Imperial College London

Reinforcement Learning
Maze Environment Coursework 1

28/10/2022 - 11/11/2022

MSc Computing in Artificial Intelligence & Machine Learning

Author :

Louis BERTHIER

Email : ldb22@ic.ac.uk

CID : 02285087

Contents

List of figures

Chapter A

Dynamic Programming

A.1 Algorithm Selection (Q1.1)

To solve this problem with a DP agent, the method chosen here is the **value iteration algorithm**. This method converges in 10 to 20 times fewer epochs than the policy iteration algorithm for this maze and the parameter below. Indeed, 48 epochs are needed for value iteration against 940 for policy iteration. Note also that this MDP is of low complexity (few possible states and actions) and finite, which **ensures convergence**. The **terminating condition** is a parameter. The assumption is that a low terminating condition ($Threshold = 0.00001$) makes it possible to get a suitable result and an optimal policy following convergence for a small number of epochs.

A.2 Optimal Policy and Optimal Value Function (Q1.2)

The optimal policy and the optimal value function are shown in figure 1 (left and right respectively).

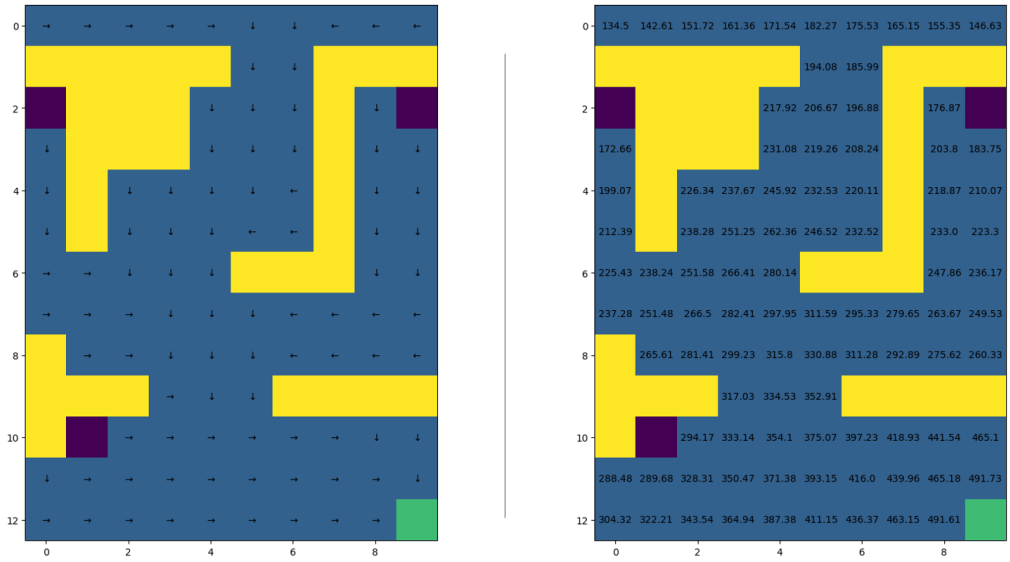

Figure 1: **Left:** The optimal policy for $p = 0.82$ and $\gamma = 0.96$. **Right:** The optimal value function for $p = 0.82$ and $\gamma = 0.96$.

A.3 Influence of discount factor γ and success probability p (Q1.3)

Multiple values of the success probability p were tested, namely: $0.15, 0.25$, and 0.55. The policy for each value of p is displayed in figure 2. It is worth noting that the more the probability tends to 0 and the more the value iteration algorithm will take time (thus epochs).

- $p < 0.25$**:** With such a probability, our policy is not moving in the right direction. Instead of taking the action that maximizes our value function, the agent takes an action that directs it to a state where the value function will be less important. Thus moving away from the expected end state.

- $p = 0.25$**:** The agent has the same probability of taking one of the four actions. Here the policy is conditioned by the starting policy and will remain the same.
- $p > 0.25$**:** In contrast to the previous cases, the policy will move in the right direction, i.e. the state that maximizes the value function. So the agent goes from an initial state to the expected final state with fewer iterations if the probability increases.

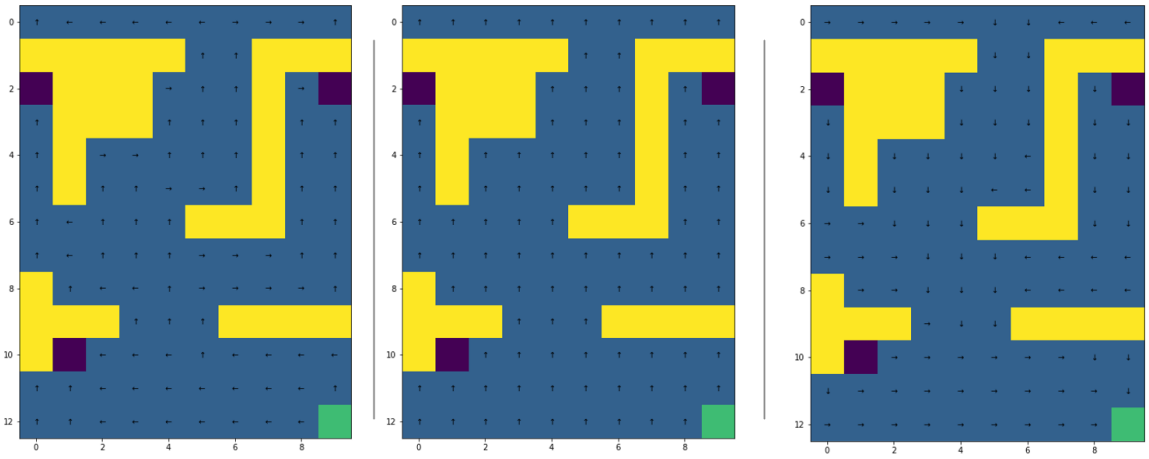

Figure 2: **Left:** The policy obtained after 193 epochs for $p = 0.15$. **Center:** The policy obtained after 206 epochs for $p = 0.25$. **Right:** The policy obtained after 101 epochs for $p = 0.55$.

Two values of the discount factor γ were tested, namely: 0.35 and 0.75. The value function for each value of γ is displayed in figure 3. It is worth noting that the more the γ tends to 1 and the more the value iteration algorithm will take epochs to converge.

- $\gamma < 0.5$**:** Such gamma values the immediate reward as it diminishes the importance given to previous rewards. A large region of states close to the initial states keeps the same values in contrast to some states close to the final state.
- $\gamma > 0.5$**:** Contrary to the case above, the more gamma tends towards 1, the more equal importance is given to the rewards regardless of their order of appearance. Here the cut-off is not as large as before, with a wider range of values taken by even more states to converge to the final state.

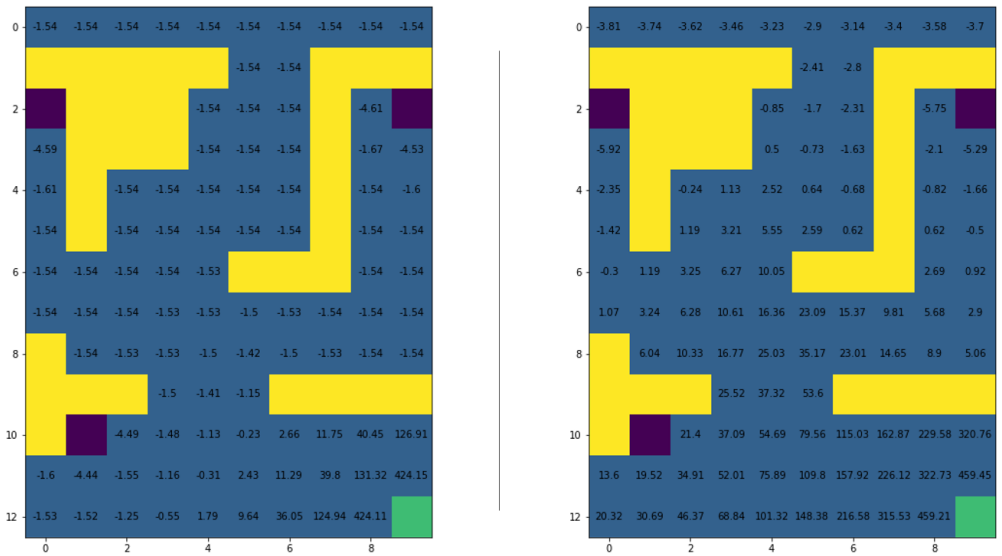

Figure 3: **Left:** The value function obtained after 16 epochs for $\gamma = 0.35$. **Right:** The value function obtained after 35 epochs for $\gamma = 0.75$.

Chapter B

Monte-Carlo Reinforcement Learning

B.1 Algorithm Selection (Q2.1)

The method selected is the **ε-soft on policy first-visit MC control algorithm**. The rewards obtained are averaged as soon as a state is visited for the first time, whereas with the every-visit MC method, the average is taken each time the state is visited. On Policy is used here rather than Off Policy since the Markov chain is stationary (probabilities depend only on the initial and arrival states).
The parameters are ε (= 0.3, little exploration to focus on the best actions) the exploration rate for the greedy-policy, α (= 0.2, the agent learns little by little) the learning rate, the **number of episodes** (= 2000, sufficient to ensure that the agent converges) and the **number of replications** (= 40, to study the variability of the agent). If α varies, it decreases with an exponential decay which is proportional to the ratio of the episode number to the total number of episodes. Finally, if ε varies, it must tend towards 0 thanks to a rate proportional to the inverse of the number of episodes which is subtracted from our epsilon. This allows us to satisfy the **GLIE condition**.

B.2 Optimal Policy and Optimal Value Function (Q2.2)

The optimal policy and the optimal value function are shown in figure 4 (left and right respectively).

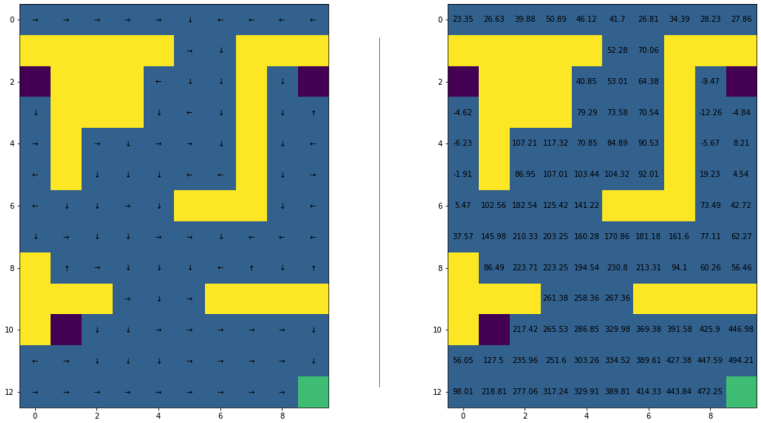

Figure 4: **Left:** The optimal policy for $\alpha = 0.2$, $\varepsilon = 0.3$ and $episodes = 2000$. **Right:** The optimal value function for $\alpha = 0.2$, $\varepsilon = 0.3$ $episodes = 2000$.

B.3 Variability in MC Algorithm (Q2.3)

The main sources of variability are the 10 **random starting states and the actions that follow from them**. Each simulation starts randomly at one of these 10 states. In the beginning, the agent explores randomly without knowing which are the best actions to take. Depending on the simulations, some states are visited only a few times. To limit the randomness of the starting point and the resulting actions, the model is **replicated a certain number of times** to obtain a reliable mean and standard deviation. It can be considered that 40 **replications** is sufficient since there are 4 actions and 10

starting states. However, such a large number of replications is very demanding on physical resources, especially if ε and α are tested for several values. Good reproducibility can also be achieved by considering only 20 replications.

B.4 Learning Curve (Q2.4)

The learning curve of the agent presented above is depicted in figure 5. It can be seen that the agent quickly converges at around 250 episodes but there is a significant standard deviation throughout the training. It was expected that the deviation would decrease as the agent converged to the final state.

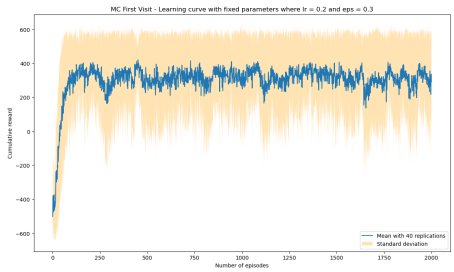

Figure 5: Agent's learning curve for $\alpha = 0.2$, $\varepsilon = 0.3$, $episodes = 2000$ and $replications = 40$.

B.5 Consequences of the Variations of ε and α on our Learning Curve (Q2.5)

By varying ε, we obtain the results shown in figure 6. The agent seems to converge quickly (slower for ε that tends towards 1) with some instability since the curve of the mean does not show a plateau but oscillations due to a high standard deviation. For low values of ε, there is a slightly lower standard deviation as the agent explores less and takes the best action.

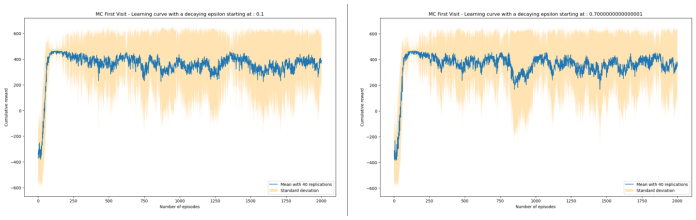

Figure 6: **Left** Learning curve with ε starting at 0.1. **Right** Learning curve with ε starting at 0.7.

The results in figure 7 are not easy to interpret or distinguish. The agent seems to be converging since, despite the variations, the curve reaches a plateau around which the values will oscillate. However, our agent does not converge toward the optimal solution (reward well below $400/500$). It might be explained by the fact that the low value of the success probability impacts the exploration phase of the agent which results in a low value of different rewards.

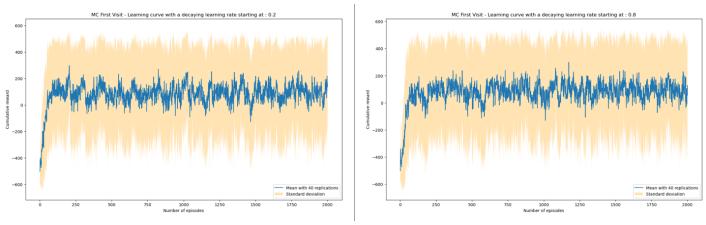

Figure 7: **Left** Learning curve with α starting at 0.2. **Right** Learning curve with α starting at 0.8.

Chapter C

Temporal Difference Reinforcement Learning

C.1 Algorithm Selection (Q3.1)

The **Q-learning algorithm** was selected for this problem. It is off-policy and favored over SARSA since the problem has 3 absorbing reward states scattered in the environment, which brings to mind the cliff walking problem. Contrary to SARSA which will take the safest path, Q-learning allows going through the most optimal path to reach the final state.
In this simulation, there are the same 4 parameters of the MC method, namely: the **number of episodes and replications**, α, and ε. The same values and decay of α and ε are retained in the case of methods with parameters that decrease over time. One assumption is that in our simplistic world, it is not necessary for α and ε to decay over time to ensure convergence.

C.2 Optimal Policy and Optimal Value Function (Q3.2)

The optimal policy and the optimal value function are shown in figure 8 (left and right respectively).

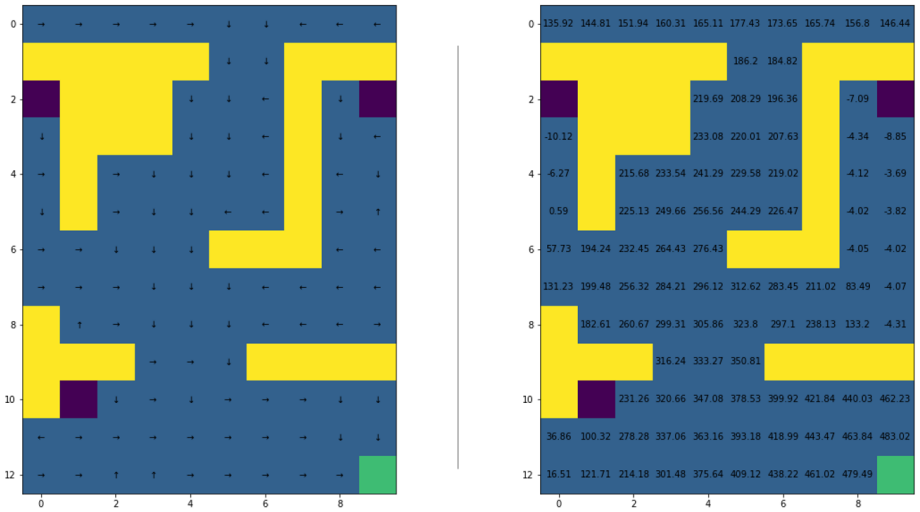

Figure 8: **Left:** The optimal policy for $\alpha = 0.2$, $\varepsilon = 0.3$ and $episodes = 2000$. **Right:** The optimal value function for $\alpha = 0.2$, $\varepsilon = 0.3$ $episodes = 2000$.

C.3 Learning Curve (Q3.3)

The agent's learning curve is depicted in figure 9. Initially, the agent explores new states, hence the high standard deviation but there is a slight decrease in the deviation as the agent trains on several episodes. However, this decrease should tend toward 0 once the agent seems to have converged to the final state (from about 150 episodes).

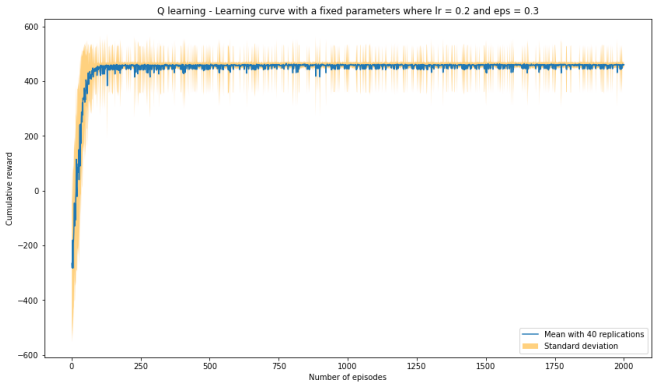

Figure 9: Agent's learning curve for $\alpha = 0.2$, $\varepsilon = 0.3$, $episodes = 2000$ and $replications = 40$.

C.4 Consequences of the Variations of ε and α on our Learning Curve (Q3.4)

The results in figure 10 show that with a Q-learning algorithm and a decreasing ε, it is important to start with a high ε. This allows the agent to explore more at the beginning of the training. As the agent learns, ε decreases so it starts to abandon exploration and converge to the final state, hence the decrease in the deviation.

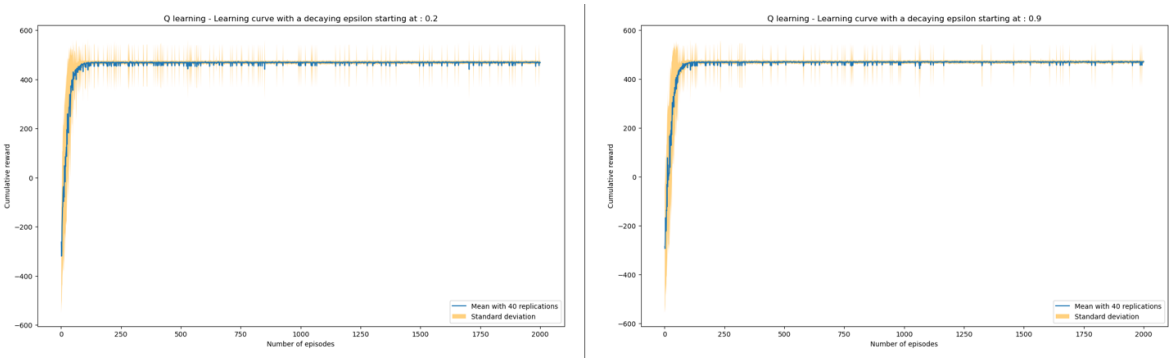

Figure 10: **Left** Learning curve with ε starting at 0.2. **Right** Learning curve with ε starting at 0.9.

By varying α, two points can be distinguished according to the results in figure 11. Firstly, if α tends towards 0, then little new information is considered, so convergence takes longer (although this is slight here). Also, a high learning rate will result in a model with a high standard deviation over all episodes because if on some simulations the agent takes actions that are not the best, these will have greater weight than for low values of α.

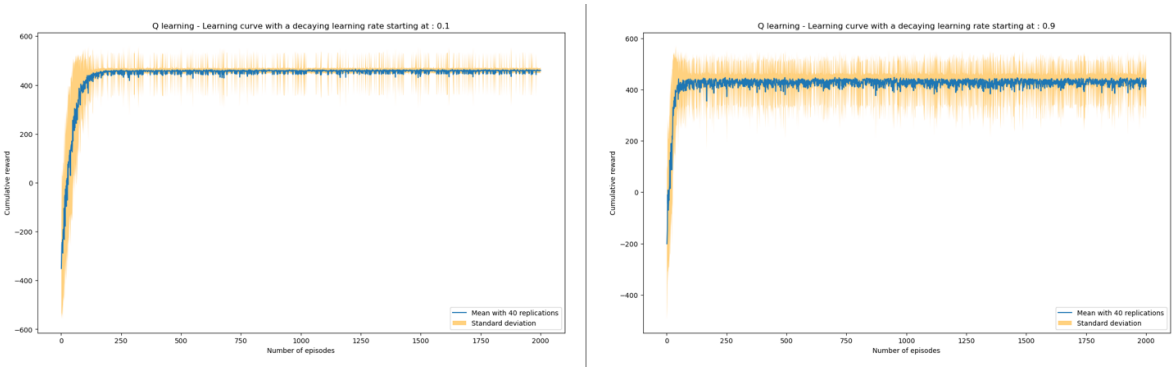

Figure 11: **Left** Learning curve with α starting at 0.1. **Right** Learning curve with α starting at 0.9.

Chapter D

Comparison of learners

D.1 Value Function Estimation Errors (Q4.1 and 3)

The graphs representing the evolution of the MSE of the two agents are shown in figure 12.

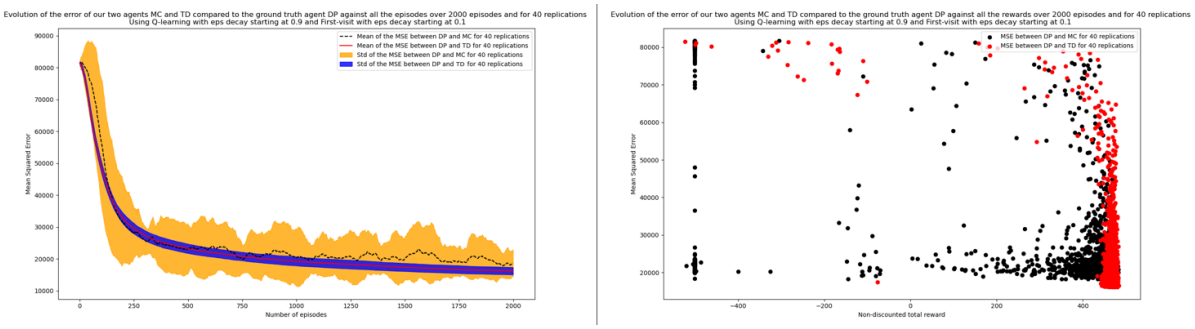

Figure 12: **Left** Mean Squared Errors w.r.t the number of episodes for both learners with 40 replications and 2000 episodes. **Right** Mean Squared Errors error w.r.t the undiscounted total reward for both learners with 40 replications and 2000 episodes.

D.2 Analysis of the MSE through the episodes (Q4.2)

From the graph on the left in figure 12, it can be seen that the TD agent converges faster than the MC agent. This difference in convergence is explained by the use of bootstrapping for the TD agent, which allows it to learn from 'incomplete' episodes (less exploration), unlike the MC agent which waits until the end of the episode to learn and update its policy.
It is worth noting that even if at a certain point the MC method seems to be more efficient than the TD method, it is the latter that still performs better from the point of view of error, convergence speed, and standard deviation, which remains less important than for the MC agent (based on cumulative rewards).

D.3 Analysis of the MSE through the undiscounted total rewards (Q4.4)

The graph on the right of figure 12 characterizes the optimality of our agent's policy with respect to the value function. It can be seen that TD agent has better values regarding the MSE thanks to bootstrapping. It is also worth noting that the TD agent is better since the great majority of the points (each representing an episode) are located in the upper range of undiscounted reward (> 400) whereas for the other agent we have a more fragmented distribution (from 0 to > 400) probably because of the difference between the standard deviations of the two methods.
It is important to have a good value function estimate to obtain a good reward because if our agent moves to the wrong states, it is likely that the resulting rewards will be low.
Finally, this graph shows that the TD agent converges despite a less good value function at times (high MSE and high reward). Contrary to TD, the MC agent will explore more and once the exploration period is over (low MSE and low reward), we obtain good results (low MSE but high reward).