ULiège
Prof. Damien Ernst

INFO8003-1
Optimal decision making
for complex problems

Project 1

Reinforcement Learning in a Discrete Domain

DOMAIN

We consider the following components:

- State space: $X = \{(x, y) \in \mathbb{N}^2 | x < n, y < m\}$,
- Action space: $U = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$,
- Reward signal: $r((x, y), (i, j)) = R(g, F((x, y), (i, j)))$,
- Discount factor: $\gamma = 0.99$,
- Time horizon: $T \to +\infty$,

where

- $g \in \mathbb{R}^{n \times m}$ for any $n, m \in \mathbb{N}^2$,
- $F((x, y), (i, j)) = (min(max(x + i, 0), n - 1), min(max(y + j, 0), m - 1))$
- $R(g, (x, y))$ returns the value of the cell located at (x, y) in g. Note that the reward signal is bounded by $B = \max_{(x,y)} R(g, (x, y))$.

From these components, we derive two different domains:

- **Deterministic domain**: The dynamics f are defined as follows:

$$f((x, y), (i, j)) = F((x, y), (i, j)), \forall ((x, y), (i, j)) \in X \times U.$$

- **Stochastic domain**: The dynamics f are defined as follows:

$$f((x, y), (i, j), w) = \begin{cases} F((x, y), (i, j)) & \text{if } w \leq \frac{1}{2}, \\ (0, 0) & \text{otherwise,} \end{cases} \quad \forall ((x, y), (i, j)) \in X \times U,$$

  where $0 \leq w \leq 1$ is drawn according to a uniform distribution. In this domain, the reward signal is redefined by $r((x, y), (i, j), w) = R(g, f((x, y), (i, j), w))$.

-3	1	-5	0	19
6	3	8	9	10
5	-8	4	1	-8
6	-9	4	19	-5
-20	-17	-4	-3	9

Figure 1: Domain instance. The number in each cell represents the reward obtained while reaching it after a move (e.g., if the agent moves up from the red cell, it receives a reward equal to 5). $(0, 0)$ corresponds to the upper-left corner of the grid. The two coordinates refer to the line and to the column number, respectively. The initial state of this instance is the red cell located at $(3, 0)$.

Figure 1 describes an instance of the domain that you must use in all sections of this assignment.

INSTRUCTIONS

You are expected to deliver (i) your source code (in Python 3.10, a file per section and named as *sectionK.py* where K is the section number - each missing one will cost you a one point penalty) and (ii) a report which is structured according to the next sections. We insist on the fact that the report is mandatory and that the source code needs to use only standard programming libraries plus, possibly, *NumPy* and *matplotlib*. We should also be able to execute your code, understand easily the code (a good documentation of each non-trivial function) and the results displayed. Both domains must be considered while addressing each section of this assignment. Non-deterministic numerical results should be averaged through 10 independent runs and displayed with standard deviation.

1 IMPLEMENTATION OF THE DOMAIN (2 POINTS)

Implement the different components of the domain instance described in Figure 1. Implement a rule-based policy of your choice (e.g., always go left, select actions always at random...) and describe it in your report. Simulate the policy in the domain through a single trajectory of 10 steps, starting by the initial state $x_0 = (3, 0)$. Show the trajectories in your report as a sequence of four tuples $(x_0, u_0, r_0, x_1), \ldots, (x_{10}, u_{10}, r_{10}, x_{11})$ (e.g., by a screenshot of your terminal).

2 EXPECTED RETURN OF A POLICY (2 POINTS)

Implement a routine to estimate J_μ in this domain, where $\mu : X \to U$ is a stationary policy. Test your implementation with your rule-based policy of Section 1. Choose a N which is large enough to approximate well the infinite time horizon of the policy and motivate your choice using the discount factor γ. Display $J_\mu^N(x)$ for each state x in a table in your report.

3 Optimal Policy (3 points)

Implement routines which compute $p(x'|x,u)$ and $r(x,u)$ that define the equivalent MDP of the domain. Afterwards, implement a routine which computes the sequence of Q_N-functions from these components using the dynamic programming principle. Determine the smallest N such that a greater value does not change the policy inferred from the last Q-function of the sequence. Derive directly μ^* from Q and show it in a table in your report. Display $J^N_{\mu^*}$ for each state x in a table in your report.

4 System Identification (5 points)

Implement a routine which estimates $r(x,u)$ and $p(x'|x,u)$ from a given trajectory $h_t = (x_0, u_0, r_0, x_1, u_1, r_1, \dots, u_{t-1}, r_{t-1}, x_t)$. Display the convergence speed of $\widehat{p}$ and $\widehat{r}$ towards p and r in a curve plot using the infinite norm through a growing trajectory generated by a random uniform policy. Compute $\widehat{Q}$ by using $\widehat{r}(x,u)$ and $\widehat{p}(x'|x,u)$ that estimate the MDP structure. Display, in your report, the infinite norm between $\hat{Q}$ and Q for each trajectory length in a table in your report. Derive $\widehat{\mu}^*$ from $\widehat{Q}$ and display it in a table in your report. Display $J^N_{\widehat{\mu}^*}$, along with $J^N_{\mu^*}$, for each state x in a table in your report. Explain the influence of the length of the trajectory on the quality of the approximation $\hat{Q}$.

5 Q-learning in a Batch Setting (8 points)

5.1 Offline Q-learning (2 points)

Sample a trajectory from a random uniform policy with a finite, large enough time horizon T. Motivate your choice of T. Implement a routine which iteratively updates $\widehat{Q}(x_k, u_k)$ with *Q-learning* from this trajectory. Run your routine with a constant learning rate $\alpha = 0.05$. Derive directly $\widehat{\mu}^*$ from $\widehat{Q}$ and display it in a table in your report. Display $J^N_{\widehat{\mu}^*}$, along with $J^N_{\mu^*}$, for each initial state x, in a table in your report.

5.2 Online Q-learning (4 points)

Implement an intelligent agent which learns Q with Q-learning using an ϵ-greedy policy, *i.e.* a policy that is greedy with a probability of $(1-\epsilon)$ and random otherwise. In the following, a replay buffer is defined as a data structure of previously seen transitions *i.e.* (s_t, u_t, r_t, s_{t+1}).

The first experimental protocol is the following. The agent trains over 100 episodes having 1000 transitions in the domain instance described in Figure 1. An episode always starts from the initial state (see Figure 1). The learning rate α is equal to 0.05 and the exploration rate ϵ is equal to 0.5. The values of α and ϵ are both constant over time. The function $\hat{Q}$ is updated after every transition. The transitions are used only once for updating $\hat{Q}$ *i.e.* there are not stored in a replay buffer.

The second experimental protocol differs from the first one only by the learning rate which is such that $\alpha_0 = 0.05$ and $\forall t > 0, \alpha_t = 0.8\alpha_{t-1}$.

The third experimental protocol is identical from the first one except that the one-step system transitions are stored in a replay buffer and that at each time-step, the function $\hat{Q}$ is updated ten times by drawing ten transitions at random from the replay buffer.

Run the three experimental protocols and display for each of them, after each episode, the value of $\|J^N_{\mu_{\hat{Q}}} - J^N_{\mu^*}\|_\infty$ – where $\mu_{\hat{Q}}$ is the policy derived from $\hat{Q}$ – in a plot. Compare the results obtained. You might also plot $\|J^N_{\mu_{\hat{Q}}} - J^N_{\mu^*}\|_2$ after each episode if needed.

5.3 Discount Factor (2 points)

Rerun the first abovementioned experimental protocols with $\gamma = 0.4$. What do you observe in terms of convergence rate of the algorithm expressed as the speed at which $\hat{Q}$ converges to the Q-function of the new domain ?
NB: Please discuss the results even tough you see no results. We are more interested in the discussion and why something is or is not working than the results itself

5.4 A policy optimization algorithm: SARSA (Bonus, 2 points)

Implement the algorithm SARSA[1] and discuss its difference with Online Q-learning. Also, discuss its results. **As this exercice is a Bonus, no questions will be answered by the TA. You can see this as a research exercise.**

[1] https://en.wikipedia.org/wiki/State%E2%80%93action%E2%80%93reward%E2%80%93state%E2%80%93action