

Optimal decision making - Assignment 1-3

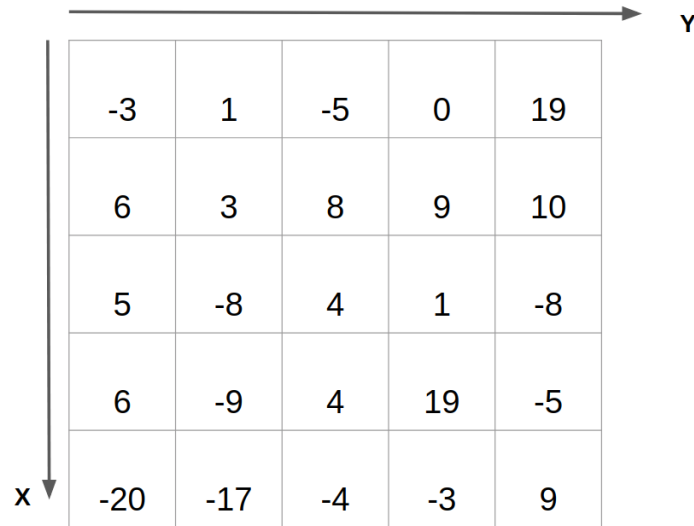
Tom CRASSET
s154416

Pr. Damien ERNST

1 Implementation of the domain

The domain was implemented and the most important detail is the layout of the axes. The axes are laid out in such a way that the origin is in the top left and the x resp. y coordinates go up when going down resp. right in the grid, as shown in each Figure with the arrows. This is done so that this representation matches the matrix representation of grids in python.

2 Expected return of the policy



The figure shows a 5x5 grid of numerical values. To the left of the grid is a vertical arrow pointing downwards, labeled 'x'. Above the grid is a horizontal arrow pointing to the right, labeled 'y'. The grid contains the following values:

-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 – Initial reward grid

After implementing the different components of the domain and testing with a simple policy, we are tasked to compute the cumulative expected return of the policy

$$\mu(x) = (0, 1)$$

, which in my case is the direction RIGHT, using the Bellman equation (1).

$$J_N^u(x) = \begin{cases} E_{w \sim P_w(\cdot|x,u)}[r(x, \mu(x), w) + \gamma J_{N-1}^\mu(f(x, \mu(x), w))], & N \geq 1 \\ 0 & N = 0 \end{cases} \quad (1)$$

We are starting from the initial grid depicted at Figure 1.

	1839	1857	1881	1900	1900
	990	997	999	1000	1000
	-779	-779	-791	-800	-800
	-471	-467	-476	-500	-500
x	849	875	888	900	900

FIGURE 2 – Consecutive iterations of the expected cumulative reward with a deterministic policy.

As show in Figure 2, the cumulative reward after 10000 iterations are shown for a deterministic policy. The choice of the number of iterations will be discussed later. As expected, the rightmost column has the highest values because after 4 iterations, the final state has already been reached and each successive application of the policy lands us in the same spot. This value is also 100 times the value of these cells in the initial grid. This is because the following : $\sum_{i=1}^{\infty} 0.99^i = 99, \bar{9}$.

In addition, we see that row-wise, the cumulative rewards are decreasing when going to the left. This is also due to the policy, as the high value obtained at the rightmost column trickles down to the previous states when updating. Finally, every row is independent of each other as the policy was to always go right.

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

As for the number of iterations set at 10000, we can compute the bounds of the difference between the return of the optimal stationary policy and the return of the numerically computed policy using the equation (2), which dynamic programming theory tells us is tending to zero when N tends to infinity. Computing this bound with $N = 10000$, $B_r = 19$ and a discount factor of $\gamma = 0.99$ gives us a value of $4.5e-41$, which is very close to 0. So the hypothesis is validated and the value of $N = 10000$ is high enough for this problem.

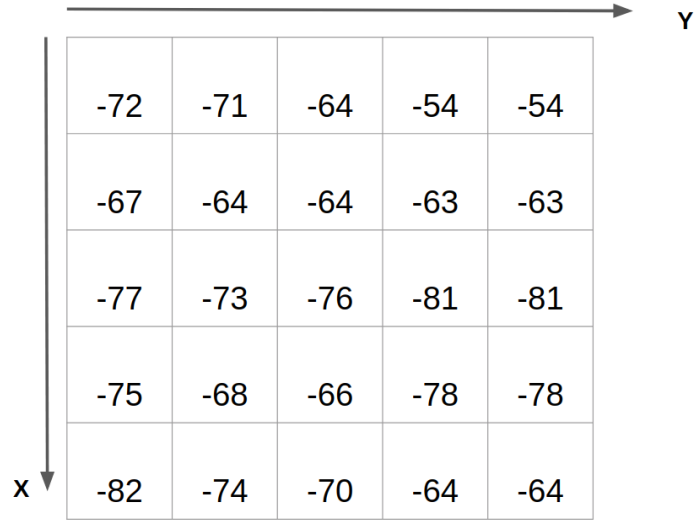


FIGURE 3 – The expected cumulative reward for a stochastic policy with beta equal to 0.7.

In Figure 3, the expected cumulative reward for 10000 iterations are shown, using a stochastic policy with a probability of $\beta = 0.5$ to use the reward of state $(0,0)$ and a probability of $1 - \beta$ to use the reward of the current state after application of the policy.

Nothing special can be deduced from the results, there isn't a row-wise gradient from left to right anymore and all the cumulative rewards are in the same vicinity. The random effect of the stochastic dynamics is mainly at fault here because we choose to take the reward of the origin wherever we are in the grid with a probability of β . This is also why, if we had chosen $\beta = 0$, we would have had the same results as the deterministic dynamics in Figure 2 and if we had chosen $\beta = 1$, each cell would have had a cumulative reward of -300, which is 100 times the reward of state $(0,0)$.