AI: Principles and Techniques
Programming task 4 (Machine Learning block)

The goal of this fourth task is to get some experience with MDPs and reinforcement learning techniques. More specifically, you are asked to implement algorithms that a) compute the value of the states in an MDP given a policy and compute an optimal policy (and demonstrate the effect of different settings), and b) learn an optimal policy using reinforcement learning.

Consider an $n \times m$ world with obstacles (walls) and end states with a (possibly negative) reward. Examples of such worlds can be seen in Section 12.1 in Poole and Mackworth (2nd edition) and on the slides (References to other sections/figures here are about this book as well).

- Implement the **value iteration** algorithm (see Section 9.5.2) to find the values of the states in such a world, and the (approximately) optimal policy that follows from these values. To **validate** your implementation, experiment with several (not too many) different settings, for example:
    - **different worlds** (e.g. the 4 × 3 world described in the chapter; a 10 × 10 world with or without obstacles and a reward of +10 at end state (10, 10); a 10 × 10 world with a reward of +10 at end state (5, 5); world with both end states with positive rewards and end states with negative rewards)
    - **different values of the discount factor** γ (γ=1, γ=0.9, γ=0.1);
    - **different values for the state penalty (i.e. -reward)** c (c=0, c=0.01, c=0.1);
    - **different settings of the transition probabilities** (deterministic (action up results in going up unless there was a wall blocking the action); probabilities 0.8 (or 0.5) for the direction corresponding to the action, 0.1 (or 0.25) for ending in the states to the left and the right of the original state.

- Implement the **Q-learning** algorithm (see Section 12.4) to find the utilities of the different state-action values in such a world when the transition probabilities and reward functions are unknown to the learner. Make sure that your code accommodates non-deterministic transition probabilities. Compare the results with the optimal policies you found in part **a.**) Try to find a good **exploration** parameter (use ε-greedy exploration to select your actions; not just randomly select an action) and **evaluate** your algorithm in terms of Section 12.6.

- Make a **short report** according to the IMRAD method, describing and explaining the findings of your experiments under both point **a.** and **b.** *Note that showing that you understand what is happening is more important than doing a lot of tests.* We prefer that you write your report in LaTeX, but it is also allowed to use Word or an open source variant. **Be sure to convert the report into a PDF.**

The **programming language** for the task is:

1. Java (version 11 or later), or
2. Python (version 3.6 or later)

Keep the interface simple (the goal is to investigate and explore the algorithms). You may use the MDP JAVA example code (see .zip file) as scaffold or example to build on (but please, properly document that/where you use, modify, etc. this code in your own source code). Choose informative names for the used variables, constants and functions. Use comments to make your code more accessible.

One **bonus point** (= 0.5 higher grade) will be given if you implement MDPs yourself (i.e., not use the example code).

It is allowed/advised to make this task in a team of (at most) 2 persons. Also check out the assessment form on Brightspace!

To hand in:
Both report (in .pdf) and (zipped) code. Do not forget your name(s), student number(s), course, number of the task, date, etc. Hand in via Brightspace. If you submit in a couple, let one of the students submit report and code and the other just a text file with the name of the other student for easy reference.

Practical session:
We again organize practical sessions (on campus and online) where you can ask questions, get feedback etc.

Deadline:
Friday 8-1-2021 23.59 via Brightspace.