COSC1125/1127 Artificial Intelligence
Assignment 2 Connect 4
Due 11:59pm Thursday 2 May 2019

This is an individual or pair assignment. It has **100 points** in total and is worth 15% of the overall course grade. You may not collude with any other individual, or plagiarise their work. Students/Pairs are expected to present the results of *their own thinking and writing*. Never copy another student's work (even if the other student "explains it to you first.") and never give your written work to others. Never copy from the web or any other resource. Remember you are meant to generate the solution to the questions by yourself. Suspected collusion or plagiarism will be dealt with according to RMIT University policy.

If you choose to work as a pair, you need to clearly spell out your individual contribution to the assignment in the report. We assume each of you will contribute equally to this assignment.

Task

You need to create a minimax agent evaluation function for the game of Connect 4 (or you can create an agent with more advanced AI techniques if you wish) the file that you need to edit can be found in **`connectfour/agents/agent_student.py`**.

In the commercial game of Connect 4, two players take turns to move on a vertical board with 6 rows and 7 columns. A move consists of placing a counter onto the top of one of the columns. The counter then falls, by gravity, to the last free position at the bottom of that column. The first player to get 4 of their counters in a line, horizontally, vertically or on a diagonally is the winner. To play a web implementation of the game, please visit https://connect4.gamesolver.org/. In the generalized game there are *nrows* rows, *ncols* columns and the winner is the first to get *k* counters in a line.

The Program

The start-up code for the Connect-4 game is available in the GitHub Project: https://github.com/StevenKorevaar/ai1901-connectfour. To set up the project simply clone the base repo and rename the new clone to "<teamname>-ai1901-connectfour" and add StevenKorevaar as a collaborator to the repository (you also should make your cloned repository private to ensure no other students can see the code). Throughout development you should follow standard Software Engineering practices in handling the repository (including regular commits), we will be checking this to ensure all team members are contributing equally.

Follow the instructions in the README.md file to install and get the code ready to run. You can then run the game with a command in the following format:

```
python -m connectfour.game --player-one RandomAgent --player-two StudentAgent
```

There are three agents that we will be marking you on for winning against that you to can test your own agent against.

- **Random:** This player just moves randomly, but will win given the opportunity. It should be very easy to beat this opponent!
- **Monte Carlo Tree Search:** This is a medium level player that uses Monte Carlo Tree Searching to find a good move.
- **Hard:** This agent will require a good understanding of minimax, heuristic design, and the game Connect 4 in order to beat.

The code for the first two agents can be found in `connectfour/agents/computer_player.py` and in `connectfour/agents/monte_carlo.py`. The final agent will take part in the contest section of the assignment so you can see how it performs against that while not having access to the internal code.

The first task for this assignment is to create your own evaluation function for a Minimax Agent:

```
        def evaluateBoardState(self, board, player)
```

This evaluates the current board position and returns a double between -1 (very bad) and 1 (very good) indicating how good it is. player is the player you are evaluating the board for (either 0 or 1).

The minimax algorithm itself has been provided for you: **you are free to modify it or use a completely different AI method if you choose provided you can justify your decision in the report**.

Your agent should also take a reasonable amount of time to make a turn, given every computer has different processing time, a soft limit of approximately 20+-5 seconds is the requirement. EG: An agent should take less than 20 seconds to compute and make a turn. This is not a strict requirement, but any agents consistently taking longer than 30 seconds will be deducted marks.

Marking

You also need to write a short (1-2 page) report describing your Agent's inner workings (may include a description of what your evaluation function looks at, or what other AI techniques you added to the agent). You must also describe the agent's performance, including its strengths and its weaknesses. Your mark in this part will depend on how well your agent plays and your understanding of what you have created:

- Pass (50+ marks): Your agent can beat the random agent
- Credit (60+ marks): Your agent can beat the Monte Carlo, and random agents
- Distinction (70+ marks): Your agent can beat the Hard, Monte Carlo, and random agents
- High Distinction (80+ marks): The same as for distinction, but your agent contains advanced heuristics and plays very strongly.

In order to ensure that your agents perform well in general in many different situations (and are not simply overfitted to counter only the agents we provide) we will be holding a round robin like contest of all student submitted agents through the assignment period so you can see how your agent performs against your peers, this will be run a few times per week to allow you to improve your agent over time. To enter your current agent you should take the version you want to submit and tag the commit on your repository with "c4-contest-submission".

Within your report make sure you include: Your name + Student number, your partner's name and students number, your team name, your contributions (breakdown of work) and a link to your GitHub Repository.

General Advice

Your evaluation function should look at the current state and return a score for it. As an example, the simple agent provided works as follows:

1. If the opponent has won this game, return *-1*.
2. If we have won the game, return *1*.
3. If neither of the players has won, return a random number.

To write your evaluation function, you need to think about how you would play a game of Connect 4 and how you would evaluate whether a particular board position was good or not.

You should put your agent in a file called `connectfour/agents/agent_student.py` - It will then be loaded when you run the command:

```
python -m connectfour.game --player-one RandomAgent --player-two StudentAgent.
```

You can play your agent against any of the included agents, or another agent (e.g., one of your fellow classmates') by copying the agent file to the agents directory, and then running the following command:

```
python -m connectfour.game --player-one RandomAgent --player-two
<new agent's python file name without '.py')>.<new agent's class name>
```

Please see the following Wikipedia page for more background information on Connect 4:

https://en.wikipedia.org/wiki/Connect_Four

Submission instructions

There are three submissions you need to make for this assignment:

1. Create a private Git Repository: a clone of the base project and add StevenKorevaar to the repository as a Collaborator and tag your release versions (you can tag and retag as much as you like up until the submission date, once that has past we will be using the code uploaded to Canvas with
"c4-contest-submission": https://github.com/StevenKorevaar/ai1901-connectfour

2. Fill out the team registration form: https://goo.gl/forms/sYe1tBK6oRImZ3es2

3. You are also required to submit via Canvas the 2 files required (see below) in a single .zip file before the due date. This will be your final submission

Your submission to Canvas should be named: *`<yourstudentnumber>.zip`*: and within it should be the following 2 files:

1. A text file called *`agent_<yourstudentnumber>.py`* containing your Connect 4 agent class implementation.
2. A PDF report called `<yourstudentnumber>.pdf` containing a description of how your Connect 4 agent works, and its weak and strong points.

++
Late Submissions: 10% of the possible marks for this assignment will be deducted for each day late and assignments submitted 5 or more days late will not be marked.
++