Guess the Elo – Predicting Chess Player Rating

Pranav Avva

Department of Computer Science

Princeton University

SML 310: Research Projects in Data Science

Jonathan Hanke, Ph.D.

April 30, 2022

Abstract

Chess is a two-player abstract strategy board game. Games are often rated, which allow players to be ranked according to their skill. Inspired by the “Guess The Elo” YouTube/Twitch series by IM Levy Rozman (a.k.a. GothamChess), this paper explores the Glicko-2 rating system used by Lichess and attempts to predict player rating given a engine-analyzed game. Models were trained using games from the Lichess games database. I found that player rating is successfully predictable to an error rate of about 159 rating points. Further analysis is possible using a variety of approaches and will expand the research in this relatively unexplored field of predicting chess player rating.

Keywords: Chess, Glicko-2, supervised learning, regression

Guess the Elo – Predicting Chess Player Rating

Background and Motivation

The Game of Chess

Chess is a two-player abstract strategy board game in which each player (called "white" and "black," respectively) take turns moving their pieces, of which each player has 16: 1 king, 1 queen, 2 knights, 2 bishops, 2 rooks, and 8 pawns. The objective of the game is to "checkmate" the opponent's king, that is, directly attack the enemy king in a way such that it cannot escape. Games may also be terminated by resignation, loss on time (in the case of timed games), stalemate (in which a player has no legal moves, yet their king is not under attack), draw by the 50-move rule, or agreement of both players to a draw.

Chess Player Rating Systems

Organized chess is governed internationally by FIDE, the International Chess Federation. Chess is also governed by national federations at the individual nation level. For example, the USCF (US Chess Federation) governs all rated games played in the United States.

Rating chess games allows the relative skill of players to be compared. This rating is summarized as a number and the exact formula is rating system-dependent. There are three major systems used today: (a) Elo (b) Glicko (c) Glicko-2. All three systems are used in many zero-sum games beyond chess, including sports and multiplayer video games.

Elo system

The Elo system is the oldest of the three systems and was adopted by FIDE in 1970. The Elo rating system calculates an "expected score" for both players in a rated game. This expected score is then used to update the player ratings after the game. A feature of the Elo system is that if two players with a rating difference of 200 play a game, the higher-rated player is expected to have a 75% chance of winning. Therefore, the expected score calculation means that the greater the rating difference between two players before a game, the greater the change in each player's rating after the game *in the event of an upset*. If the stronger player wins, their rating will increase only slightly, if at all.

In summary, the stronger a player is compared to their opponent, the more they are expected to win, and the more they are penalized if they lose, and less they are rewarded if they win. This causes players with established to be disinterested in playing rated games against those whose rating is vastly different from their own.

Glicko system

The Glicko system–also referred to as the Glicko-1 system–is an improvement on the Elo system. It introduces the concept of rating deviation (RD), a measure of how accurate a player's rating is. A player's RD is dependent on time since the last game played (which allows the system to compensate for players who may have been inactive and lost skill), consistency in performance over their recent games, and their recent opponents' RDs. The higher a player's RD is, the more their rating will change after a rated game.

The Glicko system is used by Chess.com, the most popular online chess website, as well as many competitive multiplayer video games.

Glicko-2 system

The Glicko-2 system is a further improvement on the Glicko system and introduces the concept of rating volatility (RV) which describes how erratic the player's performance has been over their recent games. A player with consistent performance would have a low RV, but if they were to suddenly improve, their RV would increase. Because of the additional variable introduced in the rating formulation, Glicko and Glicko-2 scores cannot be directly compared and would instead need to be converted.

Glicko-2 is used by Lichess, the second-most popular online chess website. Glicko-2 is also the rating system studied in this paper.

Note: the exact formulas for these rating systems are beyond the scope of this paper and are therefore not provided herein. However, they are freely available online.

Chess Engines

Computer scientists have been building algorithms to play chess since the 1950s, though computers were unable to beat the best humans until 1997, when Deep Blue beat then-World Champion GM Garry Kasparov. Since then, computers have only become more powerful. For example, the strongest engine to date is Stockfish 15 and has an estimated rating of 3542 FIDE Elo (CCRL Team, 2022). For comparison, the highest FIDE Elo rating achieved by a human was 2882, attained by current World Champion GM Magnus Carlsen.

Chess engines perform a number of different computations. Given a position, they are able to calculate which player is winning, and by how much. This score is called the "evaluation" of the position and the units are points of material. A pawn is worth 1 point, knights and bishops are worth 3 points each, a rook is worth 5 points, and a queen is worth 9 points. For example, if white and black had the same pieces except black was missing a queen (and the position is otherwise equal), the evaluation of the position would be +9 because white has 9 more points of material than black. It is important to note that pieces alone do not constitute the evaluation; various other factors such as king vulnerability to attack, ability to control squares in the enemy's side of the board, and even phase of the game all affect the evaluation.

Engines are also able to determine the best move for a player in a particular position. Stockfish specifically does this by considering the possible moves for a player in a position and evaluating the resulting position. It then evaluates the best move for the opponent and finds the best reply to that move. This search process is called minimax alpha-beta pruned tree search. Due to the recursive nature of the search and the extremely large number of possible continuations from a chess position, engines typically need an indefinite amount of time to evaluate a position and provide a true best move.

Stockfish is also able to analyze completed games. By sequentially analyzing each position in a completed game, the engine is able to determine errors that players make. These errors are found and scored in following procedure:

1. Receive a chess position as input
2. Use minimax alpha-beta pruning to find the best move
3. Analyze the position after playing this best move and calculate the evaluation score. Let this score be x.
4. Analyze the next position that was actually played and calculate the evaluation score. Let this score be y.
5. Calculate the error of the actually-played move as $y - x$

This error is called "centipawn loss" (CPL) because it is measured in 1/100-ths of a pawn. The CPL of a move is almost surely positive, thought it can be negative if the engine was only able to tree search to a shallow depth *or* if, by virtue of an irregularity of the position, Stockfish didn't correctly score the position. After analyzing the next actually-played move, Stockfish will then correctly list that move as its top choice. Taking the arithmetic mean of the CPLs yields a metric called average centipawn loss (ACPL) which represents the move-over-move performance of the player during the game.

Research Question

From the above information, many possible research questions arise. The question pursued by this paper is: "Given access to Stockfish and a complete chess game with all associated metadata, is it possible to predict the rating of each of the players? Furthermore, how accurate can the model be?"

Review of Related Literature

The author was unable to find previous work in this specific category of predicting chess player rating using machine learning techniques.

Data Acquisition

Lichess publishes all games played on their platform on a month-to-month basis. Each month's games are published in a single compressed `.pgn` file. The models in this paper used a subset January 2013 games dump. This month was chosen because it had the fewest games played (121 332 games), therefore having the shortest download and decompression time. The compressed `.pgn` was downloaded to the Princeton Adroit computing cluster. This cluster was chosen for its high CPU and GPU performance.

After decompressing the PGN archive, a SLURM batch job was created. This job would read through the PGN file and use Stockfish 14.1 to evaluate each position of each game. The evaluations and the following accompanying metadata of each game were stored in a dictionary that was then saved to disk using the Python `pickle` module.

For each game, the following data was collected:

- Date and time of the game
- White and Black's Glicko-2 ratings
- Game result (`1-0`, `0-1`, or `1/2-1/2`)
- Time control (`mm+ss` increment)
- Lichess username for White and Black
- The name of the opening played in the game, as defined in the Lichess opening book
- a list of moves with their accompanying metadata

For each move in each game, the following data was collected:

- The FEN notation for the game state at that position
- The evaluation type (`cp` for centipawn score, `mate` if a forced checkmate sequence exists)
- The evaluation score (if the evaluation type is `cp`, this is the number of centipawns advantage/disadvantage from white's perspective; if the evaluation type is `mate`, this is the number of moves remaining in the forced checkmate sequence)

- Centipawn loss (the difference in the eval score compared to the previous move)

Due to time limit restrictions of SLURM jobs on the Adroit cluster, the job was only able to run for 168 hours at a time. In this time, the job was able to analyze and pickle 31 081 games.

While additional data could have been gathered from the PGN archive, it was determined that there was sufficient data to train a variety of candidate models. The SLURM job can easily be repeated in the future with a different PGN archive to generate a larger dataset.

Data Preprocessing

There were a number of data preprocessing steps that needed to occur before piping the data into a machine learning model or performing exploratory data analysis.

First, because evaluations (and therefore centipawn losses) are calculated from the perspective of White, the value of the centipawn loss needs to be negated for moves by Black. If White makes a poor move, the recorded centipawn loss will be a negative number; if Black makes a poor move, the recorded centipawn loss will be a positive number.

Next, because the model will attempt to predict the rating of a single player, it is possible to double the number of data points by splitting each data point into two, using White's player-specific feature in one data point and that of Black in the other. Data about the game itself remain the same between both "sibling" data points. This causes the available data points to double to 62 162.

Finally, the opening name (as taken from the Lichess opening book) and time control were converted from strings to one-hot encoded columns.

Model Fitting

After performing exploratory data analysis, an 80%-20% train-test split was performed to create a dataset with which to validate the machine learning models against.

Feature Selection

Due to the large amount of data and metadata provided by the PGN archive, and due to domain knowledge, considerable feature selection was performed.

On average, a chess game lasts for 40 moves. This is a known fact dating back to the early days of organized chess under the auspices of FIDE (most rated time controls are chosen with an expected game length of 40 moves in mind). Upon inspecting the `game_length` feature, it was clear that many of the chess games in the data set exceeded the 40 move rule-of-thumb. For this reason, and because a CNN model was later iterated upon, the CPLs of the first 49 moves of each game were included as features. If games had fewer than 49 moves, the remaining moves were padded with zeros. This was done because in the perceptron formula $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$, where $\vec{x}$ is the input vector of values, $\vec{w}$ is a vector of learned weights, and $b \in \mathbb{R}$ is a learned bias term, the neurons whose inputs are zeros would output only the bias term. It is expected, but not guaranteed, that the neural network architecture would raise the bias term sufficiently to prevent the neurons whose inputs are padded from activating.

Model Selection and Iteration

A number of models were selected to iterate upon. Each model is suited for a different task. Since this project requires a complex regression, these models encompass a range of various capabilities. The candidate models used in this paper are: (a) Ordinary Least Squares linear regression (b) feed-forward neural network (c) convolutional neural network (d) random forest and extra trees ensemble (e) XGBoost.

All models were evaluated based on the root mean square error (RMSE), in units of Glicko-2[1] rating points.

[1] Recall that Lichess uses the Glicko-2 rating system.

Ordinary Least Squares Regression

Using `sklearn`'s pipeline creation tools, `StandardScaler` and `LinearRegression` were composed using the default settings for each. The `StandardScaler` causes features to be scaled and centered such that $\mu = 0, \sigma^2 = 1$.

Using just the player's ACPL and the length of the game as features, OLS regression yielded an RMSE of 199 points. Including the max CPL and min CPL of the player over the game reduced the RMSE to 197. Further including the first 49 moves reduced the RMSE to 189 points. Finally, including the time control as one-hot encoded features further reduced the RMSE to 176 points.

Feed-forward Neural Network

Multiple neural network architectures were experimented with. All FFNN models were set to train on a max of 500 epochs, with a batch size of 32, and with early stopping with a `patience` of 25 epochs. All Dense layers used the Rectified Linear Unit (ReLU) activation function. The output layer used a Linear activation. All models used the Adam optimizer with `learning_rate` = 0.01, Mean Squared Error loss, and tracked the Root Mean Squared Error metric.

Dense 64 → Dense 64 → Dense 64 → 50% Dropout. Using just the player's ACPL and length of game yielded ans RMSE of 192 points.

Dense 128 → Dense 128 → Dense 64 → Dense 32 → 50% Dropout. Using the player's ACPL, game length, and first 49 moves (zero-padded if needed) yielded an RMSE of 185 points. Including time control yielded an RMSE of 177. Further including the game's opening yieled an RMSE of 171.

An interesting pattern present in the plot of true values vs. predicted values is a line of predictions for 1500 rating points. This is likely an artifact of the Lichess rating system that the FFNN model is learning. On Lichess, new players start at a provisional rating of 1500 (recall that in the Glicko-2 system, RD and RV determine the change in rating, but new players will always start at 1500 rating points).

Convolutional Neural Network

For the CNN model, only the first 49 (zero-padded) moves were used. For each data point, the 49x1 move CPL array was reshaped into a 7x7 array. The values were then normalized to the $[0, 1]$ range and treated as 7px-by-7px images. Two pairs of 2D convolutions with ReLU activation and 2D max poolings were used. The images were flattened, then followed by three Dense layers (with ReLU activation functions) of size 128, 64, and 32 neurons, respectively. The output layer was a single Dense neuron with linear activation.

The model was compiled with the Adam optimizer (`learning_rate` = 0.001), using mean squared error for both the loss function and tracked metric. The yielded RMSE was 202. Due to impracticality of including other metadata (which was proved useful by the OLS model) in the CNN model, further trials of the CNN model were not attempted.

Ensemble Regression Methods

Both Random Forests and Extra Trees ensemble regressions were performed, thought it became evident that Extra Trees had much higher performance.

Random Forests with 1000 estimators and unlimited depth, training on the ACPL, min and max CPL, and CPL list yielded an RMSE of 183. An Extra Trees regression with the same hyperparameters and features yielded an RMSE of 181. Including the time control reduced the Extra Trees RMSE to 168. Including the opening reduced the Extra Trees RMSE to 161.

XGBoost Regression

XGBoost had the best performance and moderate hyperparameter tuning was performed. All XGBoost regressors were trained on the ACPL, min/max CPL, game length, 49 CPLs, time control, and opening feature set.

With 1000 estimators and max depth of 10, the RMSE was 159, making this model the best performing model across all candidate models. With 1000 estimators and a max depth of 25, the RMSE was 161. With 7500 estimators and a max depth of 25, the RMSE was 160.

Results and Future Steps

The XGBoost regressor with 1000 estimators and max depth of 10 had the best performance across all models trained in this project, with an RMSE of 159. Although this RMSE is still high (159 rating points differnce means the higher rated player would have a 50%-75% chance of winning), it is acceptable for the relatively simple models trained in this project.

Given that there is little to no prior work in this field, there is likely much more exploration possible surrounding this topic. Future projects in this niche can explore using the board positions, rather than just the centipawn losses, as features. Advanced machine learning techniques can also be used, such using pre-trained models (though it is unclear if such models for this task may exist) and using reinforcement learning (though this may require a vastly larger dataset).

Honor Code Statement

This paper represents my own work in accordance with University regulations.

/s/ Pranav Avva

April 30, 2022

Acknowledgements

The author is grateful to Professor Jonathan Hanke for his time on providing advice regarding this project. The author would also like to thank his peers for providing suggestions in approaching usage problems with the Scikit-Learn and Keras libraries and on providing feedback on the structure and content of this paper. Specifically, the author would like to thank the following members of the Princeton Quadrangle Club:

- Rebecca Giblon GS
- Alan Ding ’22
- Marlon Escobar ’23
- Nadia Rodriguez ’23

This project was inspired by the “Guess the Elo” series on YouTube, published by GothamChess. IM Levy Rozman, the individual behind the GothamChess YouTube channel is rated 2333 as of the May 2022 FIDE rating supplement. In the “Guess the Elo” videos, Rozman reviews a follower-submitted chess game live on stream, finds and explains mistakes, and predicts the rating of the players. He is often accurate to within 50-100 points.

For this paper, the author distilled the motivation behind “Guess the Elo” into a simple regression task: predicting player rating given a game played. The author substituted the Stockfish 15 chess engine in place of the experience of over a decade of FIDE-rated chess afforded to GothamChess.

The author would like to thank Levy Rozman for the inspiration for this project.

Finally, the author would like to thank the Lichess team on providing a forever-free and open-source platform for learning and playing chess, and for creating a public database hosting the games played on their platform.

References

CCRL Team. (2022, April 23). CCRL 40/15. https://ccrl.chessdom.com/ccrl/4040/rating_list_all.html

Chollet, F. (2022, February 3). *Keras* (Version v2.8.0). https://keras.io

Developers, T. (2022, April 21). *TensorFlow* (Version v2.9.0-rc1). Zenodo. https://doi.org/10.5281/ZENODO.4724125

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2021). Auto-sklearn 2.0: Hands-free AutoML via meta-learning. *arXiv:2007.04074 [cs, stat]*. Retrieved April 29, 2022, from http://arxiv.org/abs/2007.04074

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lichess. (2013, January). Lichess.org open database [Type: database]. https://database.lichess.org

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Reback, J., Jbrockmendel, McKinney, W., Van Den Bossche, J., Augspurger, T., Roeschke, M., Hawkins, S., Cloud, P., Gfyoung, Sinhrks, Hoefler, P., Klein, A., Petersen, T., Tratner, J., She, C., Ayd, W., Naveh, S., Darbyshire, J., Garcia, M., . . . Battiston, P. (2022, April 2). *Pandas-dev/pandas: Pandas 1.4.2* (Version v1.4.2). Zenodo. https://doi.org/10.5281/ZENODO.3509134