Assignment 2 Minimax Connect Four

Dan Blanchette

June 19, 2025

Abstract

This Connect Four game features an AI opponent that utilizes the Minimax algorithm with alpha-beta pruning. Players can choose to go first or second, with roles assigned accordingly(x's or o's for piece designations). The AI evaluates moves up to a fixed depth of 4, using a simple heuristic that favors center control and quicker paths to victory. Depth-based scoring ensures the AI prefers faster wins and delays losses when possible. Alpha-beta pruning optimizes performance by ignoring less optimal branches during the search process. The result is that the AI agent was able to beat the opponent even when playing defensively. Future optimizations to this program involve exploring different heuristics, such as favoring specific win types (vertical, horizontal, diagonal), blocking immediate threats, and experimenting with a variation where the agent does not always prioritize the center of the board.

1 Algorithm

1.1 minimax description

The minimax algorithm prioritizes the most positive score (max) for the AI agent and the most negative score (min) for the opponent. The algorithm uses a zero-sum game approach. This means that as long as each player is playing optimally, the resulting state will end in a draw. If the opponent player makes a mistake, a positive score deficit will arise between the AI agent and the player favoring the AI agent's win goal. If the AI agent makes a mistake, the score will result in a negative score deficit between the opponent and the AI agent, favoring the opponent's win state.

This is achieved using a Depth First Search algorithm to determine the most optimal search path for the AI agent to traverse. I also added the quickest route to a win as part of this method, if the AI is winning, and the longest delay to a loss as part of my approach. In other words, if the AI is winning, end the game as quickly as possible with the most optimal piece placement for scoring; otherwise, force a draw, if possible, by delaying the AI agent's loss.

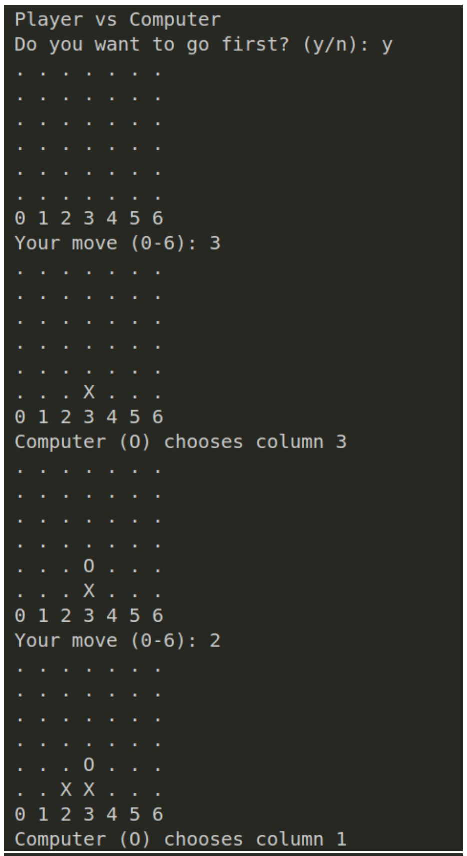
1.2 Alpha-Beta pruning

This method compares optimal move scores based on the traversed paths(branches) during the depth-first search process. If a branch scores less than other evaluated branches, the path is ignored and is no longer considered part of the minimax search goal for the AI agent. This method minimizes the amount of the tree that needs to be evaluated, thereby improving the search speed.

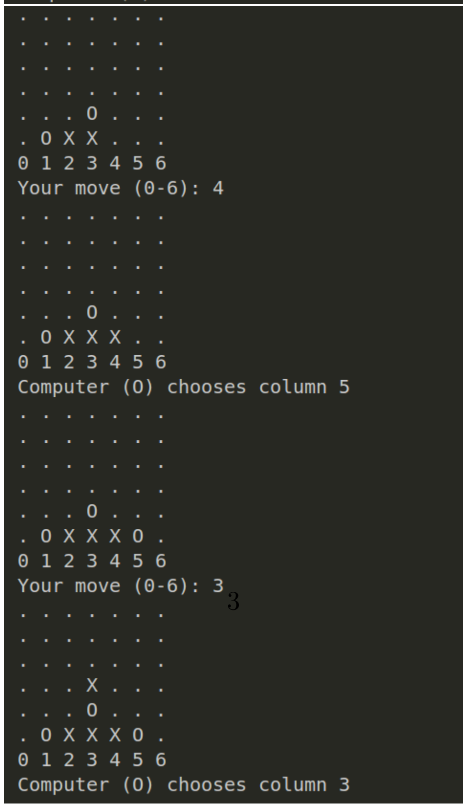
1.3 Evaluation Function

The evaluation function assesses the number of the AI's pieces in the center column. Each piece adds 3 points to the score. The intent is to encourage the AI to play toward the center, which gives more flexibility for forming horizontal, vertical, and diagonal connections.

2 Sample Gameplay

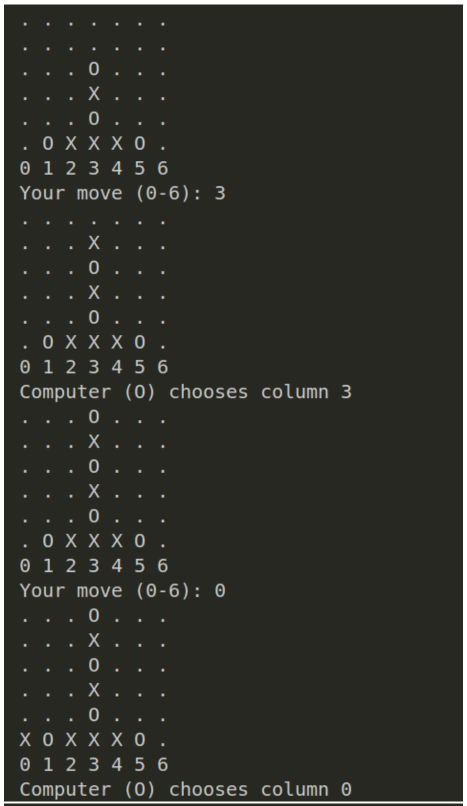


```
Player vs Computer
Do you want to go first? (y/n): y
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
0 1 2 3 4 5 6
Your move (0-6): 3
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . X . . .
0 1 2 3 4 5 6
Computer (O) chooses column 3
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . O . . .
. . . X . . .
0 1 2 3 4 5 6
Your move (0-6): 2
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . O . . .
. . X X . . .
0 1 2 3 4 5 6
Computer (O) chooses column 1
```

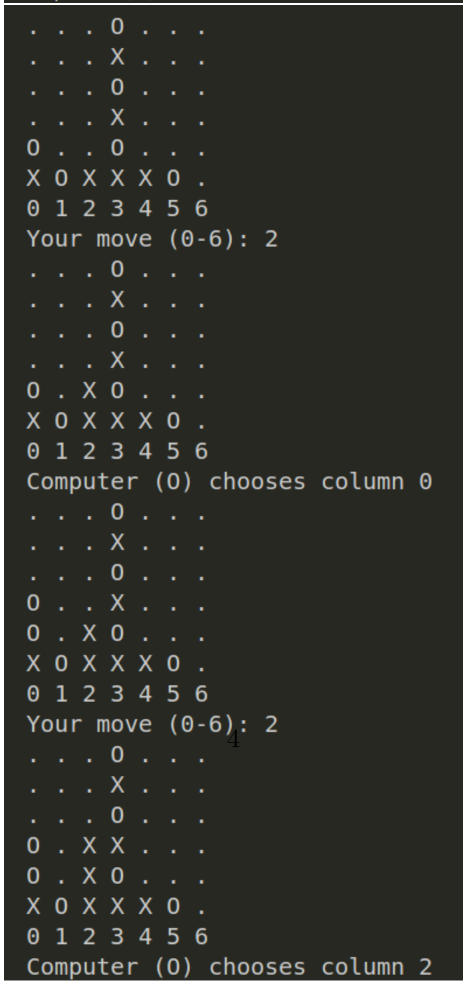


```
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . O . . .
. O X X . . .
0 1 2 3 4 5 6
Your move (0-6): 4
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . O . . .
. O X X X . .
0 1 2 3 4 5 6
Computer (O) chooses column 5
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . O . . .
. O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 3
. . . . . . .
. . . . . . .
. . . . . . .
. . . X . . .
. . . O . . .
. O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 3
```

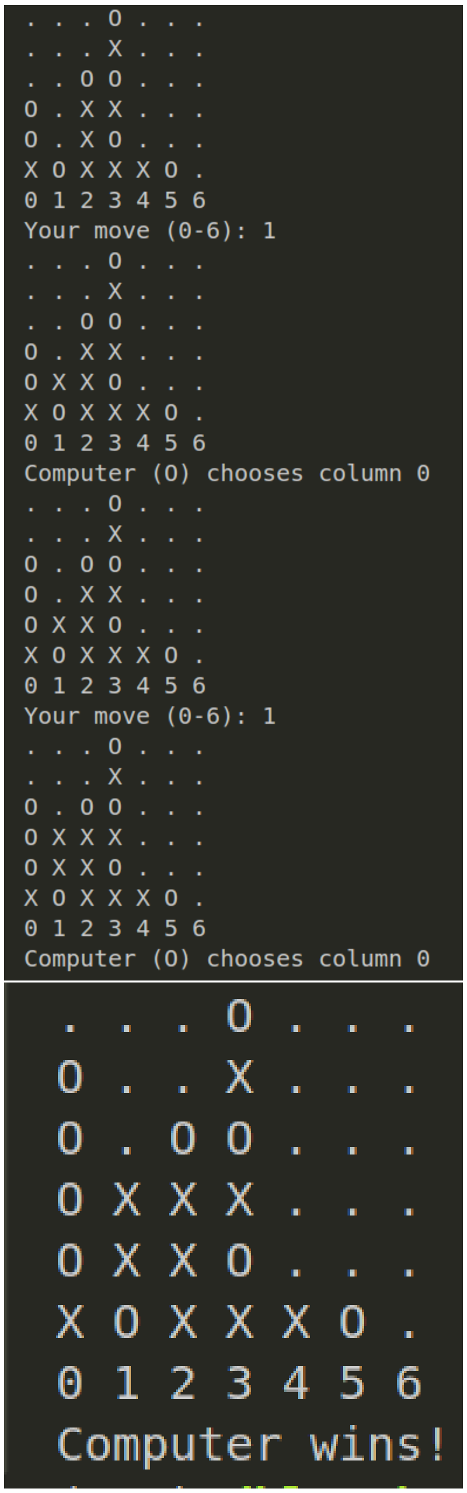
Figure 1: Sample Gameplay 1



```
. . . . . . .
. . . . . . .
. . . O . . .
. . . X . . .
. . . O . . .
. O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 3
. . . . . . .
. . . X . . .
. . . O . . .
. . . X . . .
. . . O . . .
. O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 3
. . . O . . .
. . . X . . .
. . . O . . .
. . . X . . .
. . . O . . .
. O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 0
. . . O . . .
. . . X . . .
. . . O . . .
. . . X . . .
. . . O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 0
```



```
. . . O . . .
. . . X . . .
. . . O . . .
. . . X . . .
O . . O . . .
X O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 2
. . . O . . .
. . . X . . .
. . . O . . .
. . . X . . .
O . X O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 0
. . . O . . .
. . . X . . .
. . . O . . .
O . . X . . .
O . X O . . .
X O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 2
. . . O . . .
. . . X . . .
. . . O . . .
O . X X . . .
O . X O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 2
```



```
. . . O . . .
. . . X . . .
. . O O . . .
O . X X . . .
O . X O . . .
X O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 1
. . . O . . .
. . . X . . .
. . O O . . .
O . X X . . .
O X X O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 0
. . . O . . .
. . . X . . .
O . O O . . .
O . X X . . .
O X X O . . .
X O X X X O .
0 1 2 3 4 5 6
Your move (0-6): 1
. . . O . . .
. . . X . . .
O . O O . . .
O X X X . . .
O X X O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer (O) chooses column 0
```

```
. . . O . . .
O . . X . . .
O . O O . . .
O X X X . . .
O X X O . . .
X O X X X O .
0 1 2 3 4 5 6
Computer wins!
```

Figure 3: Sample Gameplay 3

3 Conclusions/Discussion

3.1 Program Performance

The program performed well; however, it did seem to prioritize vertical wins more often. If that win condition was not possible, the program typically pursued horizontal wins. The least seen win was the diagonal win. Based on my mini-max conditions, which prioritize shortest paths to victory and minimize delayed losses, it appears that both goals are being adhered to. This comes at the cost, sometimes, of prolonging the game if the AI has made an error.

3.2 Gameplay

Overall, I was impressed with how tricky this AI agent is to beat. Most games I thought I would win ended in a draw. Mostly, games ended in losses for me, the opponent. The sample gameplay for this paper displays 21 moves until the AI won. I found that often I would end up pursuing my objective, but I would not consider that the AI was one move away from winning. Gameplay was a success in this sense as my mistakes maximized the chances for the AI to win.

3.3 Future Work

Future improvements for the program could involve exploring different heuristics, such as favoring specific win types (vertical, horizontal, diagonal), blocking immediate threats, and experimenting with a variation where the agent does not always prioritize the center of the board.

4 ChatGPT Evaluation For Assignment 2: GPT as a Coding Tutor

4.1 Application

I primarily used ChatGPT to assist with translating some of the pseudocode from the textbook. Learning that the Python math library had -inf and inf values to help clarify how to represent those values from the pseudocode helped make some sense of things.

I ended up using it more as a tutor this time around, instead of asking for code examples. Instead, I had it coach me as if I were pair programming with it through the pseudocode. GPT40 was also used for some debugging assistance when the mini-max scoring was causing the AI to perform unexpectedly.

4.2 Performance

This was an interesting approach I hadn't considered using ChatGPT for, and it wasn't too bad. Revisiting the lecture videos and notes helped solidify the tools and how they operate, and I felt like it was better as an assistant at this capacity than acting as a generator for a function or otherwise.

5 Sources

1. https://www.geeksforgeeks.org/dsa/finding-optimal-move-in-tic-tac-toe-using-minimax-algorithm-in-game-theory/
2. https://www.geeksforgeeks.org/dsa/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
3. ChatGPT for debugging and explanations