# Homework #5

**Assignment** The aim of this final homework is to train an *Agent* to move in a 10x10 checkerboard in order to reach a certain point as a goal. To do so I implement the following tasks:

- add an obstacle in the center of the map, the *Agent* needs to avoid and circumnavigate it;
- after a simple grid search to find the best parameters, training and testing the model with Q-learning, SARSA with *ε-greedy* and SARSA with *softmax* to compare the results.

**1) The set-up** The environment where the *Agent* can move is a 10x10 checkerboard with a squared obstacle in the middle starting from the position [3, 3] and ending in [6, 6]. The *Agent* can't pass through it as it is considered a *out-of-bounds* move with an assigned reward of -1. The aim of the agent is to reach the *Goal* position [0, 3] starting from a random position of the checkerboard, excluded the obstacle in the center.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | G | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | | | | | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | | | | | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | | | | | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | | | | | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 1: *The environment is a checkerboard with an obstacle in the middle. The agent starts in a random position "0" and needs to reach the Goal "G". Figure created using a graphical software just for visual representation of the problem.*

**2) Model & training** The *Agent* allowed *Actions* are: stay on the current position, move up, down, left and right. As the length of moves per episode is fixed, it is expected to have the *Agent* stay in the *Goal* position once it reaches it.

The *Agent* episode's length is **50** moves, and I use **SARSA** and **Q-learning** to investigate the different strategies learned. Q-learning has a greedy update policy ($\varepsilon$ is always zero), SARSA can have an $\varepsilon$-greedy or softmax behaviour policy.
For the $\varepsilon$ *parameter* and $\alpha$ *learning rate* values, I implement a simple grid search. The $\varepsilon$ might start from 0.9 or 0.7 and decrease each episode until it reaches the final value of **0.001**, in this way the training starts with randomly selected actions and ends with the *Agent* mainly selecting greedy actions. The $\alpha$ may change at every episode too, from **1** to **0.001** with decreasing random numbers, so initially the *Agent* relies only on the most recent information to select its next move, but with more episodes it ends up to heavily rely on learned knowledge. Otherwise $\alpha$ may be the fixed value 0.25 during the grid search.
The *discount factor* is **0.9**, so high and it might slow down the convergence, but it improves the foresightedness of the *Agent*.
I applied the grid search on the three methods and selected the parameters providing the highest average reward for them: **decreasing** $\alpha$ **[1 0.001]** and $\varepsilon$ starting from **0.7**. The tested networks during the grid search and the final network are trained for **3000** episodes.

**3) Results** Looking at the plots in Figure 2 of the *average reward* per *episode*, Q-learning increases more linearly than SARSA with $\varepsilon$-greedy, which could mean the *Agent* learns faster with this algorithm. Using SARSA with a *softmax* policy gives a higher reward starting very earlier in the episodes when compared with the $\varepsilon$-greedy policy or Q-learning.

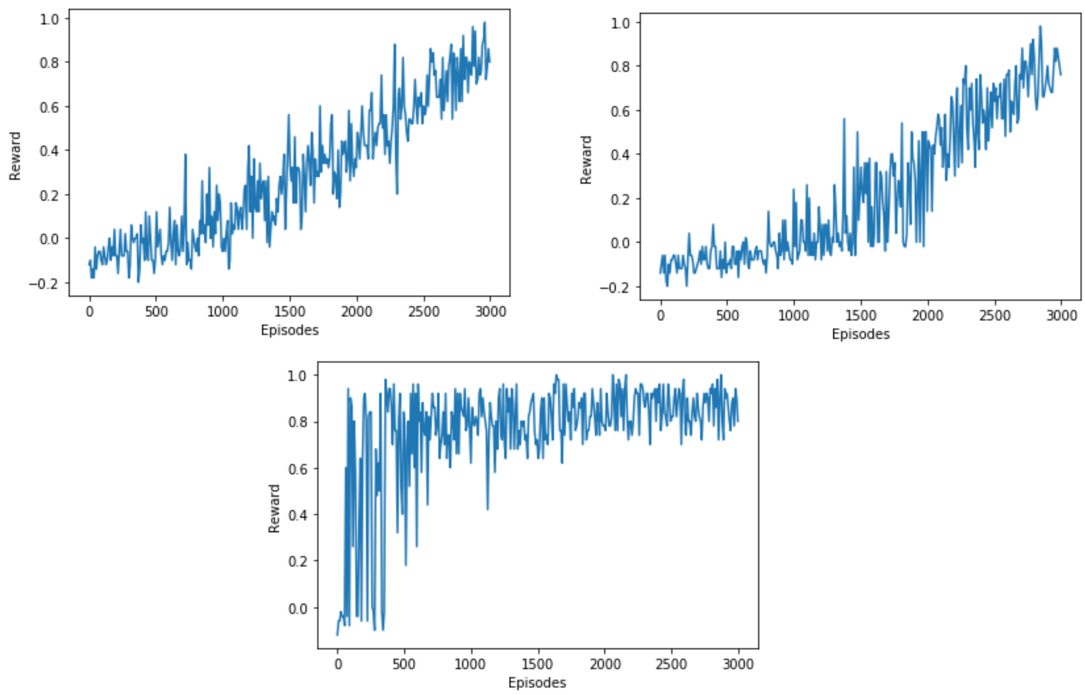


Figure 2: *Graph of the reward over the episodes for Q-learning on the top-left, SARSA with $\varepsilon$-greedy on the top-right and SARSA with softmax in the bottom-center.*

To concretely compare the results on the "field", I decided to show the movements of the *Agent* trained with Q-learning and SARSA using the same starting points. With every method, the *Agent* is able to reach the final goal avoiding the obstacle, in particular:

- using a Q-learning algorithm (off-policy) the *Agent* takes more risks, as it assumes it won't make mistake in the following step. In Figure 3 (top), the *Agent* is always selecting the shortest paths walking near the obstacle in the center;
- using a SARSA algorithm with *ε-greedy* behaviour policy the *Agent* takes less risks during the movements. In Figure 3 (middle) the *Agent* avoids coming closer to the obstacle in the center, at the cost of more moves than Q-learning requires;
- using a SARSA algorithm with *softmax* behaviour policy, the *Agent*'s actions are very similar to the ones obtained using Q-learning. In Figure 3 (bottom) the *Agent* always chooses the shortest paths.

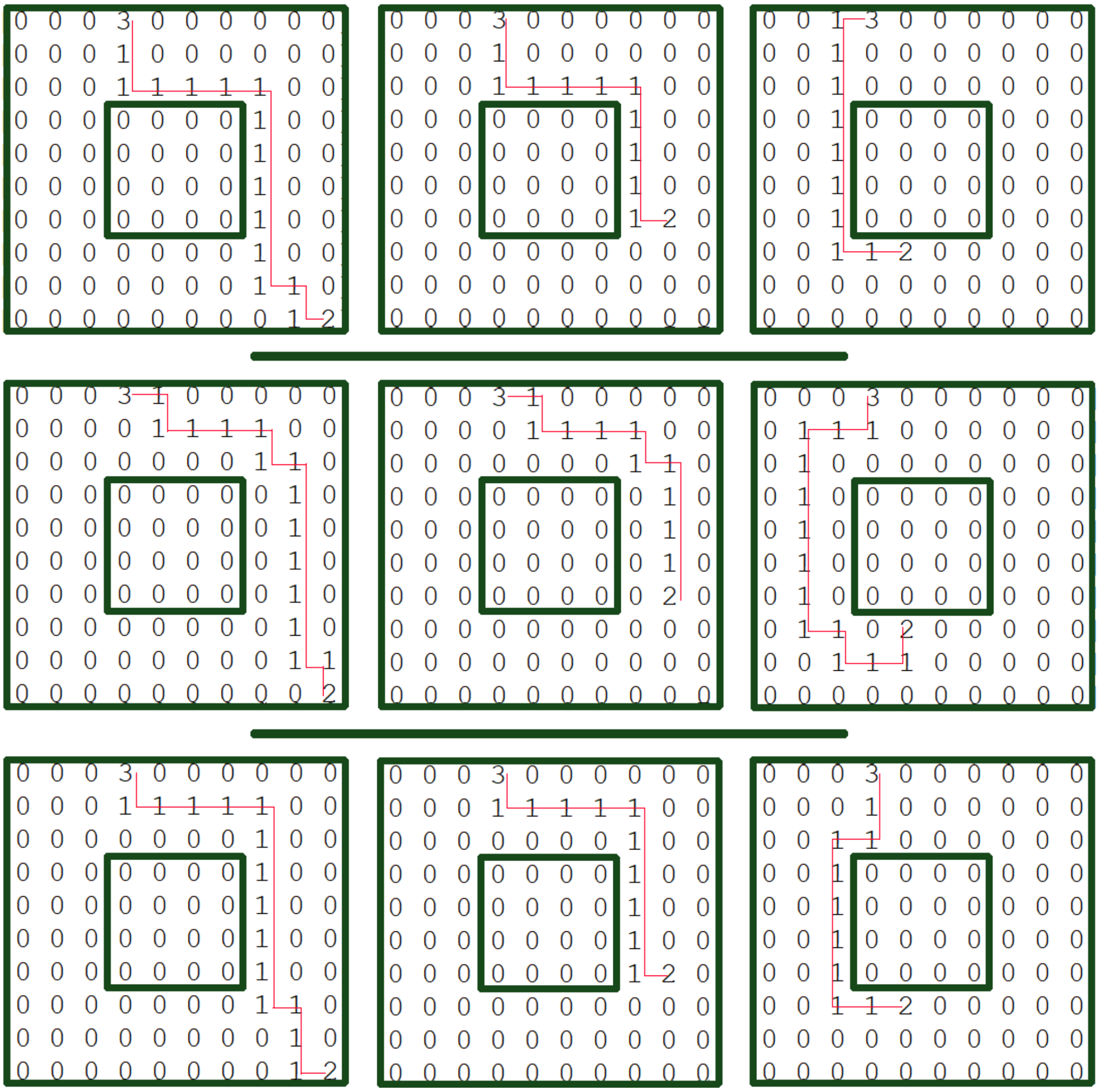

Figure 3: *Some of the final tests with the model trained using Q-learning on the top, SARSA with ε-greedy in the middle and SARSA with softmax in the bottom. The matrices have been generated from the results, the green and red lines have been added using a graphical software. "3" is the goal, "2" the start and "1" the steps.*