Practical 3: Reinforcement Learning

The goal of this practical is to familiarise ourselves with Q-learning, a basic Reinforcement Learning (RL) algorithm. RL algorithms are a family of powerful methods to solve complex problems in a semi-supervised fashion. In other words, by contrast to neural networks, which learn on the basis of examples, RL occurs by means of a scalar signal (reward).

As a general rule, you may consider that any RL implementation must abide by several requirements:

- First, the problem is defined as a Markov Decision Problem, where the actions of the agent do not in general determine the outcome (they can have a stochastic outcome).
- Second, the definition of the state is crucial, as it can vastly alter the problem, facilitating its solution or rendering it intractable. Because of this, you should devote some time to think of what the best definition of the state to use for your specific problem. For example, in the case of chess (like the one of our practical), the problem consists of finding a policy that leads to a check mate. As a state, we could use a list containing the positions and types of pieces, or the entire board. The problem may be solved in either case, but the amount of memory and calculations required varies largely in either case.
- Third, solving the problem is equivalent to finding the right policy to transition across states, starting from an initial state to a final state (which is not necessarily known beforehand).
- Fourth, since the "right" (optimal) policy is the one leading to maximum reward, the manner in which we assign reward (the so-called reward function) is also crucial to find the desired state.
- Five, RL algorithms learn on the basis of trial-and-error assessment, quantified in term of reward. In other words, learning occurs when the prediction of expected reward associated to a particular transition differs from the real reward obtained when executing the transition. Conversely, the algorithm is said to converge when the rewards obtained approach expected values.
- Sixth, the definition of the reward assignment function (the one quantifying the reward associated to each transition) often requires significant introspection, as it may yield radically different solutions to the same problem. For example, if you want to find the shortest path from point A to point B in a grid, you may assign a negative value at each intermediate step, and a large value when reaching B.

The Simulator

The same simulator you used for the previous practicals, consisting of four hierarchical classes: Aichess, Chess, Board and Piece. The latter three implement the dynamics of a chess game, which may be run by two players (run the main on the class Chess to this end). The Aichess class is a capsule of the other three, with the purpose of implementing AI algorithms to analyse and alter the dynamics of the chess game.

The Definition of the State

Although you have full autonomy as to the specific definition of state, we suggest that you start by defining the state in terms of the positions and piece types, as you did in the previous practicals.

Your Work

1. (4 pts) We will start by tackling the simple problem seen in class of finding a path from start to goal in the following scenario:

3				Goal
2		▨		
1	Start			
	1	2	3	4

Remember to provide the first, two intermediate and the final Q-table in every case.

a. Implement the Q-learning algorithm to find the optimal path considering a reward of -1 everywhere except for the goal, with reward 100.

i. (0.4 pts) Print the first, two intermediate and the final Q-table. What sequence of actions do you obtain?
ii. (0.4 pts) After trying for a bit, what is your parameter choice for alpha, gamma and epsilon? Why?
iii. (0.4 pts) How do you judge convergence of the algorithm? How long does it take to converge?

b. Try implementing the more accurate reward given by:

3	-3	-2	-1	100
2	-4	▨	-2	-1
1	-5	-4	-3	-2
	1	2	3	4

which is a function of the distance to the goal.

i. (0.4 pts) Answer the questions of the previous section for this case.
ii. (0.4 pts) What is the effect of the new reward function on performance?
iii. (0.4 pts) How does this relate to the search algorithms studied in P1? Could you apply one of those in this case?

c. The main novelty in RL algorithms with respect to the search algorithms in P1 is that they can be applied in stochastic environments, where the agent doesn't fully determine the outcome of its actions.

i. (0.6 pts) **Drunken sailor.** Your agent is now a drunken sailor trying to get to bed after a good share of whiskey and shanties: their legs don't seem to obey them all the time. Introduce stochasticity (= randomness) by enforcing that only 99% of the steps intended by the sailor are actually taken, the rest leading randomly in any other possible direction.
ii. (1 pts) Use at least one of the two rewards proposed:
1. (0.15 pts) What is your parameter choice? Why?
2. (0.15 pts) Assuming the sailor is in a state that allows learning: how many drunken nights are necessary for them to master the perilous path to bed? Compare to the previous, deterministic scenario.
3. (0.2 pts) What is the optimal path found? If we watched the sailor try to follow it, would they always follow the same path?
4. (0.5 pts) Could we apply one of the algorithms in P1 here? Why? **Hint**: think of the notions of *deterministic* vs *random* and of *path* vs *policy*.

2. (6+1 pts) Now, let's move back to the chess scenario, namely the first board configuration of P1. Remember that we have the black king, the white king and one white rook, and that only whites move. Remember to provide the first, two intermediate and the final Q-table in every case.

a. Adapt your Q-learning implementation to find the optimal path to a check mate considering a reward of -1 everywhere except for the goal (check mate for the whites), with reward 100.

i. (0.5 pts) What sequence of actions do you obtain?
ii. (0.5 pts) After trying for a bit, what is your parameter choice for alpha, gamma and epsilon? Why?
iii. (0.5 pts) How do you judge convergence of the algorithm? How long does it take to converge?

b. Try now with a more sensible reward function adapted from the heuristic used for the A* search:

i. (0.5 pts) Answer the questions of the previous section for this case.
ii. (0.5 pts) What is the effect of the new reward function on performance?

c. **Drunken sailor.** On their way to bed, our drunken sailor sees a chessboard on a table, coincidentally configured as in the previous section. They have seen the captain play with the first mate and want to give it a try, but only have a rudimentary knowledge of the rules (they know how each piece moves and what is a check mate, but not that blacks move as well).

   i. (0.5 pts) Introduce stochasticity (= randomness) by enforcing that only a given percentage of the moves intended by the sailor are actually taken, the rest taken randomly from all other possibilities.
   ii. (1 pts) Use any reward you prefer:
      1. (0.5 pts) What is your parameter choice? Why?
      2. (0.5 pts) Assuming our obsessive sailor is in a state that allows learning: how many games do they have to play before they are satisfied that they have found the best strategy and can go to bed? Compare to the previous, deterministic scenario.
      3. (0.5 pts) What is the optimal path found? If we watched the sailor try to follow it, would they always follow the same path?

d. (0.5 pts) Compare the application of Q-learning in this chess scenario with that of the grid of exercise 1. How do the two scenarios differ? How does that translate into your results?

e. (1 pts) Compare the use of Q-learning with that of search algorithms of P1 for the chess scenario seen here, both in the deterministic and stochastic case.

f. (*Voluntary: 1 pts*) Use your Q-learning implementation of the drunken sailor on the second board configuration of P1, and try to find the best parameter combination for quick convergence. How robust was your previous parameter choice? Discuss your results.

Submission Deadlines & Interviews

- **Groups B and F** should submit their work (code + report) by **Saturday, 13/12, at 23:59**. The respective interviews will take place on **Tuesday, 16/12**, during each group's usual laboratory session.

- **Group A** should submit its work (code + report) by **Sunday, 14/12, at 23:59**. The respective interview will take place on **Wednesday, 17/12**, during the usual laboratory session.