CIS 412 Artificial Intelligence Spring'16

Project 2 Team assignment

Development of AI for Ultimate Tic-Tac-Toe.

Assigned March 11 Due April 1, 2016

You are given a basic game of Ultimate Tic-Tac-Toe (two players can play on one computer). You are provided with a board to play the game and get used to the rules. The game rules as well as explanation on the provided files is given below. Your task is to develop the following three versions:

1. "Dumb" game, meaning implementing depth-bound min-max search with evaluation function, so the computer can play with a human player.
2. Somewhat intelligent game, meaning implementing alpha-beta pruning with e.g. 2-ply lookahead.
3. Very intelligent game, meaning alpha-beta pruning and improving on the number of lookahead levels (a move should not take more than 2.5 minutes), evaluation function, pruning.

A competition will be organized between the "Very intelligent" AIs. The winning team will be exempt from Final exam. The presentations to instructor will be held between April 4 and 8, 2016. The competition will be held on April 8th and 11th, 2016.

Elaboration:

1. You are provided with a Javascript version of the board and game. It is strongly recommended you use that. In case you prefer to code yours or find something in another language, keep in mind that there can be NO intelligence in the found version. Otherwise, your team will be disqualified and the grade will reflect that.

2. You will deliver:
   a. Three functional versions of AI-ed game as described above (or one version with three options), i.e. commented code and applicable files.
   b. Documentation on strategies, lookahead levels, evaluation functions used, etc.
   c. Demonstrate your game and its capabilities in competition.

3. Since this is a team effort, during the demonstration you will be asked to explain the contribution of each team player.

Grading:

1. "Dumb" version - 15 points for working game with documented code and necessary documentation. 7 points for code, 8 points for report on depth-bound min-max.
2. "Somewhat intelligent" version - 25 points for the same as above and design of evaluation function. 10 points for code, 15 points for report with all elements (2b above).
3. "Very intelligent" version - 20 points for evaluation function, strategy, etc. 7 points for code, 13 points for report, finding and final conclusions on strategy comparison.
4. 10 points for presentation to instructor.
5. 10 points for competition participation.

Game particulars

What each file does:

UltimateTicTacToe.html

Open this to play game. Links all other files, writes title text, and calls functions to draw the game board and start the game.

UltimateTicTacToe.css

Contains formatting info for centering everything on the page and setting the title text font.

UltimateTicTacToe.js

Contains all functions for playing the game. drawBoard() draws the board using SVG elements. startGame() sets up all the interactive elements of the board and gets ready for player 1 to play. Anytime a player clicks on a valid cell, markCellForPlayer() is called, which updates the board and the scoring in selected and wonCells, then checks for a victory, and if there is none, sets up for the next player to go. The two global variables, player1IsHuman and player2IsHuman determine whether human players are playing or AI players. If either of them are false, their corresponding AI functions in UltimateTicTacToePlayerAI.js are called (calculateAIPlayer1Move and calculateAIPlayer2Move, respectively).

UltimateTicTacToePlayerAI.js

This contains the two functions mentioned above for player 1 and player 2 AI. These work by calling other functions for actually calculating what move to play. Three such functions should be implemented. This files is provided empty for your own writing.

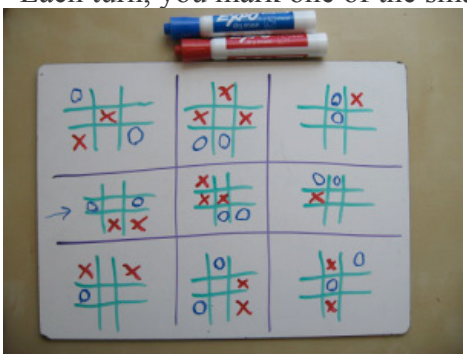
To write your own AI:

Since all of the game AI is contained within UltimateTicTacToePlayerAI, writing your own AI to play the game just requires erasing the contents of that file and writing your own calculateAIPlayer1Move and calculateAIPlayer2Move functions.

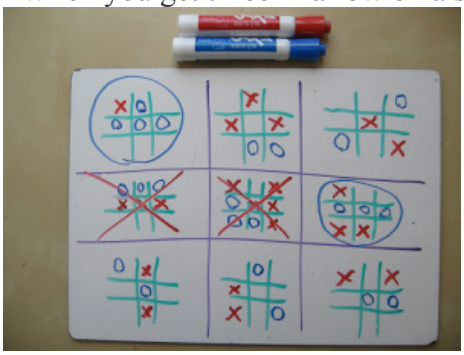
Since the game is written in JavaScript and meant to run in a web browser, it is restricted slightly by the tendency of internet browsers to kill pages that try to use too much memory. The running speed is also limited by the fact that JavaScript is interpreted, not compiled. This requires that any algorithm for playing the game be written with efficiency in mind, because it will not work if it's not efficient.

Game rules

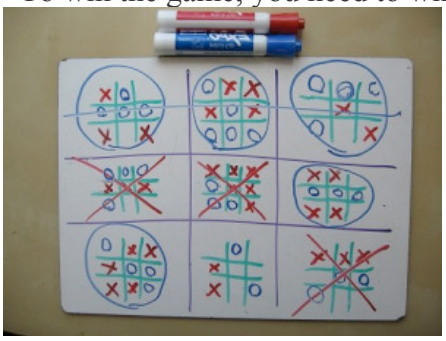
1. Each turn, you mark one of the small squares.



2. When you get three in a row on a small board, you've won that board.

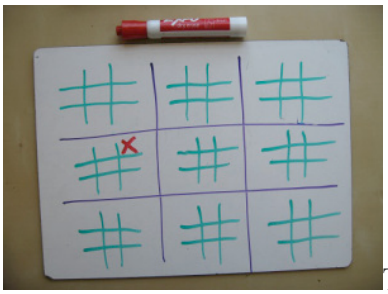


3. To win the game, you need to win three small boards in a row.

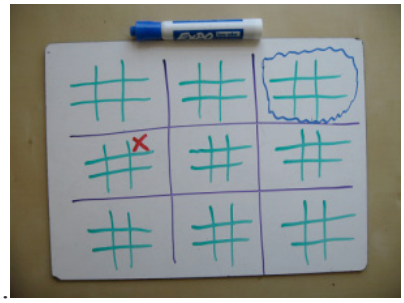


You don't get to pick which of the nine boards to play on. That's determined by your opponent's previous move. **Whichever *square* he picks, that's the *board* you must play in next.** (And whichever square *you* pick will determine which board *he* plays on next.)

For example, if I go here…

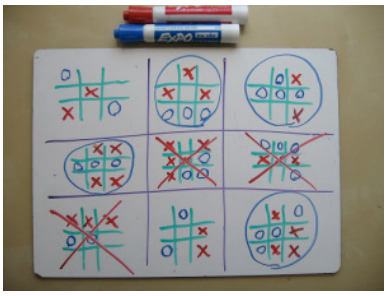


Then your next move must be here…



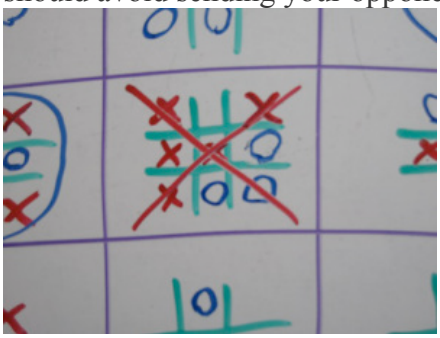
This lends the game a strategic element. You can't just focus on the little board. You've got to consider where your move will send your opponent, and where his next move will send you, and so on.

The resulting scenarios look bizarre. Players seem to move randomly, missing easy two- and three-in-a-rows. But there's a method to the madness – they're thinking ahead to future moves, wary of setting up their opponent on prime real estate. It is, in short, vastly more interesting than regular tic-tac-toe.



A few clarifying rules are necessary:

1. *What if my opponent sends me to a board that's already been won?* In that case, congratulations – you get to go anywhere you like, on any of the other boards. (This means you should avoid sending your opponent to an already-won board!)



2. *What if one of the small boards results in a tie?* I recommend that the board counts for **neither** X nor O. But, if you feel like a crazy variant, you could agree before the game to count a tied board for **both** X and O.