Win-Loss Prediction Based on Opening A Player Chooses for a Game of Chess

Akshay Jain, Nidhi Tholar, Sashank Pidur Kuppuswamy, San José State University

October 2021

Abstract

Introduction

A Game of Chess is played between two players on an 8x8 Square board,16 pieces on each side, and with millions of board positions and possibilities. It was invented fifteen centuries back in India and has been spread all over the world it lots of changes in the game's rules over time until they were finalized around the 1880s which is also the romantic era of chess. There are millions of games played and recorded in various places like books, newspapers, and online databases over time and the players use them to improve and learn the game of chess. In this project, we are trying to create an algorithm that helps chess players to learn about the different types of chess openings and statistics associated with it like winning percentage over the years, use of it at grandmaster (expert at the game) level, etc. A game of chess is stored in a form of a PNG (portable game notation) file. To store and process a chess game we first need to understand the notations used in a PNG file, a sample PNG looks like –

[Event "New York"]
[Site "New York"]
[Date "1918.??.??"]
[Round "?"]
[White "Capablanca, Jose Raul"]
[Black "Marshall, Frank James"]
[Result "1-0"]
[WhiteElo "2589"]
[BlackElo "2632"]
[ECO "C89"]

1.e4 e5 2.Nf3 Nc6 3.Bb5 a6 4.Ba4 Nf6 5.O-O Be7 6.Re1 b5 7.Bb3 O-O 8.c3 d5 9.exd5 Nxd5 10.Nxe5 Nxe5 11.Rxe5 Nf6 12.Re1 Bd6 13.h3 Ng4 14.Qf3 Qh4 15.d4 Nxf2 16.Re2 Bg4 17.hxg4 Bh2+ 18.Kf1 Bg3 19.Rxf2 Qh1+ 20.Ke2 Bxf2 21.Bd2 Bh4 22.Qh3 Rae8+ 23.Kd3 Qf1+ 24.Kc2 Bf2 25.Qf3 Qg1 26.Bd5 c5 27.dxc5 Bxc5 28.b4 Bd6 29.a4 a5 30.axb5 axb4 31.Ra6 bxc3 32.Nxc3 Bb4 33.b6 Bxc3 34.Bxc3 h6 35.b7 Re3 36.Bxf7+ 1-0

Important notations and jargons to learn – A chessboard consists of 8files(columns) which are denoted from 'a' to 'h' and 8 ranks(rows) which are denoted from 1-8. The moves are denoted starting with white pieces and followed by black pieces. Pieces and notations – there are 6 different pieces in a game of chess 1)A pawn – there are 8 pawns with each side, A pawn moves only in one direction and captures diagonally.it can also move one or two steps in the first move and after that, it can only one step. It is denoted with the file name on which it is present for example: 1.e4 denotes on the first move white as pushed the pawn on 'e' file from his current position to e4. 2)A Bishop – there are 2 bishops with each side namely light squared(white) bishop and dark-squared(black) bishop they move diagonally in any direction and any number of steps. It is denoted with a capital letter 'B' for example: 3.Bb5 denotes white has moved the bishop to b5. 3) A Knight – there are 2 knights witch each side and they move in an 'L' in any direction and a knight is the only piece that can move over other pieces. It is denoted with the capital letter 'N' for example: 4.Ba4 Nf6 denotes black has moved to the knight to f6. 4)A Rook – there are 2 rooks with each side and they move in a straight line both horizontally and vertically. It is denoted with the capital letter 'R' For example: 12.Re1 denotes white has moved to the rook to e1. 5)A Queen- there is one queen per side and is the most powerful piece on the board which can move in all directions (vertically, horizontally, and diagonally) any number of steps.it is denoted with the capital letter 'Q' for example: 14.Qf3 Qh4 white has moved the queen to f3. 6)The King-there is one king per side and it is the most important piece on the board one it is captured the game ends and it can be moved in all direction but only one step. It is denoted with the capital letter 'K' for example: 24.Kc2 denoted white has moved to the king to c2. Elo is rating associated with the player, higher rating indicates a better player(mostly).

Other important notations – castling – it is denoted by 'O-O' or 'O-O-O' depending on the side the king moves.

Checks- it is denoted by a plus sign 'Rae8+', it denoted there is a direct attack to the opposite side king.

Captures- it is denoted by a small letter 'x' eg: Nxe5 knight has captured the piece on e5.

Data

We utilized a data-source[1] that provided thousands of games with respect to an Opening. We picked 15 opening/variation for our analysis. The data files were in text format (.txt). We transformed the data into a dataframe with the help of python string manipulation and regex. We also manually created a python dictionary that contained a mapping of an Opening to the move pattern.

Openings we chose: Four Knights, Caro Kann Classic, Qid4e3 (Queen's Indian,4e3, Sicilian Rossolimo, Sicilian dragon other6, KingsGambit, RuyLopezMarshall, ScotchGambit, GiuocoPiano, Reti2b3, Caro-Kann2c4, Caro-KannAdv, Nimzowitsch-Larsen, Caro-KannEx, Modern.

Missing values: The missing values were high for Elo rating of Players. Since it's an important factor in the analysis, replacing these missing values with mean or some other value, might have harmed the analysis. Hence, we decided to drop the rows containing missing values.

Elo Rating: We considered only those games where Elo Rating of both the players were above 2000

Openings: Though the data files were for a respective opening, there were some errors. We dropped the rows where the opening didn't belong to the 15 openings we chose.

Methods

The Dataset created has multiple features ranging from the game's player, the date the game was played to the moves played for that game's particular round. The final outcome of the game is noted as white black or draw in the result column. Since the problem's target variable is categorical and the data is available in columns, the possibilities of applying machine learning techniques are considered. The simplest classification technique is considered initially to determine the outcome.

Decision Tree

Since we wanted to analyze the amount of influence an opening had for a game's result, we decided to use Decision Tree.

Decision tree is a supervised learning method used for classification. It is used to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The library considered for the implementation is from sklearn[2].

Decision Tree Trained on Opening and Player's rating

Here we considered features of opening, player's Elo rating and the result of the game to train the model. We utilized 80% of the data to train and the other 20% to test. Since Decision Tree doesn't work with categorical data, we encoded the opening name with numbers. We built the decision trees varying in depths to determine the outcome of the data considered. As the game of chess is a complex game and a game's result is affected by lot of factors other than opening. The classifier model's accuracy supported this inference. The accuracy we obtained with depth 12 to 40 was around 50%. The accuracy of the model showed that it is difficult to predict the game result based on opening and player's rating.

Decision Tree Trained on First X moves, Opening and Player's rating
Since an opening would cover an average of 5 to 10 moves, we decided to extend the number of moves we considered to explore the effect of moves on the games result. In this approach, we selected first N moves from each record of games where first N moves represented a both the player's moves. We used a python dictionary, where the combination of moves represented in a key, and the value marked against this was a list of games that had same first N moves. We built the decision tree for first 10 and 20 moves, but the accuracy of the model was not observed to improve.

Approach 2
From the obtained dataset, the major data to determine the game's outcome is the game's moves that are played. The player irrespective of their quality, tend to repeat a set of moves, and by analyzing their games played with the games of other player's we could determine the similar moves that are played predominantly and can be used to determine a set of better possible moves towards the middle game. To achieve this, we are implementing the technique of finding the similarity between all the possible games considered, and the resulting similarities are stored in an array of played game indexes. With a given threshold, the games that spanned out similar moves would be grouped, and the remaining unique moves would be considered as possible moves. This is implemented using the python inbuilt library of difflib and with the help of the Sequence matcher module. The sequence matcher module compares two sets of strings and returns the similarity object from which the percentage of similarity or the ratio can be determined for any two given games. This technique would have been an ideal solution to provide numerical values to the game moves if the intensity on the hardware is not excess. While running this technique for a simple set of 30 thousand games, resulted in exhausted memory. Further scope for improvement on this technique is currently being observed and explored.

Approach 3
Since the problem considered involves chess moves as data and the major output variable considered is win-lose, We are formulating the problem as determining the output category as 'win', lose' or 'draw' based on the previous games played. However the game data is of varying length, and being coded texts, choosing a model that performs better with this data had been a major concern. Most of the machine learning models work with numerical values for better prediction results. Hence If there could be a way to incorporate numerical values into the given text, then the data could be generalized for other data models. The initial suggestion would be to consider the bag of words method or the word to vector techniques. However, these techniques will consider the occurrences of the word in that document as a unique occurrence with unique meaning and would assign a numerical value to the word. The Chess moves data could not be considered as unique words with unique meanings as the moves preceding the current move will determine the flow of the game and this would change dynamically as the game moves further into the finale. So to create a unique meaning to the moves that are made, we are considering an approach inspired by both words to vector method and the alpha zero's algorithm. This technique assigns unique positional code to each and every piece that is part of the game and provides a 32-character wide code for the current alignment of pieces on the board. The advantage of this considered technique is the previously played games would look only at the possibility of the move, completely ignoring the previous moves that led to that particular board formation. Further, the best possible move from the current alignment of pieces can be figured irrespective of the opening of the game. From the data perspective, the frequent pairs could be easily achieved by traversing through the game

moves across different games and could be stored as a key-value pair for determining the next possible better move. The difference in the length of data has also been handled. All the improvements and hurdles are tackled at expense of extra information for every move. The data considered for every game increases drastically and the only improvement that is being considered currently is to update the historical game library from the given dataset simultaneously as the code for every arrangement is generated. Further improvements on this technique are being explored.

Comparisons

Example Analysis

Conclusions

References

[1] https://www.pgnmentor.com/files.html#modking
[2] https://scikit-learn.org/stable/modules/tree.html