University of Liège

Assignment 1

Optimal decision making for complex problems

Maxime Meurisse (s161278)
François Rozet (s161024)

Academic year 2020-2021

1 Implementation of the Domain

Our rule-based policy is defined as a *clockwise-lap stationary policy*, *i.e.* the agent follows the domain border in a clockwise fashion when it is on a border and goes right everywhere else. Formally,

$$\mu(x, y) = \begin{cases} (1, 0) & \text{if } x < n - 1 \text{ and } y = m - 1 \\ (0, -1) & \text{if } x = n - 1 \text{ and } y > 0 \\ (-1, 0) & \text{if } x > 0 \text{ and } y = 0 \\ (0, 1) & \text{else} \end{cases} \tag{1}$$

where $(x, y) \in X$ is the state of the agent and $n \times m$ the size of the domain instance.

In the deterministic domain, the action proposed by the policy is always applied. However, in the stochastic domain, it is applied only if the noise $w \leq 0.5$. Otherwise, the state is updated to $(x, y) = (0, 0)$.

Because w is draw randomly from $\mathcal{U}(0, 1)$, both events have the same probability (0.5). Hence, this mechanism is equivalent to choosing w as 0 or 1, with an equal probability. Formally, $w \sim \mathcal{U}\{0, 1\}$. Furthermore, the deterministic domain can be seen as a specialization of the stochastic one where w is always 0 ($w \sim \mathcal{U}\{0\}$). These considerations allow to simplify the implementation of the domain components.

The policy (1) has been simulated in the two domains through a single trajectory of 11 steps (x_0 to x_{11}), starting at initial state $x_0 = (3, 0)$, which is illustrated in Listing 1.

```
Deterministic domain
--------------------
((3, 0), (-1, 0), 5.0, (2, 0))
((2, 0), (-1, 0), 6.0, (1, 0))
((1, 0), (-1, 0), -3.0, (0, 0))
((0, 0), (0, 1), 1.0, (0, 1))
((0, 1), (0, 1), -5.0, (0, 2))
((0, 2), (0, 1), 0.0, (0, 3))
((0, 3), (0, 1), 19.0, (0, 4))
((0, 4), (1, 0), 10.0, (1, 4))
((1, 4), (1, 0), -8.0, (2, 4))
((2, 4), (1, 0), -5.0, (3, 4))
((3, 4), (1, 0), 9.0, (4, 4))

Stochastic domain
-----------------
((3, 0), (-1, 0), 5.0, (2, 0))
((2, 0), (-1, 0), 6.0, (1, 0))
((1, 0), (-1, 0), -3.0, (0, 0))
((0, 0), (0, 1), 1.0, (0, 1))
((0, 1), (0, 1), -5.0, (0, 2))
((0, 2), (0, 1), -3.0, (0, 0))
((0, 0), (0, 1), 1.0, (0, 1))
((0, 1), (0, 1), -5.0, (0, 2))
((0, 2), (0, 1), -3.0, (0, 0))
((0, 0), (0, 1), -3.0, (0, 0))
((0, 0), (0, 1), 1.0, (0, 1))
```

Listing 1 – Simulated trajectories by applying policy (1) in both domains.

2 Expected Return of a Policy

From the theory [1], we know that

$$\|J^\mu - J_N^\mu\|_\infty \leq \frac{\gamma^N}{1-\gamma} B_r \tag{2}$$

where $B_r = \|r\|_\infty$ and $\gamma \in [0; 1)$ is the discount factor. It shows that we can estimate, up to a certain precision, J^μ by the functions J_N^μ, which are defined by the recurrence equation

$$J_N^\mu(x) = \underset{w \sim p_w(\cdot|x,u)}{E} \Big[r(x, \mu(x), w) + \gamma J_{N-1}^\mu(f(x, \mu(x), w)) \Big], \quad \forall N \geq 1 \tag{3}$$

where $J_0^\mu(x) \equiv 0$. To implement J_N^μ, we decided to follow the dynamic programming (DP) principles and built a routine (a function) that iteratively computes J_{i+1}^μ (for all $x \in X$) based on the values of J_i^μ. Also, since we know that $w \sim \mathcal{U}\{0, 1\}$ (or $\mathcal{U}\{0\}$), the expectations were computed by averaging over the possible values of w (either 0 or 1).

For the choice of N, we decided to bound the infinite norm of the difference between J^μ and J_N^μ by the threshold $\epsilon = 10^{-3}$ since this value is numerically insignificant with respect to the rewards. Hence, we have to find N such that

$$\begin{aligned} \frac{\gamma^N}{1-\gamma} B_r &\leq \epsilon \\ \gamma^N &\leq \frac{\epsilon}{B_r}(1-\gamma) \\ N &\geq \log_\gamma \left(\frac{\epsilon}{B_r}(1-\gamma) \right) \end{aligned} \tag{4}$$

The terms γ and B_r being fixed at 0.99 and 19, respectively, we derive the smallest possible value of N, *i.e.*

$$N = \left\lceil \log_\gamma \left(\frac{\epsilon}{B_r}(1-\gamma) \right) \right\rceil = 1439. \tag{5}$$

For both domains, we computed $J_N^\mu(x, y)$ for the stationary policy (1) at each state $(x, y) \in X$ (*cf.* Tables 1 and 2).

x \ y	**0**	**1**	**2**	**3**	**4**
0	−48.474	−49.974	−45.428	−45.887	−65.542
1	−50.989	−47.328	−55.887	−65.542	−76.305
2	−44.479	−69.797	−74.542	−76.305	−68.995
3	−39.035	−44.812	−49.305	−68.995	−64.642
4	−32.644	−52.318	−68.795	−72.107	−74.386

Table 1 – $J_N^\mu(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1439$.

x \ y	**0**	**1**	**2**	**3**	**4**
0	−121.103	−121.529	−116.330	−110.876	−119.052
1	−122.892	−114.805	−115.876	−119.052	−126.476
2	−119.277	−120.604	−123.552	−126.476	−123.294
3	−117.988	−115.369	−112.976	−123.294	−119.895
4	−116.850	−129.286	−133.943	−129.747	−127.171

Table 2 – $J_N^\mu(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1439$.

3 Optimal Policy

The equivalent *Markov Decision Process* (MDP) of the domain(s) can be obtained by computing the transition probability $p(x' \mid x, u)$ and the expected reward $r(x, u)$ using, respectively,

$$p\left(x' \mid x, u\right) = \underset{w \sim p_w(\cdot|x,u)}{E}\left[I_{\{x'=f(x,u,w)\}}\right] \quad \forall x, x' \in X, u \in U \tag{6}$$

$$r(x, u) = \underset{w \sim p_w(\cdot|x,u)}{E}[r(x, u, w)] \quad \forall x \in X, u \in U \tag{7}$$

where $I_{\{logical\}} = 1$ if *logical* is true and 0 else. Computing the transition probability and the expected reward is straight forward.

Then, the state-action value functions $Q_N(x, u)$ are evaluated with the recurrence equation

$$Q_N(x, u) = r(x, u) + \gamma \sum_{x' \in X} p(x' \mid x, u) \max_{u' \in U} Q_{N-1}\left(x', u'\right), \quad \forall N \geq 1 \tag{8}$$

with $Q_0(x, u) \equiv 0$. Similarly to J_N^μ, we used to DP principles to implement a routine generating the sequence of Q-functions. Interestingly, equation (8) can be expressed as basic matrix operations like element-wise sums and scalar products, which we took advantage of in order to make the routine efficient.

Concerning N, we have to find N such that $\mu_{N'}^* \equiv \mu_N^*$ for all $N' \geq N$ where

$$\mu_N^*(x) \in \arg\max_{u \in U} Q_N(x, u). \tag{9}$$

At first sight, we could think that finding the smallest N such that $\mu_{N+1}^* \equiv \mu_N^*$ is sufficient, but it is not. In fact, there is no way to guarantee that the inferred policy won't change latter on. There even could be an infinite periodic oscillation between different policies having the same return. However, we have the relation

$$\left\| J^{\mu^*} - J^{\mu_N^*} \right\|_\infty \leq \frac{2\gamma^N}{(1-\gamma)^2} B_r \tag{10}$$

to impose a bound on the difference of the returns from μ_N^* and μ^*. With the same development (and parameters) as (4), we obtain

$$N = \left\lceil \log_\gamma \left(\frac{\epsilon}{2B_r}(1-\gamma)^2 \right) \right\rceil = 1966. \tag{11}$$

The inferred policies μ_N^* for both domains are presented in Tables 3 and 4.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
1	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
2	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(-1, 0)
3	(-1, 0)	(0, 1)	(0, 1)	(-1, 0)	(-1, 0)
4	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(0, -1)

Table 3 – $\mu_N^*(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1966$.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(-1, 0)
1	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
2	(-1, 0)	(0, 1)	(-1, 0)	(1, 0)	(-1, 0)
3	(0, -1)	(0, 1)	(0, 1)	(0, -1)	(0, -1)
4	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(1, 0)

Table 4 – $\mu_N^*(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1966$.

We also computed $J_N^{\mu_N^*}$ with the previously implemented routine (*cf.* Tables 5 and 6).

x \ y	**0**	**1**	**2**	**3**	**4**
0	1842.031	1857.19	1881	1900	1900
1	1854.576	1870.279	1880.09	1891	1900
2	1842.031	1855.576	1870.279	1881.09	1891
3	1828.61	1848.01	1863.646	1863.279	1864.09
4	1816.324	1826.52	1849.01	1863.646	1842.01

Table 5 – $J_N^{\mu_N^*}(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1966$.

x \ y	**0**	**1**	**2**	**3**	**4**
0	159.446	159.637	163.052	172.13	172.13
1	159.637	163.052	164.903	167.63	172.13
2	159.446	160.137	163.052	167.213	167.63
3	159.259	162.196	167.313	162.196	167.213
4	159.259	155.713	162.196	167.213	162.229

Table 6 – $J_N^{\mu_N^*}(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1966$.

4 System Identification

From a given trajectory $h_t = (x_0, u_0, r_0, x_1, u_1, r_1, \ldots, u_{t-1}, r_{t-1}, x_t)$, we can estimate $r(u, x)$ and $p(x' \mid x, u)$ by using, respectively,

$$\hat{r}(x, u) = \frac{1}{|A_h(x, u)|} \sum_{i \in A_h(x,u)} r_i, \tag{12}$$

$$\hat{p}(x' \mid x, u) = \frac{1}{|A_h(x, u)|} \sum_{i \in A_h(x,u)} I_{\{x_{i+1}=x'\}}, \tag{13}$$

where $A_h(x, u)$ is the set of indices $\{i \mid x_i = x, u_i = u\}$. However, this procedure is computationally expensive. Instead, we implemented the algorithm described in the course (slide 29 of the first lecture) to update iteratively $\hat{r}$ and $\hat{p}$.

However, this algorithm doesn't handle the case where a state-action pair (x, u) isn't represented in the trajectory. In this case, we assumed that the transitions from (x, u) to any state x' were equally likely, *i.e.*

$$p(x' \mid x, u) = \frac{1}{|X|} = \frac{1}{nm} \quad \forall x' \in X. \tag{14}$$

We applied the algorithm on a fixed trajectory h_T of length $T = 10^6$, generated by a random uniform policy. To study the convergence of $\hat{r}$ and $\hat{p}$ towards r and p, we computed them on truncated sections h_t of h_T, with $t = 1, 100, 1000, \ldots, 10^6$. The convergence of $\hat{p}$ and $\hat{r}$ towards p and r, for both domains, are shown in Figures 1 and 2.

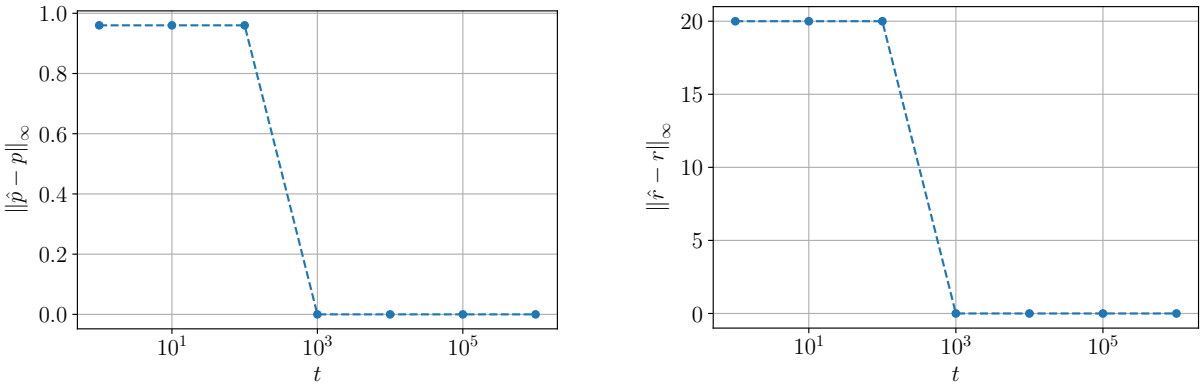


Figure 1 – Convergence of $\hat{p}$ and $\hat{r}$ in the deterministic domain.

In the deterministic domain, both $\hat{p}$ and $\hat{r}$ converged after 10^3 steps in the trajectory.

In the stochastic domain, even after 10^6 steps, $\hat{p}$ and $\hat{r}$ have not yet fully converged. This behavior is easily explained. In order to have accurate estimates of r and p, the trajectory needs to visit at least a few times every state-action pairs. Using the Markov chain theory, one can easily prove that, in the deterministic domain, a random uniform policy will have visited all pairs with high probability in the order of 10^3 steps. But, in the stochastic case, there is a 0.5 chance to be teleported to $(0, 0)$ at all time. Hence, the probability to reach the furthest states (*e.g.* $(4, 4)$) is significantly lower. Using the same theory, it can be shown that it requires to the order of 10^7 steps to have visited every pair with

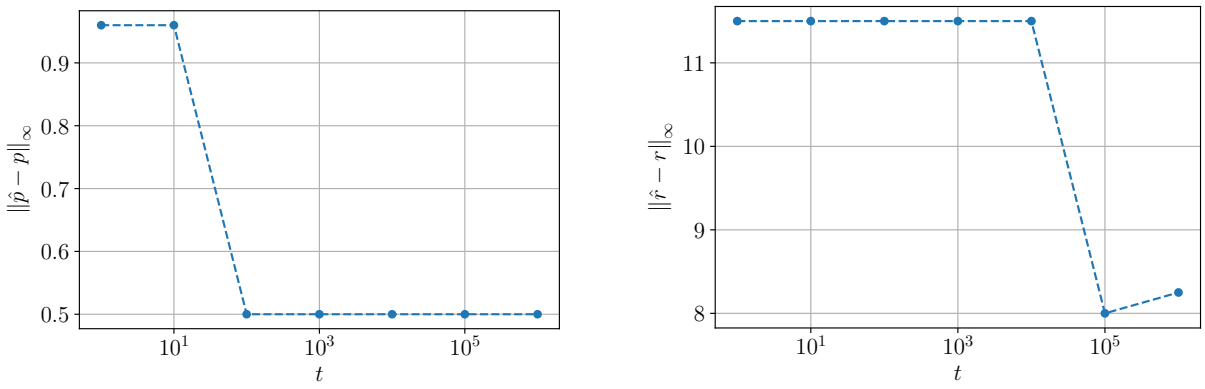


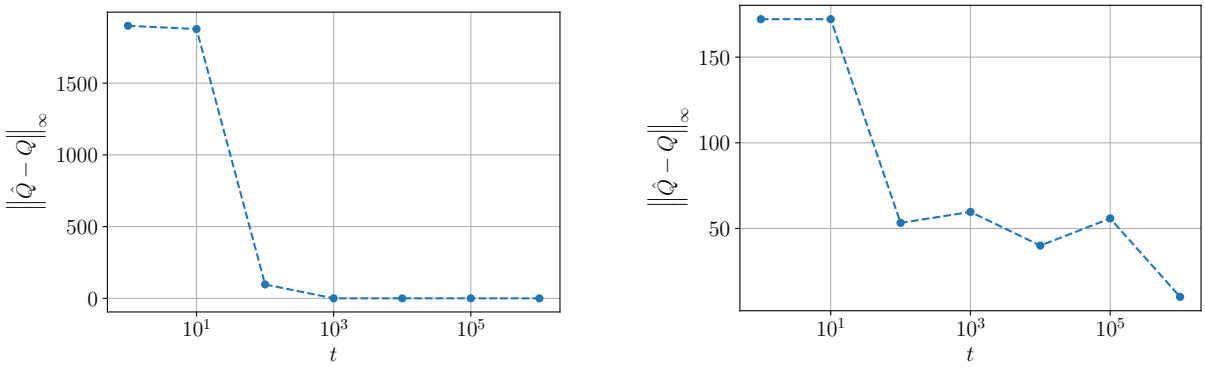
Figure 2 – Convergence of $\hat{p}$ and $\hat{r}$ in the stochastic domain.

high probability, which would have required a lot of memory (assuming the trajectory is stored) and computations. Furthermore, increasing the number of visits, and therefore the trajectory length, will improve $\hat{r}$ and $\hat{p}$ as they are approximations of expectations. Although, in the deterministic domain, a single visit is enough. In our case, we settled for a trajectory length of 10^6 as it is large enough to observe a bit of convergence in the stochastic domain.

From estimates $\hat{r}$ and $\hat{p}$, we can compute the $\hat{Q}$-functions exactly like Q-functions by replacing r and p by respectively $\hat{r}$ and $\hat{p}$ in relation (8).

Using the routine(s) implemented in previous sections, we computed $\hat{Q}_N$ with $N = 1966$ from relation (11). Once again, to study its convergence towards Q_N, we computed $\hat{Q}_N$ on the same truncated sections h_t of h_T.

The infinite norms between $\hat{Q}_N$ and Q_N on both domains are shown in Figure 3.



(a) Deterministic domain.

(b) Stochastic domain.

Figure 3 – Convergence of $\hat{Q}_N$ towards Q_N.

Again, one can see that, in the deterministic domain, $\hat{Q}_N$ converged to Q_N after 10^3 steps while in the stochastic case, it didn't even after 10^6 steps.

An approximated optimal policy $\hat{\mu}^*_N$ can be directly derived by replacing Q_N by $\hat{Q}_N$ in relation (9) and the best approximation is reached when $\hat{Q}_N$ has converged to Q_N. For the deterministic case, this length can be chosen as 10^3 but, for the sake of simplicity, we computed $\hat{Q}_N$ with the highest trajectory length (10^6) for both domains.

The obtained policies are shown in Tables 7 et 8.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
1	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
2	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(-1, 0)
3	(-1, 0)	(0, 1)	(0, 1)	(-1, 0)	(-1, 0)
4	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(0, -1)

Table 7 – $\hat{\mu}^*_N(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1966$.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(1, 0)	(1, 0)	(0, 1)	(0, 1)
1	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
2	(-1, 0)	(0, 1)	(1, 0)	(1, 0)	(-1, 0)
3	(0, -1)	(0, 1)	(0, 1)	(-1, 0)	(1, 0)
4	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(1, 0)

Table 8 – $\hat{\mu}^*_N(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1966$.

Using the routine inspired from (3), we get $J_N^{\mu^*_N}$ (*cf.* Tables 5 and 6), previously computed, along with $J_N^{\hat{\mu}^*_N}$ (*cf.* Tables 9 and 10) for each state (x, y) in both domains.

x \ y	**0**	**1**	**2**	**3**	**4**
0	1842.031	1857.19	1881	1900	1900
1	1854.576	1870.279	1880.09	1891	1900
2	1842.031	1855.576	1870.279	1881.09	1891
3	1828.61	1848.01	1863.646	1863.279	1864.09
4	1816.324	1826.52	1849.01	1863.646	1842.01

Table 9 – $J_N^{\hat{\mu}^*_N}(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1966$.

x \ y	**0**	**1**	**2**	**3**	**4**
0	159.446	159.637	163.052	172.13	172.13
1	159.637	163.052	164.903	167.63	172.13
2	159.446	159.472	161.709	166.229	167.63
3	159.259	161.709	166.229	160.209	162.229
4	159.259	155.472	161.709	166.229	162.229

Table 10 – $J_N^{\hat{\mu}^*_N}(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1966$.

One can see that values of $J_N^{\mu_N^*}$ and $J_N^{\hat{\mu}_N^*}$ are the same in the deterministic domain while they are slightly different in the stochastic case. This can be explained by the fact that, because $\hat{Q}_N$ did not converged to Q_N, the policy $\hat{\mu}_N^*$ isn't equivalent to μ_N^* in the stochastic domain.

5 Q-learning in a Batch Setting

From a trajectory $h_t = (x_0, u_0, r_0, x_1, u_1, r_1, \ldots, u_{t-1}, r_{t-1}, x_t)$, we can directly infer $\hat{Q}$, without estimating parameters of the equivalent MDP, with the *Q-learning* algorithm, presented in the course (slide 30).

Method 1. Q-learning algorithm

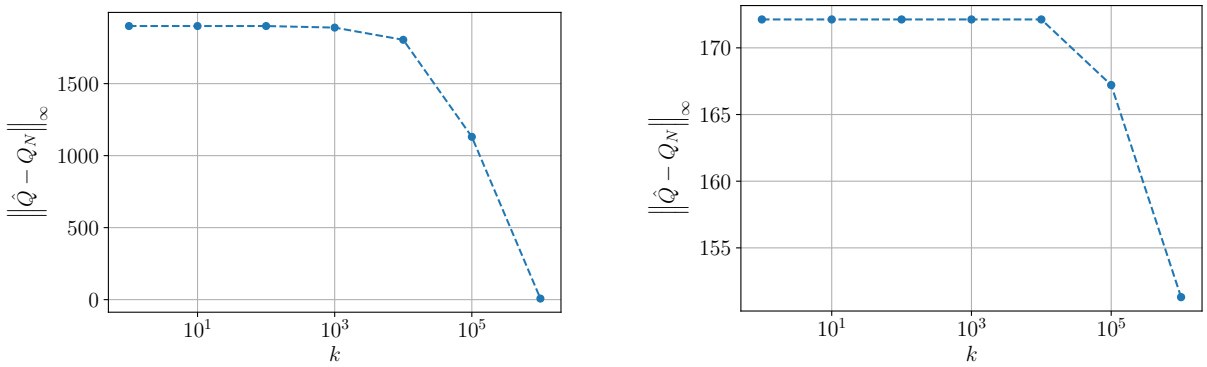
1. Initialize $\hat{Q}(x, u)$ to 0 everywhere and set $k = 0$.
2. Until $k = t$,

$$\hat{Q}(x_k, u_k) \leftarrow (1 - \alpha_k)\hat{Q}(x_k, u_k) + \alpha_k \left(r_k + \gamma \max_{u \in U} \hat{Q}(x_{k+1}, u) \right)$$
$$k \leftarrow k + 1$$

If the trajectory h_t is computed and stored in advance, the algorithm is known as *offline* Q-learning. Conversely, if the one-step transitions $l_k = (x_k, u_k, r_k, x_{k+1})$ are computed during the learning, and possibly depend on the values of $\hat{Q}$, the algorithm is known as *online* Q-learning.

5.1 Offline Q-learning

Like in the previous section, we simulated a trajectory h_T of length $T = 10^6$ with a random uniform policy. We then applied our implementation of 1. To observe the converge of $\hat{Q}$ towards Q_N, we computed $\left\|\hat{Q} - Q_N\right\|_\infty$ when $k = 1, 10, \ldots, 10^6$ (*cf.* Figure 4).



(a) Deterministic domain.

(b) Stochastic domain.

Figure 4 – Convergence of $\hat{Q}$ towards Q_N; $N = 1966$.

One can see that the chosen time horizon $T = 10^6$ is (just) sufficient to have convergence of $\hat{Q}$ towards Q_N in the deterministic domain. Conversely, it seems convergence was not

reached in the stochastic domain as the infinite norm barely decreased. As before, for the sake of simplicity, we kept the last of $\hat{Q}$ for the rest of our calculations.

An approximated optimal policy $\hat{\mu}^*$ can be directly derived from $\hat{Q}$ by replacing Q_N by $\hat{Q}$ in relation (9). The obtained policies, for both domains, are shown in Tables 11 et 12.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
1	(0, 1)	(0, 1)	(0, 1)	(0, 1)	(-1, 0)
2	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(-1, 0)
3	(-1, 0)	(0, 1)	(0, 1)	(-1, 0)	(-1, 0)
4	(-1, 0)	(0, 1)	(-1, 0)	(-1, 0)	(0, -1)

Table 11 – $\hat{\mu}^*(x, y)$ for all $(x, y) \in X$ in the deterministic domain.

x \ y	**0**	**1**	**2**	**3**	**4**
0	(1, 0)	(1, 0)	(1, 0)	(1, 0)	(-1, 0)
1	(1, 0)	(0, -1)	(0, 1)	(0, -1)	(-1, 0)
2	(-1, 0)	(0, -1)	(-1, 0)	(-1, 0)	(-1, 0)
3	(-1, 0)	(0, -1)	(-1, 0)	(0, -1)	(0, -1)
4	(-1, 0)	(-1, 0)	(-1, 0)	(0, 1)	(1, 0)

Table 12 – $\hat{\mu}^*(x, y)$ for all $(x, y) \in X$ in the stochastic domain.

Using the routine inspired from (3), we get $J_N^{\mu_N^*}$ (*cf.* Tables 5 and 6), previously computed, along with $J_N^{\hat{\mu}^*}$ (*cf.* Tables 13 and 14) for each state (x, y) in both domains.

x \ y	**0**	**1**	**2**	**3**	**4**
0	1842.031	1857.19	1881	1900	1900
1	1854.576	1870.279	1880.09	1891	1900
2	1842.031	1855.576	1870.279	1881.09	1891
3	1828.61	1848.01	1863.646	1863.279	1864.09
4	1816.324	1826.52	1849.01	1863.646	1842.01

Table 13 – $J_N^{\hat{\mu}^*}(x, y)$ for all $(x, y) \in X$ in the deterministic domain; $N = 1966$.

Once again, due to the convergence issue, one can see that values of $J_N^{\mu_N^*}$ and $J_N^{\hat{\mu}_N^*}$ are the same in the deterministic domain while they are different in the stochastic case.

5.2 Online Q-learning

For this section, the trajectory has to be generated according to an ϵ-greedy policy, *i.e.* an exploratory policy that selects with a probability ϵ a random action and a probability

x \ y	**0**	**1**	**2**	**3**	**4**
0	133.445	132.11	136.081	136.415	146.644
1	133.11	133.445	136.415	136.081	146.644
2	133.445	133.11	136.081	136.415	142.144
3	133.11	133.445	133.915	132.843	139.813
4	133.445	126.11	132.843	136.743	136.743

Table 14 – $J_N^{\hat{\mu}^*}(x, y)$ for all $(x, y) \in X$ in the stochastic domain; $N = 1966$.

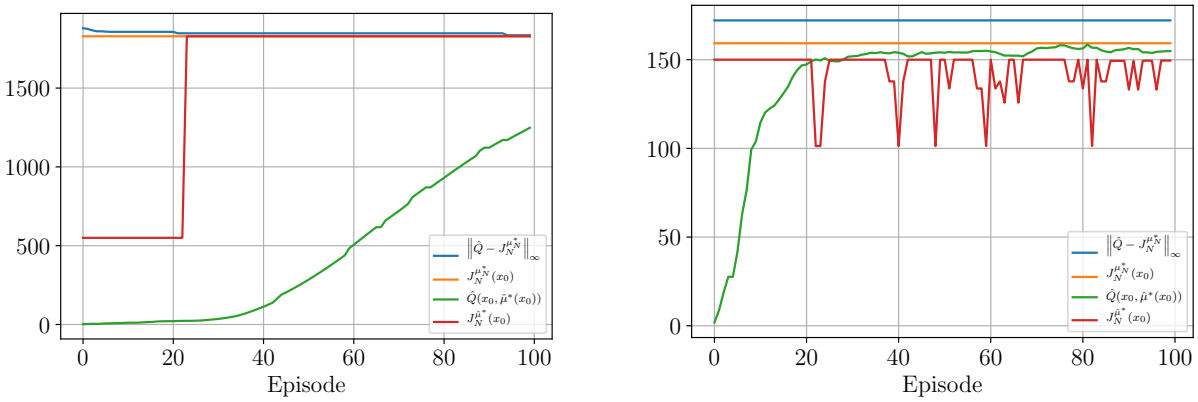
$1 - \epsilon$ the action recommended by the approximated policy $\hat{\mu}^*$.

$$\begin{cases} u_k \sim \mathcal{U}\{U\} & \text{with probability } \epsilon \\ u_k = \hat{\mu}^*(x_k) & \text{else} \end{cases} \tag{15}$$

Because this policy uses $\hat{\mu}^*$, it requires to know the *current* values of $\hat{Q}$. Therefore, we have to implement an online Q-learning protocol. In order to keep the code simple, we implemented this protocol as a function with a few parameters such that the three requested experiments were easy to setup and repeat.

It was also requested to measure the performance of the agent(s) with the metric $\left\|\hat{Q} - J_N^{\mu_N^*}\right\|_\infty$ after each episode. However, Q and J_N^μ don't have the same domain and therefore shouldn't be compared directly with the infinite norm. To allow this comparison, we decided to *project* $\hat{Q}$ with respect to $\hat{\mu}^*$, *i.e.*

$$\left\|\hat{Q} - J_N^{\mu_N^*}\right\|_\infty = \sup_{x \in X} \left|\hat{Q}(x, \hat{\mu}^*(x)) - J_N^{\mu_N^*}(x)\right|. \tag{16}$$



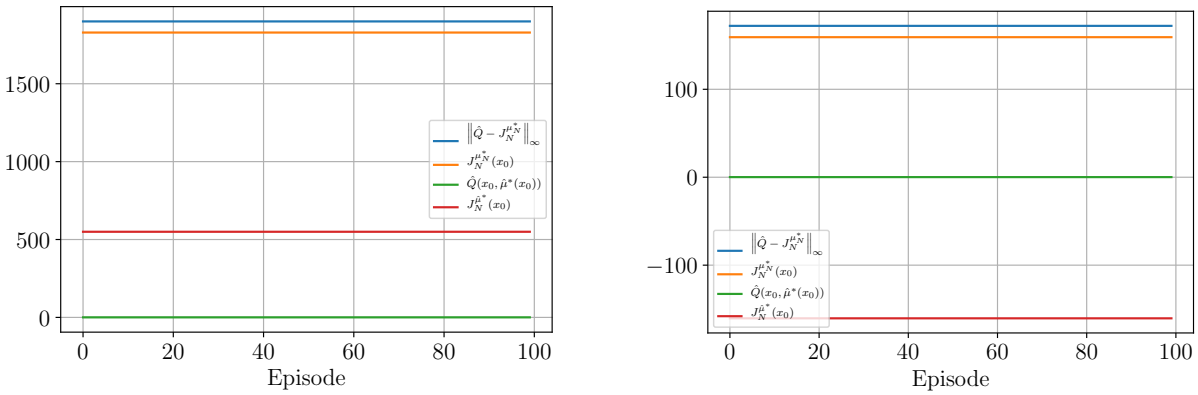
(a) Deterministic domain.

(b) Stochastic domain.

Figure 5 – Performance of the agent for the first experimental protocol.

Unfortunately, as can be seen in Figures 5, 6 and 7, this metric doesn't seem to converge towards zero as the agent learns. In fact, this metric isn't well suited to measure the performance of the agent(s).

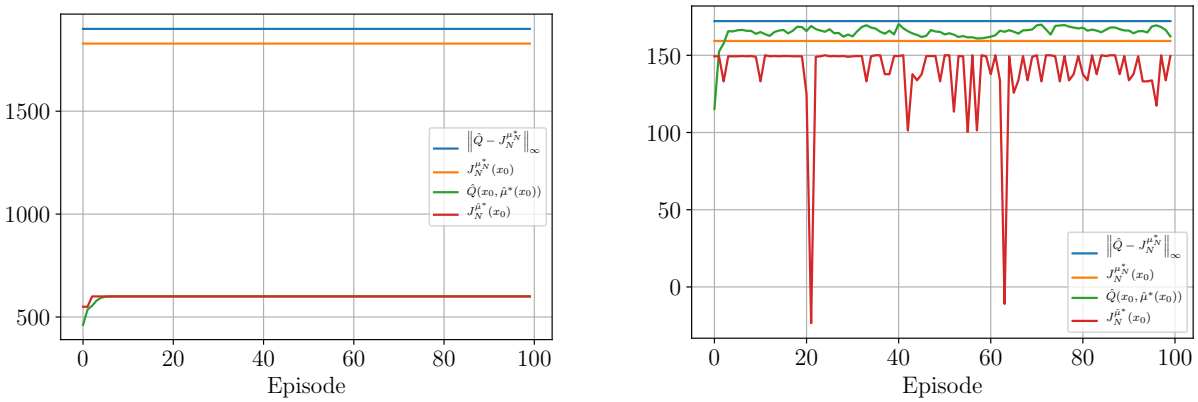
Indeed, a drawback of the ϵ-greedy policy is that it will much more rarely explore state-action pairs (x, u) leading to low rewards than a random uniform policy. As a consequence,



(a) Deterministic domain.

(b) Stochastic domain.

Figure 6 – Performance of the agent for the first experimental protocol.



(a) Deterministic domain.

(b) Stochastic domain.

Figure 7 – Performance of the agent for the first experimental protocol.

the value of $\hat{Q}(x, u)$ for such pair is very far from $J_N^{\mu_N^*}(x)$ since the algorithm 1 requires a large (infinite) number of visits to be accurate.

However, the ϵ-greedy policy has the advantage to explore quite well the "good" trajectories. Therefore, as the initial state is $x_0 = (3, 0)$, it should determine accurately the expected return of this state and the optimal policy starting from it. These two quantities are respectively measured as $\hat{Q}(x_0, \hat{\mu}^*(x_0))$ and $J_N^{\hat{\mu}^*}(x_0)$ and should ideally both converge towards $J_N^{\mu_N^*}(x_0)$.

As can be seen in Figure 5, these two quantities do seem to converge in the first experiment although less consistently in the stochastic domain. For the second experiment, however, no convergence can be observed whatsoever. This can be easily explained as α_k decreases exponentially[1] quickly with k. Indeed, by $k = 100$, *i.e.* a tenth of the first episode, $\alpha_k \simeq 10^{-11}$ which is numerically insignificant. Hence, $\hat{Q}$ is only modified on the first few updates.

For the third experiment (*cf.* Figure 7), the metrics seem to reach a plateau after a few episodes. It could be explained by the fact that, in the buffer, which is simply the

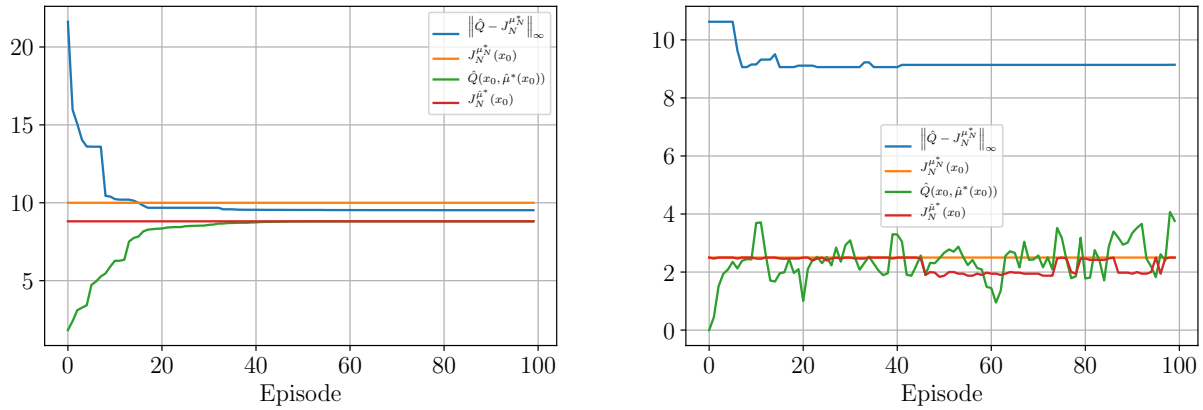
[1]In fact, $\alpha_k = 0.8^k \alpha_0$ isn't a valid strategy as it is required that $\sum_{k=0}^{T} \alpha_k$ diverges to ∞ as T grows.

past trajectory, a lot of sub-optimal transitions are represented since they were chosen (partly) using a sub-optimal $\hat{Q}$. Therefore, sampling from this buffer further anchors those transitions in $\hat{Q}$ and a lock is reached. Theoretically, this lock should eventually resolve as the values of $\hat{Q}$ cannot grow past $\frac{B_r}{1-\gamma}$.

A way to resolve the lock faster, is to increase the stochasticity of the transitions. For example, one could increase ϵ to 0.5, reduce the number of transitions sampled from the buffer, limit the size of the buffer, clear the buffer after each episode, etc. We tried some of these in our implementation but didn't have the time to analyze them.

5.3 Discount factor

As requested, we have re-run the first experimental protocol of Section 5.2 with $\gamma = 0.4$ (*cf.* Figure 8).



(a) Deterministic domain.

(b) Stochastic domain.

Figure 8 – Performance of the agent for the first experimental protocol; $\gamma = 0.4$.

It seems like the metrics converge faster towards their goal. It might be explained by the fact that the goal is lower and can be attained in shorter trajectories as γ is much lower.

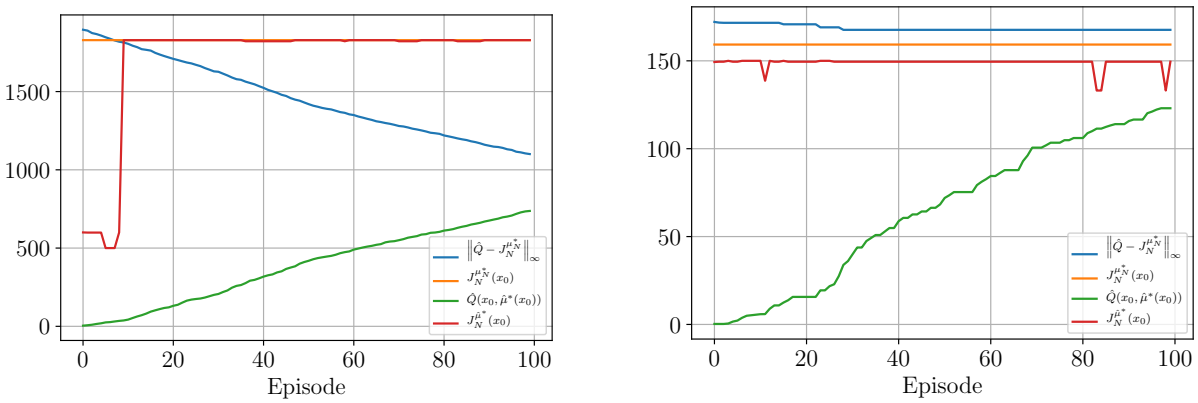
5.4 Q-learning with Another Exploration Policy

We choose, as an other exploration policy, the *Boltzmann exploration policy*. In this policy, the probability of selecting an action u in a state x is given by

$$\frac{\tau \exp \hat{Q}(x,u)}{\tau \sum_{u' \in U} \exp \hat{Q}(x,u')}. \tag{17}$$

In our implementation we chose $\tau = \frac{1-\gamma}{B_r}$ as it is (the inverse of) a bound for the values of Q. This prevents to have completely skewed distributions. This policy is more stochastic that the ϵ-greedy one, and therefore should converge more consistently.

We have re-run the first experimental protocol of Section 5.2 with this new exploration policy (*cf.* Figure 9).



(a) Deterministic domain.

(b) Stochastic domain.

Figure 9 – Convergence speeds of $\hat{Q}$ towards $J_N^{\mu^*}$ for the first experimental protocol; Boltzmann exploration policy.

As expected, the metrics converge very consistently in both domains, although not as fast as with ϵ-greedy. This is not very surprising as, in our setting, this policy is very similar to a random uniform policy.

References

[1] Damien Ernst. "Optimal sequential decision making for complex problems". URL: http://blogs.ulg.ac.be/damien-ernst/wp-content/uploads/sites/9/2020/02/RL-1.pdf (page 2).