Adaptive MinMax - Lodi Alessandro

Matricola: 274425

Library used

I used various libraries for this homework

- python-chess as a chess library that comes with move generation, move validation, and support for common formats (eg. FEN).
- chess-board that allows the visualization of the board using the fen descriptor
- keras to implement the model used for the advanced implementation
- pandas to manage the dataframes

Basic implementation

I made an implementation of the min-max algorithm with alpha-beta pruning and I used four heuristics in order to evaluate the chessboard:

- **Material heuristic**: This is the most basic heuristic for chess. Basically, we assign a value to each piece type and then we calculate the sum of all values (negative values for the opponent pieces)

- **Piece Square Tables**: This is a simple way to assign values to specific pieces on specific squares. We have a table for each piece type where each cell represent a square on the game board and we assign a value to each of these cells

- **Pawn Structure**: Pawns are considered relatively static pieces and their structure largely determines the strategic nature of the position. So this heuristic evaluates the positions of all the pawns on the board based on some chess knowledge such as doubled pawn, isolated pawn, etc...

- **Mobility Bonus**: This bonus is computed based on the mobility of the knight, bishop, rook, and queen. For each piece and for each number of possible moves for that piece is defined a bonus (negative or positive) added to score

The evaluation of the board is made weighting these four heuristics together:

$$H = material(board) + 0.3 * pst(board) + 0.3 * pawn_structure(board) + 0.5 * mobility(board)$$

and the results are quite good because it can make a good game with a player with a chess rating of about 1000.

In the Min-Max, these heuristics are always computed from the point of view of the white player and if the heuristic player is black it will minimize the score of the board instead of maximizing.

Advanced implementation

First part

In the first part, the model is used to predict the heuristic score at depth L for each of the legal moves.

To implement the model I used a Keras Regressor that takes as input a dataset formatted as follow:

- **h1, h2, h3, h4**: the weighted heuristics explained before, computed after the move
- **H** : the score computed by the Min-Max at level L

Then the regressor was trained while playing with the basic implementation and the Min-Max algorithm is used to make the dataset. In details for each player turn are computed the heuristic with the MinMax and added to the dataset and after two games (white player swap) the model is trained with the dataset

In order to improve the model learning, the dataset is normalized

Second part

In the second part, the model is used in the MinMax algorithm and the board evaluation at level L is substituted with the model prediction. With the model prediction the MinMax algorithm is (theoretically) capable to predict the board score at level $2L$

Results

During the training were tracked the game results and as expected the player that used the model never won. On 184 games the model lost 159 games and made 25 draws.

Unfortunately, the model did not train well because the loss was very high. There are two possible reasons for this problem:

- The dataset is not enough big and there is needed more training data
- The train is not feasible due to the high variability of H

Because of this, even the implementation of MinMax that uses the model can not win against the basic implementation.