CS224W Report: Analyzing Chess Results Network

Charles Burlin , Matthew Creme and Yoann Le Callonec

December 10, 2017

1 Introduction

This project will seek to analyze the structure of game results networks and how they can be used to predict future outcomes of games. Specifically, we will be looking at a network of previous results of chess matches and will attempt to predict future matches based on our findings. Historically, ELO ratings have been used to rank chess players and have even been used in other fields such as Go[1] and football[2]. These ELO ratings are simply ratings, and therefore may not capture the correlation between certain players. This project will use the network structure of the data in order to create a more thorough classification system.

2 Review of Prior Work

2.1 Edge Weight Prediction in Weighted Signed Networks

Predicting the outcome of chess games can be done via some edge weight prediction technique. Indeed, we can consider a network where the nodes are the chess players who evaluate each other. In that case, a negative edge from player u to player v could mean that u beat v, whereas a positive edge from v to u would mean that u beat v. The magnitude of these weights can depend on several factors, such as the number of times these players played each other, what colors they were playing with, etc.

The article [3] suggests a method to predict the weight of future edges, that is given two nodes u and v in the network, we try to predict the weights $W(u, v)$ and $W(v, u)$ of the directed edges (u, v) and (v, u) respectively. Before actually trying to predict the weights, the authors propose to compute two different but dependent metrics for each node v: goodness $g(v)$ and fairness $f(v)$. The goodness score, which captures how much the node is trusted or liked by other nodes lies in the $[-1, 1]$ interval, whereas the fairness score, which takes its values in $[0, 1]$, measures how fair the node is when it comes to judging other nodes.

Goodness and fairness are mathematically defined as follows

$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u) \times W(u, v)$$

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|W(u, v) - g(v)|}{R}$$

Given two nodes $(u, v) \in V^2$, the weight $W(u, v)$ is predicted by $f(u) \times g(v)$.

Our idea is to analyze the chess network as a social network, with weights given by games outcomes. We could then compute goodness and fairness scores for the players, and predict the weights of missing edges, i.e. future games.

However, while the article simply suggests to estimate $W(u, v)$ with fairness and goodness scores, using $\hat{W}(u, v) = f(u) \times g(v)$, we will fit a logistic regression on the data to predict the outcome of future games.

2.2 HITS and PageRank Link

In their paper "*Comparative Study of HITS and PageRank Link based Ranking Algorithms*" [4] Pooja Devi, Ashlesha Gupta, Ashutosh Dixit present and compare PageRank and HITS as ranking algorithms. Having made the success of the Google search engine, PageRank is based on a web page being given importance when important pages point to it. The actual PageRank score is defined as:

$$PR_i = (1-d) + d \sum_{j:j\to i} \frac{PR_j}{Q_j}$$

where $\{j : j \to i\}$ is the set of pages that point to i and Q_j the number of outlinks of j. Last, d is the damping factor that makes for pages with no outlinks. An iterative method described in the paper allows to find a distribution PR that fits the equation above.

2.3 Bradley-Terry model applied to Chess Network

Ferreira presents one method for predicting future chess results in his paper "*Predicting the Outcome of Chess Games based on Historical Data.*" [5] His approach involves using a modified version of the Bradley-Terry model. This model estimates the probability of player X defeating player Y as:

$$P(\text{X beats Y}) = \frac{1}{1+\frac{\gamma_Y}{\gamma_X}}$$

Where γ_X and γ_Y respectively represent the rating of player X and Y. Then Ferreira notes that the difficulty arises in estimating γ_X, as X only plays against competition with unknown ratings as well. In order to solve this problem, Ferreira considers a reference player Z and estimates:

$$\gamma_X = P(\text{X beats Z}) = \frac{1}{1+\left(\frac{1}{P(\text{Y beats Z})}-1\right)\left(\frac{1}{P(\text{X beats Y})}-1\right)}$$

Now, in order to take into account all the opponents of X , Y can be taken to be all the opponents of X and therefore P(X beats Y) can be taken to be the average over all the elements in Y. Now, in order to actually solve for these quantities, an iterative method is used to update the probabilities based on previous values.

3 Data Collection and Analysis

Our data was retrieved from kaggle [6]. The dataset contains 4 columns containing the month in which the match was played, the ID of the white player, the ID of the black player as well as the score. The score is either 1 which represents a white win, 0.5 which denotes a tie or 0 which represents a black win. The dataset has 8631 nodes along with 65,053 edges representing matches that had been played. In addition, there is a cross-validation dataset which contains the results of matches between months 96-100, as well as a test dataset that consists of results during the months 101-105.

Before explaining our methods for future match results prediction, we will explore some additional features of the dataset. As the white player always has the first move in chess, the white player has an advantage. This phenomenon is illustrated in the dataset as the white white player averages 0.5464 points per match. This will be an important property for our prediction methods. In addition, there were 12,994 pairs of players who played each other more than once, and combined these players played a total of 31302 games against one another. However, when two players replayed one another, 47.30% of games had the same result as the previous game. This is much higher than $\frac{100\%}{3}$, which would be the probability of two successive having the same result. Therefore, it is possible that this could be a useful feature as well.

Now, the percentage of balanced triads and unbalanced triads was studied. The overall score of each pair was calculated as the average score a player received against this opponent. A balanced triad was therefore any triad that respected a hierarchy of these scores. It was found that 48.66% of triads were balanced while 51.33% of triads were unbalanced. Now, it seems as if this network is

highly unbalanced. However, the addition of many ties makes it difficult for a triad to be balanced as if players X and Y are tied overall, then any mutual neighbour must have the same result against both of them in order to create a balanced triad.

Lastly, we have a visualization of the network. A random connected subset of the graph was chosen, and each node was sized based on how well the player performed against other nodes in the network.

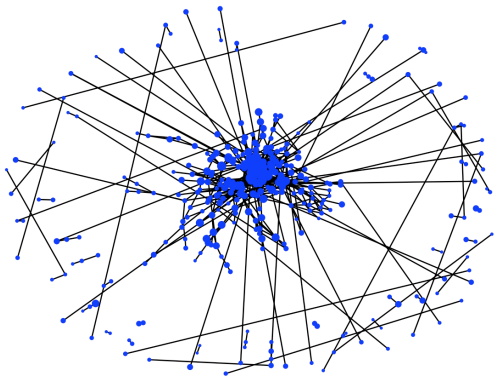

Figure 1: Random Subset of Chess Results Network

4 Methods

4.1 Fairness and Goodness

As explained in the litterature review, our goal is to compute a fairness and a goodness score for every player in the chess network. First, we create a weighted directed graph G as follows:

- Create a node for every chess player in the dataset.
- For each game between white player p_1 and black player p_2 with result $r \in \{0, 0.5, 1\}$, create an edge (p_1, p_2) with weight $1 - r + \epsilon$ and en edge (p_2, p_1) with weight r. This means that the white player evaluates the black player's strength with the grade $1 - r + \epsilon$, whereas the black player evaluates the white player with the grade r. The positive ϵ accounts for the fact that playing with whites gives a little advantage.

As said previously, we compute a fairness and a goodness score for every player in the chess network, using the following algorithm (see [3]):

For all the nodes $v \in V$ initialize $f^0(u) = 1$ and $g^0(u) = 1$

do

$$g^{t+1}(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f^t(u) \times W(u, v), \quad \forall\, v \in V$$

$$f^{t+1}(u) = 1 - \frac{1}{2|out(u)|} \sum_{v \in out(u)} |W(u, v) - g^{t+1}(v)|, \quad \forall\, v \in V$$

while $\sum_{u \in V} |f^{t+1}(u) - f^t(u)| > \epsilon$ or $\sum_{v \in V} |g^{t+1}(u) - g^t(u)| > \epsilon$

return $f^{t+1}(u)$ and $g^{t+1}(u)$, $\forall\, u \in V$

Figure 2 shows the distribution of fairness and goodness scores among chess players. We can now use these scores as features for a logistic regression, and try to predict the outcome of future games. In other words, our multi-class logistic regression would take as input $(f_{\text{white}}, g_{\text{white}}, f_{\text{black}}, g_{\text{black}})$ and output three scores corresponding to the events "white wins", "draw" and "black wins".

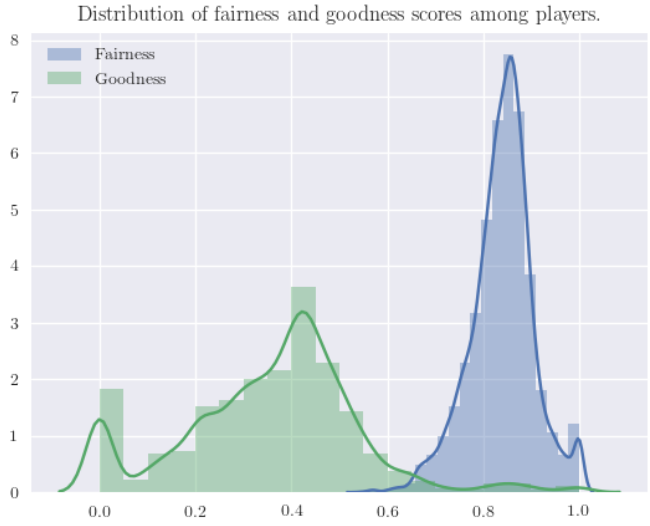

Figure 2: Distribution of fairness and goodness scores in the chess network

4.2 Estimating Player Strengths Model

As in the model explains by Ferreira, the strengths of each player are calculated using:

$$\gamma_X = P(\text{X beats Z}) = \frac{1}{1 + \left(\frac{1}{P(\text{Y beats Z})} - 1\right)\left(\frac{1}{P(\text{X beats Y})} - 1\right)}$$

Where an iterative method is used until these probabilities converge. Explicitly, the strength of each player is initially set to 0.5. Until all values have converged, use the above formula to update each γ_X. $P(\text{Y beats Z})$ is the strength of Y, and the value from the previous iteration is used, while $P(\text{X beats Y})$ is calculated as shown below.

Next, in order to calculate $P(\text{X beats Y})$, we arise at an issue as if X has only played one game and lost that game then this value would be zero and therefore X's strength would be zero. This would result in the algorithm considering X as a player that can never win. To combat this, we introduce 2 constants α and β. and set

$$P(\text{X beats Y}) = \frac{\alpha * \beta + \sum_{y \in Y} w_{x,y} P(X > y)}{\alpha + \sum_{y \in Y} w_{x,y}}$$

Intuitively, the means that we introduce α fake opponents to X, where X scores an average of β in these games. Now α and β were calculated for the entire dataset using an iterative method. In addition, a time-weighting factor $w_{x,y}$ is added to each game. This value is calculated as $\frac{100}{100+\gamma(M-m_{x,y})}$. Where M is set to 100, the number of months in the training set and $m_{x,y}$ is the months in which the game took place. Lastly, γ is a hyper parameter that was calculated with α and β

Figure 3 shows the distribution of fairness and goodness scores among chess players. Now, once the strengths of each player have been calculated, these can be used as the features in a logistic regression to predict the results of future chess games. In addition, a simple neural network with softmax regression can be trained on these features to try and obtain better results.

Lastly, the strengths of each player can be combined with the fairness, goodness and page-rank of each node in the graph to create a combined model. The combined features can then be used as inputs into a logistic regression in hopes of obtaining superior results.

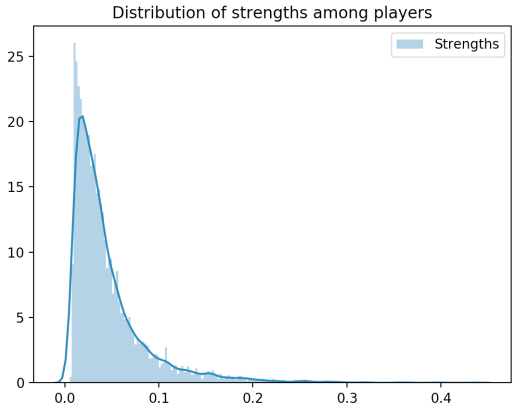

Figure 3: Distribution of player strengths in the chess network

4.3 PageRank

We try to predict matches outcome using the PageRank algorithm. In order to do so, we decided to use two approaches in order to build a graph from our training set data.

In the simple approach, each node of our graph represents a player and we add a directed edge from player A to player B if B has beaten A more often in all the matches they played. Although this model is simplifying as it does not capture the frequency at which players beat one another, it is an interesting baseline to start with.

As described previously, we then use the SNAP function to compute PageRank scores with a damping factor of $d = 0.85$. In order to predict the outcome of a future match, we predict a win for A if $\text{PageRank}_A > (1+\epsilon)\text{PageRank}_B$, a win for B if $\text{PageRank}_A < (1-\epsilon)\text{PageRank}_B$ and a draw otherwise, for a certain value of ϵ.

In the more complex approach, we try to take advantage of the time at which each game was played in order to construct our directed graph. In fact, we decide to draw a directed edge from A to B if the following quantity is positive:

$$s_{AB} = \sum_{s \in \mathcal{G}_{AB}} s e^{\frac{T-t_t}{\tau}}$$

where for each score s (1 for win, 0 for draw and -1 for loss) in the games played between A and B $\mathcal{G}_{AB}$, we weight by the quantity $e^{\frac{T-t_t}{\tau}}$ where t is the time at which the game was played and τ and T some hyperparameters. This allows for more recent game to count more than old games.

Once the graph is built, we use the method described above in order to make predictions.

5 Results

5.1 Fairness and Goodness Results

The multi-class logistic regression using fairness and goodness scores as features provided encouraging results, as displayed on table 1. The confusion matrix of the classifier gives the number of test examples (games from the test set) classified as white victory, draw or black victory as a function of the real results. One observation is that the algorithm tends to predict a white or a black victory nearly 81% of the time even though 44% of the games end up in a draw in the test set.

We also present the ROC curves (Receiver Operating Characteristic) of the logistic regression for the three labels "white wins", "draw" and "black wins" on figure 4. The ROC curve presents the true positive rate (proportion of positive labels predicted as positive) versus the false positive rate (proportion of negative labels predicted as positive) as the discrimination threshold varies.

	Predicted White wins	Predicted draw	Predicted Black wins
White wins	1104	144	324
Draw	820	920	1123
Black wins	342	178	1550

Table 1: Confusion matrix when applying the logistic regression on a validation set.

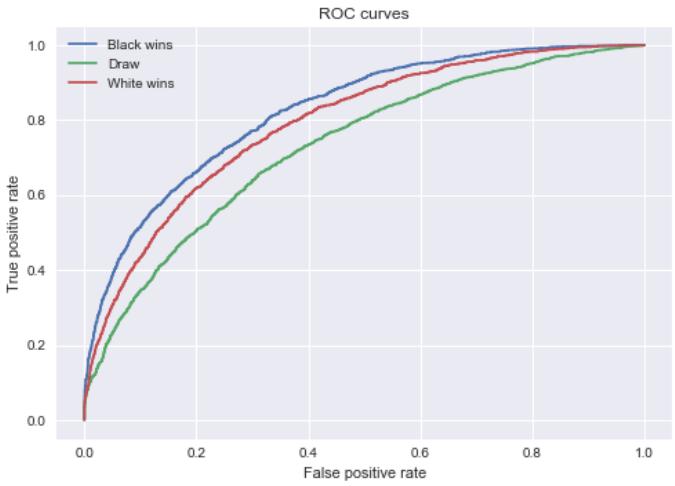

Figure 4: Receiving operator characteristic of the logistic regression.

5.2 Player Strengths Model Results

The multi-class logistic regression using estimated player strengths showed encouraging results, as the confusion matrix for the classification was:

	Predicted White wins	Predicted draw	Predicted Black wins
White wins	1006	1541	245
Draw	833	3135	739
Black wins	371	2052	1260

Table 2: Confusion matrix when applying the logistic regression.

With a weighted F-1 score of 0.4712. F-1 score is a metric defined for each label between 0 and 1 as:

$$\frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{where precision } = \frac{t_p}{t_p + f_p}$$

$$\text{and recall } = \frac{t_p}{t_p + f_n}$$

Where t_p represents the true positive rate, f_p the false positive rate and f_n the false negative rate. Lastly, in order to calculate the weighted F-1 score, the F-1 score of each label is weighted by the number of occurrences of the label.

The simple neural network with softmax regression obtained fairly comparable results. When analyzing the results, it is clear that the algorithm is too often predicting ties. The original kaggle challenge was scored not based on classification but rather on root mean square error. Therefore, using the logistic regression probabilities, respectively the neural network probabilities, the expected score in each game could be computed. Using these values, a root mean square error, as calculated in the competition, of 0.69605 was obtained. This value would have ranked among the top 10 in the initial challenge and therefore is an extremely encouraging result.

5.3 PageRank Results

The initial PageRank method shows some interesting results for win-loss predictions. With the same calculation method as above, we obtain a weighted F-1 score of 0.4507. We summarize the results in table 3.

	Predicted White wins	**Predicted draw**	**Predicted Black wins**
White wins	530	315	147
Draw	480	632	431
Black wins	126	210	313

Table 3: Confusion matrix on the validation set when using the simple PageRank method.

However, using the more complex PageRank method described above with $T = \tau = 100$ only slighlty improves the results, as shown in table 4. Indeed, the new F-1 score is: 0.4639.

	Predicted White wins	**Predicted draw**	**Predicted Black wins**
White wins	525	317	150
Draw	478	637	428
Black wins	122	218	309

Table 4: Confusion matrix on the validation set when using the advanced PageRank method.

5.4 Combined Features Results

We just evaluated the performances of a multi-class logistic regression when using three distinct sets of features. It makes now sense to combine these features and feed them to our logistic regression classifier. We eventually end up with 8 features: fairness score, goodness score, PageRank score and strength of the two players.

We get the confusion matrix from table 5.

	Predicted White wins	**Predicted draw**	**Predicted Black wins**
White wins	802	516	254
Draw	322	1910	631
Black wins	208	628	1234

Table 5: Confusion matrix on the validation set when using all the features together.

We notice that combining the features actually helps predicting the games outcome, which is not really surprising. The fairness and goodness scores worked well when the real results were a black or a white victory, whereas the strength score was very efficient when predicting draws.

6 Challenges

Our main issue was the lack of data, as many players in our dataset only took part in a few games, and predicting the outcome of a game when one of the two players only took part in a few games in the training set is quite challenging.

The figure 5 shows the distribution of the number of games played per each player in the training set.

7 Conclusion

During this project, we tried to apply some network analysis techniques to the problem of predicting the outcome of chess games. We believe our approach of seeing chess players and chess games as the nodes and edges of a network was original.

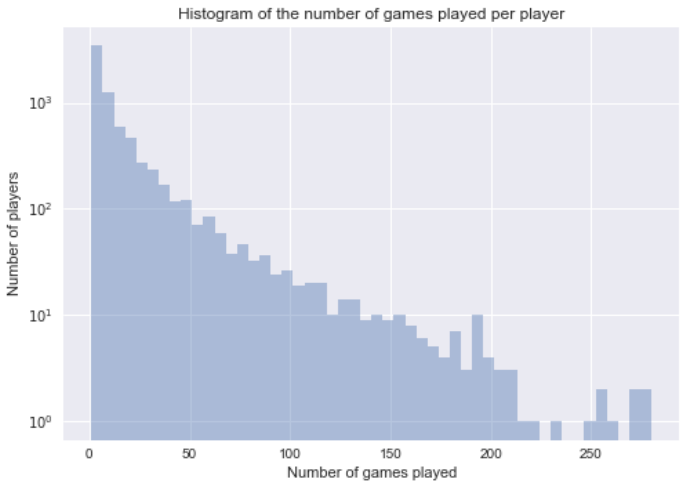

Figure 5: Histogram of the number of games played per player.

The results we obtained are definitely encouraging, even though our predictions could probably be even more accurate if we had focused more on the machine learning aspect of the project. However, the main objective of our project was to find interesting features, such as fairness, goodness, strength and page-rank.

Historically, ELO ratings have been used to predict outcomes in chess matches. These predictions are solely based on the rankings of players, and they are unable to capture some of the intricacies of the network. We have come up with more robust features that can hopefully capture the interdependencies and correlation between nodes in the network.

References

[1] Rémi Coulom. "*Computing Elo Ratings of Move Patterns in the Game of Go.*" Computer Games Workshop, Jun 2007, Amsterdam, Netherlands.

[2] Lars Magnus Hvattum and Halvard Arntzen. "*Using ELO ratings for match result prediction in association football.*" International Journal of Forecasting. Volume 26, Issue 3, July–September 2010, Pages 460-470.

[3] Kumar, Srijan and Spezzano, Francesca and Subrahmanian, V.S. and Faloutsos, Christos. "*Edge Weight Prediction in Weighted Signed Networks.*" Data Mining (ICDM), 2016 IEEE International Conference on.

[4] Pooja Devi, Ashlesha Gupta, Ashutosh Dixit "*Comparative Study of HITS and PageRank Link based Ranking Algorithms.*" International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 2, February 2014

[5] Diogo R. Ferreira. "*Predicting the Outcome of Chess Games based on Historical Data*". November 25, 2010.

[6] Kaggle. "*Chess ratings - Elo versus the Rest of the World.*" www.kaggle.com/c/chess/data.