Beating Elo

Jeffrey S. Pennington

December 10, 2010

1 Introduction

The Elo system is a rating system of competitive head-to-head games. It or small modifications of it are used by the United States Chess Federation (USCF), the World Chess Federation (FIDE), American college football and basketball and Major League Baseball.

The Elo system assigns a single rating to each player and calculates expected outcomes based on the rating differences:

$$E_A = \frac{1}{1 + e^{(R_A - R_B)/\Lambda}}, \quad E_B = \frac{1}{1 + e^{(R_B - R_A)/\Lambda}} \tag{1}$$

where E_A and E_B are the expected outcomes for players A and B, and R_A and R_B are their ratings. Here we designate a win by 1, a loss by 0, and a draw by 0.5. The spread of the ratings is set by Λ, which for chess ratings is usually taken to be $\Lambda = 400/\log(10) = 173.72$.

The result of each game updates each player's rating according to an update rule. The update rule is usually based on the difference between the actual outcome and the expected outcome,

$$R_A \leftarrow R_A + K(I_A - E_A), \tag{2}$$

where I_A is the actual outcome of the game. The factor K is a constant which, given Λ, determines the volatility of each player's rating. The different Elo chess rating systems use different values of K, usually between 16 and 32. Most systems take $K = K(R_A)$ to be a decreasing function of the rating, so that higher rated player's ratings are less volatile.

2 Elo versus the Rest of the World

The website kaggle.com hosted a competition to find an approach that predicts the outcomes of chess games more accurately than the Elo rating system. The contest provides a dataset of 65,053 games from 100 consecutive months between 8,631 of the top chess players. Each element of the dataset contains four pieces of information: the month, a number identifying the white player, a number identifying the black player, and the score. The score is 0 for a white loss, 0.5 for a draw, and 1 for a white win. The test data contains 5 additional months of data for the same players. The submissions are evaluated based on the RMSE of the players' predicted total scores per month and their actual total scores per month during those 5 months.

3 Feature Selection

We illustrate the difficulty of selecting good features with a simple linear regression model. A little thought suggests that perhaps some good features to include for each player are:

$$F_1 = \left\{ \begin{array}{c} \text{percent of games won/lost/drawn} \\ \text{total number of games played} \end{array} \right.$$

This gives very high training error, and indeed F_1 does not really capture the information very well. One might consider,

$$F_2 \quad = \quad \begin{cases} F_1 \text{ as white} \\ F_1 \text{ as black} \end{cases}$$

Or, better, yet,

$$F_3 \quad = \quad \begin{cases} \text{the mean of } F_2 \text{ of all opponents defeated as white,} \\ \text{the mean of } F_2 \text{ of all opponents drawn as white,} \\ \text{the mean of } F_2 \text{ of all opponents lost to as white,} \\ \text{the mean of } F_2 \text{ of all opponents defeated as black,} \\ \text{the mean of } F_2 \text{ of all opponents drawn as black,} \\ \text{the mean of } F_2 \text{ of all opponents lost to as black} \end{cases}$$

We select the $n = \{1, \ldots, 19\}$ most relevant features from set F_3 using stepwise regression and plot the learning curve in figure1.

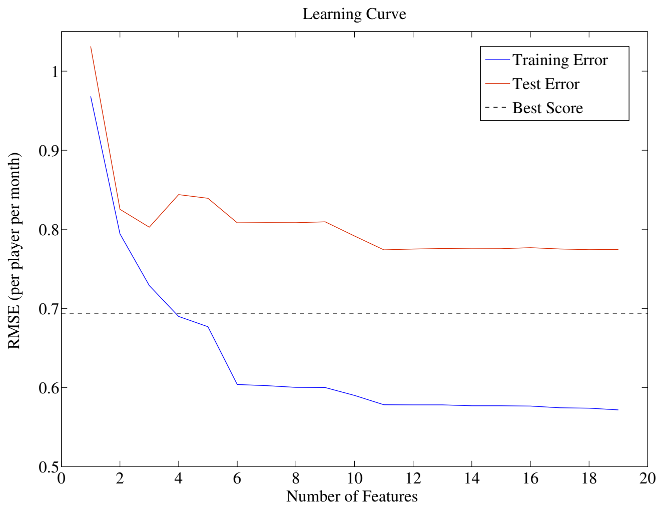

Figure 1: Learning curve for linear regression. The dotted line shows the competition's best submission. The test error was calculated after the full data set became public.

The learning curve shows that the test error is high even though the training error is relatively low, an indication that our feature set is suboptimal. F_3 has 36 elements and it has only included the relevant data of each players' opponents. To add more features, we would need to include some information about the opponents of each player's opponents. Actually, the number of features is exponential in the recursion depth, d:

$$|F| = 2 \times 6^d \tag{3}$$

It is impractical to include features for a recursion depth larger than about 5, but, in principle, d should be as large as the total number of games played by the player with the most games, which in this data set is over 100. This obstacle is not limited to linear regression and will be present with any model. It is therefore necessary to consider an entirely different approach.

3.1 Optimal Ratings

One observation about the fitted linear regression model is that the coefficients of the features associated with the white player are roughly equal and opposite in sign to the coefficients of the features associated with the black player. This indicates that a good model might be to model the predicted outcomes as,

$$p_i = \beta_0 + r_{w_i} - r_{b_i}, \tag{4}$$

where p_i is the predicted outcome of the ith game, $\beta_0 = 0.5468$ is the mean score, w_i and b_i are the white and black players in game i, and r_{w_i} and r_{b_i} are their ratings, which we will attempt to fit.

This model has $8,631$ features and $65,053$ training examples. It is only numerically tractable because the model is linear and the design matrix is quite sparse, which allows for certain iterative least squares algorithms to succeed. We solve this problem using the lsqr function in MATLAB. We find,

$$\text{training error} = 0.5630, \quad \text{test error} = 0.7377 \tag{5}$$

which is a substantial improvement from the previous model.

4 Time Dependence

Performance in chess, like all games, varies over time. For this reason, one would expect a player's recent games to be the best indicator of his future performance. The most convenient method to account for this phenomenon is by weighting each month differently in the least squares calculation,

$$\text{error} = \sum_i w_i(t_i)(y_i - p_i)^2, \tag{6}$$

where y_i is the actual outcome of the ith game, p_i is given as in (4) and $w_i(t)$ is a weight function. The variable t_i is discretized by month, as dictated by the data. As a model for $w_i(t)$, we choose a monotonically decreasing function. The simplest such choice is,

$$w(t) = t^\alpha. \tag{7}$$

Post-analysis shows that the optimal value for α is 0.64, which gives some improvement:

$$\text{training error} = 0.5929, \quad \text{test error} = 0.7293 \tag{8}$$

Unfortunately, fitting α by cross-validation is difficult. This is due to an unfortunate feature of the training data, namely that it is heavily end-weighted. There are many more games in the last 10 months than in the remainder of the data set. This can be seen in figure (2).

To make matters worse, the function $w(t)$ turns out to be crucially important to achieving optimal scores, and the simple power law model is actually quite poor. As a demonstration of this, figure (3) shows the optimal weighting function as determined by post-analysis. Such a model achieves a test error of 0.692, well below the winning score for this competition. We expect that a weight function which is not monotonically decreasing will not generalize to other contexts, but such a model may be necessary to win the competition.

5 Optimization

5.1 Reduce over-fitting

There are several further techniques that should be employed to optimize predictions. First, there are many players who engage in very few games, and as such their optimized ratings will be skewed. There are a number of ways to reduce such over-fitting, but we are constrained by the very large size of the design

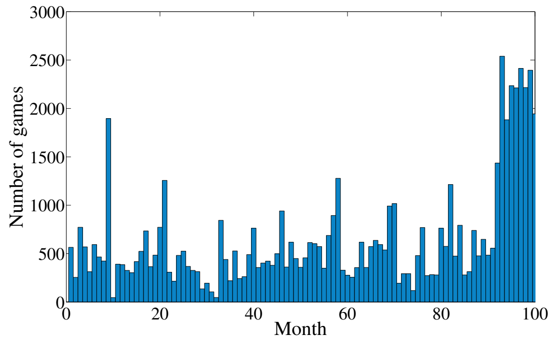

Figure 2: The number of training examples per month. The last months are more populated, making cross-validation difficult.

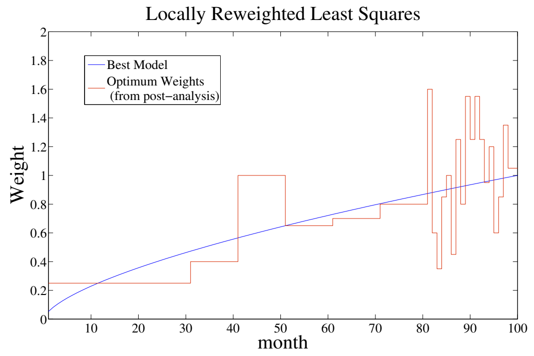

Figure 3: The optimal weight function and the power law approximation. The optimal weight function produces a test error of 0.692.

matrix. One method which is computationally efficient is to introduce a damping factor to the least squares fit,

$$\min_r |Ar - y|^2 + d^2 |r|^2. \tag{9}$$

The damping factor d governs the penalty for ratings which differ from zero. A quadratic penalty is not necessarily optimal, but such a mechanism is simple to incorporate into an iterative least squares algorithm and the problem remains numerically tractable.

In addition to penalizing outlying ratings, it is also possible to penalize outlying predictions. If Br gives the predicted values for the test set, then we can optimize the following,

$$\min_r |Ar - y|^2 + d_1^2 \, |r|^2 + d_2^2 \, |Br|^2. \tag{10}$$

The above problem remains feasible, and post-analysis shows that the optimal values for the parameters are $d_1 = 0$ and $d_2 = 0.70$. This produces a substantial increase in performance, and the resulting scores are now,

$$\text{training error} = 0.5925, \quad \text{test error} = 0.7025, \tag{11}$$

which would put the model into 15th place in the competition.

5.2 Include quadratic ratings

Evidence shows that at the highest level of chess competition, draws are more frequent. That is to say, the prediction p_i should tend towards 0.5 as r_w+r_b increases. This behavior cannot be captured by the current linear model, which only uses the rating difference r_w-r_b.

This suggests we introduce a more complicated model, but unfortunatley we are computationally limited. Running a single minimization with a non-linear model (using gradient descent) takes approximately fifteen minutes on modern hardware. While this is reasonable for the computation of a final prediction, this is prohibitive for the lengthy process of model- and parameter-selection.

One compromise is to employ a non-linear model to the linearly-determined ratings. One might question whether this is a reasonable procedure since the ratings were already optimized based on their linear differences. Figure (4) shows that indeed there is additional information in the ratings beyond their linear differences. Notice that $r_A - r_B$ is a good predictor for the overall score, but that as $r_A + r_B$ increases, the likelihood of a draw increases. This is in accord with the phenomenon we expected to observe.

So, given the linearly-fitted ratings as features, we can generate a non-linear fit to the data to improve our results. It turns out that a quadratic model has the best performance, and decreases the error a small amount:

$$\text{training error} = 0.5917, \quad \text{test error} = 0.7011, \tag{12}$$

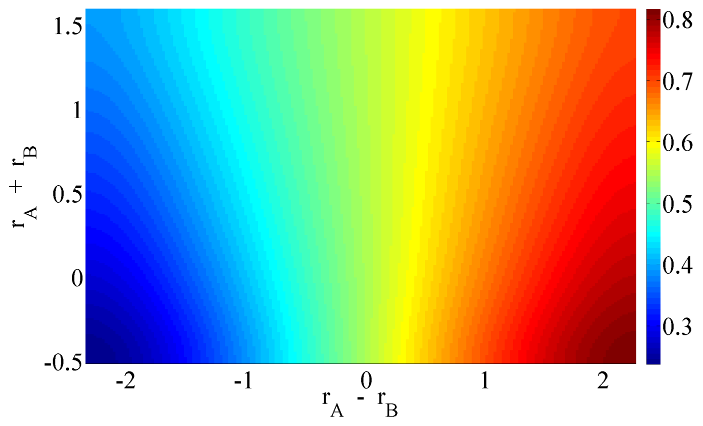

Figure 4: The observed score as a function of the linearly-fitted ratings. The likelihood of a draw increases with the sum of the ratings.

5.3 Post-processing

Finally, a bit of post-processing can be performed on the predictions for further optimization. Binning the predictions near 0, 0.5, and 1, to those values helps a small amount. Curiously, it turns out that previous games between the same opponents are actually very poor indicators of future outcomes. This problem is exacerbated with the optimal ratings because the ratings are chosen to fit past outcomes. This particular limitation can be alleviated to some extent by rescaling the predictions between players who have previously played each other closer toward the mean. All together, these post-processing procedures can reduce the overall error to the final result:

$$\text{training error} = 0.6035, \quad \text{test error} = 0.6992, \tag{13}$$

6 Conclusions

The Elo benchmark for this competition was 0.7438. This benchmark was easily surpassed with a simple implementation of the optimal ratings method. Further modifications and optimizations allowed this model to achieve a minimum score of 0.692, well above the winning score for this competition. Unfortunately, the parameter choices for this particular model are unintuitive, unlikely to generalize, and impossible to obtain without the full data set on hand. Nevertheless, the basic framework allows for the construction of very good models with reasonable parameter choices. The best model we found with realistic parameter values obtained a score of 0.6992, which would finish in the top ten of the competition. Unfortunately, many of the improvements to the model came after the competition ended, so they couldn't be submitted!

One potential drawback to the optimal ratings method is that for large datasets it is computationally impractical. This problem has a fairly straightforward remedy. Once the overall scale and mean of the ratings are set, ratings which are calculated within inclusive sub-populations are comparable. This means that optimal ratings can be calculated in separate, manageable blocks. Moreover, the final ratings could be constructed from an average of several different partitions of the entire population into such blocks. In this way, ratings could be assigned regardless of how large the population happens to be.