Chess Bot

Daniel McColm

Department of Computer Science
Western University

London Ontario, Canada

dmccolm@uwo.ca,

Navid Khan

Department of Computer Science
Western University

London Ontario, Canada

nkhan296@uwo.ca

Ryan Hayter

Department of Computer Science
Western University

London Ontario, Canada

rhayter2@uwo.ca

Abstract:

The project problem was to create a chess bot using neural networks. To create this, we researched different algorithms such as minmaxing and alpha-beta pruning to understand how it works. We also observed how previous iterations of this were processed and used our knowledge to come to the solution. The bot was created using Python. The results of the bot were successful; it's able to successfully play chess while staying within the rules. It did not quite reach our ELO goal however we would still say that it is a success. Takeaways from this project were that the results did match up to what we expected for the most part. The use of alpha-beta pruning cuts and minmaxing yielded the ability for

the computer to optimize the playable results. For future work, the idea would be to optimize the code so that it can run higher depth levels faster, and to improve the evaluation function to more accurately evaluate chess positions.

Introduction:

Chess is one of the most popular games in the world, with a history spanning hundreds of years and with players from everywhere in the world. Computers have been playing chess for much shorter than humans but have improved massively in the past couple decades, becoming much stronger than any human could possibly be. Research was done into how modern chess engines work, using decision trees to view every possible permutation up to a given depth and picking the move that has the best end result for the engine. We ended up deciding to use a minimax algorithm with alpha beta pruning to implement our engine.

Background & Related Work:

There are many chess engines in today's world, the premier one being stockfish, which analyzes each move at an average depth of 24. The top chess engines are very complicated and advanced, so we had no delusions about trying to measure up with such engines. However, chess engines do not need to be this advanced to be competent, or able to beat the average human player. It was in 1997 when chess engines truly reached this level, when the top chess engine at the time, IBM’s Deep Blue, beat reigning world champion Gary Kasparov in a publicized match. Since then computers have completely eclipsed humans in chess ability, with engines stronger than deep blue able to run on phones, airplanes, treadmills, and more. Today making a chess bot has never been easier, which is why we decided to make our own.

Methods:

The chess bot was created in python using a minimax algorithm with alpha beta pruning. Although we had planned to use neural networks we decided that that was not actually necessary for the implementation. The bot uses the python chess library to check legal moves and make moves on a board. The bot checks a decision tree of all possible moves from each position and picks the move with the best ultimate outcome for the engine, assuming that the opponent will then play the move with the worst outcome for the engine. An evaluation function is used to determine how good or bad a position is for the engine. The evaluation function is very straightforward, it subtracts the total value of the opponents pieces from the total value of the engines pieces. Of course the evaluation function also checks for checkmate, and there is a small element of randomness to keep things exciting.

Research Objectives:

The research objective for this project was to identify how minmaxing can be used to select strong chess moves, and therefore affect a computer's ability to stand its ground against either a player or another computer in a game of chess.

Research Methodology:

After doing research on both the minimax algorithm and alpha beta pruning we decided to use them for our chess bot. The rationale was that the minimax algorithm was particularly suited to chess where you would want to "maximize" on the engine's turn and "minimize" on the humans, as the engine will always assume the player will make the best move available to them. Alpha beta pruning was used because there is a lot of computations to be done when analyzing potential moves, pruning allows the computer to not compute

branches it knows is not the best option, saving time and computation power.

Results:

The chess bot works, and is able to play a game of chess, playing only legal moves and identifying checkmate. Its strength of play depends on the depth selected for the tree, however going deeper makes the bot significantly slower. At depth 4 the bot played a handful of games against chess bots on chess.com. Our bot lost to a 2000 rated bot however still played strongly, with an accuracy score of 78.5 and an estimated elo of 1200. The bot beat a 1000 and 1200 rated bot and lost to a 1200 and 1300 rated bot, all in close games. Based on this we would estimate the bots playing level to be around an ELO of 1000 at depth 4. Although this does not meet the ELO rating we sought we would still consider it a success overall, and with some optimization that goal is still easily in sight.

Conclusions & Future Work:

In conclusion, the minimax algorithm with alpha beta pruning is an adequate method for building a chess bot. In the future improvements could be made to improve the run time and the evaluation function, making the bot even stronger.

References

D. Yao, “25 years ago today: How deep blue vs. Kasparov changed Ai forever,” *AI Business*, 24-Oct-2022. [Online]. Available: https://aibusiness.com/ml/25-years-ago-today-how-deep-blue-vs-kasparov-changed-ai-forever. [Accessed: 10-Apr-2023].

“Minimax algorithm in Game theory: Set 1 (introduction),” *GeeksforGeeks*, 13-Jun-2022. [Online]. Available: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/. [Accessed: 10-Apr-2023].

“Artificial Intelligence: Alpha-beta pruning - javatpoint,” *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/ai-alpha-beta-pruning. [Accessed: 10-Apr-2023].

“Minimax algorithm and alpha-beta pruning | Mathspp.” [Online]. Available: https://mathspp.com/blog/minimax-algorithm-and-alpha-beta-pruning. [Accessed: 11-Apr-2023].