Assignment 1 - Reinforcement Learning in a Discrete Domain

Antoine Louis (s152140)

---

1 Implementation of the domain

We consider the rewards matrix in Figure 1, where the origin cell is located in the upper left corner (corresponding to a reward of -3). To test our implementation, we consider an agent following the simple policy "always go up", that always selects the action (-1, 0). Starting from the initial state in position (3, 0), coloured in red in Figure 1, the cumulative discounted reward that the agent gets after 3 time steps is computed, for both deterministic and stochastic setting (with $\beta = 0.5$ for the stochastic domain). The results are presented in Table 1.

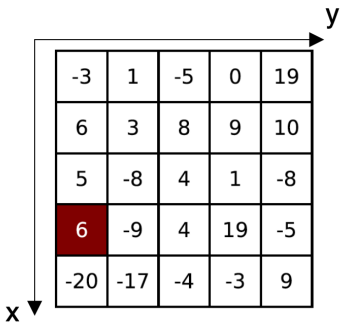


Figure 1 – Rewards matrix

	Sequence of states	Discounted cumulative reward
Deterministic	(3,0) $\rightarrow$ (2, 0) $\rightarrow$ (1, 0) $\rightarrow$ (0, 0)	$0.99^0 * 5 + 0.99^1 * 6 + 0.99^2 * (-3) = 7.9997$
Stochastic	(3,0) $\rightarrow$ (2, 0) $\rightarrow$ (0, 0) $\rightarrow$ (0, 0)	$0.99^0 * 5 + 0.99^1 * (-3) + 0.99^2 * (-3) = -0.9103$

Table 1 – Discounted cumulative reward of a simple agent after 3 time steps.

2 Expected return of a policy

To compute the expected cumulative reward for each state of our domain, the dynamic programming principle can be used, where a sequence of functions J_N ($N = 0, 1, ...$) are computed using the Bellman equations. For a deterministic domain, the function is given by :

$$J_N^\mu(x) = r(x, \mu(x)) + \gamma J_{N-1}^\mu(f(x, \mu(x)))$$

where $r(x, \mu(x))$ and $f(x, \mu(x))$ are respectively the reward signal and the transition function for doing the action told by the policy μ in state x. For a stochastic domain, we have :

$$J_N^\mu(x) = \mathop{\mathrm{E}}_{w \sim P(.|x,u)}[r(x, \mu(x), w) + \gamma J_{N-1}^\mu(f(x, \mu(x), w))]$$

where the disturbance w is drawn from a uniform distribution between [0,1] at each time step. In that case, we now have to consider the expected reward value that the agent will get by following the policy in the stochastic environment. This expected reward can be deduced from

the dynamics of the stochastic setting and will depend on the value of the parameter β such that :

$$E[r(x, \mu(x), w)] = (1 - \beta)r(x, \mu(x)) + \beta r(x_0)$$

where x_0 is the origin cell (0,0) with a reward $r(x_0) = -3$. Indeed, in a given state, the agent will either jump to the origin cell with a probability of β or follow the policy with a probability of $(1 - \beta)$. That way, the closer β will be to 0, the more our agent will behave as if it was in the deterministic environment, always following the given policy. On the contrary, the closer β will be to 1, the more often it will jump to the origin cell (0,0), accumulating the discounted rewards of -3.

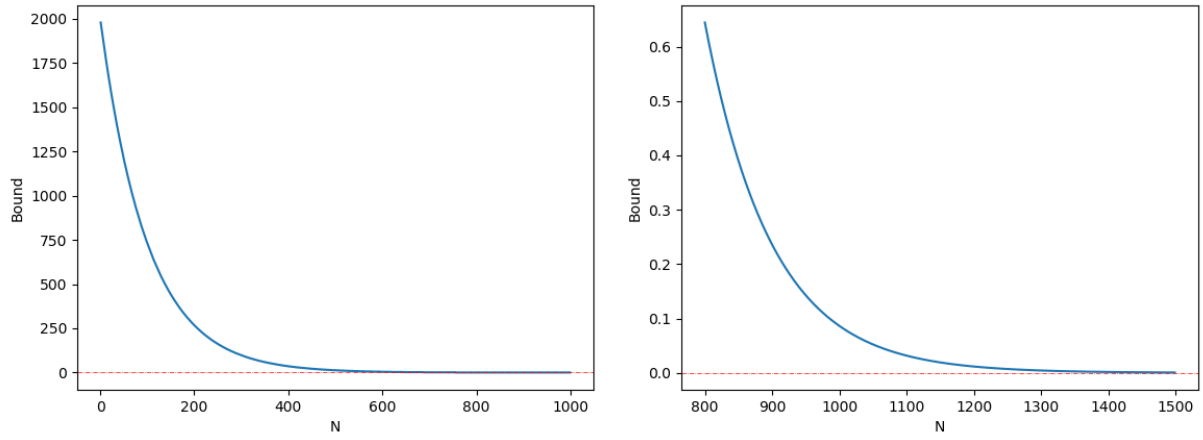
Although the expected return is computed over an infinite time horizon, we can stop the recurrence process after a few iterations. Indeed, from the dynamic programming theory, we will have that :

$$\lim_{N \to \infty} ||J_N^\mu - J^\mu||_\infty \to 0$$

The resulting bound will be :

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

where B_r is a constant that bounds the reward function. By plotting the bound in function of the time horizon, we see in Figure 24 that it progressively converges towards 0 and becomes almost negligible when $N = 1400$. Therefore, we can consider a time horizon of $N = 1400$ to compute the expected return of each state.



(a) From N=0 to N=1000

(b) From N=800 to N=1500

FIGURE 2 – Bound of the expected return depending on the value of the time horizon

The expected return of each state is presented in Figure 3, for both deterministic and stochastic settings (with $\beta = 0.5$ for the stochastic domain).
For the deterministic setting, the expected returns of a given row (same x index) are in general pretty close to each other. This can be explained by the fact that, under the policy "always go right" (policy that always take the action $(0, 1)$), the agent will get "stuck" in the rightmost cell of the corresponding row and will then start to collect the discounted rewards of that cell at infinite.
For the stochastic setting, the expected returns are all pretty much of the same order of magnitude. This is explained by the fact that, half the time ($\beta = 0.5$), the agent exactly behaves the same in each state and jumps to the origin cell to get the discounted -3 reward. As said earlier,

the expected returns will depend on the value of β. In the extreme case of $\beta = 0$, they will all exactly be as those of the deterministic setting. In the extreme case of $\beta = 1$, the agent will always go to the origin cell and the expected return of each state will be equal to the following sum : $\sum_{t=0}^{\infty} 0.99^t * (-3) = -300$.

Deterministic					**Stochastic** (beta=0.5)				
1839.62	1857.19	1881	1900	1900	-72.02	-71.46	-64.25	-54.75	-54.75
990.04	997.01	999	1000	1000	-67.78	-64.91	-64.16	-63.66	-63.66
-779.3	-779.09	-791	-800	-800	-77.41	-73.26	-76.99	-81.49	-81.49
-471.57	-467.24	-476	-500	-500	-75.35	-68.08	-66.52	-78.52	-78.52
849.37	875.12	888	900	900	-82.34	-74.12	-70.65	-64.65	-64.65

FIGURE 3 – Expected return of each state

3 Optimal policy

We now consider the equivalent Markov Decision Process of the domain, defined through the transition probabilities $p(x'|x,u)$ and the reward function $r(x,u)$ $\forall x, x' \in X,\ u \in U$, together with γ and T.
To compute the Q-function for each pair of state-action of our domain, the dynamic programming principle can be used. The functions Q_N for a MDP structure are defined by the following recurrence equation :

$$Q_N(x,u) = r(x,u) + \gamma \sum_{x' \in X} p(x'|x,u) \max_{u' \in U} Q_{N-1}(x',u'), \qquad \forall N \geq 1$$

with $Q_0(x,u) \equiv 0$. Given this estimate of $Q(x,u)$, the optimal policy μ_N^* can be derived such that :

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

The value function $J_N^{\mu^*}$ can also easily be computed for each state of the domain :

$$J_N^{\mu^*}(x) = \max_{u \in U} Q_N(x,u)$$

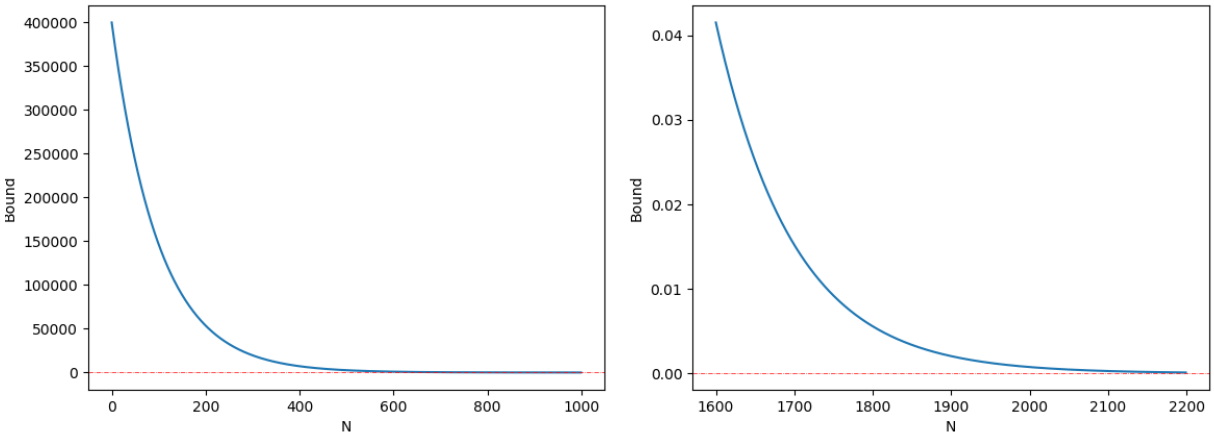
Notice that, although the Q-functions are theoretically computed over an infinite time horizon, we can stop the recurrence process after a few iterations. Indeed, from the dynamic programming theory, we have that :

$$\lim_{N \to \infty} ||Q_N - Q||_\infty \to 0$$

Moreover, the bound on the suboptimality of μ_N^* with respect to μ^* will be :

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

where B_r is a constant that bounds the reward function.
By plotting this bound in function of the time horizon, we see in Figure 24 that it progressively converges towards 0 and becomes almost negligible when $N = 2000$. Therefore, considering a time horizon of $N = 2000$ will lead to a good approximation of the Q-function.

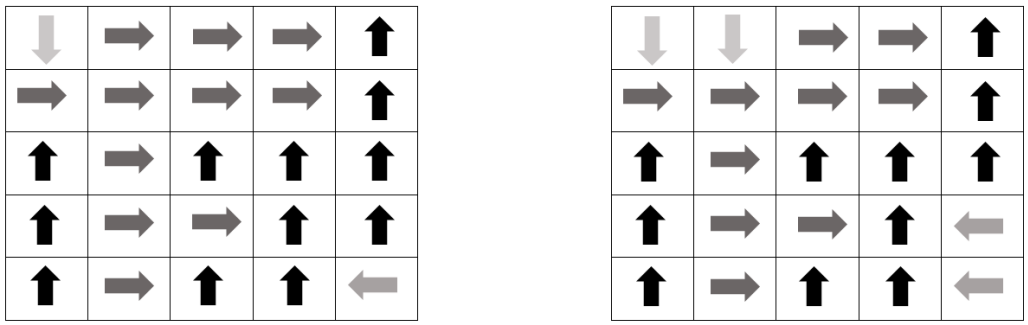


(a) From N=0 to N=1000

(b) From N=1600 to N=2200

FIGURE 4 – Bound of the value function depending on the value of the time horizon

Figure 12 shows the actions given by the optimal policy μ_N^* in each state of the domain for both deterministic and stochastic settings, and Figure 6 presents the corresponding $J_N^{\mu^*}$. Note that these actions will always be the same for the deterministic domain, but they may change in the stochastic domain, depending on the value of β.
The strategy adopted by the optimal policy is clear : move towards the top rightmost cell in $(x, y) = (0, 4)$ and, once there, always go up to accumulate the discounted rewards of 19. That is why, in the deterministic domain, all the $J_N^{\mu^*}$ are approximately equal to : $\sum_{t=0}^{\infty} 0.99^t * 19 = 1900$. In the stochastic domain, the strategy stays the same but the $J_N^{\mu^*}$ values will be lower than 1900 due to the fact that, 10% of the time ($\beta = 0.1$), the agent will jump back to the origin cell in $(x, y) = (0, 0)$, to get a discounted reward of -3.



(a) Deterministic domain ($\beta = 0$)

(b) Stochastic domain ($\beta = 0.1$)

FIGURE 5 – Action given by the optimal policy μ_N^* for each state x

Deterministic domain					Stochastic domain (beta=0.1)				
1842.03	1857.19	1881	1900	1900	1201.92	1209.68	1228.68	1245.78	1245.78
1854.58	1870.28	1881.09	1891	1900	1209.68	1221.43	1229.56	1237.68	1245.78
1842.03	1855.58	1870.28	1881.09	1891	1201.92	1210.58	1221.43	1229.56	1237.68
1828.61	1849.01	1863.65	1863.28	1864.09	1194.1	1207.94	1218.47	1215.13	1218.47
1816.32	1826.52	1849.01	1863.65	1842.01	1188.03	1191.37	1207.94	1218.47	1201.64

FIGURE 6 – $J_N^{\mu^*}$ values of each state

4 System identification

In the system identification phase, we must now estimate an approximate of the reward function $\hat{r}(x,u)$ and an approximate of the probability distribution $\hat{p}(x'|x,u)$, given one (or multiple) trajectory $h_t = x_0u_0r_0x_1...x_{t-1}u_{t-1}r_{t-1}x_t$. A trajectory is generated by first choosing randomly an initial state in the domain and then, by selecting at each time step a random action in order for our agent to explore the domain and collect information on it. This information will then serve to compute the estimations of the reward function and the probability distribution. Indeed, an estimation of the reward function can be computed such that :

$$\hat{r}(x,u) = \frac{\sum_{k\in A(x,u)} r_k}{\#A(x,u)}$$

where $A(x,u) = \{k \in \{0,1,...,t-1\} | (x_k,u_k) = (x,u)\}$ and $r_{k1}, r_{k2}, ..., r_{k\#A(x,u)}$ are values of the random variable $r(x,u,w)$ which are drawn independently. On the same way, an estimation of the probability distribution can be computed such that :

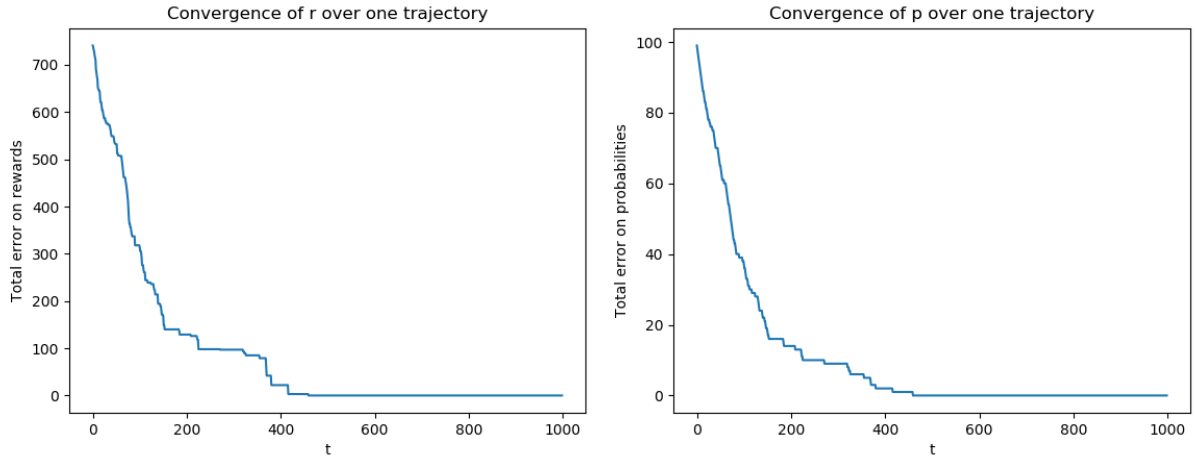
$$\hat{p}(x'|x,u) = \frac{\sum_{k\in A(x,u)} I_{\{x_{k+1}=x'\}}}{\#A(x,u)}$$

where $I_{\{x_{k_1+1}=x'\}}, I_{\{x_{k_2+1}=x'\}}, ..., I_{\{x_{k_{\#A(x,u)}+1}=x'\}}$ are values of the random variable $I_{\{x'=f(x,u,w)\}}$ which are drawn independently.
Notice that, these estimators will converge towards the real r and p if N is large enough. Indeed, if $\theta_1, \theta_2, ..., \theta_n$ are n values of the random variable θ which are drawn independently, the strong law of large numbers states that :

$$\lim_{n\to\infty} \frac{\theta_1 + \theta_2 + ... + \theta_n}{n} \to \mathop{\mathrm{E}}_{P}[\theta]$$

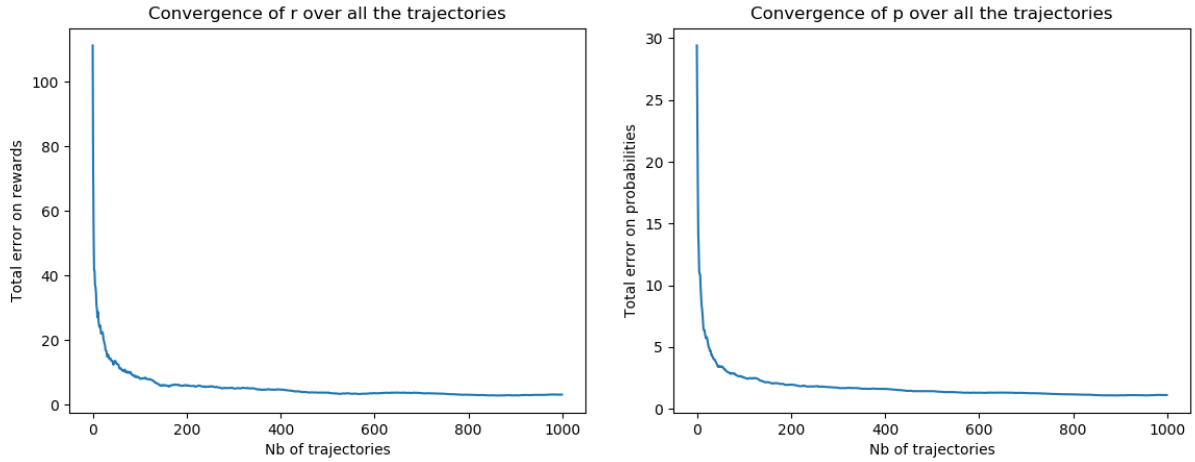
To analyse the convergence speed of these estimators towards r and p, we define the "total error" as being the sum of all the absolute differences between the estimations and the real values of r and p (those used in Section 1), at either a time step t (if only one trajectory considered) or for a trajectory i (if multiple trajectories considered).
For the deterministic domain, only one trajectory is considered to compute the estimations because we see in Figure 7 that, if we consider a trajectory of 1000 time steps, the total error on both rewards function and probability distribution is almost equal to 0, making our estimations very precise.
For the stochastic domain, it might be more efficient to consider multiple trajectories, as the agent will often jump (with a probability of β) to the cell $(0,0)$. Indeed, if only one trajectory was considered, the agent might not be able to explore the full domain correctly, it will rather

get a lot of information on the surrounding cells of $(0, 0)$ and much less on the furthest ones. By considering multiple trajectories, the agent will start its exploration at a new random state each time and will then be able to explore the domain on a more uniform way. In Figure 8, we see that considering 1000 different trajectories of 1000 time steps for each one of them will lead to a small total error and therefore to quite good estimations.



(a) Convergence speed towards r (b) Convergence speed towards p

FIGURE 7 – Convergence speed towards r and p for the deterministic domain ($\beta = 0$)



(a) Convergence speed towards r (b) Convergence speed towards p

FIGURE 8 – Convergence speed towards r and p for the stochastic domain ($\beta = 0.1$)

By computing $\hat{Q}_N$ with the estimations $\hat{r}(x, u)$ and $\hat{p}(x'|x, u)$, the estimated optimal policy $\hat{\mu}^*_N$ can directly be derived. It turns out that $\hat{\mu}^*_N$ is exactly the same as μ^*_N, presented in Figure 12, for both deterministic and stochastic settings. However, the $J_N^{\hat{\mu}^*}$ values differ a bit for the stochastic domain. These differences, presented in Figure 9, are due to the fact that, unlike the deterministic setting, the estimations of the rewards and the probabilities in the stochastic domain are not perfectly precise. Notice that, for the deterministic domain, the $J_N^{\hat{\mu}^*}$ values are exactly the same as the $J_N^{\mu^*}$ values, presented in Figure 6.

$J_N^{\mu^*}$				
1201.92	1209.68	1228.68	1245.78	1245.78
1209.68	1221.43	1229.56	1237.68	1245.78
1201.92	1210.58	1221.43	1229.56	1237.68
1194.1	1207.94	1218.47	1215.13	1218.47
1188.03	1191.37	1207.94	1218.47	1201.64

$J_N^{\hat{\mu}^*}$				
1248.42	1256.62	1276.94	1294.45	1295.18
1256.67	1268.92	1277.63	1286.55	1295.1
1248.46	1257.7	1268.97	1278.02	1287.06
1240.39	1255.09	1266.21	1263.53	1266.15
1234.16	1239.09	1255.09	1266.07	1249.29

FIGURE 9 – $J_N^{\mu^*}$ (left) and $J_N^{\hat{\mu}^*}$ (right) values of all states in the stochastic domain ($\beta = 0.1$)

5 Q-learning in a batch setting

5.1 Implementation of the Q-learning algorithm

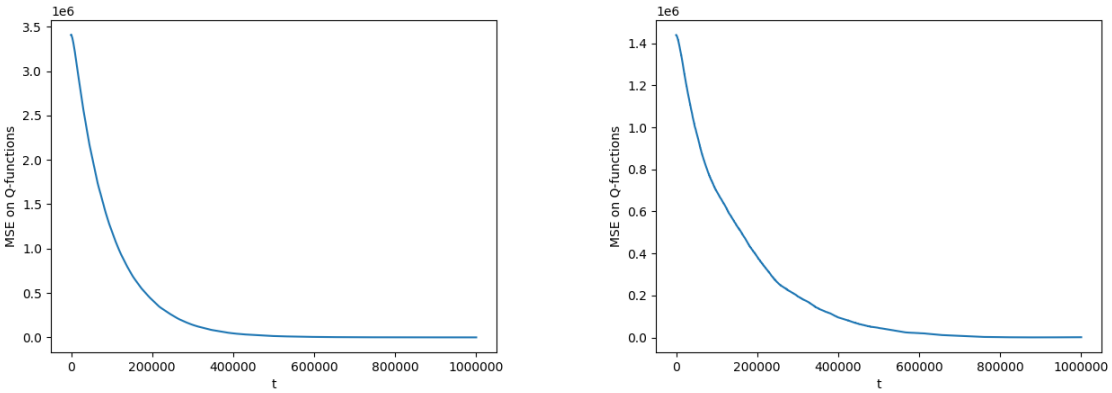
The *Q-learning* algorithm is implemented in such a way that it handles single and multiple trajectories and allows to do experience replay. As an experiment, the $\hat{Q}(x_k, u_k)$ are computed with experience replay using a total of 10^6 samples drawn from a uniform distribution over a set of 1000 trajectories, each one containing 1000 transitions in the domain instance, where each action has been selected randomly. The learning ratio α_k is set to 0.05, $\forall k$.

To show that $\hat{Q}$ converges towards Q, the mean squared error (MSE) between the matrix of the estimated $\hat{Q}(x_k, u_k)$ and the matrix of the real $Q(x_k, u_k)$ is computed at each iteration. The results are shown in Figure 10 for both deterministic and stochastic settings. As expected, the $\hat{Q}(x_k, u_k)$ progressively converge towards the real values $Q(x_k, u_k)$. Indeed, the following convergence conditions are respected :

1. Conditions on the the learning ration α_k :

$$\lim_{t\to\infty} \sum_{k=0}^{t-1} \alpha_k \to \infty \qquad \text{and} \qquad \lim_{t\to\infty} \sum_{k=0}^{t-1} \alpha_k^2 < \infty$$

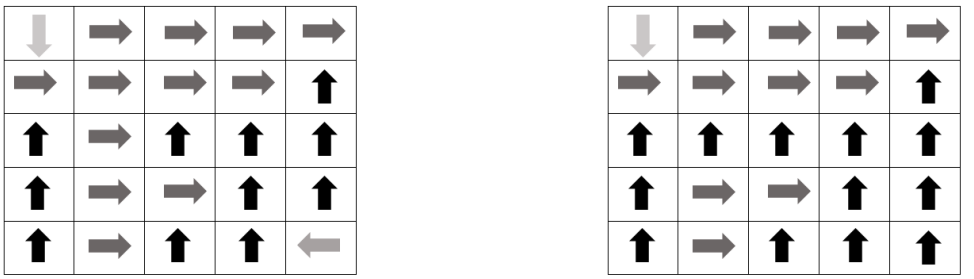
2. Condition on the history h_t : When $t \to \infty$, every state-action pair needs to be visited an infinite number of times. By considering here 1000 trajectories of 1000 random transitions, this condition is clearly satisfied.



(a) Deterministic setting ($\beta = 0$) (b) Stochastic setting ($\beta = 0.1$)

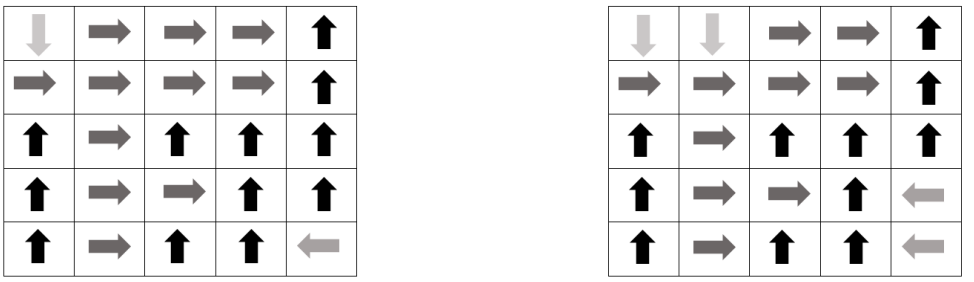
FIGURE 10 – MSE between $\hat{Q}$ and Q

By deriving the policy $\hat{\mu}^*$ from the $\hat{Q}(x,u)$ computed with the *Q-learning* algorithm, we got the policies presented in Figure 11. As a reminder, the optimal policies for each domain are shown in Figure 12. It can be seen that the policy computed for the deterministic domain corresponds to the optimal one, whereas the one computed for the stochastic domain differs a bit.



(a) Deterministic domain ($\beta = 0$) (b) Stochastic domain ($\beta = 0.1$)

FIGURE 11 – Action given by the estimated optimal policy $\hat{\mu}^*$ for each state



(a) Deterministic domain ($\beta = 0$) (b) Stochastic domain ($\beta = 0.1$)

FIGURE 12 – Action given by the optimal policy μ^* for each state

Finally, Figure 13 and Figure 14 show the differences between $J_N^{\hat{\mu}^*}$ and $J_N^{\mu^*}$ respectively for the deterministic and the stochastic settings. As can be seen, these estimations are not far from the reality but could be a bit more precise (a better precision is achieved when we consider a batch size of 10^7 but then, the computational time becomes quite large).

$J_N^{\mu^*}$				
1842.03	1857.19	1881	1900	1900
1854.58	1870.28	1881.09	1891	1900
1842.03	1855.58	1870.28	1881.09	1891
1828.61	1849.01	1863.65	1863.28	1864.09
1816.32	1826.52	1849.01	1863.65	1842.01

$J_N^{\hat{\mu}^*}$				
1834.32	1849.57	1873.38	1892.38	1892.37
1846.87	1862.58	1873.44	1883.36	1892.34
1834.33	1847.88	1862.6	1873.43	1883.35
1820.87	1841.27	1855.9	1855.57	1856.44
1808.56	1818.72	1841.23	1855.91	1834.28

FIGURE 13 – $J_N^{\mu^*}$ (left) and $J_N^{\hat{\mu}^*}$ (right) values of all states in the deterministic domain ($\beta = 0$)

$J_N^{\mu^*}$					$J_N^{\hat{\mu}^*}$				
1201.92	1209.68	1228.68	1245.78	1245.78	1214.64	1222.39	1238.95	1248.27	1252.49
1209.68	1221.43	1229.56	1237.68	1245.78	1219.1	1232.93	1236.69	1242.25	1251.09
1201.92	1210.58	1221.43	1229.56	1237.68	1212.24	1222.58	1228.8	1237.85	1246.35
1194.1	1207.94	1218.47	1215.13	1218.47	1204.17	1220.75	1231.78	1226.71	1231.99
1188.03	1191.37	1207.94	1218.47	1201.64	1199.68	1204.46	1222.2	1231.69	1215.11

FIGURE 14 – $J_N^{\mu^*}$ (left) and $J_N^{\hat{\mu}^*}$ (right) values of all states in the stochastic domain ($\beta = 0.1$)

5.2 Implementation of an intelligent agent using an ϵ-greedy policy

Let's consider an agent that uses an ϵ-greedy policy to learn Q. With this kind of policy, the agent will explore new random states with a probability of ϵ and will greedily exploit already visited states with a probability of $(1 - \epsilon)$. The choice of ϵ defines then the trade-off between exploration and exploitation. In order to be able to determine which value of ϵ leads to the best results, we first have to determine an ideal batch size for experience replay, in a sense that it gives good results in a reasonable amount of time.

In the following sections, we will consider an agent that learns (with a learning ratio $\alpha_k = 0.05, \forall k$) from 100 episodes, each one having 1000 transitions in the domain instance and starting from state $(3, 3)$.

5.2.1 Batch size for experience replay

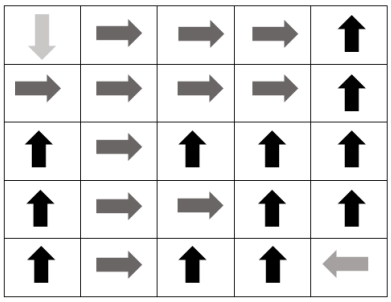
Let's test multiple values of batch sizes for a fixed $\epsilon = 0.5$ in a deterministic setting. The Table 2 presents the MSE between the matrix of the estimated $\hat{Q}(x_k, u_k)$ and the matrix of the real $Q(x_k, u_k)$, for multiple batch sizes, after the 100 episodes.

Due to the ϵ-greedy policy, the agent will explore the domain by either selecting an action that he knows will lead to the maximum cumulative reward according to the policy computed at the previous episode, or by selecting a random action. As a consequence, if the batch size of the experience replay is not large enough, a lot of state-action pairs (x_k, u_k) will stay non-visited, meaning that $\hat{Q}(x_k, u_k) = 0$, which explains the huge value of the MSE when the batch size is small, in Table 2. Therefore, in addition to the MSE between the two matrices, the resulting policy $\hat{\mu}^*$ as well as the corresponding value functions $J_N^{\hat{\mu}^*}$ are considered to determine the ideal batch size.

Batch size	100	1000	10 000	50 000	100 000
MSE	3 126 620	1 722 250	734 399	18 138	2 559

TABLE 2 – MSE between the matrix of $\hat{Q}(x_k, u_k)$ and the matrix of $Q(x_k, u_k)$

After making some tests, it turns out that, from a batch size of 20000, we begin to get the correct optimal policy but the $J_N^{\hat{\mu}^*}$ are still not very precise compared to the $J_N^{\mu^*}$. Figure 15 shows the policy and the $J_N^{\hat{\mu}^*}$ that we get with a batch size of 30000. Considering this size leads to the real optimal policy in the deterministic domain, with the $J_N^{\hat{\mu}^*}$ being really close to the true $J_N^{\mu^*}$. Therefore, this is the one we choose as default in the next sections.



(a) Action given by policy $\hat{\mu}^*$ for each state

1839.62	1857.19	1881.	1900.	1900.
1854.58	1870.28	1881.09	1891.	1900.
1838.22	1855.58	1870.28	1881.09	1891
1821.52	1849.01	1863.65	1863.28	1864.09
1808.08	1826.52	1849.01	1863.65	1842.01

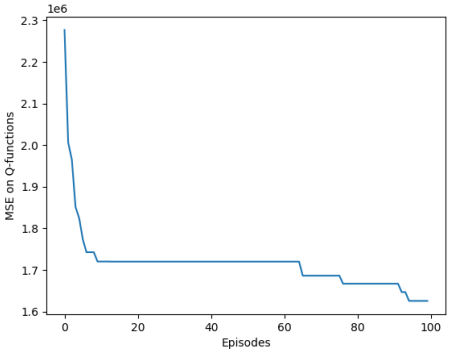
(b) Value functions $J_N^{\hat{\mu}^*}$ for each state

FIGURE 15 – Estimated $\hat{\mu}^*$ and $J_N^{\hat{\mu}^*}$ computed by Q-learning with a batch size of 30 000

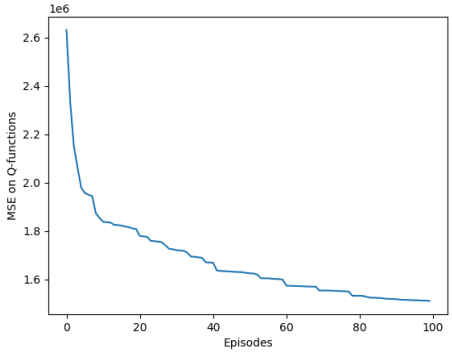
5.2.2 Value of ϵ

We will now test different values of ϵ and analyze the consequences on the resulting $\hat{Q}(x_k, u_k)$, the optimal policy $\hat{\mu}^*$ and the $J_N^{\hat{\mu}^*}$. For that, we will consider ϵ to be equals to 0.05, 0.2 and 0.5 and try these values in both deterministic and stochastic settings. As 100 episodes are not much for learning, we will always consider experience replay in what follows.

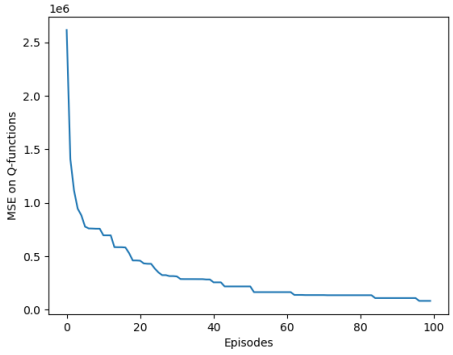
Let's first analyze the consequences of varying ϵ in the deterministic setting. The MSE between the matrix of the $\hat{Q}(x_k, u_k)$ and the matrix of the $Q(x_k, u_k)$ is shown in Figure 16. It can be seen that, except in the case of $\epsilon = 0.5$, $\hat{Q}$ does not converge towards Q after the 100 episodes considered. This is due to the fact that, the smaller the ϵ, the more state-actions pairs (x_k, u_k) will stay non-visited, letting the value 0 to the corresponding $\hat{Q}(x_k, u_k)$ in the matrix and thus leading to a large MSE between the two matrices.



(a) $\epsilon = 0.02$



(b) $\epsilon = 0.2$

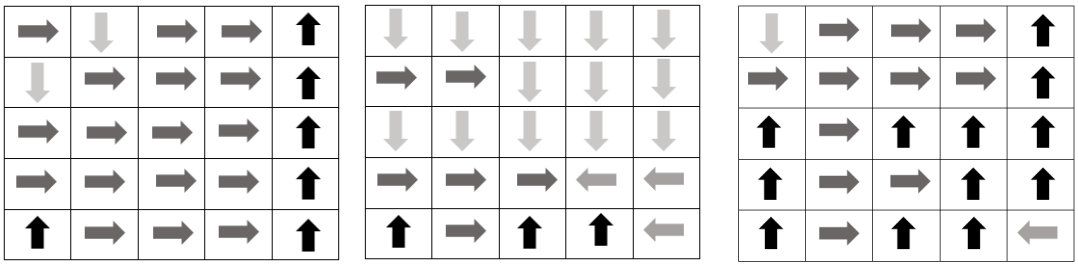


(c) $\epsilon = 0.5$

FIGURE 16 – MSE between $\hat{Q}$ and Q for different ϵ in the deterministic domain ($\beta = 0$)

Figure 17 shows the optimal policies that we got in a deterministic setting for the different values of ϵ. We see that the ones computed with $\epsilon = 0.02$ and $\epsilon = 0.2$ are far from the real optimal one. By making several tests for these values, it can also be noticed that these policies are very variable. This can be explained by the fact, as the exploration is rare for small values of ϵ, the $\hat{Q}$ will strongly be influenced by the few actions that the agent will choose randomly.

When considering $\epsilon = 0.5$, we got the real optimal policy for the deterministic setting, presented on the left in Figure 12. This can partially be explained by the fact that, here, the agent will explore a lot more the domain and so, after the 100 episodes, will get a more precise knowledge of it.



(a) $\epsilon = 0.02$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.5$

FIGURE 17 – Action given by the estimated optimal policy $\hat{\mu}^*$ for each state depending on ϵ in the deterministic domain ($\beta = 0$)

The $J_N^{\hat{\mu}^*}$ values corresponding to the policies presented in Figure 17 are shown in Figure 18. As can be seen, the $J_N^{\hat{\mu}^*}$ computed with a $\epsilon = 0.5$ are very close to the real ones, presented on the left in Figure 13. This is absolutely not the case for the two others values of ϵ, except in the states in which the action told by the policy is the same than the one of the optimal policy.

296.05	329.15	380.6	1900.	1900.
301.18	369.1	405.78	1891.	1900.
327.05	365.61	398.62	1864.09	1891.
316.89	359.07	388.63	1840.45	1864.09
303.3	320.36	1420.27	1831.04	1840.45

(a) $\epsilon = 0.02$

223.81	235.2	239.53	238.22	847.2
230.99	249.08	777.07	740.86	948.41
989.73	995.01	1015.3	1022.94	1004.61
994.79	1015.45	1022.5	1014.08	1022.76
990.68	1002.08	1016.27	1022.94	1009.71

(b) $\epsilon = 0.2$

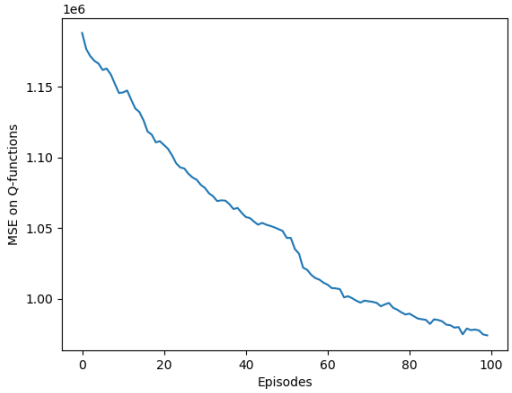
1839.62	1857.19	1881.	1900.	1900.
1854.58	1870.28	1881.09	1891.	1900.
1838.22	1855.58	1870.28	1881.09	1891
1821.52	1849.01	1863.65	1863.28	1864.09
1808.08	1826.52	1849.01	1863.65	1842.01

(c) $\epsilon = 0.5$

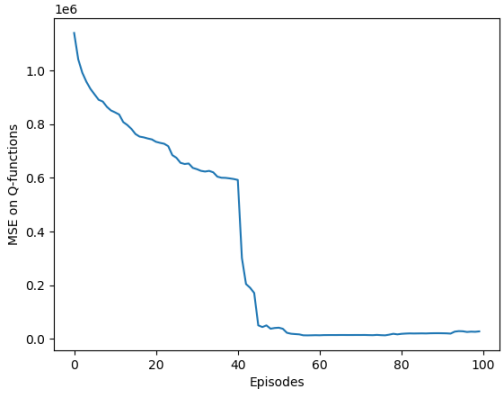
FIGURE 18 – $J_N^{\hat{\mu}^*}$ for each state depending on ϵ in the deterministic domain ($\beta = 0$)

Let's now consider the case of a stochastic domain (with $\beta = 0.1$). We begin by computing the MSE between the matrix of the $\hat{Q}(x_k, u_k)$ and the matrix of the $Q(x_k, u_k)$, shown for every value of ϵ in Figure 19. Once again, we see that, for the same reason as in the deterministic case, the MSE takes huge values when ϵ is very small. However, it converges here more rapidly for

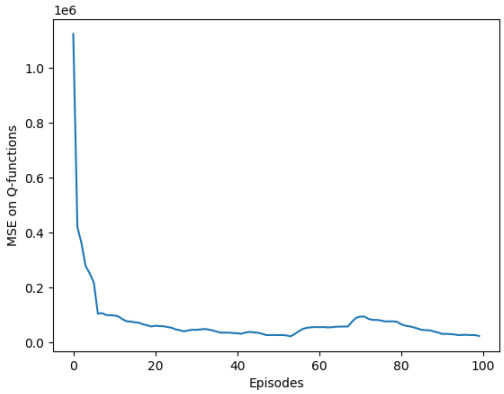
the two other values.



(a) $\epsilon = 0.02$



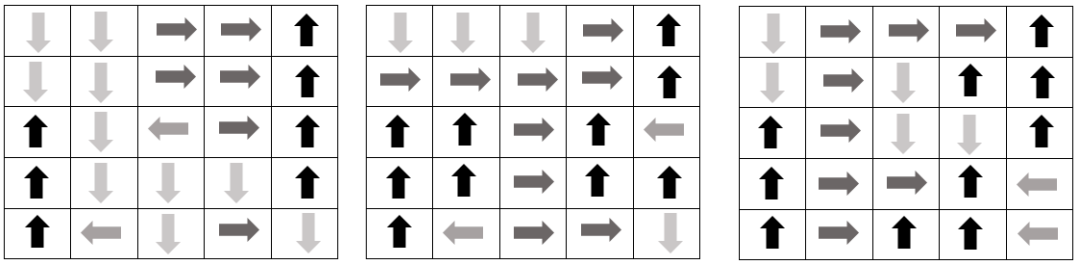
(b) $\epsilon = 0.2$



(c) $\epsilon = 0.5$

FIGURE 19 – MSE between $\hat{Q}$ and Q for different ϵ in the stochastic domain ($\beta = 0.1$)

Figure 20 shows the optimal policies that we got in a stochastic setting for the different values of ϵ. We see that none of these are close to the real optimal policy presented on the right in Figure 12. This is however the policy computed with an $\epsilon = 0.5$ that is the closest.



(a) $\epsilon = 0.02$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.5$

FIGURE 20 – Action given by the optimal policy $\hat{\mu}^*$ for each state depending on ϵ in the stochastic domain ($\beta = 0.1$)

The $J_N^{\hat{\mu}^*}$ values corresponding to the policies presented in Figure 20 are shown in Figure 21. It can be noticed in the case of $\epsilon = 0.2$ and $\epsilon = 0.5$ that, despite the fact that the policies are not optimal, he $J_N^{\hat{\mu}^*}$ are not so far from the real $J_N^{\mu^*}$, presented on the left in Figure 14.

456.25	391.94	166.71	175.74	178.21
455.77	410.27	153.22	161.81	178.32
455.66	445.71	125.58	117.65	159.61
453.56	399.97	226.07	463.79	101.14
449.65	423.54	184.98	471.51	496.11

(a) $\epsilon = 0.02$

1376.29	1374.99	1395.96	1408.75	1419.82
1384.17	1395.12	1402.64	1408.7	1412.77
1376.27	1375.06	1376.34	1389.33	1392.27
1367.28	1349.82	1360.28	1359.87	1376.03
1359.43	1335.49	1352.04	1368.7	1372.21

(b) $\epsilon = 0.2$

1339.13	1337.85	1359.19	1377.01	1376.98
1346.59	1350.63	1368.4	1370.85	1386.64
1355.22	1353.65	1376.56	1386.95	1385.37
1348.83	1365.36	1384.88	1377.26	1384.95
1341.59	1349.04	1361.41	1386.78	1369.91

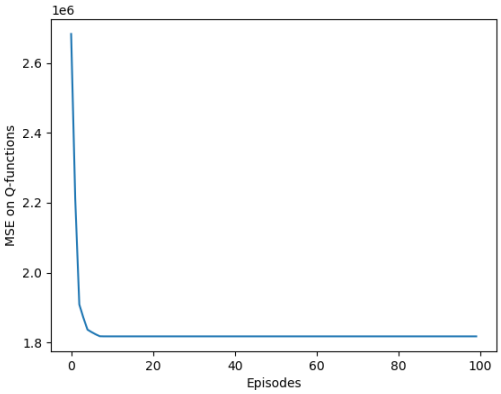
(c) $\epsilon = 0.5$

FIGURE 21 – Value functions $J_N^{\hat{\mu}^*}$ for each state depending on ϵ in the stochastic domain ($\beta = 0.1$)

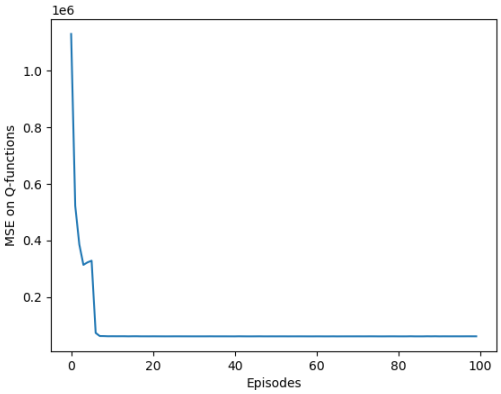
From all these observations, we conclude that an ϵ equals to 0.5 is the best choice to make among the three possibilities, because it is the one that mostly gives the best policy with the $J_N^{\hat{\mu}^*}$ being the closest to those computed with the real optimal policy μ_N^*.

Until now, we considered an ϵ that was constant during the learning process. In practice, ϵ is initially fixed to a certain value when beginning the learning process, value that, in general, gives priority to exploration. Then, as the learning process goes on, the value of ϵ progressively decreases in order to give more importance to the exploitation now that the agent has some knowledge about the domain.

For the test, let's consider a set of ϵ's that starts from 0.5 and decreases linearly. As 100 episodes are considered, the value of ϵ is being decreased of 0.05 after each episode. The MSE bewteen $\hat{Q}$ and Q are shown for both deterministic and stochastic settings in Figure 22. In both settings, $\hat{Q}$ does not converge towards Q, although the MSE in the stochastic domain is lower than the one in the deterministic domain (but this can be due to chance). As convergence has not been reached, the resulting policies are far from being optimal.



(a) Deterministic domain ($\beta = 0$)



(b) Stochastic domain ($\beta = 0.1$)

FIGURE 22 – MSE between $\hat{Q}$ and Q when ϵ decreases of 0.05 at each episode.

5.3 Softmax policy

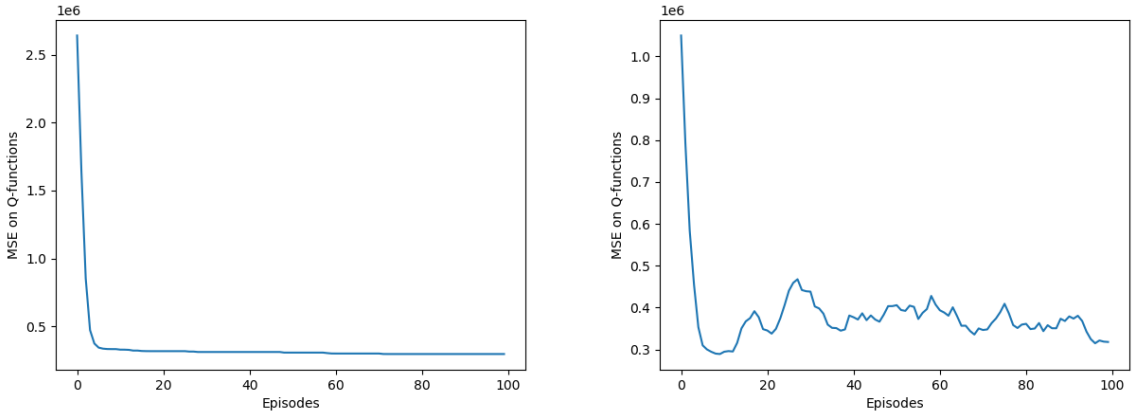
Although the ϵ-greedy policy for action selection is an effective and popular way of balancing exploration and exploitation in reinforcement learning, it has the drawback that, when it explores, it chooses equally among all actions. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action.

To solve this problem, we could use another popular policy for action selection in reinforcement learning, called *softmax*. The softmax policy chooses an action using Boltzmann distribution and explores all possible actions in every visited state. It chooses the action u on the τ^{th} play with the following probability :

$$\pi(x,u) = \frac{e^{Q(x,u)/\tau}}{\sum_{\forall u'} e^{Q(x,u')/\tau}}$$

where τ is a positive parameter called the temperature. Note that high temperatures cause the actions to be all (nearly) equiprobable, whereas low temperatures cause a greater difference in selection probability for actions that differ in their value estimates. In the limit as $\tau \to 0$, the probability of selecting the greedy action tends to 1, and no more exploration is considered.

For the test, we fixed a τ relatively large, equals to 1000 in order to favour exploration. With this value, the agent computed an optimal policy really close to the real optimal one in both deterministic and stochastic settings but with $J_N^{\hat{\mu}^*}$ values being still a bit far to the $J_N^{\mu^*}$. The MSE bewteen $\hat{Q}$ and Q are shown for both deterministic and stochastic settings in Figure 23. It can be seen that, in both settings, $\hat{Q}$ does not converge towards Q, particularly in the stochastic domain where the $\hat{Q}$ vary a lot from one episode to the other.



(a) Deterministic domain ($\beta = 0$)

(b) Stochastic domain ($\beta = 0.1$)

FIGURE 23 – MSE between $\hat{Q}$ and Q for an agent using a softmax policy with $\tau = 1000$

6 Discount factor

We will now study the effects of the discount factor on all the elements implemented in this assignment. Previously, the discount factor was set to 0.99. We now change the domain and set γ to 0.4.

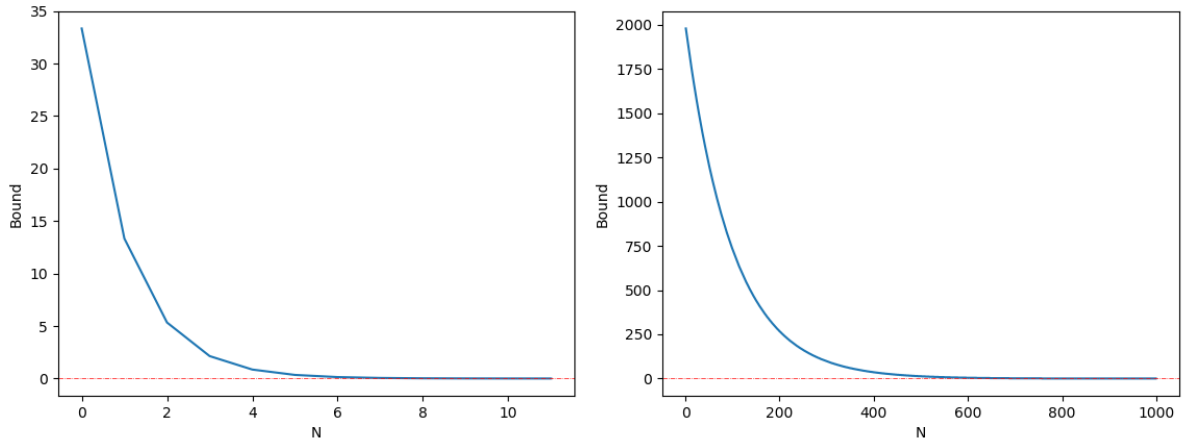
6.1 Expected return of a policy

As the discount factor is now smaller, we expect the expected returns of a given policy to also be smaller. Moreover, the iterating process can now be stopped with less iterations than

before. Indeed, the bound directly depends on the discount factor γ :

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

where B_r is a constant that bounds the reward function. By plotting the bound in function of the time horizon, we see in Figure 24 that it rapidly converges towards 0 after a few iterations and becomes almost negligible when N=10. In comparison, when $\gamma = 0.99$, the bound begins to be negligible when $N = 1400$. We also notice that the initial bound is much smaller when considering $\gamma = 0.4$.



(a) $\gamma = 0.4$ (b) $\gamma = 0.99$

FIGURE 24 – Bound of the expected return depending on the value of the time horizon

The comparison between the expected returns considering each γ is shown in Figure 25 and Figure 26 respectively for the deterministic and for the stochastic setting ($\beta = 0.5$), for a policy that always takes the action (0, 1), i.e. that always select the action "go right".

Gamma = 0.99					Gamma = 0.4				
1839.62	1857.19	1881	1900	1900	1.02	0.06	12.66	31.66	31.66
990.04	997.01	999	1000	1000	8.7	14.26	15.66	16.66	16.66
-779.3	-779.09	-791	-800	-800	-7.09	2.27	-4.33	-13.33	-13.33
-471.57	-467.24	-476	-500	-500	-4.89	10.27	15.67	-8.33	-8.33
849.37	875.12	888	900	900	-18.12	-2.8	3.	15.	15.

FIGURE 25 – Expected return of each state for the policy "go right" in the deterministic domain ($\beta = 0$)

Gamma = 0.99					Gamma = 0.4				
-72.02	-71.46	-64.25	-54.75	-54.75	-2.37	-4.49	-0.09	9.41	9.41
-67.78	-64.91	-64.16	-63.66	-63.66	0.06	2.68	3.28	3.78	3.78
-77.41	-73.26	-76.99	-81.49	-81.49	-6.09	-0.57	-2.97	-7.47	-7.47
-75.35	-68.08	-66.52	-78.52	-78.52	-6.21	1.31	6.41	-5.59	-5.59
-82.34	-74.12	-70.65	-64.65	-64.65	-11.38	-4.54	-2.84	3.16	3.16

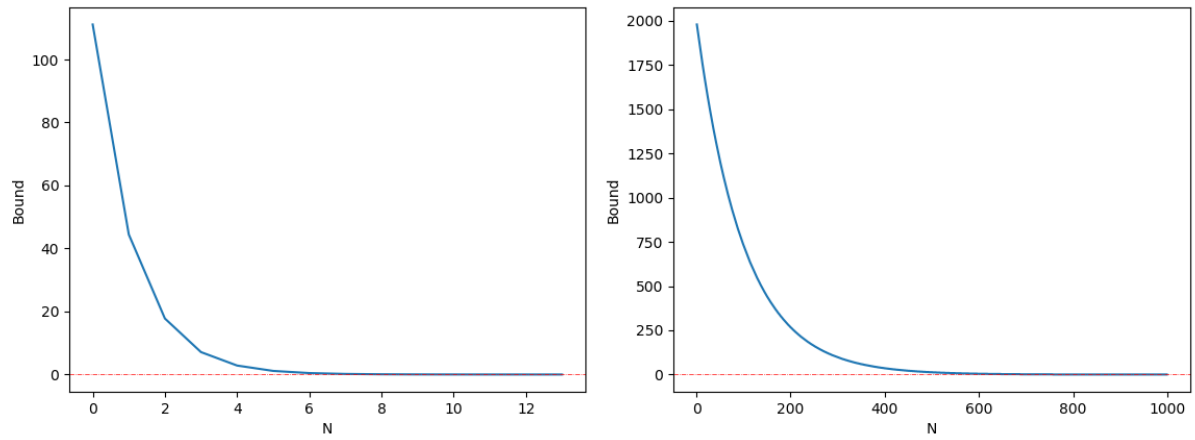
FIGURE 26 – Expected return of each state for the policy "go right" in the stochastic domain ($\beta = 0.5$)

6.2 Optimal policy

Here again, the iterating process for computing the Q-functions can be stopped with less iterations than before. Indeed, the bound on the suboptimality of μ_N^* with respect to μ^* is given :

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

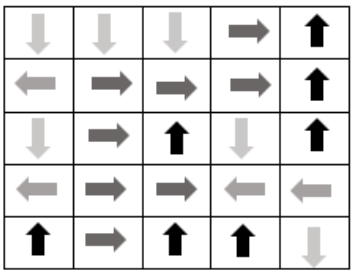
By plotting this bound in function of the time horizon, we see in Figure 27 that it progressively converges towards 0 and becomes almost negligible when $N = 12$, which is much less than the previous $N = 2000$ when considering $\gamma = 0.99$.



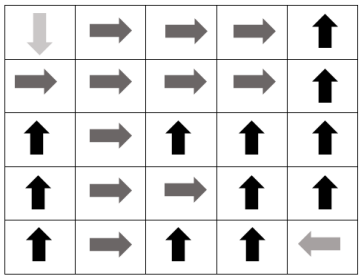
(a) $\gamma = 0.4$ (b) $\gamma = 0.99$

FIGURE 27 – Bound of the expected return depending on the value of the time horizon

The comparison between the optimal policies considering each γ is shown in Figure 28 and Figure 29 respectively for the deterministic and for the stochastic setting ($\beta = 0.1$). We can notice that the optimal policies are the same in the deterministic and stochastic cases when $\gamma = 0.4$. Figure 30 and Figure 31 makes the comparison between the corresponding $J_N^{\mu^*}$ respectively for the deterministic and stochastic settings. It's interesting to notice that the $J_N^{\mu^*}$ values between the deterministic and stochastic settings are much closer in the case of $\gamma = 0.4$ than when $\gamma = 0.99$.

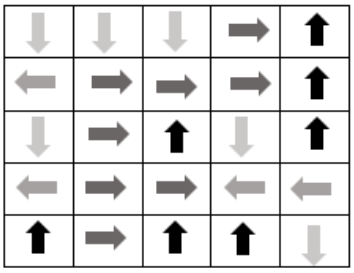


(a) $\gamma = 0.4$

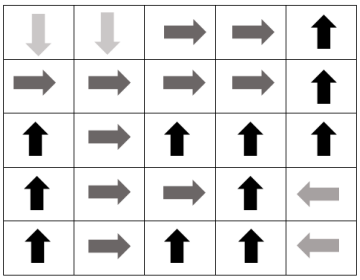


(b) $\gamma = 0.99$

FIGURE 28 – Action given by the optimal policy μ^* for each state in the deterministic domain ($\beta = 0$)



(a) $\gamma = 0.4$



(b) $\gamma = 0.99$

FIGURE 29 – Action given by the optimal policy μ^* for each state in the stochastic domain ($\beta = 0.1$)

Gamma = 0.99					Gamma = 0.4				
1842.03	1857.19	1881	1900	1900	10.	9.09	15.23	31.67	31.67
1854.58	1870.28	1881.09	1891	1900	10.	15.23	18.07	22.67	31.67
1842.03	1855.58	1870.28	1881.09	1891	10.	10.09	15.23	24.52	22.67
1828.61	1849.01	1863.65	1863.28	1864.09	10.	13.81	24.52	13.81	24.52
1816.32	1826.52	1849.01	1863.65	1842.01	10.	1.52	13.81	24.52	15.

FIGURE 30 – $J_N^{\mu^*}$ values of each state in the deterministic domain ($\beta = 0$)

Gamma = 0.99					Gamma = 0.4				
1201.92	1209.68	1228.68	1245.78	1245.78	8.5	7.27	12.59	26.78	26.78
1209.68	1221.43	1229.56	1237.68	1245.78	8.5	12.59	14.87	18.68	26.78
1201.92	1210.58	1221.43	1229.56	1237.68	8.5	8.17	12.59	21.2	18.68
1194.1	1207.94	1218.47	1215.13	1218.47	8.5	11.27	21.2	11.27	21.2
1188.03	1191.37	1207.94	1218.47	1201.64	8.5	0.5	11.27	21.2	12.72

FIGURE 31 – $J_N^{\mu^*}$ values of each state in the stochastic domain ($\beta = 0.1$)

6.3 System identification

By considering multiple trajectories of 1000 transitions in the domain, where each action is being selected randomly, the agent perfectly identifies the domain, in both deterministic and stochastic settings and computes the same policies as the optimal ones shown in Figure 28 and Figure 29.

6.4 Q-learning in a batch setting

Here also, considering the same batch size of 30000 for experience replay, the agent will compute the real optimal policy in both settings, with the $J_N^{\hat{\mu}^*}$ values being really close to the $J_N^{\mu^*}$ values.