# Practical 3: Reinforcement Learning

The goal of this practical is to familiarise yourselves with basic Reinforcement Learning (RL) algorithms, which is a family of powerful methods to solve complex problems in a semi-supervised fashion, by means of a scalar signal (reward). Taking as a starting point the design and inspiration provided by the heuristic and algorithms engineered in our practical 2 (adversarial games), we propose to endow our simulated players with the ability of adaptively learning novel strategies to act in their chess-like environment.

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

- First, the problem must be defined in terms of states composing a Markovian space, which must be navigated from an origin to a final state.
- Second, the definition of the state is crucial, as it can vastly alter the problem, facilitating its solution or rendering it intractable. For example, in the case of the chess solving problem ---like the one of our practical, the main problem consists of finding the set of transitions such that they lead 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 set of transitions across states, starting from an origin state to final state. Typically, we know the origin, but not the end-state.
- Fourth, since the “right” criterion 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. This is the basis of the Temporal Difference (TD) algorithm.

## The Simulator

As for practicals 1 and 2, you will use the same chess simulator, consisting of four hierarchical classes: aichess, chess, board, piece. The latter three implement the dynamics of a chess game, which may be run by two players. Please, run the main on the class Chess to this end. Finally, the Aichess class is a capsule of the former three, with the purpose of implementing AI algorithms to analyse and alter the dynamics of the chess game.

## The Definition of the State and the Initial State

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

**Your Work**

1. RL algorithms may operate both in stochastic and deterministic environments. This first part of the practical consists of implementing a Q-Leaning algorithm that learns to solve the same problem proposed in practical 1, where only the white pieces may move. This is an example of the simpler case of deterministic environments. Your instruction is to implement a Q-learning algorithm, updated via Temporal Difference, which finds the path towards check-mate in the least number of moves possible. Use a table to store the corresponding Q-Values. Comment the code accordingly (6p).

2. Using the code of practical 2 as a starting point, program two RL agents (Whites & Blacks), and make them learn to compete each other until check-mate. Use Q-Learning on either side (4p).

    a. Which differences can you describe of the behaviour of these agents with respect to that of point 1.
    b. How long does it take to learn?
    c. What happens if you vary their relative learning rates?

**Documentation**
The python code provided is documented, and provides a straightforward structure for you to study and analyse. Although you do not have to understand every single detail, you need to build sufficient intuition about the structure of the underlying classes to be able to solve the problems here described.