CS152 Final Project

Rita Kurban

December 21, 2018

1 Problem Definition

Tic-tac-toe is a game for two players, X and O, who take turns marking the spaces in a grid. The player who succeeds in placing three of their marks horizontally, vertically, or diagonally wins the game. If no one won, and the board is full, it is counted as a draw.

Tic-tac-toe has multiple extensions where players can play on bigger boards and mark more consecutive tiles. Extensions make the game less predetermined and, thus, more fun.

For this paper, I programmed an AI opponent for a generalized tic-tac-toe game in Python. In order to make the opponent unbeatable, it was necessary to use an algorithm that could evaluate all the possible moves available for the computer player and use some metric to determine the best one. The minimax algorithm seemed to be the right choice for the job.

After implementing the AI logic as well as the tic-tac-toe class, I also coded a GUI in tkinter to make it user-friendly and demonstrate how the AI performs against a human player.

2 Solution Specification

The minimax algorithm performs a complete depth-first exploration of the game tree. If the maximum depth of the tree is d and there are m legal moves at each point, then the time complexity of the minimax algorithm is $O(m^d)$. Thus, the number of game states the algorithm needs to examine is exponential in the depth of the tree. This complexity is impractical for real-time games. For example, a naive estimate of the number of possible tic-tac-toe games on a 3x3 board is (3x3)! which equals 362,880 games. Even after the exclusion of games that end with 5, 6, 7, or 8 moves, we are still left out with 255,168 games (Bottomley 2001). Even though these calculations do not take symmetry of the board into consideration, the magnitude of the number indicates that even reasonably small boards such as 6x6 are almost impossible to calculate in a reasonable time.

Unfortunately, there is no way to get rid of the exponent. However, I managed to cut the complexity in half with the help of the alpha-beta pruning. The logic of this approach is that if there is a node n somewhere in the tree and a rational player has a better choice m either at the parent node of n or at any choice point further up, then n will never be reached in actual play. Therefore, it can be pruned with no consequences (Russell and Norvig 2016). Since I only have three outcomes: 1, -1, and 0, I also implemented another optimization that immediately returns the outcome in case the winning path is found.

3 Analysis of Solution

I created a user-friendly GUI in tkinter that supports different board sizes and consecutive square numbers. Figure 1 illustrates how it looks.

I separated the logic of the game from the GUI part so that they can be manipulated independently and it's impossible to change one of them by changing the other which was the case in my original implementation. Apart from that, my original implementation was pretty slow even for the 3x3

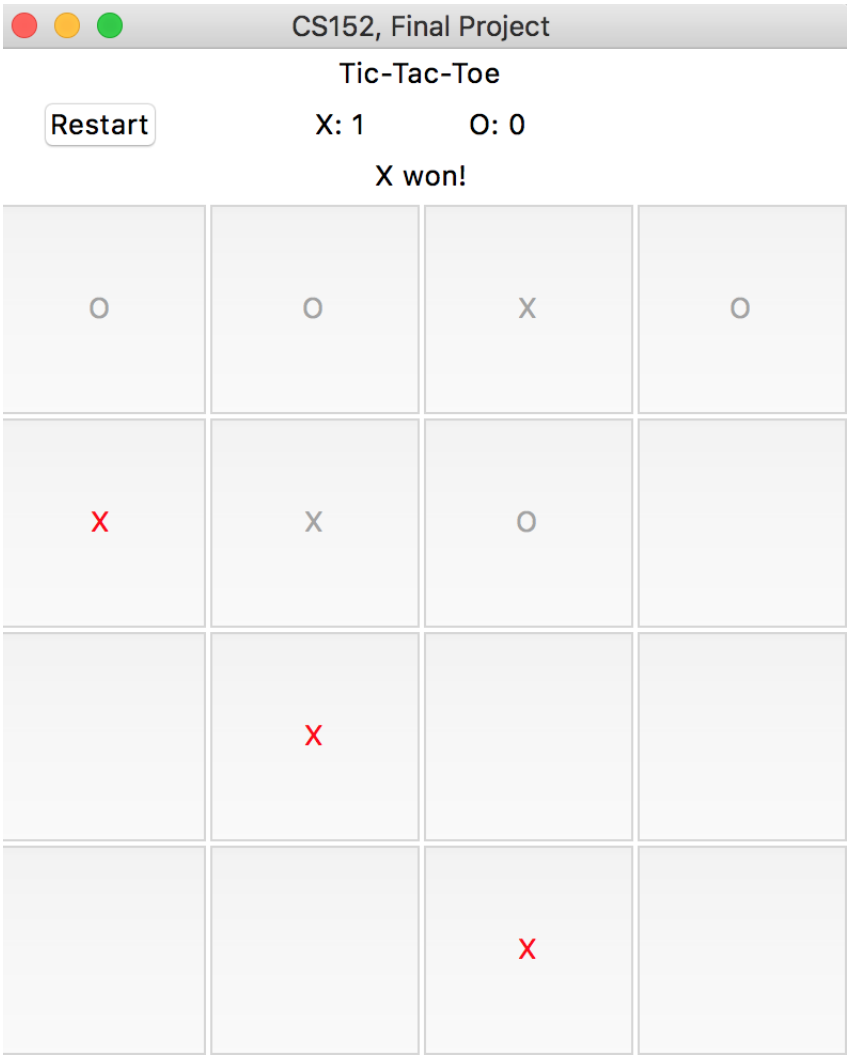

Figure 1: 4x4 Tic-tac-tor board

board which, of course, is not appropriate for the real-life games. I rewrote the game over function to only check whether the last move led to the end of the game instead of iterating through every single square. This optimization led to a significant improvement in speed. Unfortunately, it's still impossible to play on the 4x4 board in real time. However, 3x3 is currently really fast:

Board size	**Running time of the first AI move**
3x3	0:00:00.140841
4x4	0:01:19.463197

Table 1: Running times

4 Conclusion

I implemented an AI opponent for the tic-tac-toe game with the help of the minimax algorithm with alpha-beta pruning (#search). I also added two other optimizations that significantly increased the running time as well as the GUI for better user experience. One of the major limitations of my game, especially the extended version, is in the running time which can potentially be further improved by reducing the complexity of the code (e.g., decreasing the number of loops.)

References

Bottomley, Henry (2001). *How many tic-tac-toe (noughts and crosses) games are possible.*

Russell, Stuart J and Peter Norvig (2016). *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,