Deloitte/FIDE Chess Rating Challenge

Andrew Cotter (team "George")

May 11, 2011

1 Overview

My approach to this challenge was initially influenced by what I knew of the Netflix prize competition ([1]), so my early efforts were focused on creating a large number of very different skill measures, and then "blending" them in order to create a robust skill measure which would take into account all of the various aspects considered by its components. My hope was that the primary constraints to this approach would be my own patience, and implementation time–I was confident that I could come up with any number of bad ideas for how to measure the skill of a chess player, and that the performance of my final, blended classifier would continue to improve as the number of these bad ideas multiplied.

I say that this was my *initial* approach because implementing it proved to be a greater challenge, in practice, then I had at first hoped–when blending different skill measures, I found that the top one or two components completely determined the final result, with the other, worse-performing components being almost entirely ignored, even those which I had reason to believe were capturing aspects of the data which were not accounted for adequately by the other components. In the end, I wound up using only two: a logistic-regression based approach (section 2.1) which I believe does a good job of finding the absolute "skill" of a player, but a poor job accounting for either uncertainty or variations in skill over time; and also a modified version of Glicko ([2] and section 2.2), which I believe does a decent job of accounting for uncertainty and changes in skill over time, but never looks back over its predictions in order to re-evaluate them in light of more recent games, and therefore doesn't "fit" the data as well as it might.

If time had permitted, I would have attempted to implement a variant of Trueskill Through Time ([3, 4] and as described by Jeremy Howard in the Kaggle forums for the "Chess ratings - Elo versus the Rest of the World" challenge), which I believe would have outperformed Glicko, while still capturing the same essential information. My blending model is described in section 2, but my suspicion is that many other teams attempted to implement essentially the same approach, and did so better than I did. The main reason why my entries performed well in the competition was the technique which I used to exploit what has come to be called "future scheduling".

I'm not a chess player (at least, not a good one), and have never participated in the chess tournament. After a bit of research, however, I learned that chess tournaments are generally organized according to the "Swiss system", in which, at each round, players are matched against other players with similar records in the tournament. Elimination-style tournaments evidently do take place, but are far less common–I am unsure if any such tournaments are present in the dataset. The upshot of this is that if a weak player, in a particular month, is matched against a large number of stronger players, then it is likely that they are all participants in the same Swiss-style tournament, and furthermore that the weak player is, for this month at least, outperforming what we previously believed to be his or her skill level.

It is immediately clear that the use of this information should dramatically improve performance. The main difficulties in making use of this information are:

1. The dataset is noisy: the test dataset has a number of spurious games added, and also probably does not include all games by all players in a particular tournament.

2. There are a large number of players who played few enough games in the test dataset that any "future scheduling"-based inferences we attempt to make about their performance will be based on few examples, and will therefore be unreliable.

Hence, when attempting to perform future scheduling, the approach must be, to the greatest extent possible, *robust*. My solution was to break this problem into two subproblems. In the first of these, the task is to answer the question "is this player a participant in a Swiss-style tournament this month, and do we have enough data to make use of this information", and the second is to either predict the player's skill without using future scheduling (if the answer is "no"), or to do so while making use of future scheduling (if the answer is "yes"). The precise technique used to solve both of these subproblems is logistic regression, using my previously-mentioned "blended" skill measure, as well as a number of features derived from the testing data which I believed would include much of the relevant tournament information. This will be described in section 3.

2 Blended skill measure

2.1 Logistic model

In the setting of this competition, we have n games, each of which is played between white and black players $w_i, b_i \in \{1, \ldots, m\}$ at time $t_i \in [-1, 1]$ (the "times" in the dataset are actually quantized to the level of months, but these may be easily scaled to any desired range). The outcome of the ith game is $y_i \in \{0, 1/2, 1\}$, denoting a loss, draw, or win by the white player, respectively. The task is to predict $e_i \in (0, 1)$ for $i \in \{1, \ldots, n\}$ minimizing:

$$f(e) = \sum_{i=1}^{n} \left(y_i \log_{10} \frac{1}{e_i} + (1 - y_i) \log_{10} \frac{1}{1 - e_i} \right) \tag{1}$$

If $y_i \in \{0, 1\}$, then this would be the logit loss, which suggests the use of some variant of logistic regression. While successful models such as Glicko, Chessmetrics and Trueskill all optimize no objective in particular, it seemed to me that since we were given a clear objective to minimize, it would only be sensible to be sure to be optimizing exactly this objective wherever possible. As a result, I made frequent use of logistic regression in my approach, of which this is the first example. The main assumption behind this model is that the "skill" of the jth player at time t is a quadratic function of t. This is obviously a very coarse approximation, and likely results in skill estimates which are overly smooth. Still, it works pretty well:

$$s_i(t) = \alpha_i^{(0)} + t_i \alpha_i^{(1)} + t_i^2 \alpha_i^{(2)} \tag{2}$$

With this skill measure in hand, we may assume that the log-odds of a victory for the white player are given by the difference of the two players skills, and parametrize e_i as:

$$e_i = \frac{\exp_{10}\left(\beta_0 + \beta_1 \left(s_{w_i}(t_i) + s_{b_i}(t_i)\right) + \left(s_{w_i}(t_i) - s_{b_i}(t_i)\right)\right)}{1 + \exp_{10}\left(\beta_0 + \beta_1 \left(s_{w_i}(t_i) + s_{b_i}(t_i)\right) + \left(s_{w_i}(t_i) - s_{b_i}(t_i)\right)\right)}$$

Here, β_0 and β_1 are coefficients which represent the advantage of the white player as a linear function of the average skill of the two players. Adding in regularization penalties $\lambda_0, \lambda_1, \lambda_2$ on $\alpha_i^{(0)}, \alpha_i^{(1)}, \alpha_i^{(2)} : i \in \{1, \ldots, n\}$ yields the

regularized objective:

$$\begin{aligned}
f(\beta,\alpha) = & \sum_{i=1}^{n}\left(y_i \log_{10}\frac{1}{e_i} + (1-y_i)\log_{10}\frac{1}{1-e_i}\right) \\
& + \frac{\lambda_0}{2}\sum_{i=1}^{m}\left\|\alpha^{(0)}\right\|_2^2 + \frac{\lambda_1}{2}\sum_{i=1}^{m}\left\|\alpha^{(1)}\right\|_2^2 + \frac{\lambda_2}{2}\sum_{i=1}^{m}\left\|\alpha^{(2)}\right\|_2^2 \\
= & \sum_{i=1}^{n}\left(\log_{10}\left(1+\exp_{10}\left(\beta_0+\beta_1\left(s_{w_i}(t_i)+s_{b_i}(t_i)\right)+\left(s_{w_i}(t_i)-s_{b_i}(t_i)\right)\right)\right)\right. \\
& \left. -y_i\left(\beta_0+\beta_1\left(s_{w_i}(t_i)+s_{b_i}(t_i)\right)+\left(s_{w_i}(t_i)-s_{b_i}(t_i)\right)\right)\right) \\
& + \frac{\lambda_0}{2}\sum_{i=1}^{m}\left\|\alpha^{(0)}\right\|_2^2 + \frac{\lambda_1}{2}\sum_{i=1}^{m}\left\|\alpha^{(1)}\right\|_2^2 + \frac{\lambda_2}{2}\sum_{i=1}^{m}\left\|\alpha^{(2)}\right\|_2^2
\end{aligned}$$

This is pretty horrendous-looking, but its derivatives are quite simple:

$$\begin{aligned}
\frac{\partial}{\partial\beta_0}f(\beta,\alpha) &= \sum_{i=1}^{n}(e_i-y_i) \\
\frac{\partial}{\partial\beta_1}f(\beta,\alpha) &= \sum_{i=1}^{n}\left(s_{w_i}(t_i)+s_{b_i}(t_i)\right)(e_i-y_i) \\
\frac{\partial}{\partial\alpha_j^{(0)}}f(\beta,\alpha) &= \sum_{i=1}^{n}\left(\beta_1\left(\delta_{w_i=j}+\delta_{b_i=j}\right)+\left(\delta_{w_i=j}-\delta_{b_i=j}\right)\right)(e_i-y_i)+\lambda_0\alpha_j^{(0)} \\
\frac{\partial}{\partial\alpha_j^{(1)}}f(\beta,\alpha) &= \sum_{i=1}^{n}\left(\beta_1\left(\delta_{w_i=j}+\delta_{b_i=j}\right)+\left(\delta_{w_i=j}-\delta_{b_i=j}\right)\right)t_i(e_i-y_i)+\lambda_1\alpha_j^{(1)} \\
\frac{\partial}{\partial\alpha_j^{(2)}}f(\beta,\alpha) &= \sum_{i=1}^{n}\left(\beta_1\left(\delta_{w_i=j}+\delta_{b_i=j}\right)+\left(\delta_{w_i=j}-\delta_{b_i=j}\right)\right)t_i^2(e_i-y_i)+\lambda_2\alpha_j^{(2)}
\end{aligned}$$

Here, δ is the Kronecker delta function. I optimized this objective using the conjugate gradient algorithm, yielding the values of the coefficients β_0, β_1 and $\alpha_i^{(0)}, \alpha_i^{(1)}, \alpha_i^{(2)} : i \in \{1,\ldots,n\}$, permitting the skill of each player, at each time, to be found by evaluating equation 2. Models of this same basic form will be used repeatedly in the remainder of this writeup (but will be described in less detail). The values of the regularization penalties are $\lambda_0 = 0.5, \lambda_1 = 8, \lambda_2 = 4$, and may be found in the ".conf" files in my implementation.

I actually made a few minor tweaks to this model in the final version. For one thing, I trained not just on the primary training data, but also on the secondary and tertiary data. In these two additional data sources, it is unknown which players were white, and which were black, so the β_0 and β_1 terms (representing the advantage of the white player) were not included. Also, in order to reduce overfitting, $y_i = 1$ were set to $y_i = 0.98$, and $y_i = 0$ to $y_i = 0.02$ (which were found empirically to work best)–the intuition behind this is that there should be a little bit of an aspect of "you are who you play"–someone who loses to very good players is likely to be at least a pretty good player himself.

2.2 Glicko

My implementation of Glicko is identical to that described on Mark Glickman's web page ([2]). As I did for the logistic regression-based model, I trained the Glicko model using all three of the primary, secondary and tertiary training datasets. For the first of these, I modified Glicko very slightly to give a small (25 point) advantage to white, by replacing replacing the following expression for the expected game outcome (from the Glicko web page):

$$E(s \mid r, r_j, RD_j) = \left(1+10^{-g(RD_j)(r-r_j)/400}\right)^{-1}$$

with:

$$E\left(s \mid r, r_j, RD_j\right) = \left(1 + 10^{-g(RD_j)(25+r-r_j)/400}\right)^{-1}$$

assuming that player j is the black player (otherwise, the constant is -25). As Jeff Sonas suggested in his forum post, the constant c, which controls how quickly our uncertainty in the skill of an idle player grows, was set to 15.8.

2.3 Blending

One concern which I had, once I settled on my "multi-tier" approach of first attempting to learn player skills based only on some simple models, then blending them, and finally adjusting these predictions based on future scheduling (section 3) was the risk of "using up" the training data. If I used the entire dataset, future and past alike, to predict every game, then the predictions, on the training data, would naturally depend on the entirety of the training data. On what independent data source, then, would I train the blended and future scheduling predictor?

My solution was to incrementally train both the logistic regression and Glicko models based only on the past. More precisely, for each month t, I trained a version of each of these models using all months $< t$, and then saved their predictions (along with some additional quantities, such as the Glicko deviation) on month t. The complete set of these predictions were then used, along with the known game outcomes, to train a logistic regression blend with the objective of equation 1 plus a regularization term $\frac{\lambda}{2}\left\|w\right\|_2^2$, where:

$$e_i = \frac{\exp_{10}\left(b + w^T x_i\right)}{1 + \exp_{10}\left(b + w^T x_i\right)}$$

where x_i is a feature vector for the ith game (the features themselves will be listed in section 2.4). Once more, this model was trained using the conjugate gradient algorithm. For this model, I trained using $\lambda = 1$, and only used the last two years of the primary training data, and only on those games for which both players had played at least 12 games in the previous two years. Also, similarly to the logistic regression skill model, I decreased $y_i = 1$ to $y_i = 0.99$ and increased $y_i = 0$ to $y_i = 0.01$. Once more, these choices were based on empirical performance on the validation set, and have no other justification.

The final output of this model is the value of $b + w^T x_i$ for each i in the training and testing sets. While the logistic regression skill model and Glicko model were both trained using only the past, the coefficients chosen by this blending procedure depend on the entirety of the training set, and therefore are not "independent" of any of the training data. I chose to ignore this, and use the output of this blending model on the training data in order to train the future scheduling predictor to be described in section 3. My justification for this decision was that while the optimal blending coefficients (contained in the bias b and weight vector w) may depend on all of the training data, the quantities which it is blending (the feature vectors x_i) do not, and therefore that the dependency introduced between each training example, and its blended prediction, would be weak enough that it could still be used for training.

2.4 Features

The features x_i used for the ith game in the logistic blending model were all derived from the output of the logistic regression skill model of section 2.1, the Glicko model of section 2.2, and some general statistics for the players participating in the game. They are:

- The skills found by the logistic skill model for both players
- The skills found by the Glicko model for both players
- The deviations found by the Glicko model for both players

- The product of the Glicko deviances with each of the two logistic skills
- The product of the Glicko deviances with each of the two Glicko skills
- The Glicko deviances, times -1 if the black player's logistic skill is higher than the white player's
- The product of the Glicko deviances with each of the two logistic skills, times -1 if the black player's logistic skill is higher than the white player's
- The product of the Glicko deviances with each of the two Glicko skills, times -1 if the black player's Glicko skill is higher than the white player's
- The average logistic skill of the white player's opponents over the past 24, 12, 6 and 3 months
- The average logistic skill of the black player's opponents over the past 24, 12, 6 and 3 months
- The proportion of wins by the white player over the past 24, 12, 6 and 3 months
- The proportion of wins by the black player over the past 24, 12, 6 and 3 months
- The average logistic skill of the white player's opponents over the past 24, 12, 6 and 3 months, times $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, respectively, where n_k is the number of games played by the white player over the past k months
- The average logistic skill of the black player's opponents over the past 24, 12, 6 and 3 months, times $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, respectively, where n_k is the number of games played by the black player over the past k months
- The proportion of wins by the white player over the past 24, 12, 6 and 3 months, times $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, respectively, where n_k is the number of games played by the white player over the past k months
- The proportion of wins by the black player over the past 24, 12, 6 and 3 months, times $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, respectively, where n_k is the number of games played by the black player over the past k months
- $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, where n_k is the number of games played by the white player over the past k months
- $\frac{1}{128}\max(n_{24}, 128)$, $\frac{1}{64}\max(n_{64}, 64)$, $\frac{1}{32}\max(n_{32}, 32)$ and $\frac{1}{16}\max(n_{16}, 16)$, where n_k is the number of games played by the black player over the past k months

3 Future scheduling

3.1 Mixture model

The main idea behind my technique for performing future scheduling was to break the problem into two stages, the first of which determines whether future scheduling should be performed on a particular game, and the second actually performing it. In order to accomplish this, I used something akin to a mixture model, in that both of these subproblems were solved in parallel. Once more, I used what is essentially logistic regression, with the regularized objective:

$$\begin{aligned} f(b, w, a_1, v_1, a_2, v_2) = &\sum_{i=1}^{n} \left(y_i \log_{10} \left(p(b, w; x_i)\, q_1\left(a_1, v_1; y_i^{(1)}\right) + (1 - p(b, w; x_i))\, q_2\left(a_2, v_2; y_i^{(2)}\right)\right)\right. \\ &\left. + (1 - y_i) \log_{10} \left(p(b, w; x_i) \left(1 - q_1\left(a_1, v_1; y_i^{(1)}\right)\right) + (1 - p(b, w; x_i)) \left(1 - q_2\left(a_2, v_2; y_i^{(2)}\right)\right)\right)\right) \\ &+ \frac{\lambda}{2}\|w\|_2^2 + \frac{\gamma_1}{2}\|v_1\|_2^2 + \frac{\gamma_2}{2}\|v_2\|_2^2 \end{aligned}$$

here, $p(b, w; x_i)$ is the probability that mixture component 1 should be used, and is based on the feature vector x_i for game i containing features which I believed would be useful in determining whether future scheduling was worth performing, as well as coefficients b and w. Similarly, $q_1\left(a_1, v_1; y_i^{(1)}\right)$ and $q_2\left(a_2, v_2; y_i^{(2)}\right)$ are the predictions of the two mixture components, each of which is based on different feature vectors. I experimented with the choice of these features, but found that the best performance was achieved when $y_i^{(1)}$ was left as only the output of the blended predictor of section 2, while $y_i^{(2)}$ contained these predictors, along with a number of features meant to capture the structure of any tournament in which the participants of the ith game may have participated. These functions were taken to have the form:

$$p(b, w; x_i) = \frac{10^{b+w^T x_i}}{1 + 10^{b+w^T x_i}}$$

$$q_1\left(a_1, v_1; y_i^{(1)}\right) = \frac{10^{a_1 + v_1^T y_i^{(1)}}}{1 + 10^{a_1 + v_1^T y_i^{(1)}}}$$

$$q_2\left(a_2, v_2; y_i^{(2)}\right) = \frac{10^{a_2 + v_2^T y_i^{(2)}}}{1 + 10^{a_2 + v_2^T y_i^{(2)}}}$$

As before, the objective includes regularization penalties $\lambda, \gamma_1, \gamma_2$ to the weights w, v_1, v_2, but not to the biases b, a_1, a_2, with $\lambda = 16$, $\gamma_1 = 1$ and $\gamma_2 = 64$ (the blended skill features in $y_i^{(2)}$, but not the others, were scaled by a factor of 8 in order to compensate for this large regularization penalty). As was the case in section 2.1, the derivatives of this objective with respect to the biases and weights are fairly simple. Optimization was performed by performing a fixed number of conjugate gradient iterations to optimize b, w, then a_1, v_1, then a_2, v_2, and finally repeating. As was the case for the blending model of section 2.3, I decreased $y_i = 1$ to $y_i = 0.99$, increased $y_i = 0$ to $y_i = 0.01$, and trained only on those games in the last two years of the primary training set for which both players had played at least 12 games in the previous two years.

3.2 Features

I discussed the three different feature vectors, $x_i, y_i^{(1)}, y_i^{(2)}$ used by the future scheduling predictor. These vectors are each composed of "subsets" of features, which are conceptually related. These are the output of the blended model of section 2.1 (included in all three feature vectors), "indicator" features designed to determine whether future scheduling should be performed (included in x_i and $y_i^{(2)}$), and "tournament" features designed to use future scheduling to improve the predictions (included only in $y_i^{(2)}$).

3.2.1 "Indicator" features

Many of these features are of a number of "soft indicators" for whether a certain quantity has at least a certain value. What I mean by a "soft indicator" is this: suppose that the white player has played 14 games this month, and we have soft indicators for whether the player has played at least 4, 8, 16 or 32 games. Then, the soft indicator for having played at least 4 or 8 games will be 1, that for having played at least 32 will be zero, and that for 16 will be $\frac{14-8}{16-8} = \frac{3}{4}$.

The features included in this feature set are:

- Soft indicators for whether the white player has played at least 2, 3, 4, 6, 8, 12, 16 or 32 games this month
- Soft indicators for whether the black player has played at least 2, 3, 4, 6, 8, 12, 16 or 32 games this month
- Soft indicators for whether the white player's opponents this month have played an average of at least 2, 3, 4, 6, 8, 12, 16 or 32 games

- Soft indicators for whether the black player's opponents this month have played an average of at least 2, 3, 4, 6, 8, 12, 16 or 32 games
- Soft indicators for whether the white and black player have played each other at least 2, 3, 4, 6 or 8 times this month

Also included are some "neighborhood" features, which attempt to capture the broader tournament structure surrounding each player. Each of these features has an associated depth d. The depth-d tournament feature is calculated by, starting from the white player, and *ignoring all games played between the white and black player this month*, finding all of the white players opponents this month, all of their opponents, and so on, out to depth d. These will be called the "depth-d opponents" of the white player. Then, the same is done for the black player. If the two players participated in the same tournament, then one would expect the amount of overlap to be large. Hence, the proportion of the depth-d opponents of the white player who are also depth-d opponents of the black player is found, and saved as a feature. The same is done for the proportion of the depth-d opponents of the black player who are also depth-d opponents of the white player. The complete set of these features are:

- Depth 1, 2, 3 and 4 neighborhood features

3.2.2 "Tournament" features

As with the "neighborhood" features, each "tournament" feature has an associated depth d. First, for each game d, we have a approximately linear measure of the difference in skill between the two players, as predicted by the blended predictor of section 2 ($b + w^T x_i$), and will define the "relative skill" of the white player to be half this quantity, and of the black player to be $-\frac{1}{2}$ times this quantity. For, say, depth 3 tournament features of game i, we would find all of the white player's opponents opponents opponents this month, and find the relative skills of these players compared to the white players opponents opponents. The same would then be done for the black player, and various percentiles of these sets of relative skills outputted. Actually, two sets of percentiles are outputted, one of which includes games which are repeated in the list of relative skills, and the other of which does not (I call the latter "reduced percentiles"). The idea here is just what was described in the overview section: the quality of the white player's opponents indicates how well this player is performing this month, the quality of *their* opponents indicates how well they are performing, and so on. The features included in this feature set are:

- 25.1th, 50th and 74.9th percentiles and reduced percentiles at depths 1, 2, 3 and 4

4 Implementation

Source code can be found in the implementation directory, although it was all coded in a rush, and is therefore very difficult to read (there's quite a bit of cut-and-paste code, for example). Most of the programs take a ".conf" file as a command-line parameter, which includes the parameters to the various models, and also input and output files for each. There are three configuration files: one for the validation set, one for the testing set, and one for the followup dataset, and the best reference for the parameters which I used in my submission is these files. One should run all of the programs in sequence (the order in which they are listed below will work), because many of the programs require the output of earlier programs as input–for example, many of the models depend, directly or indirectly, on the logistic regression skill model of section 2.1 (which is created by the "model_logistic_skill" program).

- partition_training_data - creates a validation set (and a training set with the validation set removed) from the training data
- model_glicko - trains the Glicko model (section 2.2)
- model_logistic_skill - trains the logistic regression skill model (section 2.1)

- model_combined_skill - creates features from the Glicko and logistic skill models (section 2.4)
- model_logistic_blend - creates a blended predictor (section 2.3)
- model_tournament_indicator - creates the "soft indicator" features (section 3.2.1)
- model_neighbor - creates the "neighborhood" features (section 3.2.1), as well as some alternative tournament features which I ultimately found to not be of use
- model_tournament_skill - creates the "tournament" features (section 3.2.2)
- model_logistic_tournament - trains the future scheduling model (section 3.1)

All of these files also require datasets on which to run, which are not included. Please follow the instructions in the README file in the "sample_datasets" directory to see how to include the data files correctly. The output of these programs will be placed in the "sample_output" directory.

References

[1] BellKor's Pragmatic Chaos Netflix Prize Solution. http://www.netflixprize.com//community/viewtopic.php?id=1537.

[2] Mark E. Glickman. The Glicko System. http://www.glicko.net/glicko/glicko.doc/glicko.html.

[3] Ralf Herbrich and Thore Graepel. "TrueSkill: A Bayesian Skill Rating System". MSR-TR-2006-80. 2006.

[4] Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. "TrueSkill Through Time: Revisiting the History of Chess". Advances in Neural Information Processing Systems 20. MIT Press. 2008.