AI Playing Connect 4

By Klarissa Navarro, Edward Cardenas, Erika Dickson

1. Problem

Our project will implement an AI agent that plays a competitive game of Connect 4 using Minimax with Alpha-Beta Pruning. The goal is to build an intelligent agent capable of making strategic moves against a human or another AI opponent. Connect 4 is a two-player, zero-sum game where the players alternate placing discs into a vertical grid, aiming to get four in a row either horizontally, vertically, or diagonally. The challenge is to create an efficient evaluation function that can analyze the game board and predict the best possible move, given the opponent's potential future actions. The Minimax algorithm will allow our AI to explore the game tree to make decisions, while Alpha-Beta Pruning will optimize it by reducing the number of nodes it evaluates.

2. Programming Language

2.1. **What Language:** Python

2.2. **Why:** Our team will use Python for this project because it is the preferred language for the course and well-suited for AI development. Python's high-level, readable syntax makes it ideal for quickly implementing complex algorithms. It also offers extensive libraries and tools that can streamline our development process. Given its popularity in AI and machine learning, Python will enable us to efficiently build and test our Minimax algorithm with Alpha-Beta Pruning, while focusing more on the logic and performance of our AI rather than low-level implementation details.

3. Datasets

3.1. **Will there be datasets**: No

3.2. **Why:** No datasets are required since this is a game-playing algorithm and not a machine-learning model. This project implementing Minimax with Alpha-Beta Pruning does not need to use datasets to learn patterns from data to make decisions.

4. Existing code & will we use it?

While open-source implementations of Connect 4 with Minimax exist, we will design and code our own version. In cases where we reference existing implementations for inspiration, we will ensure our solution introduces improvements and make sure to reference them in our project, such as:

- Implementing Alpha-Beta Pruning for efficiency
- Designing a custom evaluation function to assign utility values based on board states
- Adding an intuitive user interface to allow easy interaction between human players and the AI agent

5. Algorithm Applied/Approach

Our approach involves using the Minimax algorithm with Alpha-Beta Pruning to efficiently explore the game tree. In this course, we have studied various adversarial search algorithms that simulate

two-player games with perfect information, which makes Minimax an appropriate choice. Minimax will help the AI anticipate its opponent's moves and select the optimal strategy. Alpha-Beta Pruning will enhance performance by reducing unnecessary exploration of the game tree.

6. Timeline

Week	Tasks
10/27 - 11/02	● Complete project proposal ● Research project algorithm, implementation details ● Design plan (determine requirements, identify technologies to be used, development tools, etc.,) ● Determine team structure & roles
11/03 - 11/09	● Plan out high-level architecture of software implementation, components ● Make necessary diagrams to describe user experience ● Design UI ● Outline of test plan
11/10 - 11/16	● Software implementation ● Build UI ● Debug/Test software ● Make a comprehensive list of test cases
11/17 - 11/23	● Software implementation ● UI refinement
11/24 - 11/30	● Debug, test, evaluate software
12/01 - 12/07	● Complete software implementation, evaluation ● Complete testing & debugging ● Deploy code ● Project presentation ● Code submission
12/08 - 12/14	● Submit written report

7. Roles & Responsibilities

- **Klarissa Navarro:** Lead developer for game rules and evaluation function design
- **Edward Cardenas:** Responsible for Minimax with Alpha-Beta Pruning implementation and algorithm optimization
- **Erika Dickson:** Testing and debugging the AI agent, ensuring performance through test cases, and developing the user interface.

8. Special Implementation Details

- PyGame
- Numpy