Machine Learning Project Proposal

Joseph Green
jcg0064@auburn.edu

Tripp Isbell
cai0004@auburn.edu

James Lee
jyl0003@auburn.edu

For our project we propose to build a chess engine.

Introduction

The game of chess has been around for many centuries, and with the emergence of computers in the 20th century came the idea of designing a machine to algorithmically play chess. Chess is a two-person, zero-sum minimax game [1] with perfect information, so we should (not we as in the project group, we as in 'humanity') be able to design a machine that plays optimally. Our ability to do so, however, is limited by the complexity of the rules and the combinatorial explosion of the possible board states. Claude Shannon was one of the first to propose an algorithm that minimaxes using a restricted set of actions evaluated by the current board positions [2].

Since then, many advancements have been made in the pursuit of an optimal chess engine. In 1997, IBM's Deep Blue famously defeated then No. 1 chess player Garry Kasparov. The current state of the art chess engine is StockFish, which frequently wins the Top Chess Engine Championship. Both Deep Blue and StockFish are conventional chess engines, in that they employ alpha-beta pruning to narrow the search space of possible moves [3].

Deep reinforcement learning has presented a new approach to building an optimal chess engine. Most notably, DeepMind has developed AlphaZero [4], a chess (as well as shogi and Go) which uses reinforcement learning and has achieved state of the art performance in chess, defeating top engines like StockFish. AlphaZero uses their self-play learning approach, which uses Monte Carlo tree searches optimized with deep neural networks, an approach generalized from their AlphaGo Zero[5].

We are neither good at chess, nor good at reinforcement learning (and also we don't have near the resources of DeepMind), so we don't expect to design an engine to achieve near the performance of these others. However, we'd like to develop a modest engine using reinforcement learning that performs somewhat well and can be trained in a reasonable amount of time. Since StockFish does provide varying levels of handicaps (achieved by limiting the amount of lookahead moves in their search), we can use it as both a baseline (StockFish level 1) and an oracle (StockFish unleashed). We hope to build something maybe competitive with StockFish 3 or so.

Input/Output and Data

In a high-level sense, in this project we want to be able to give the state of a chess board as input to the AI and have it output the best action it could take from all its possible moves, or at least avoid the worst ones. The board state will update in response to that action, the opponent will take their action and the AI will use the new board state as its next input, repeating each turn over the course of a game. We'll be using the gym-chess environment made for OpenAI Gym by software developer and GitHub user `genyrosk` to train our AI-player. The AI will continuously play games against bots in this environment to train it and since we'll be using a deep reinforcement learning approach, these games and their results will serve as our dataset.

Since this behavior is nice and general, we can modify initial board states and play other variants like Fischer random chess or possibly some variant of Minichess if it turns out too expensive to run on a full-size board.

AlphaZero

The algorithm used in the AlphaZero algorithm utilizes deep neural networks and a tabula rasa reinforcement learning algorithm. Furthermore, instead of using alpha-beta search with domain-specific enhancements, AlphaZero uses a more general search algorithm—the Monte Carlo tree search algorithm. Also, in terms of self-play, the games are generated by using the latest parameters for this neural network which skips the selection of a best player which is used by AlphaGo Zero a predecessor of AlphaZero. Lastly, the generalized AlphaZero algorithm has managed to outperform many of its competitors such as Stockfish in chess after just 4 hours and Elmo in shogi after only 2 hours. Thus, we believe that we can utilize some of the ideas used in the AlphaZero algorithm to our chess engine. However, given our limited resources, we will have to adapt/decide on what is feasible and what is not in this paper and algorithm.

References

[1] J. v. Neumann, "Zur theorie der gesellschaftsspiele (the theory of parlor games)," *Mathematische Annalen*, vol. 100, no. 1, pp. 295–320, 1928. [Online]. Available: https://doi.org/10.1007/BF01448847

[2] C. E. Shannon, "Programming a computer playing chess," *Philosophical Magazine*, vol. Ser.7, 41, no. 312, 1959.

[3] T. Hart and D. Edwards, "The alpha-beta heuristic," M.I.T., USA, Tech. Rep., 1961.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017. [Online]. Available: https://doi.org/10.1038/nature24270