Chess AI Project Proposal

Problem Statement:

There have been many chess engines made, and there are even some that can play better than any human. Some open-source examples of these very high-level engines include Stockfish and Leela Chess Zero [1], [2]. We would like to try to make our own chess engine to see how well we can replicate their success, although we definitely will not have the resources to make anything quite so refined. We plan to try a few different approaches, and take notes on which ones produce the best results. This way, we will have a chance to learn why certain algorithms can work better for some tasks. Additionally, if we have time, we might experiment with some arbitrary constraints to see which approaches can adapt more easily.

Approach:

We intend to use multiple approaches. We intend to build a chess AI that uses reinforcement learning to determine high-level strategy and uses search to determine which moves to make in order to complete the high-level strategy. The problem relates to both algorithms we chose, because they are used by two of the best chess bots that currently exist; Stockfish uses search and Leela Chess Zero uses reinforcement learning [1], [2]. The libraries we intend to use are NumPy, TensorFlow, and PyTorch. The NumPy library will be vital for quickly searching through large arrays containing potential board layouts [3]. TensorFlow is going to be very important for reinforcement learning, because it can be used to easily manage deep neural networks that are common with reinforcement learning [4]. We intend to use PyTorch to help with the search part of our AI, since the PyTorch library is able to make computational graphs that can be changed at runtime [5].

Breakdown of Work:

February 26th - Remington

1. Build a chess environment that allows us to make moves and simulates a chess game.
    a. Create a chess game that holds a chess board and position and has a function to make a move and a function to get the chess position.
    b. Create chess piece classes that each return all the legal moves for the respective pieces.
    c. Create a main program that draws the chess positions and allows a user to play a game of chess.

March 4th - Jacob

2. Create an agent that uses a greedy algorithm with a simple heuristic function that totals the piece values using the traditional rating system (pawn = 1, bishop = 3, knight = 3, rook = 5, queen = 9).

March 25th - Jacob, Remington

3. Explore alternative heuristic functions and search algorithms.
    a. Controlling the center of the board

b. How exposed the kings are
c. Number of legal moves
d. Number of pieces threatened
e. Pawn value increases as they move forward

March 25th - Eric, Remington

4. Explore using reinforcement learning to help evaluate board positions.
    a. More details will be added as we learn about reinforcement learning in class.

April 15th - All group members

5. Write the project report of all of our findings including how fast each of the methods make moves, and how good those moves are.

April 15th - All group members

6. Prepare and give a presentation on our findings.

Note that these are just general roles. We will still help each other when necessary, and we will also divide unanticipated work.

Team Structure:

This team consists of Eric Lykins, Jacob Seikel, and Remington Ward. All three team members know the rules of chess. The entire team has taken an algorithms course, which will be helpful for optimizing the AI models. Eric Lykins and Remington Ward have taken a course in which they learned the basics of machine learning, which will be useful for building the AI models.

References:

[1] Stockfish (2024) Stockfish [source code]
https://github.com/official-stockfish/Stockfish

[2] LeelaChessZero (2024) Lc0 [source code]
https://github.com/LeelaChessZero/lc0

[3] NumPy Team, “NumPy.” https://numpy.org

[4] TensorFlow, “TensorFlow.” https://www.tensorflow.org

[5] The Linux Foundation, “PyTorch.” https://pytorch.org