CS7IS2: Artificial Intelligence Assignment 2

In this assignment you will implement Minimax and Reinforcement Learning algorithms for playing 2 games, Tic Tac Toe and Connect 4, and compare their performance. This assignment is worth **25% of your overall CS7IS2** mark.

Please note this is an **individual** assignment. You are allowed to discuss ideas with your colleagues but do not share code. Any suspected plagiarism will be dealt with in line with University policies https://www.tcd.ie/teaching-learning/academic-policies/plagiarism/

Submissions are via Blackboard, and due by **Thursday April 6th 5pm**.

You have an option to **penalty-free submit assignment until Sunday April 9th** evening (midnight) but please note there will be no Blackboard/email/TA/lecturer support for assignment or any other technical issues past the official deadline. Any requests for extensions (on medical or other similarly serious grounds) have to be received before the official deadline.

Assignments submitted later than Sunday 9th will be **lowered by 20% of the achieved mark per day**.

Assignment specification:

1. Implement in Python (or reuse existing open source) games of Tic Tac Toe and Connect 4
2. Implement Minimax algorithm (with alpha-beta pruning) and tabular Q-learning Reinforcement Learning algorithm for playing each of the two games above
3. Implement a default opponent that will play these games against your algorithms. It does not have to be very advanced but it has to be better than fully random (eg, opponent must select a winning move if it exists. It must select a blocking move (i.e. prevent opponent from winning) if it exists.
4. Compare the following, presenting graphs and analysis/conclusions, making sure you let agents play a sufficient number of games to be able to draw conclusions:
    - How do your algorithms compare to each other when playing against default opponent in Tic Tac Toe
    - How do your algorithms compare to each other when playing against default opponent in Connect 4
    - How do your algorithms compare to each other when playing against default opponent overall
    - How do your algorithms compare to each other when playing against each other in Tic Tac Toe
    - How do your algorithms compare to each other when playing against each other in Connect 4
    - How do your algorithms compare to each other when playing against each other overall

    Deliverables: Python code, document **analysing and discussing the performance comparisons**, including visual comparison/graphs/tables/screenshots as required. This

document also needs to include **justification for any design choices** you make in the implementation (eg RL state space, rewards)

5. Record a brief (max 5 minutes) screen-grab (and if required voiceover) video demo of your maze solvers (showing input parameters, output screens etc)

Mark breakdown

1. While no marks are awarded specifically for the demo, the demo is a **mandatory** part of the assignment, without which the rest of the assignment will not be marked
2. Algorithm implementation – 12.5% per algorithm per game, for the total of 50% of the mark
3. Analysis document 50% (proportional to algorithms implemented, eg if only 1 game is implemented rather than both, maximum achievable mark for the document is 25%).

Important scalability note and instructions

You'll notice that connect 4 problem is a lot more complex than tic-tac-toe, due to the number of states to visit/maximum length of the game (9 vs 42 moves). Executing connect 4 minimax will not be feasible on your machine. First confirm this, by executing it for eg 30 minutes and counting how many moves (out of how many max moves) did it visit (reporting the findings). From this point on, there are multiple ways to proceed – preferred way would be to do a depth-limited search combined with a heuristic function (try looking only 5 moves ahead rather than full game), but you could also reduce the size of the board, hardcode certain number of initial moves to visit only a subset of the overall space etc. Justify your choice and report the findings. Similarly, training RL for connect 4 would require a very large number of training episodes, so you'll need to use techniques similar to the above to reduce the search/training space. Instead of playing against a semi-intelligent opponent, use completely random opponent, as it is easier to beat, so even somewhat trained RL agent should be performing better than random.

Submission instructions

Please submit a **single zip file** containing all python code, **readme.txt** with command lines use to run each of your algorithms, **pdf of the performance analysis document**, and the **video** of the demo. In addition, code for the functions (as text, not screenshot/image) containing implementations of the 4 algorithms (2 games/2 algorithms each) **need to be added as the appendices 1-4 to the document**. Please note the maximum file size for the whole submission is 100mb, so you'll need to lower the quality of your demo video if it defaults to more than this (or include only a link to a publicly accessible file).

Additional Important Notes:

- If reusing existing open source implementation of the games, make sure to credit it in your design document (including link to the repository). Make sure that the licence under which the code is distributed allows you to reuse it.

- The metrics which to use to analyze the performance are not pre-specified, it is up to you to identify suitable metrics for each algorithm, and in the design document to justify your choices.

- The purpose of the demo is to demonstrate that your code is fully functional and meets the brief requirements, in case we are unable to execute your source files due to compatibility issues. Please ensure that the video sufficiently captures this.