12200 Project Proposal for Ginormous Anteaters

Members: Catalina Raggi, Ishaan Bhojwani, Kushal Haran, Priya Lingutla

Introduction

One of the most well known tools in the chess world is the Free Internet Chess Server (FICS), a program which matches chess players from around the world to play against each other in a variety of variations of chess. The FICS database (ficsgames.org), a record of chess games played through FICS since 1999, contains a wealth of information, including the moves of millions of individual games and the ratings of the players in the match.

Goals

We want to leverage this data to help analyze chess positions at a given player rating. We will be pulling all human vs human standard time control (so in general, games without time pressure hampering decision making) games from FICS. Our project has 4 stages -- depending on instructor feedback for the scope of this project, any number of these stages (after the first) may be rearranged or omitted. The stages, elaborated upon below, are as follows:

1. Use historical Chess.com user data to recommend the *n* best possible moves in a given position
2. Fill gaps in data with open source engine evaluation
3. Analyze well known positions to reexamine current accepted interpretation
4. *(stretch goal)* Implement GUI and time control

Details

Stage 1: Recommend n *best moves (2-3 people, 2 weeks)*

When given a particular position, the program will search through the historical data and catalog all moves users had played. It will then look at how often each move led to a win (percentage *w*), and list the *n* number of best moves, alongside each move's *w* and frequency of play (number of games the move appears in (*p'*) over total games in this position (*p*)). More common moves will be given some sort of preference, most likely in the form of a *p'* cutoff *c*. If a potential move does not have a *p'* value greater than *c* (such that $p' > c$), then there is not enough data to ascertain the quality of the move. If a position appears in which there are not enough potential moves which satisfy this condition, then the program returns None (that is, if less than *i* moves exist with $p' > c$). One potential tool could be to filter the data by current player's rating, so recommendations could be skewed towards those which are effective at that particular level. Tries and trees, as used in PA1, will be useful for this stage.

Stage 2: Fill gaps with engine evaluation (2 people, 1 week, requires stage 1)

In the previous stage, when the program reaches a position for which there is not enough data (defined as any position with less than *i* number of potential moves with $p' > c$), the program would simply return None. There are ways to build functionality past this without resorting to building a fully fledged chess engine (which would defeat the purpose of using FICS

data). One potential way is to integrate an existing engine which would take over position evaluations only in positions without user data. This becomes especially useful in the endgame, when each game has a high likelihood of evolving into a unique position. It also allow us to easily judge the quality of our project's output, comparing the numerical evaluation of the engine versus the evaluation of our programs recommended move. This would simplify optimization and improvement of the project. A good candidate engine is Sunfish, a simple, open source Python engine written in only 111 lines of code. Sunfish provides us a simple system, built in Python, to continue providing move recommendations to players and to help our team's project optimization and improvement.

Stage 3: Analyze well known positions (2 people, 1 week, requires stage 1)

The potential analytical power of such a system could be applied to opening positions to reveal practical nuances. While most openings are fairly well studied, these lines of play are only useful to those who have enough experience to memorize these moves. An ELO 1200 rated player (a practicing beginner by most standards) may be able to recite the most common variations of a common opening such as the Ruy Lopez, but would have to rely on their own (error prone) analysis for most other positions. So, while in many openings, optimal lines are well known, at lower levels these are seldom followed. At any sub-grandmaster level, deviations from 'optimal play' are bound to occur, and analysis of historical data can reveal the most effective lines with these deviations in mind, as opposed to the most effective lines within the context of perfect play. Organizing our data by player rating would allow us to examine how opening lines develop differently according to skill.

Stage 4: Implement GUI and time control (1-2 people, 1-2 weeks)

Finally, as a stretch goal, implementing a simple GUI would make it convenient for a user to interact with the program and the chess position. This would require a board representation, and if it was to be complete, the program would have to have knowledge of piece movement rules. There are python chess libraries which may help, such as python-chess.

Conclusion

Chess is one of the most popular board games in the world. Given the resources already available, what would make this different from a website such as chesstempo, or chessbase? The main difference is that these websites focus on high level play, only using data from players rated ~2200 and above. While optimal play is something to strive towards for any player, at lower levels more intuitive and simpler strategies can lead to victory while being less blunder prone. Through analysing past games, we seek to create a move recommendation tool for all levels of play which can find the move which is most effective at that players particular level.