Project

Based on the game tic-tac-toe, how can you develop an MCTS tree search algorithm that can reliably beat a random agent ?

- Create the environment along with the terminal conditions where the reward is 1 for the winner and -1 for the looser.
- In the main part of the project, the opponent is purely random at all times, the play of the opponent can be seen as part of the transition function.
- You play first.
- You can restrict your code to starting from this position :

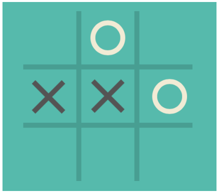

FIGURE – Illustration of the starting state for tic tac toe.

Project

Your need to provide :

- A report of maximum 2 pages pdf for your results (+figures ok). In particular, you should mention the probability of victory for all five possible actions in the restricted case, given the optimal policy. Your report should also discuss the results and provide meaningful information about the convergence process.
- You should provide the source code of your scripts (python). You are allowed to use existing code for the tic-tac-toe game (not necessarily easier !) as long as you mention the source explicitly both in the code and in the report.
- groups of 2, different from previous projects.

To get more than 8/10 :

- 2/10 points : consider a new setting where you assume that the opponent plays optimally. Hint : make use of the “maximin” principle.

Hints for the project

Due to the limited setting required based on the given starting position, you don't necessarily need to code the whole game.

- You can "hardcode" the only three winning positions for the crosses, and the only two winning positions for the "circles".
- One simple way to encode the state of the game is a list of up to 5 scalars $\in [0, 4]$ where each scalar corresponds to one free position. For instance, when the list is for instance {2,3}, it means that you choose to put a cross at position 2, the opponent at position 3. The next branching factor is 3 and the next action has to be picked from {0,1,4}.
- You have only a sequence of two actions to consider (out of 5 positions and then out of 3 positions), your last action is automatically determined by the fact that there is only one position left (for the main part of the project, not for the last 2 points).