Homework 2

Agent navigation in a maze using TD Reinforcement Learning

1 Algorithm Description

In the main script, two classes are present: the **Environment** class and the **TDAgent** class. Those two classes are used in the **main()** function to allow the TD Reinforcement Learning algorithm to take place. The **main()** function takes the following parameters (of which, part of them are used by the two classes):

- num_episodes: number of episodes for updating the policy.
- α: weight given to the TD error, to update the policy online.
- γ: discounting factor for the future state-action reward.
- ϵ: probability (normalized by 100) of making a random action in the training using the ϵ-greedy approach.
- import_maze_csv: when True, imports an example maze from a .csv file.
- show_training: when True, shows the training steps that the agent make, for every episode till the end of all the episodes. It is useful to visualize the improvement of the agent between the episodes.

1.1 Training

After the instantiation of the **Environment** and **TDAgent** objects, a **training phase** is started.
First the policy $Q(S, A)$ is initialized with random values, then a **for loop cycles each episode** e where at the start of them the state of the agent is set to the initial one S_0.
The following pseudocode **loop** (**for each episode**) is implemented:

1. Given the state S, select an action A using a ϵ-greedy policy $Q_\epsilon(S, A)$.
2. Given the selected action A, perform the action in the environment and compute the reward R, the next state S' and the "is_over" flag (to understand if the agent has reached the exit of the maze).
3. Update the policy matrix using the $TD(0)$ update: $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
4. Repeat until the episode terminates (agent reaches the maze exit).

1.2 Testing and Results

At the end of the training, to show the results, a version of the maze with the learned policy is printed. After pressing ENTER, a small simulation is executed to show how the agents follows the learned policy to reach the end of the maze. A simple loop where only the greedy policy $Q(S, A)$ is used for this purpose. The algorithm was also tested with a custom maze from the *labyrint.csv* file. The images below show the maze used, the agent in action and the learned policy.

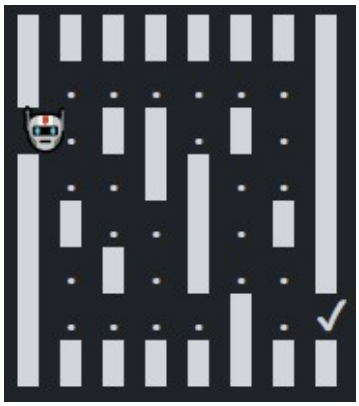

Figure 1: Simulation at start

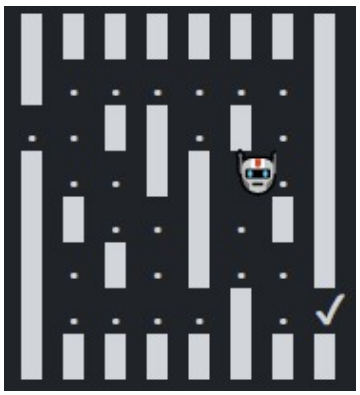

Figure 2: Agent moving

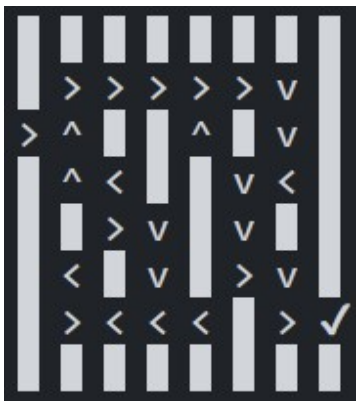

Figure 3: Learned policy

1.3 Important Notes

More comments and explanations are present in the H2_CANDELORO_python.py script. The following code was adapted and reformatted from a previous exercise done for the course of Machine Learning and Deep Learning (2020-2021) at the University of Modena and Reggio Emilia. The original project and code can be found at this link.