Projet Optimal Decision making 1.2

Bardhyl Miftari

February 2020

1 Section 5

For this part of the project, 2 new functions were introduced : $reward\_state\_action\_MDP()$ and $proba\_state1\_action\_state2\_MDP$ which respectively compute the expected reward associated to a given action taken in a given state and the proba to land in a given state when in another state and doing a given action, from experience (accumulated from trajectory) From these new functions, we were able to derive the Q function, which we modified from our previous code so that if we set the MDP variable to true when creating a *game* object, and if the chosen policy is "Q", then we end up using these functions to approximate p and r instead of the actual p and r functions.

The trajectory used to build the MDP is built from several games, each of them long of a given amount of steps. In this section, we decided to set this number of step to a very low number (5) and, if we want to get a performing MDP, a high number of games played.

The reason is simple : since we have at any time one chance out of 2 to get back to the same state (0,0), where we have only 2 chances out of 8 to get to an adjacent state (50% chance of staying still, and hitting a wall means staying still as well), as soon as we teleport back to this state (that we will call DS for default state), we don't learn much about the rest of the domain. It is therefore better to multiply the number of games where we start at random positions and do a few moves, rather than to make a few long games where we are sure to stay in the same global area, and therefore learn very little about the domain.

This is backed up by our results, where we can observe by varying the number of steps per game and the number of games that only the number of games really impacts the precision of the markov model in a positive way.

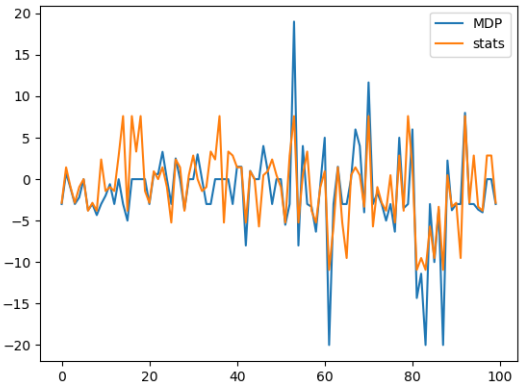

Figure 1: plot of the R evaluation for MDP vs the stats, with 5 steps per game and 100 games

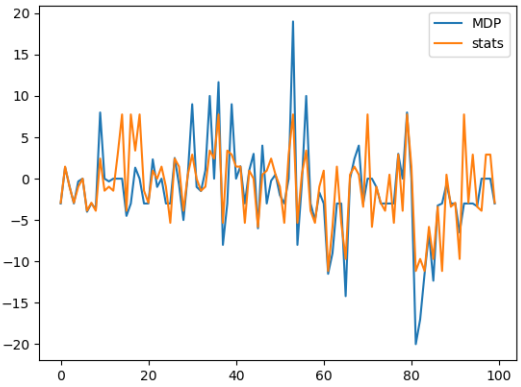

Figure 2: plot of the R evaluation for MDP vs the stats, with 3 steps per game and 100 games

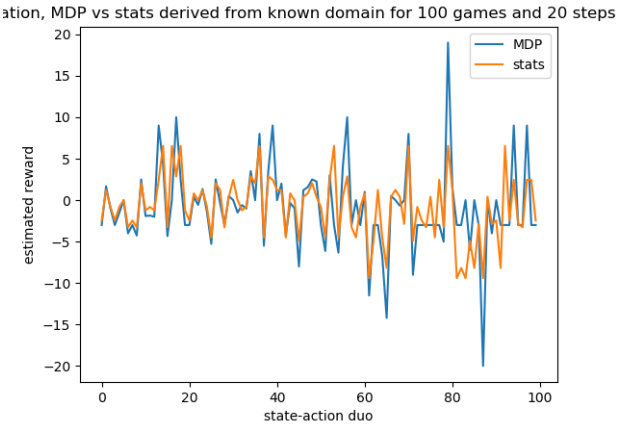

Figure 3: plot of the R evaluation for MDP vs the stats, with 3 steps per game and 100 games

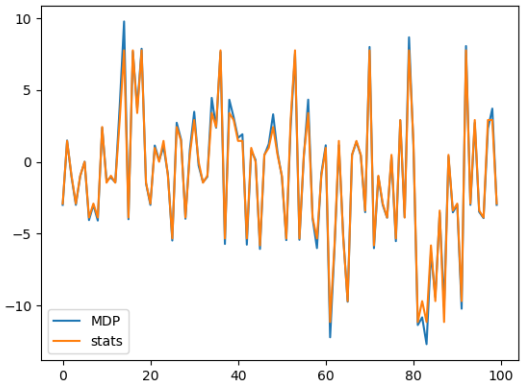

Figure 4: plot of the R evaluation for MDP vs the stats, with 3 steps per game and 10000 games

We can see in the 4 first figures that increasing the number of steps per game didn't have much effect, however increasing the number of games has great effect on the evaluation of R by the MDP.

The same kind of results can be found for the evaluation of P :

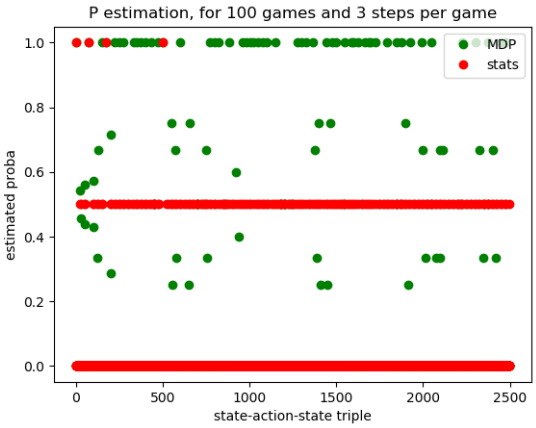

Figure 5: plot of the P evaluation for MDP vs the stats, with 3 steps per game and 100 games

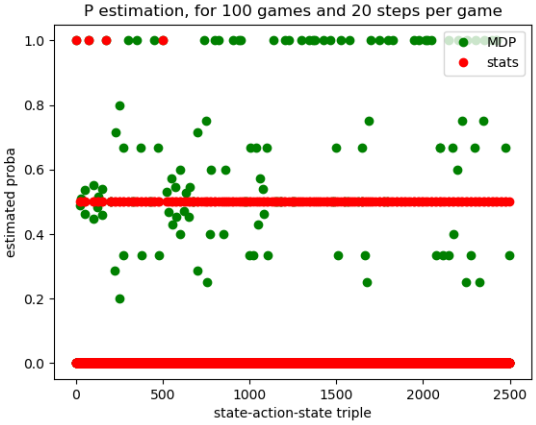

Figure 6: plot of the P evaluation for MDP vs the stats, with 20 steps per game and 100 games

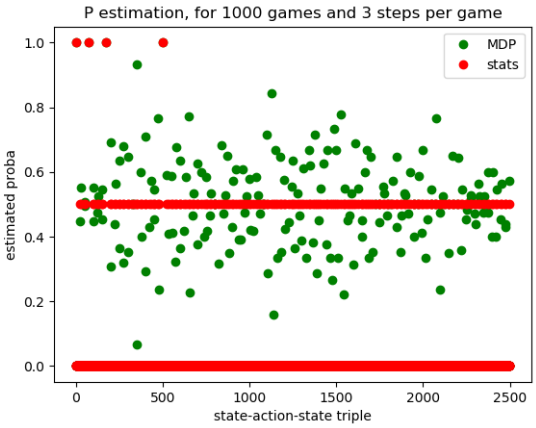

Figure 7: plot of the P evaluation for MDP vs the stats, with 3 steps per game and 1000 games

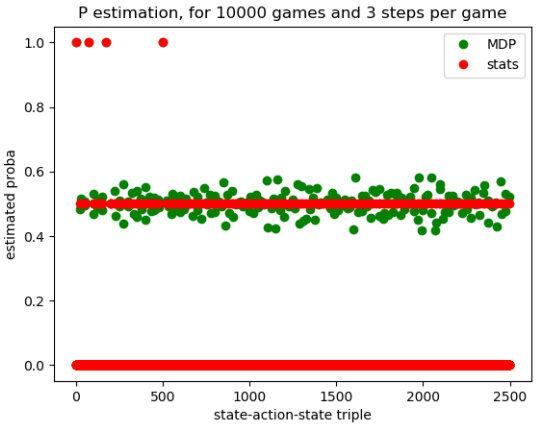

Figure 8: plot of the P evaluation for MDP vs the stats, with 3 steps per game and 10000 games

As for the speed of convergence, here are some results about the average squared error for R and P for varying numbers of game played, all for 3 steps per games:

Nb of Games	R	P
100	18.7	0.013
500	2.8	0.0019
1000	2.3	0.0012
5000	0.27	0.0002
10000	0.18	0.000009

As expected, the more games we play to infer our MDP, the more accurate we get, and converge towards average errors of 0.

1.1 Discussion on the importance of the length of the trajectory

As stated earlier, the best way to infer a good MDP is to have a lot of small trajectories, given our circumstances were chances to go back to a specific state are high at each move. However, increasing the length of a single trajectory still improves the overall precision of the model, just not as uniformally as the many-games-few-steps approach does. Indeed, since we have high chances to go back to the (0,0) state each move we make, the probability that we get far from that tile as moves go by is low, and therefore we have low chances to gather data about what is going on far from that state. But if we take a trajectory large enough, by sheer amount of trials, we will manage to sometimes get fairly far from this DS (default state), and therefore gather some data about other state-actions pairs. But the further from the DS we will look, the fewer data we will have gathered through our tries, due to the low chances to get there. Therefore, the further from the DS the state-action pair we will want to evaluate is, the worst our MDP model will get for this specific pair. However, we expect the quality of our MDP to get extremely high when the evaluated pairs will be close to this DS, as these will have been tried a lot of times.

To illustrate this, we have tried to run tests were we run a single game with varying amounts of steps to build our MDP, and have displayed the corresponding graphs for the R evaluation of the MDP to see how the model evolved. The R estimations has been chosen due to the fact that there is less cases to consider, and it is therefore easier to come up with explicit results, but the conclusions are valid for both R and P.

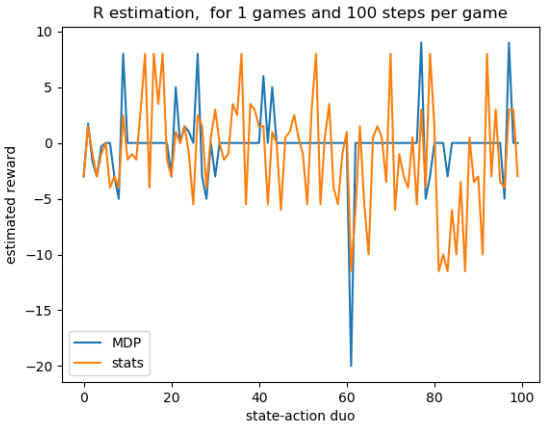

Figure 9: plot of the R evaluation for MDP vs the stats, with 100 steps per game and 1 Game

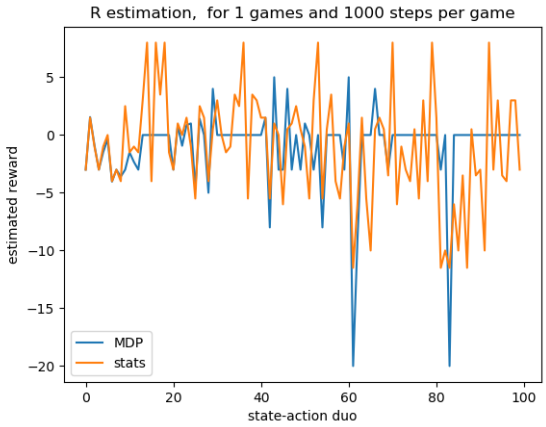

Figure 10: plot of the R evaluation for MDP vs the stats, with 1000 steps per game and 1 Game

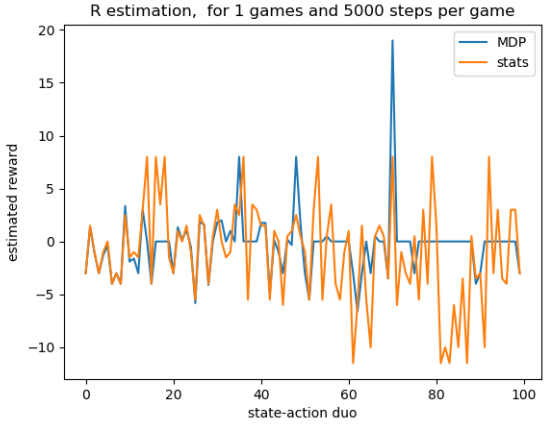

Figure 11: plot of the R evaluation for MDP vs the stats, with 5000 steps per game and 1 Game

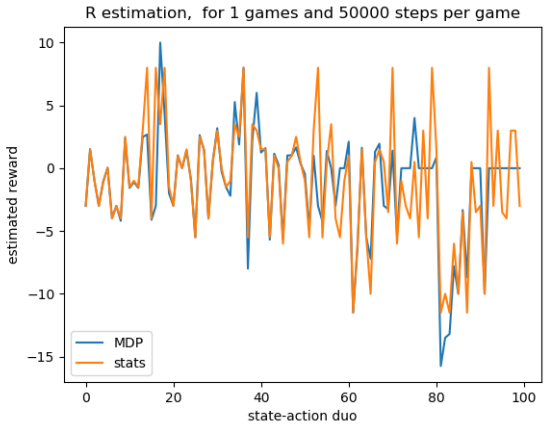

Figure 12: plot of the R evaluation for MDP vs the stats, with 50000 steps per game and 1 Game

As we can see, as the number of steps in the single game we run to build our model increases, the MDP starts to fit the actual probabilities more and more. It is also possible to notice that the difference between the MDP and the actual probabilities are vanishing way faster on the left side of the graph. This is expected, as the left part of the graph represents the states-actions pairs closer to (0,0), our default state. This region of the domain is far more explored than the far aways states on the right side of the graph, infering a better model. But as the length of the trajectory increases, the chances that we explore regularly all state-action pairs increases as well, meaning that we will slowly converge toward the correct probabilities. In other words, if we take a trajectory of size infinite, we will infer a perfect MDP, and our approximated Q function will converge toward the real Q function.