as

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,(R(s, a) + \gamma Q(s^j, a^j) - Q(s, a))$$

- For Q-Learning, it should use the following equation

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,\Big(R(s, a) + \gamma \max_{a^j} Q(s^j, a^j) - Q(s, a)\Big)$$

We choose $\alpha = \frac{1}{N_{s,a}+1}$, $N_{s,a}$ is the access frequency at (s, a) and $\gamma = 0.9$.

Testing and Outputs In your testing, generate 10000 trials for each algorithm (SARSA and Q-Learning). We initialize the Q-values for each trial at any state-action as 0 except for 50 at the goal state for each action. Right after both 1000 trials and 10000 trials, report the following three outcomes for each algorithm:

- the access frequency at each state-action $N_{s,a}$;
- the Q-value function at each state-action $Q(s, a)$;
- the optimal action at each state-action.

The format of the access frequency output should look like this (for format purpose only, not the correct solution). Due to the space limitation here, only the top-left 4 × 5 states are shown:

```
   1500       1500       1500       1500       1500       1500
1500 1500  1500 1500  1500 1500  1500 1500  1500 1500  1500 1500
   1500       1500       1500       1500       1500       1500

   1500       1500       1500       1500       1500       1500
1500 1500  1500 1500  1500 1500  1500 1500  1500 1500  1500 1500
   1500       1500       1500       1500       1500       1500

   1500       1500       1500       1500       ####       ####
1500 1500  1500 1500  1500 1500  1500 1500  #### ####  #### ####
   1500       1500       1500       1500       ####       ####

   1500       1500       1500       1500       ####       1500
1500 1500  1500 1500  1500 1500  1500 1500  #### ####  1500 1500
   1500       1500       1500       1500       ####       1500
```

The format of the Q-value output should look like this (for format purpose only, not the correct solution). Due to the space limitation here, only the top-left 4×5 states are shown:

```
   12.0      12.0      12.0      12.0      12.0      12.0
12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0
   12.0      12.0      12.0      12.0      12.0      12.0

   12.0      12.0      12.0      12.0      12.0      12.0
12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0
   12.0      12.0      12.0      12.0      12.0      12.0

   12.0      12.0      12.0      12.0      ####      ####
12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  #### ####  #### ####
   12.0      12.0      12.0      12.0      ####      ####

   12.0      12.0      12.0      12.0      ####      12.0
12.0 12.0  12.0 12.0  12.0 12.0  12.0 12.0  #### ####  12.0 12.0
   12.0      12.0      12.0      12.0      ####      12.0
```

The format of the optimal action output should look like this (for format purpose only, not the correct solution):

```
>>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  GGGG

>>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  ^^^^

>>>>  >>>>  >>>>  >>>>  ####  ####  ####  ####  >>>>  >>>>  ^^^^

>>>>  >>>>  ^^^^  >>>>  ####  >>>>  >>>>  ####  >>>>  >>>>  ^^^^

>>>>  >>>>  ^^^^  ####  ####  VVVV  >>>>  ####  >>>>  >>>>  ^^^^

>>>>  >>>>  ^^^^  <<<<  <<<<  VVVV  ####  ####  >>>>  >>>>  ^^^^

>>>>  >>>>  >>>>  >>>>  >>>>  VVVV  >>>>  >>>>  >>>>  >>>>  ^^^^

>>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  >>>>  ^^^^
```

where `<<<<`: moving westward; `^^^^`: moving northward; `>>>>`: moving eastward; `VVVV`: moving southward; `SSSS`: the source; `GGGG`: the goal.