BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Trying to master the game of chess with help of neural networks

– MIRPR report –

Team members
Tudor Petru Cheregi, tudor.cheregi@stud.ubbcluj.ro
Victor Mihai Macinic, victor.macinic@stud.ubbcluj.ro

2021-2022

Abstract

Throughout chess engines history, there have been proposed lots of different strategies in order to come up with the best algorithm possible. In comparison with very complex and general models that are capable of adapting to various environments (games), our approach is to create a neural network that is not capable of playing chess by itself, but rather is very good at evaluating random chess positions. Since there exist known metrics for evaluating a given position/configuration of pieces we can create our model on a fully supervised manner. The data needed to train the model is comprised of random chess positions from different games and different ELO ranges which can be found in the Big Chess DataBase, and their evaluation was obtained with the help of Stockfish (one of the best chess engines available for free).

Contents

Chapter 1

Introduction

Chess is an abstract strategy game and involves no hidden information. It is played on a square chessboard with 64 squares arranged in an eight-by-eight grid. At the start, each player (one controlling the white pieces, the other controlling the black pieces) controls sixteen pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. The object of the game is to checkmate the opponent's king, whereby the king is under immediate attack (in "check") and there is no way for it to escape. There are also several ways a game can end in a draw.

One of the goals of early computer scientists was to create a chess-playing machine. In 1997, Deep Blue became the first computer to beat the reigning World Champion in a match when it defeated Garry Kasparov. Though not flawless, today's chess engines are significantly stronger than even the best human players, and have deeply influenced the development of chess theory.

In comparison with Deep Blue's strategy which was comprised of several heuristic methods and a minimax tree search based algorithm, our approach is a little bit closer to what modern engines (Stockfish NNUE, Alpha-Zero) make us of: neural networks.

Chapter 2

Scientific Problem

As we have mentioned before the game of chess is quite complex both in its rules and in its strategy and it is a pretty difficult task to take on with a conventional algorithm that simply uses a set of defined rules according to the current position. The reason of that is primarily due to our limited knowledge of how chess should be played. In essence the engine's level would be gaped to the potential of that set of rules. Thus, if we want to overcome this threshold and truly to achieve a breakthrough, deep learning would be our best chance. Give a description of the problem.

Chapter 3

State of the art

In 2017, a revolutionary strategy was introduced by people from DeepMind who manage to create an algorithm that is able to generally play and adapt to any sort of game, given the necessary set of rules. Their approach was completely based on a deep reinforcement learning model.

Differences between DeepMind's approach and ours:

- Instead of using a reinforcement learning model, ours is based on a fully supervised method.
- Our model is not able to cope with any different environment besides the game of chess since it was specifically trained for this task.

Chapter 4

Approach

4.1 Regression vs Classification

Regression: We first tried to mimic Stockfish's behaviour in terms of evaluating a position, therefore our models are supposed to predict a real value representing the exact evaluation of the current position.
Classification: Due to the complexity of the regression problem we tried to simplify the task and express the regression problem as a multi-label classification one. Thus, the evaluation of the position (the label) was converted into 3 classes based on which color is supposed to have a better position (0 = black is better, 1 = equal position, 2 = white is better).

4.2 Preprocessing

Our experiments have been conducted using three main tensor input representations, the first and the last one being inspired by **Matthia Sabatelli et al, 2018** [1].
Algebric representation - An 8x8 matrix, each square having its piece value (scaled to [-1,1]) or 0 if there is no piece on the square.
Sparse representation - An 6x8x8 tensor representing an 8x8 matrix for every piece (Rook,Knight,Bishop, Queen,King,Pawn), each square having a value of -1 (if there is a black piece on the square), 0 (if there is no piece on the square) or 1(if there is a white piece on the square).
Bitmap representation - An 12x8x8 tensor representing an 8x8 matrix for every piece (Rook,Knight,Bishop, Queen,King,Pawn) of each color (Black or White), each square having a value of 0 (if there is no piece on the square) or 1(if there is a piece of the corresponding color on the square).

4.3 Models

As we have mentioned before, our goal is to create a decent evaluator for a given chess position rather than a model that is actually able to make a move.
We selected a large dataset of chess games from numerous grandmasters. Each game is comprised of a particular sequence of moves(positions) that we labeled using one of the strongest chess engines at the moment (Stockfish). Every position is afterwards turned into a FEN string representation and saved to our train data file.
Our experiments have been conducted using 2 main architectures based on exactly what has been proposed by **Matthia Sabatelli et al, 2018** [1]:
Model 1 A network with fully connected layers that has an input layer of dimension width x height x channels for each tensor representation, 6 hidden layers(having 64,64,32,32,16,16 neurons), and one output layer with either 1 neuron for the regression task or 3 neurons for the classification one.
Model 2 An FCN (fully convolutional network) to which have been added some fully connected layers in order to provide additional computation in hope of achieving better performance. The architecture is similar to the VGG architecture and it's output layer is the same as the one in the first model.

Algorithm 1 Training

```
INPUT
B - data batches
X - input data
Y - labels for input data
f - the network with parameters $\theta$
BEGIN
for i=1 TO numberOfEpochs do
  for j = 1 TO numberOfBatches do
    b $\leftarrow$ B[j];
    XBatch $\leftarrow$ all X from batch b;
    YBatch $\leftarrow$ all Y from batch b;
    predictions $\leftarrow$ f(XBatch, $\theta$);
    loss $\leftarrow$ MSE(predictions, YBatch);
    update $\theta$ with the obtained loss using an optimer, e.g. Adam
  end for
end for
return $\theta$
END
```

Algorithm 2 Choose best move

```
INPUT
network: The neural network responsible for evaluating the position
currentPosition: the current position represented as a FEN string
OUTPUT
bestFen: The FEN representation of the next best move
BEGIN
nextMoves ← getAllNextMoves(currentPosition);
bestFen ← NULL;
bestEval ← NULL;
for i=1 TO nextMoves.size() do
  if bestFen = NULL then
    bestFen ← nextMoves[i];
    bestEval ← network.predict(bestFen);
  else if network.predict(nextMoves[i]) > bestEval then
    bestFen ← nextMoves[i];
    bestEval ← network.predict(bestFen);
  end if
end for
return bestFen;
END
```

Chapter 5

Results

References

[1] Matthia Sabatelli, Francesco Bidoia, Valeriu Codreanu and Marco Wiering
Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead

[2] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis
Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

[3] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, David Silver
Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model