Building an Unbeatable Tic Tac Toe AI

After starting my journey of learning machine learning, I decided to embark on a project to test my newly acquired skills. The goal was to develop an AI capable of never losing a game of tic-tac-toe.

The Initial Approach

At first, I created a graphical interface using Tkinter. Since the interface itself wasn't my main focus, I based it on the tutorial available https://www.geeksforgeeks.org/tic-tac-toe-game-with-gui-using-tkinter-in-python/. After that, I encapsulated the logic into a class and implemented a simple random algorithm to play against itself.

Using this randomized approach, I generated data to analyze game behavior. I then attempted to build a prediction model using various machine learning algorithms, such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Neural Networks (NN), among others.

Challenges with ML Models

Initially, I approached the problem without any external research to challenge myself. However, after testing, evaluating, and analyzing the data through graphs and tables, I realized these types of models weren't effective for solving this specific problem.

Discovering Minimax

I then turned to research and found the Minimax algorithm, which optimizes decision-making by exploring a tree of possible outcomes. After watching a few tutorial videos, such as https://www.youtube.com/watch?v=5y2a0Zhgq0U&t=185s, I decided to implement it in my project.

My Unique Implementation

I made some optimizations to enhance the algorithm:

1. Scoring System:
    - Winning faster is prioritized.
    - If the computer wins, the score is calculated as:
      score = ±1 * (9 - round + 1)
      The ± depends on whether the AI is maximizing (computer starts) or minimizing (player starts).
2. Pruning Branches:
    - If an optimal result is found early (e.g., a branch yields the highest possible score), the algorithm skips further exploration of that branch.

This approach significantly reduced processing time and ensured the machine was unbeatable.

Results and Data Analysis

The results are stored in the jogadas.csv file. The columns in the CSV include data from my earlier failed attempts to build a machine learning model. However, this file proved useful for analyzing game patterns and verifying the accuracy of the Minimax implementation.

Visual Explanation

As illustrated in the image below:

- The Max player (AI) tries to maximize the score, while the Min player (human) tries to minimize it.
- The score reflects the best possible result at each step, with branches being pruned when optimal results are found.