Programming Project
Polar Tic-Tac-Toe

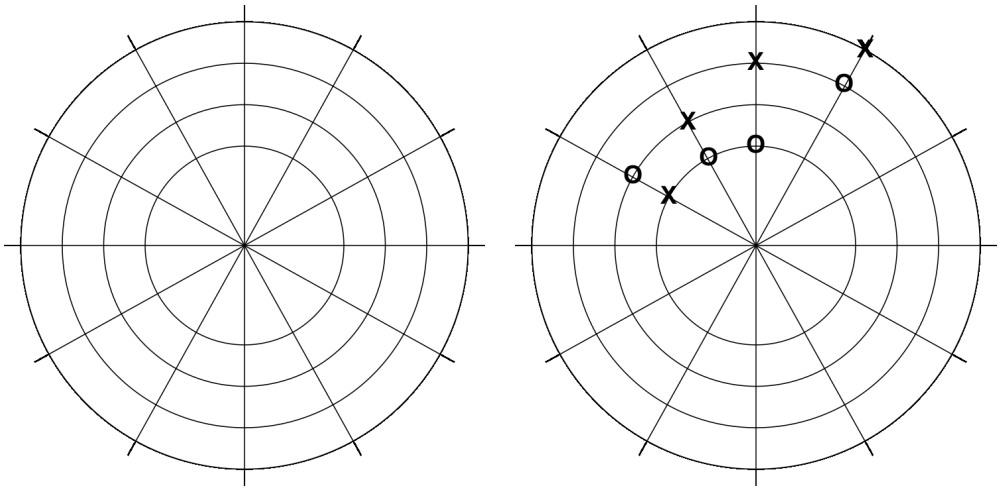
CSCI 446 – Artificial Intelligence – Fall 2014

Abstract

The goal of this programming assignment is to implement the game of Polar Tic-Tac-Toe as a running project and then create an "intelligent" player using a variety of techniqus from AI. Students will prepare a short paper as described in Section 3.

1 The Game of Polar Tic-Tac-Toe

Polar Tic-Tac-Toe is a variation of Tic-Tac-Toe where players alternate placing X's and O's on cross-points of the polar grid as shown in Figure 1(a). The first to place four X's or four O's consecutively, either along a cross line, around a circle, or "diagonally" on a spiral wins. Note that an X or O cannot be placed in the center crossing. For example, Figure 1(b) shows a diagonal win for X. In this game, we assume player X always goes first. Game play continues until either four in a row is achieved or it is impossible to obtain four in a row. In the latter case, a draw is declared.



(a) (b)

Figure 1: **(a)** A Polar Tic-Tac-Toe board with 12 radial lines (from the center) and four circles before play has begun. **(b)** A Polar Tic-Tac-Toe board demonstrating a win for X along a diagonal/spiral path.

2 Program Requirements

2.1 Submission Requirements

There are three specific deliverables for this project, all due at the end of the semester. These deliverables include the following:

1. A design document that details the software architecture and design decisions made.
2. The actual program, implementing all of the requirements listed in the next section.
3. A project report describing the results of experiments run that test the various options explored in this course.

You are free to use any computer and any programming language for this project; however, we advise you to avoid languages and systems such as SQL, Matlab, etc. Your program should be submitted as an archive, either a zip file or a tarball, with the filename consisting of team name and the assignment name, delimited by underscores, all lower case. For example, '`red_team_project.tar.gz`'. You should submit your archive through the dropbox in Desire2Learn.

Your archive should contain any and all files required to build and run your program in case the instructor wishes to run the program. It should also contain a README file, in which you list the files in your archive and a brief description of what each file contains. It should also contain any relevant notes on the functioning of your program, such as known bugs, parameters that result in excessive runtime or poor behavior, etc. You will still lose points if your program does not perform correctly, but if you document the failure modes, you can get partial credit. Your archive should also contain a `./data/` subdirectory, which should contain output from sample runs, timing information, and other information as requested. Finally, the archive should also include a PDF of your design document and a PDF of your final project report.

2.2 Feature Requirements

Your program must correctly implement the rules of Polar Tic-Tac-Toe, allowing all legal and only legal moves to be made, and correctly terminating and listing a winner when the game is over. Your game must allow for each player to be either a human or a computer-based agent. The user must be able to select one of these values for each player at runtime. For AI players, the user must be able to select the various configuration options discussed below; you should describe behavior for different configuration values in your README, and suggest reasonable values to use. Your AI players should be able to take the role of either X or O.

Human players should be prompted for a move on their turn; if they provide an invalid move, they should be prompted again until they give a valid move. AI players should never attempt to make an invalid move. The current state of the board should be displayed after each move, along with which player's turn it is. For AI players, the maximum search depth reached, number of nodes evaluated, and the time taken to decide on a move should be

displayed each time a move is selected. Project teams have complete latitude deciding how to render the board state for the user.

The principal requirements for your project are as follows. Note that your ability to satisfy these requirements will most likely emerge through the course of the semester, so you are *strongly encouraged* to work on the project as capabilities are developed. *Do not wait until the last minute or you will not be able to complete the project!!* From here on, we will refer to the game project as PTTT

- **R1:** PTTT shall implement the basics of Polar Tic-Tac-Toe as described above with one significant restriction. The first move may appear anywhere on the board. After the first move is made, future legal moves shall be restricted legal to be positions adjacent to an existing X or O on the board. (Note that this restriction might be relaxed at the end of the semester.) A legal move checker shall be included in the implementation and only legal moves shall permitted to be played.

- **R2:** PTTT shall incorporate a "win checker" using resolution (with unification). Thus, to satisfy this requirement, a set of first-order rules shall be developed to characterize what constitutes a winning state. These rules shall be used with the resolution win checker to examine the current board state and prove whether or not that board state contains a win. The predicate to be proven shall be denoted $win(player)$, where $player \in \{\mathrm{X}, \mathrm{O}\}$.

- **R3:** The computer-based player in PTTT shall implement a heuristic function that evaluates the quality of a board state. An example heuristic function that might be useful as a starting point can be found in Exercise 5.9 in the course textbook. PTTT may include multiple heuristic functions. Should multiple heuristic functions be included, the choices shall be provided as configuration options to the user. All heuristic functions shall be described fully in the design document. The performance of each heuristic function shall be reported in the final project report.

- **R4:** PTTT shall include a classifier (e.g., naïve Bayes, decision tree, nearest neighbor) to evaluate any state and predict whether or not that state will lead to a win or loss. The classifier shall be used as an alternative heuristic function and shall be compared to the heuristic function(s) developed in response to R3. The design of the classifier shall be described in detail in the design document, including any parameter settings. The results of comparing the classifier's performance to the heuristic function(s) developed in response to R3 shall be described fully in the final project report.

- **R5:** PTTT shall include a temporal difference neural network to evaluate any state and predict the expected outcome of the game from that state. The TD neural network shall be used as an alternative heuristic function and shall be compared to the heuristic functions developed in response to R3 and R4. The design of the TD neural network shall be described in detail in the design document, including any parameter settings. The results of comparing the neural networks' performance to the heuristic function(s) developed in response to R3 and R4 shall be described fully in the final project report.

- **R6:** The computer-based player in PTTT shall examine neighboring states for each ply, applying the heuristic function to those states. For a given ply, PTTT may restrict the neighborhood to a subset of available next moves. Should this option be exercised, the design and rationale for this restriction shall be described fully in the design document, and the effects of the restriction shall be described fully in the final project report. If this is a configuration option, the ability to select the neighbor shall be provided to the user.

- **R7:** The computer-based player in PTTT shall implement minimax search with and without alpha-beta pruning to play the game from either the X or the O perspective. Selecting whether or not to use alpha-beta pruning shall be offered to the user as a configuration option. The effects of using or not using alpha-beta pruning, including the effects of searching with various search depths, shall be described fully in the final project report.

3 Reporting Requirements

While this is a computer science course, writing is emphasized because it is an important part of being a good scientist or engineer. Having brilliant ideas is useless if you are unable to communicate those ideas to others in the scientific community. For this assignment, you will prepare two different documents—a design document and a scientific paper. The design document shall address the design elements that satisfy each of the requirements in the previous section. A template for the scientific paper will be provided for you to use. The template will include the expected formatting, as well as some suggestions and examples of what kind of content you should include. The written assignments will be graded on proper formatting, quality of writing, and quality of content.

The template will be available in LaTeX source and as a PDF generated from that source. If you use LaTeX for your report, you are welcome to use the source directly; otherwise, it is your responsibility to create a document that fits the template using the program of your choice. The writeup you turn in should be a PDF.

The template itself will be posted separately within Desire2Learn, and will contain more detailed information on what is expected in the paper.

4 Due Date

All deliverables (design report, fully commented code with sample runs, and project report) are due Friday, December 4, 2014 by the start of class.