Reinforcement Learning
TD 2

Fabien Pesquerel
fabien.pesquerel@inria.fr

Philippe Preux
philippe.preux@inria.fr

September 20, 2021

Alongside your notes, remember that the main reference for this course is always accessible on Philippe's website. Remember that you are invited to reproduce the Figure 3[1] of the main reference.

Students can hope to come back to this practical session later in the course and test what they will learn (algorithms, methods, heuristics). In particular, Deep RL ideas might be tested using the following exercise. Students are assumed to be familiar with *python 3.X* and usual scientific libraries. In particular, it is recommended to install *numpy*, *scipy*[2], *gym* and *pyTorch* (that will be used later in the course).

Gym & Frozen Lake

Frozen Lake[3] is a gym environment that we will use to implement some algorithms that were learned during the lessons. In the following, we will use a discount factor of $\gamma = 0.9$.

1. To check that everything is installed on your computer/environment run the small script **check_env.py**. It will try to import the necessary libraries and print current versions of them. You can use those versions to help you with any troubleshooting.

2. To better grasp the Frozen Lake environment, read, understand and run **discover.py**. Describe the MDP associated to this environment and formalize mathematically the goal of an agent.

3. Using the aforementioned environment, get a Monte-Carlo estimation of the value function at s_0 of a simple deterministic policy (s_0 is the initial state). For instance, this simple policy could be to always choose the `RIGHT` action. This is the implemented example in **hint_question3.py**.

4. (**Expected value method**) Write a function that takes a deterministic policy π as an input and outputs its value function V^π. The function should use Monte-Carlo estimation to compute the value function V^π.

5. (**Linear system method**) If $M = (S, A, p, r)$ be a finite MDP, then the value function V^π of a policy π is a vector in $\mathbb{R}^{|S|}$ (tabular case). Write V^π as the solution of a the product of a matrix with a vector (both known once π is). Write a function that takes a deterministic policy π as an input and outputs its value function V^π. Compare the result of this question with the Monte-Carlo estimation of the previous question and check for consistency.

[1] page 15 of the document as of September 2021.

[2] Those that are looking for performance might want to take a look at sparse representation of matrices (*scipy.sparse.csr_matrix*) and computation's techniques with arrays (BLAS and LAPACK are used as routine in *scipy*).

[3] Warning: The documentation is not up to date with the git repository. To import the Frozen Lake environment, you should `import FrozenLake-v1` rather than `import FrozenLake-v0` (which won't work).

6. (**Bellman operator method**) Using the contraction property of the Bellman operator, implement a function that iteratively applies the Bellman operator to compute the value function V^π of a policy π. What could be a good stopping criterion for your algorithm?

7. Using the contraction property of the **optimal Bellman operator**, implement a function that iteratively applies the optimal Bellman operator to compute the optimal value function V^*.

8. Compute an (approximation of an) optimal policy for this environment using **Value Iteration** (use the previous question).

9. Compute an (approximation of an) optimal policy for this environment using **Policy Iteration**.

10. Compare the two methods.

11. Using the contraction property of the **optimal Bellman operator**, implement a function that iteratively applies the optimal Bellman operator to compute the optimal value function Q^*.

12. Using the previous question, write a function that computes and an optimal policy from it.

13. Render a trajectory from start to finish and print the discounted cumulated reward associated to it.