Hardware

An amusing thing to note is that all the submissions made by team Planet-Thanet in this competition where achieved on a barebones machine costing GBP 238.96. To be specific this was a1 x BB-6404GA Novatech Barebone Bundle - AMD II X4 640 Quad Core - 4GB DDR3 1333Mhz - AMD 760G Motherboard With Onboard ATI Radeon 3000- 4 Bay ATX Tower Case and PSU. 1 x WD-10EARS Western Digital Caviar Green Power 1TB 64MB Cache Hard Disk Drive SATAII 300MB/s . The machine was running Linux Mint and all the code was written in java.

The Solution

At the hub of the solution is a model where each player i is assigned a single strength x_i and the expected outcome when player i as white meets player j as black is simply:

$$0.5 + 0.5 * tanh(x_i - x_j + 0.05) \tag{1}$$

The 0.05 is for white advantage. This model is trained using stochastic descent. We iterate through all the games chronologically (within a month we randomly ordered the games in order not to introduce a bias according to the player number ordering) and for each game we make an adjustment to the strength of the two contestants. If the outcome of game n is o_n and the prediction from the current estimates of the player strengths is p_n then we would update the strength of the white player by $(o_n - p_n) * learningrate$.

We iterate through the dates many times until we have converged to a solution (in practice 30 iterations was sufficient). For the learning rate we used 0.0449. In order to avoid overfitting we used also decayed all the strengths towards zero at the beginning of each iteration. The decay factor we used was 0.9912.

Once we had our update rule and predict formula it was easy to make some pragmatic changes of things that seemed to improve the results, by cross-validation. For instance in the update rule it proved useful to update the score of each player slightly just by virtue of the fact that they played at all. This played at all bonus was $4e^{-3}$.

We also updated the score of the player who outperformed relative to the prediction, pulling their weight towards the weight of the other player. It is

not clear (as in we don't remember) how much value this term added. In terms of the prediction formula, we experimented with the tanh functional form. We saw some improvement using a combination of two tanh functions and in the end hit on $0.5 + 0.515 * (1.1 * tanh(x) - 0.1 * tanh(2x))$.

Despite this formula being more messy we did not find the need to change our simple update rule, which makes sense since the second term can be considered a perturbation on the main sigmoid term.

Player strengths vary over time, but more importantly the goal of the competition was to predict results at the end of the time period. We weighted the observations in the stochastic descent as follows (the numbers being found by cross-validation). Up to month 90 we used a weight of 0.5, from that pointup to month 117 we used a weight of 0.7, then 0.9 after that.

In terms of the predict formula we found it beneficial to add (to the probability) a factor dependent on how many games white played in the month and subtract the same term for how many games black had played. We are rather ashamed to admit this was a pure look up in a list found by fitting the term to all the training data. Hey ho! Note that the probability can move outside the zero-one bound. We only truncated to the zero-one range when we made our submission.

So far we had only used the primary data set. We got some value from the secondary data set (the set for which who played white is unknown) but the value was limited and we used these games with a quarter of the weight that we applied to the primary games. It should be noted that already at this point we were using future schedule information since we were using as a factor the number of games a player played each month. It was at this point that we were struck by an evil idea. We could not only use this term in prediction, we could feed it back into the calibration of the factors themselves. From this point on we did this with any future schedule information with significant benefit (in fact we benefitted by feeding back in an amplified term and curiously this even seem to assist in regularizing the solution and preventing over-ftting). We were cognisant that our day of reckoning would arrive when we had to justify our actions! Jeff had however inadvertently given carte blanche to this approach in a related post on the forum. Having thus succumbed to the dark side, we wondered what else we could find in the treasure trove of the future schedule and we hit upon the now well-documented swiss-event hack. This hack stems from the observation that in Swiss style events, winners tend to play winners and losers play losers. Hence if you see a player playing stronger players than normal in a month, it

probably means that he has won games that month, and vice versa. We did this work in probability space. For a given game prediction we considered all the games that white played that month and all the games black played that month. For both these lists we determined a mid probability. The best functional form we had for this was a kernel weighted figure. We looked at the index of each game in the list and the distance of that index from the mid-index. If a player played n games and this game was at index i in the sorted probability list then the distance from the centre of the list was defined as $w_i = i + 0.5 - 0.5 * n$. The weighted mid probability was defined as

$$\frac{\sum_i e^{-0.5w_i^2} p_i}{\sum_i e^{-0.5w_i^2}}$$

We then reverted each prediction to the distance of this mid value from fair value, for which we found the best figure by cross-validation to be 0.58. We also threw in a more powerful cubic reversion term (simply because it worked – hey we don't claim to be experts!) and a $n/(n + sigma)$ term in order to damp this term when there were insufficient observations for it to make much sense. Funnily enough we found that this correction term worked best when it was multiplied by the following formula of the predicted score p, namely $4p(1 - p)$.

Having seen the value of the indicators so far (namely the number of games white and black had played in a month and the mean probability of white and blacks games in a given month) we decided, what the heck, maybe there is some mysterious non-linear value in these indicators that we have not mined so far. Consequently we split our validation set into two and trained a neural network on 6 inputs. The net topology was 6x20x5x1 and the inputs were $x_i - x_j$, the current prediction p_i,, the mean probability of all white/black games this month and $e^{-games/3}$ for that month for white and black. Sadly the java neural net package we used did not allow weight decay, but nevertheless after running it enough times to get a solution that was not overfit this gave us a huge improvement which we found surprising.

Pushing the future schedule idea even further there are times when you can really mine the data. For instance suppose we look at a game between 2 players and for white this is the strongest of all his opponents this month (of say 10 opponents) and for black it is the weakest opponent. You can be quite certain (unless it was their last game in the month) that this was a win to black since any other result would likely have lead to B playing another

weaker player and A another stronger player. We found a good indicator here was to once more line up the player probability scores for the month and use a very severely peaked function of how far this game is from the edges for both players (the fifth power). We also had a lot of success with a term dependent on how far the probability of the game was from the position in the list. For instance if the probability of player A winning this game was 0.7 and this was 0.8 way through the sorted list of probabilities we would revert according to the difference (0.1). All these factors were modified by a term to adjust for when there were too few games and had arbitary linear transformations where the coefficients were determined by cross-validation.

Finally we threw in a whole bunch of player by player regressions on the following features: predicted score (some players have wider variances), average age of games for player (how old is the data for this player), how many games has a player played this month. We also included a prediction for players who had met before, using their previous results.

Things That Did Not Work

TBD – Paper Scissor Stones Idea (you know the situation where player A always seems to beat player B who beats player C who beats player A – cyclical rating patterns). Lots more to come here!