CSC 215-01 Artificial Intelligence (Fall 2019)

Mini-Project 4: Solving Tic-Tac-Toe and Wild Tic-Tac-Toe using Minimax Search

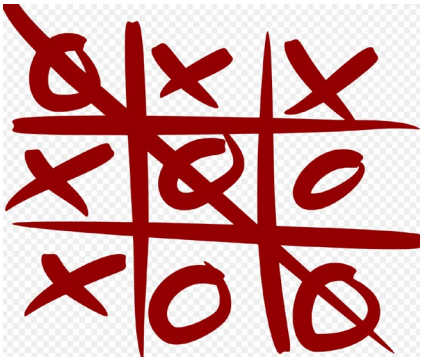
Due at 5:30 pm, Wednesday, November 13, 2019

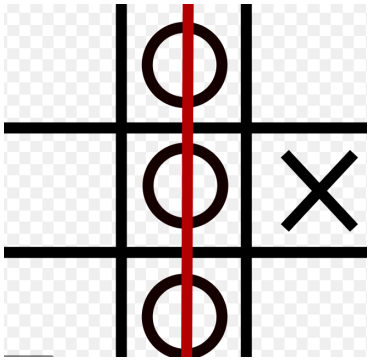
Demo Session: class time, Wednesday, November 13, 2019

1. Problem Formulation

Download the two notebooks on Canvas. This project is twofold:

1. Complete the minimax function for the Tic-Tac-Toe and Wild Tic-Tac-Toe.
2. Verify the characteristics of each game by simulations. You may play 100 games for each settings (i.e., optimal vs optimal; optimal vs random), observe the results, and include your findings in your report.





1.1 Tic-Tac-Toe

Rules:

- The first player plays x and the second player plays o. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. If all cells are used without someone getting a three in a row, the game is a draw.

Characteristics:

- If both players play optimally, the game will always be a draw.
- If one player is optimal and the other is random, the optimal player will win most of the games and the random player will never win.

1.2 Wild Tic Tac Toe

Rules:

- On each turn a player can play either x or o. The player who gets the first three (x's or o's) in a row wins the game. If all cells are used without someone getting a three in a row, the game is a draw.

Characteristics:

- If both players play optimally, the first player can always win if the middle cell is chosen as his/her first move, and if the other cells are chosen, the game will be a draw.

- If one player is optimal and the other is random, the optimal player will win most of the games and the random player will never win.

I wrote three independent functions for each game **in the provided two notebooks on Canvas**:

- optimal_vs_optimal: both players are optimal. It is pure optimal, meaning that even the first move is optimal. Our minimax function always returns the first best score and plays that move.

- random_vs_optimal: the first player plays randomly, and the second player plays optimally.

- you_vs_optimal: user gets to play as the first player against an optimal player.

2. Output examples

In [10]:
```
optimal_vs_optimal()
```

```
Player x and Player o Both play optimally.

Board :

  0  |  1  |  2  |
-----------------
  3  |  4  |  5  |
-----------------
  6  |  7  |  8  |
-----------------

x :

  x  |  1  |  2  |
-----------------
  3  |  4  |  5  |
-----------------
  6  |  7  |  8  |
-----------------

o :

  x  |  1  |  2  |
-----------------
  3  |  o  |  5  |
-----------------
  6  |  7  |  8  |
-----------------

x :

  x  |  x  |  2  |
-----------------
  3  |  o  |  5  |
-----------------
  6  |  7  |  8  |
-----------------



o :

  x  |  x  |  o  |
-----------------
  3  |  o  |  5  |
-----------------
  6  |  7  |  8  |
-----------------

x :

  x  |  x  |  o  |
-----------------
  3  |  o  |  5  |
-----------------
  x  |  7  |  8  |
-----------------

o :

  x  |  x  |  o  |
-----------------
  o  |  o  |  5  |
-----------------
  x  |  7  |  8  |
-----------------
```

```
x :

  x  |  x  |  o  |
-----------------
  o  |  o  |  x  |
-----------------
  x  |  7  |  8  |
-----------------

o :

  x  |  x  |  o  |
-----------------
  o  |  o  |  x  |
-----------------
  x  |  o  |  8  |
-----------------

x :

  x  |  x  |  o  |
-----------------
  o  |  o  |  x  |
-----------------
  x  |  o  |  x  |
-----------------

Draw!
Out[10]: 'Draw'
```

3. Grading breakdown

You may feel this project is described with some certain degree of vagueness, which is left on purpose. In other words, **creativity is strongly encouraged**. Your grade for this project will be based on the soundness of your design, the novelty of your work, and the effort you put into the project.

Use the evaluation form on Canvas as a checklist to make sure your work meet all the requirements.

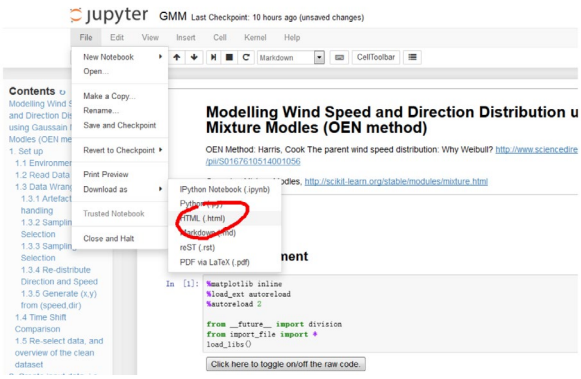
Implementation	**70 pts**
Your report	**20 pts**
In-class demo	**10 pts**

4. Teaming

Students must work in teams of 2 people. Think clearly about who will do what on the project. Normally people in the same group will receive the same grade. However, the instructor reserve the right to assign different grades to team members depending on their contributions. So you should choose partner carefully!

5. Deliverables

(1) The HTML version of your notebook that includes all your source code. Go to "File" and then "Download as". Click "HTML" to convert the notebook to HTML.



5 pts will be deducted for the incorrect file format.

(2) Your report in PDF format, with your name, your id, course title, assignment id, and due date on the first page. As for length, I would expect a report with more than one page. Your report should include the following sections (but not limited to):

- **Problem Statement**
- **Methodology**
- **Experimental Results and Analysis**
- **Task Division and Project Reflection**

In the section "**Task Division and Project Reflection**", describe the following:

- who is responsible for which part,
- challenges your group encountered and how you solved them
- and what you have learned from the project as a team.

10 pts will be deducted for missing the section of task division and project reflection.

All the files must be submitted **by team leader** on Canvas before

5:30 pm, Wednesday, November 13, 2019

NO late submissions will be accepted.

6. In-class Demo:

Each team member must demo your work during the scheduled demo session. The following is how you should allocate your time:

- Model design (1 minute)
- Findings/results (2 minute)
- Task division (1 minute)
- Challenges encountered and what you have learned from the project (1 minutes)

Failure to show up in demo session will result in **zero** point for the project.

7. Hint:

Cheatsheet will be provided

8. Think beyond the project

- What other characteristics are there for the two games?
- Play with your AI using the function ***you_vs_optimal()***, please let me know how well you perform between an optimal player and a random player.
- In this project, each move in the games is optimal because we use minimax function (notice that we search the entire game tree since Tic-Tac-Toe has a relatively small game tree). We did not limit search depth or use any heuristics as terminal evaluation functions or do any alpha-beta pruning. A more efficient game can be implemented by limiting search depths, using terminal evaluation functions (heuristics or machine learning models), or adopting alpha-beta pruning.