# Project 1 – 3×3 Tic-Tac-Toe Report

Rohman Sultan

October 3, 2021

## 1 Introduction

This project is about applying minimax search algorithm to a tic-tac-toe game. The objective is for the artificial intelligent agent to make the best possible move based on the given input file supplied by the user and output the result to a file. The caveat here is the user should be making the optimized move for the minimax search to be most effective.

## 2 Problems

I'm most productive in C++ and Python but I decided MATLAB would be a great opportunity to use for this project. I've encountered numerous problems and most being trivial. Arrays start at one for instance. The time spend tackling the language outweighed the problem itself.

The minimax search algorithm has a time complexity of $O(b^m)$ and space complexity of $O(bm)$ where b is the number of legal moves and m is the maximum depth of the tree. Even with a small problem size like tic-tac-toe, the execution speed of the program is noticeable slow on MATLAB.

I couldn't figured out how to run the program through a command-line interface and pass arguments to it, so I created an object that takes two parameters, input file and output file.

## 3 Approach

The program will exit after doing a single move (x,y) and the result be written to a file given by the user. If there is an immediate win in one of the available moves, take it. Otherwise, do a minimax on all possible moves.

$$Minimax(Position) = \begin{cases} 1 & \text{if X is the winner (ai),} \\ -1 & \text{if O is the winner (user),} \\ 0 & \text{if game is a draw and full,} \\ max(Minimax(Position_1), \ldots, Minimax(Position_n)) & \text{if it is X's turn,} \\ mini(Minimax(Position_1), \ldots, Minimax(Position_n)) & \text{if it is O's turn} \end{cases}$$

# 4 Class diagram

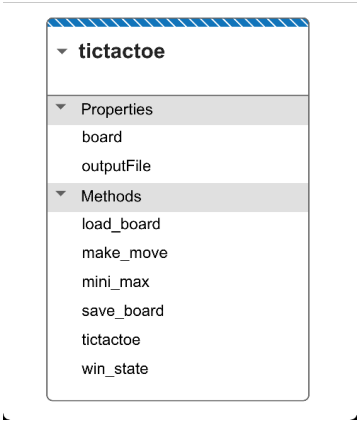


Figure 1: Class diagram for 3×3 Tic-Tac-Toe

Use the table and tabular environments for basic tables — see Table , for example. For more information, please see this help article on tables.

# 5 Results and discussions

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & -1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ -1 & -1 & 0 \end{bmatrix}$$

The minimax search makes a single legal move and saves the result to the specified output file. If there is a immediate win, take it.