Predicting the outcome of a Chess game given a move number and board position

Kieran Human, Nayoun Ham, Namki Kwon

June 27, 2023

Abstract

We blended two papers to replicate while using real-world data. The first paper was the main paper, about time management in chess, was the main paper that we replicated, but we also drew on ideas from a second paper that was about predicting the outcome of a chess game between specific chess professionals. Using our own processed data, we found some interesting results that run counter to what traditional chess logic dictates. We also found that the most accurate model was the LSTM Neural Network.

1 Introduction

Chess is a popular and one of the oldest games that have fascinated people for centuries. Although computers can beat even the best of humans, accurately predicting game outcomes remains challenging. One reason for this is because a chess engine plays very differently to a human. Humans value pieces differently and do not always make the most optimal move. Even if the computer could win a position 100 times out of 100, it does not mean that a human will always win it. There are often chess positions that the engine believes is winning for one colour, but most humans would have a higher chance of winning with the other colour. One potential solution is to use the ELO rating system, which rates players based on their game performance and can theoretically predict their chance of winning. Individual strengths and weaknesses of players can impact game outcomes. For example, chess World Champion Magnus Carlson (currently rated 1st in the world by ELO) has a 14-1 with 25 draws win rate against Hikaru Nakamura (currently ranked 5th in the world by ELO). Going solely by ELO, Magnus Carlson is expected to be 26-6 with 8 draws. As we can see, ELO does not always match reality. This is especially true at each end of the tail, which professionals are.

Machine learning models can be built to predict each player's win rate throughout the game by considering factors such as the total material, number of bishops, etc. To begin this project, a large dataset of chess games with player and board information must be collected. Machine learning modeling can then be applied to learn relevant patterns and strategies from the data. Ultimately, the trained model can be used to predict the outcome of new chess games based on player abilities and board positions. In our research, we explored how to increase the chance of winning (and therefore what features are important).

A chess game can be broken up into three sections: the opening, middle game, and end game. High level players generally have their opening memorized, so they do not need to consider much during those moves. The middle game starts once most of the pieces have been 'developed' (moved from their starting position into a more favourable one). While there is no specific time that the end game starts, it is generally considered to be when each player has three or less minor or major pieces. Rooks and Queens are major pieces; knights and bishops are minor pieces. We decided to train our models using the 15th move as the middle game is generally considered to start between the 10th and 15th move.

To do this, we drew from existing literature 'Time management in a chess game through machine learning'. We focused on the features that they found to be useful and we also implemented ones that they wanted to explore in the future, such as the number of queens on the board and the number of pawns on the board. We also added features such as whether white or black castled, have the right to castle, or lost the right to castle. The paper 'Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques' used the board position at move 20; however, we did not see much difference between models trained using the

15th or 20th move, so we decided to go with the 15th move to reduce computational overhead and increase the number of games that we could train it with.

There are two main websites for online chess: Chess.com and Lichess.org. We decided to use data from Lichess.org due to all of it being readily available online. Our system collects data and prepares it for use in our ML pipeline. First, we downloaded Lichess' entire game history in PGN format for high-rated players. From there, we created a program to iterate through each move to get to the desired move number. From there, we converted the board state into a Forsyth–Edwards Notation (FEN) string — which is the current chess board state in string format. Using the FEN, we extracted the desired features. After that, we used feature engineering to maximize the effectiveness of our features in our model and trained it with Logistic Regression, Decision Tree, CNN, and LSTM. Our strongest model, LSTM, had an accuracy of 0.61.

2 Important Definitions

2.1 Data

We have prepared a comprehensive dataset containing 100,000 games and the extracted features from the FEN (from the PGN). This dataset has various features to enhance modeling accuracy and usability. Our dataset includes the ELO ratings for both white and black players, results, and PGN. Using the PGN, we were able to extract the number of legal moves available for each player, castle rights for each player, material point value difference, the number of queens and pawns on the board, as well as each bishop, knight, and rook pairs.

whiteElo	blackElo	numLegalMoves	numQueens	numPawns	bishPair	knightPair	rookPair	whitePtV	blackPtV	ptVDiff	whiteCR	blackCR	result
2869	2820	42	2	14	6	3	6	35	35	0	0	0	1
2869	2834	46	2	14	6	3	6	35	35	0	0	0	0
2874	2831	58	2	12	6	6	6	37	37	0	0	0	1
2866	2860	39	2	12	3	5	6	31	34	-3	0	1	1
2867	2859	38	0	14	1	5	6	23	23	0	0	0	1

Figure 1: Dataset

2.2 Prediction Target

We want to predict the probability of winning or losing a chess game using an ML model based on the extracted data. The ML model uses extracted features such as the number of legal moves and the number of pieces to predict the probability of winning or losing a game.

Being able to predict the probability of winning or losing a chess game through an ML model can help assist in decision-making and playing against opponents during a chess game. For example, if the ML model predicts a high chance of losing, the player can try to come back by changing strategy, and if the ML model predicts a high win rate, the player can keep the original strategy and continue playing. While a player would not be able to look at this during the game, it can definitely help them study. Many players study 'engine lines' and will memorize the best moves for up to 20 in a row. However, they are the 'best moves' when played by a computer. In contrast, our model will accurately predict a *human's* chance at winning the game. There is no point in getting into a 'good position' if it is only able to be won by a computer.

The objects of these forecasts inform the entire forecasting process and support decision-making.

2.3 Variables or Concept in our data

The variables or concepts in our study are features extracted from Forsyth-Edwards Notation (FEN) strings in a dataset of 100,000 chess games. These features include ELO ratings for both white and black players, results, PGN, number of legal moves available to each player, castle privileges for each player, significant point value differences, number of queens and pawns on the board, and other features. It's possible. We used these to train a machine learning model to predict the outcome of a new chess game based on player ability and board position. We analyzed the correlation of these variables as follows.

	numLegalMoves	numQueens	numPawns	ptVDiff	whiteCR	blackCR	result	eval	eloDiff	newKnightPair	newBishopPair	newRookPair
numLegalMoves	1.000000	0.226861	-0.136632	-0.046646	0.088830	0.093426	0.042423	0.200482	0.000909	0.010050	-0.040227	-0.069230
numQueens	0.226861	1.000000	0.114934	-0.030076	0.209658	0.230268	-0.014520	0.031047	-0.003127	-0.005680	-0.043873	-0.044652
numPawns	-0.136632	0.114934	1.000000	0.015729	0.084872	0.077783	-0.001300	-0.004904	-0.001596	-0.024702	-0.014636	-0.104186
ptVDiff	-0.046646	-0.030076	0.015729	1.000000	-0.048915	0.045974	0.045627	0.217419	0.011178	0.010185	0.004207	0.054199
whiteCR	0.088830	0.209658	0.084872	-0.048915	1.000000	0.194680	0.014347	0.048460	-0.003098	-0.011693	-0.057135	-0.080234
blackCR	0.093426	0.230268	0.077783	0.045974	0.194680	1.000000	-0.012109	-0.083759	0.008311	-0.021871	-0.045761	-0.090542
result	0.042423	-0.014520	-0.001300	0.045627	0.014347	-0.012109	1.000000	0.233608	0.071523	-0.001009	0.007311	0.006410
eval	0.200482	0.031047	-0.004904	0.217419	0.048460	-0.083759	0.233608	1.000000	0.034081	-0.023976	0.022110	-0.004189
eloDiff	0.000909	-0.003127	-0.001596	0.011178	-0.003098	0.008311	0.071523	0.034081	1.000000	-0.011835	0.003253	-0.002014
newKnightPair	0.010050	-0.005680	-0.024702	0.010185	-0.011693	-0.021871	-0.001009	-0.023976	-0.011835	1.000000	0.049602	-0.005680
newBishopPair	-0.040227	-0.043873	-0.014636	0.004207	-0.057135	-0.045761	0.007311	0.022110	0.003253	0.049602	1.000000	0.006713
newRookPair	-0.069230	-0.044652	-0.104186	0.054199	-0.080234	-0.090542	0.006410	-0.004189	-0.002014	-0.005680	0.006713	1.000000

Figure 2: Features Correlation

2.4 Problem Statement(Given, Objective, Constraints)

Our given is a large data set of a chess game with player and move information. Our objective is to predict the outcome of a chess game given a move number and board position. To achieve this goal, we use machine learning modeling to learn relevant patterns and strategies from the data and predict the outcome of future chess games. Our constraint is the choice to use move 15 as an intermediate game rather than move 20 to reduce computational overhead and increase the number of games we can train on. We are also constrained in that chess is a game of skill and humans make mistakes. There is no way to ever get 100% prediction rate as no matter what, someone will make a mistake and they will lose when they shouldn't. From our research into similar literature, a model accuracy of anywhere between 55 to 70% seems to be the goal.

3 Overview of Proposed Approach/System

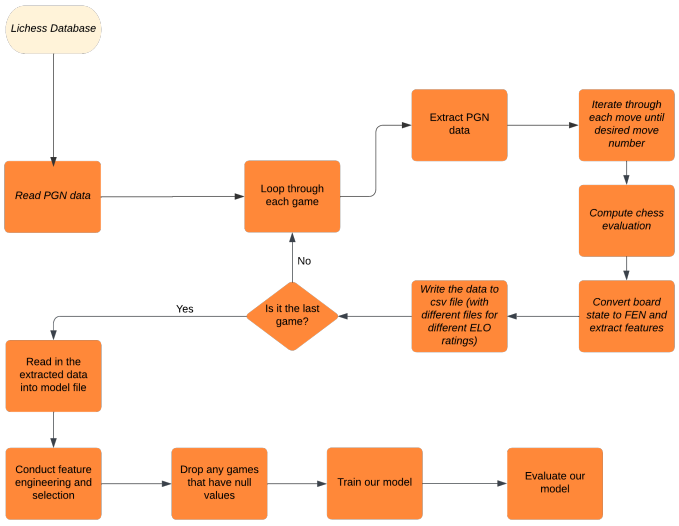

Figure 3: Our Approach Flowchart

We tried various methods such as logistic regression, decision tree, random forest, Convolutional Neural Network and LSTM. We assumed that learning based deep neural networks would be more efficient than the simple logistic regression and decision tree, and we were correct. The reason for that we assumed this was because modern (good) chess engines use deep neural networks to calculate the evaluation of the position, best move, etc. Since it is the best method for chess engines to win, we thought that it would be best at

predicting the win rate. We extracted the important features using the decision boundary based learning, and used this features to train the deep networks.

4 Technical Details of Proposed Approaches/Systems

We found https://database.nikonoel.fr/, which is a collection of high ELO games from Lichess' database that can be found at https://database.lichess.org/. Each month, Lichess can have over 100 million games total, so it is important to have a smaller and more feasible dataset. From there, we looped through each game to extract our features. Due to the sheer size of the database, we first ran into issues regarding memory as we were attempting to keep everything in memory. As this was obviously infeasible, we fixed the issue by extracting the feature for each game and appending to the csv. We implemented Stockfish 15.1 to calculate the evaluation with the given board position. As Stockfish is an open-source software, it can be found at https://stockfishchess.org/download/.

For the training, we set the minimum of the accuracy as the results of logistic regression and decision tree. We found the important features using the random forest. As shown in the Figure 4, there are less important features such as numQueens and rookPair. So we tried exluding these features, but as shown in the Table 1, accuracy was not improved remarkably.

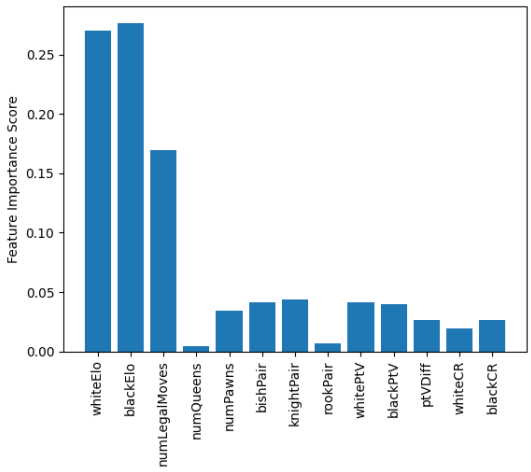

Figure 4: LSTM Feature importance

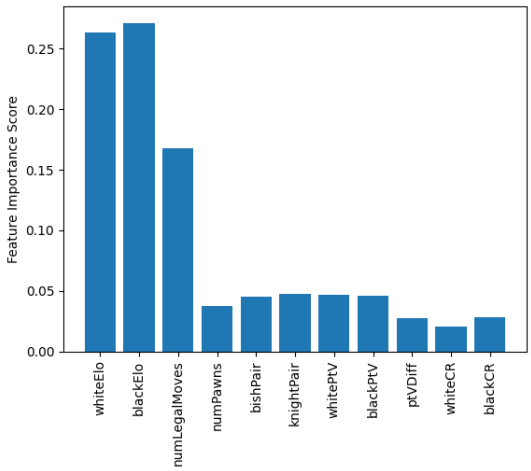

Figure 5: Feature importance with less important features removed

To train the deep neural networks, we tried Convolutional Neural Network first, one of the most famous networks. However, its train loss and accuracy was not good. We assumed the reason is, CNN is not appropriate to the Chess dataset. CNN is usually used for image recognition tasks. This network is good for capture the features of the pixels, but Chess dataset contains series of the data. We adopted the LSTM (Long Short-Term Memory), type of recurrent neural network. LSTM is good for store information over extended periods of time. Even though we don't predict time series, we could get benefit from its memory span. Our training schedule used 0.2 Dropout, and RMSProp optimizer. Batch size was 64 and trained for 50 epochs.

5 Experiments

Our LSTM model had an accuracy of 61%. We could reduce the training loss during the 50 epochs as well as the validation loss.

Figure 6: Training and validation loss

Our experiments found some data that was interesting: Chess engines significantly prefer legal moves. The computer evaluation had a correlation of 0.2 to legal moves; however, the result was only 0.04. This means that humans are not as good at winning with more legal moves than computers are. The same is true for the point value difference. The engine evaluation correlation to point value difference was 0.22 whereas result to point value difference was 0.045. Humans are more affected by the ELO difference in a game, which could be because of a psychological issue of playing against someone that is better than them.

5.1 Baseline methods for Comparison

We believe that the last column, 'With only evaluation' to be near the maximum accuracy achievable. As our dataset only includes high level players that generally do not make extremely large mistakes, we can use the engine evaluation to get the best accuracy of their chance of winning.

Without less important features	With less important features	With only evaluation
0.6100798872180451	0.6092575187969925	0.6482316851615906

Table 1: Model Accuracy

6 Related work

There has been a lot of work and development around chess engines, such as open-source works Stockfish and Leela Chess Zero and from prominent companies such as Google with AlphaZero. For predicting the result of a chess game, the literature was usually focused on predicting the result between specific professional players. That used a completely different technique as it was no longer just about the game and more about player match-ups. One example of this is 'Chess Game Result Prediction System' by Zheyuan Fan, Yuming Kuang, and Xiaolin Lin. They predicted the outcome of a rematch between players in 6 months. Their model had an accuracy of 55.64%, which they praised given the three possible outcomes of chess. The two main papers that our work relates to are 'Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques' by Héctor Apolo Rosales Pulido and 'Time management in a chess game through machine learning' by Guga Burduli and Jie Wu. Pulido's paper did not look at extracting features from the game. Instead, he looked at information in the PGN (such as white's ELO, result, etc.) and the FEN. To look at important features, we turned to Burduli and Wu's paper. Using their paper to recreate, we used many of their features, added the suggested features for future work, and added some that we thought would help, such as the castle right for each colour.

7 Conclusion

We were able to create a model with 61% accuracy, close to the near maximum of 65%. Our model beats previous works where they predicted the (6 months in the future) head-to-head win rate for professional chess players rather than predicting the win rate through board state. We predict that we may be able to improve the accuracy of our model with more epochs or more fine-tuned networks. We could also look at other features, such as how badly the person is playing. We could calculate that by evaluating the position at each move and summing the loss between moves. This would, unfortunately, require an extremely powerful computer and would likely need to have parallelization implemented to run through multiple games at the same time due to the extremely computationally expensive task it would be doing. As we implemented the future works suggested by Burduli and Wu, we were able to have a better understanding of what features are important and what are not. They had suggested that the number of queens could affect the model much; however, we confirmed that it does not. Their suggestion to use board assessment and number of pawns on the board were great, though. Unlike previous works where they focused on professional players only, our model was trained using data spanning years and from strong Lichess players. This means that can be used applied to any player above 2000 ELO on Lichess, not just the top players in the world.

References

1. Guga Burduli Jie Wu (2023) Time management in a chess game through machine learning, International Journal of Parallel, Emergent and Distributed Systems, 38:1, 14-34, DOI: 10.1080/17445760.2022.2088746

2. Héctor Apolo Rosales Pulido (2016), Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques, Universitat Politécnicade Catalunya Barcelonatech Facutat D'Informatica de Barcelona,
https://upcommons.upc.edu/bitstream/handle/2117/106389/119749.pdf

3. Zheyuan Fan, Chess Game Result Prediction System, Stanford University,
http://cs229.stanford.edu/proj2013/FanKuangLin-ChessGameResultPredictionSystem.pdf

Source code:
https://drive.google.com/drive/u/1/folders/10tGsl1ev7hqiAKsqmVepkErec8xrloyW

Game database:
https://database.nikonoel.fr/