CS 182 Final Project Proposal

Jacob Lurye, Walter Martin, Ryan Wallace

October 28, 2016

1 Topic

We intend to construct a program that plays losing chess (also known as "suicide" chess), a variant of standard chess in which the goal is to lose all of your pieces. This is a minor variation on the classic game that is interesting for artificial intelligence research for multiple reasons. First is that the game is relatively under-researched compared to standard chess. To our knowledge, there are fewer than five losing chess engines in existence. Second, the branching factor of the game in certain game states is significantly less than in standard chess because pieces capable of making a capture must do so. This results in a potentially more tractable problem than standard chess, since AIs might be able to search the state space a greater number of plies into the future.

We hope to compare the performance of various adversarial search algorithms as well as state-evaluation functions in this context. We intend to progressively "build out" our system by first implementing a general framework for losing chess, (including a state representation and function for executing a given move), then layering on top various search algorithms (depth-limited minimax, depth-limited expectimax, alpha-beta pruning), and evaluation functions (linear combinations of important features, deep-learning).

There are multiple ways we may compare the performance of our algorithms. To compare the variations on search, distinct nodes expanded per time step is an important measure. To compare evaluation functions, pitting one evaluation function against another with both agents using the same search method and observing the win/loss rate could provide a ranking of the evaluation methods.

2 Course Topics Investigated

This project provides an excellent review of numerous course topics:

- Adversarial search: depth-limited search, minimax, expectimax, alpha-beta pruning.
- Evaluation functions: game state evaluation.
- Reinforcement learning.

Exploring deep reinforcement learning as an evaluation method would also gain us exposure to a natural extension of the topics of reinforcement learning covered in class. Additionally, if time permits, we could explore the performance of iterative deepening with node ordering, a topic only briefly touched upon in lecture.

3 Expected Behavior

We hope to construct some interface, graphical or otherwise, that would allow a human to play a game of losing chess against the program. We aim for the program to play reasonably well against an average opponent. Specifically, we intend the program to consistently follow the standard rules of losing chess, and to avoid obvious tactical blunders.

We hope that the search system achieves performance capable of exploring multiple plies from every state within the standard time constraints on moves.

4 Expected Issues

Representing the game programmatically (ideally in a way that allows humans to play against the computer) will be a non-trivial task. Due to the high number of features in the game state and the large size of the state space, we will need to focus on the data structures used to represent the game in order to avoid unnecessary slow-downs or space consumption when searching through the state space.

Additionally, the dataset of losing chess games available contains games on only full-sized boards, and deep learning is no cakewalk on data of this size. Given that the other published deep learning application to chess had significant time and space requirements (multiple days running on GPUs) to train the neural network, we anticipate needing to either compromise on training time or reduce the dataset size to reach results within the time and resource constraints of the project.

Additional issues may center around processing and vectorizing the data from the online game repository for use in the deep learning algorithm. Interfacing between the state representation needed for game state search and that needed for the neural network in a clean and efficient way may also prove difficult.

5 Division of Work

All team members are interested in contributing to all aspects of the project. Naturally, all three hope to contribute to the deep learning evaluation component, given that it is a new exciting topic. Given that the framework needs to be developed before any search or evaluation functions are added, all three also intend to work together to perfect the basic framework, since no other work can happen before this is completed. Tentatively, Ryan then intends to work on the search functionality, while Jacob explores the standard evaluation functions, and Walter works on setting up the deep learning framework. This is a tentative plan that is subject to change as specific demands arise.

6 References

1. https://arxiv.org/pdf/1509.01549v2.pdf
2. https://erikbern.com/2014/11/29/deep-learning-for-chess/
3. http://www.ficsgames.org/download.html
4. http://magma.maths.usyd.edu.au/ ~ watkins/LOSING_CHESS/losing.pdf

Description of references:

1. The first deep RL based chess engine. Note that this is a 2015 article.
2. This engine uses the same methods, but seems to be less refined. Nevertheless, there is some valuable math background provided.
3. Database of around 300 million chess games move-by-move. Approximately 1 million of these are losing chess games.
4. Losing chess has been weakly solved for white. From 1. e4 there are no winning black responses. Nevertheless, this is only the tip of the iceberg of a complete solution.