1 S6 ratings calculation: Technical document

1.1 Overview

Season 6 uses as an evaluative metric a kind of "Elo" rating. The typical idea of such ratings is to attempt to predict the outcome of future games from the results of past games by assigning to each player a single number (which is supposed to be a coarse representation of that player's skill).

The core assumption of our ratings model, known as a *Bradley-Terry* model, is that in a game between players A, B with ratings E_A, E_B, the probability that player A wins is given by

$$P(A \text{ beats } B) = \frac{1}{1 + 10^{(E_B - E_A)/400}} \tag{1}$$

The method we use for computing the ratings of players is Bayesian. For each player, we begin with some prior belief about the possible values of that player's rating; in our specific case, the belief is that the rating is normally distributed with some fixed standard deviation σ, and a mean μ that depends on the player (and is determined by some mix of season 5 performance and current Cadence PB).

We then add the data of a set of games G (the games that will be played during season 6). **The set of ratings we assign to players is the set of ratings that maximizes the probability of obtaining the given game results according to our prior beliefs and the equation (1).**

To find these maximum-probability ratings E, we use Bayes's formula, which says

$$P(E|G) \propto P(G|E)P(E).$$

Here $P(E|G)$ is the "posterior" probability of obtaining the ratings E given the results G; $P(G|E)$ is the probability of obtaining the results G given the ratings E; and $P(E)$ is the "prior" probability of the ratings E.

$P(G|E)$ is explicitly calculable from equation (1) (it's simply the product of one term of the form of equation (1) for every game in our data set). So we are left with the mathematical problem of finding the ratings that maximize $P(G|E)P(E)$. This is what the rating code attempts to do.

1.2 Mathematical Details

Instead of the ratings E, we work with ratings γ defined by

$$\gamma = 10^{E/400}.$$

We wish to maximize

$$\log\big(P(\gamma|G)\big) = \log\big(P(G|\gamma)\big) + \log\big(P(\gamma)\big) + \text{const.} \tag{2}$$

We will denote the right-hand side of this equation by $\ell(\gamma)$ after dropping some annoying constants. Let the players be indexed from 1 to n, and let γ_i be the rating of the i-th player. The probability that player i beats player j is given by

$$P(i \text{ beats } j) = \frac{\gamma_i}{\gamma_i + \gamma_j}.$$

Let w_{ij} denote the number of times player i has beaten player j in the set of games G (w_{ii} is zero by convention), and let $N(x; \mu, \sigma)$ be the standard normal distribution. Our prior $P(E)$ is such that each player's rating is normally distributed with some mean ν_i. We let $\mu_i = 400\nu_i / \log(10)$ (this converts ν_i to "natural" units).

Expanding (2) gives

$$\begin{aligned}\ell(\gamma) &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \log\left(\frac{\gamma_i}{\gamma_i + \gamma_j}\right) + \sum_{i=1}^{n} \log\left(N(\log\gamma_i; \mu_i, \sigma)\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \log\left(\frac{\gamma_i}{\gamma_i + \gamma_j}\right) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\log\gamma_i - \mu_i)^2\end{aligned}$$

In the second equation, we have again dropped a constant; we take this final expression as the official definition of ℓ. Maximizing this expression suffices to maximize $P(\gamma|G)$.

Elementary calculus shows that $x \log x - x \geq -1$ for $x > 0$, with equality only if $x = 1$. Substitute y/x for x and rearrange to get

$$-\log x \geq 1 - \log y - \frac{x}{y} \qquad x > 0, y > 0. \tag{3}$$

Given a set of ratings $\lambda \in \mathbb{R}^n_+$ define

$$Q_\lambda(\gamma) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\Big[\log\gamma_i - \frac{\gamma_i + \gamma_j}{\lambda_i + \lambda_j} - \log(\lambda_i + \lambda_j) + 1\Big] - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\log\gamma_i - \mu_i)^2$$

From (3) (with $y = \gamma_i + \gamma_j$ and $x = \lambda_i + \lambda_j$) it follows that

$$Q_\lambda(\gamma) \leq \ell(\gamma).$$

In particular, if γ is a maximizer for Q_λ, then

$$\ell(\lambda) = Q_\lambda(\lambda) \leq Q_\lambda(\gamma) \leq \ell(\gamma). \tag{4}$$

This suggests an iterative procedure for attempting to find a maximizer for ℓ. Namely, we begin with a candidate maximizer $\gamma^{(1)}$, and given a candidate maximizer $\gamma^{(k)}$, we let $\gamma^{(k+1)}$ be a maximizer for $Q_{\gamma^{(k)}}$.

The upshot of this method is that Q_λ is a sum of functions of the form $q_{i,\lambda}(\gamma_i)$, and so we may attempt to maximize each of these functions individually. Explicitly, we define

$$q_{i,\lambda}(\gamma_i) = \sum_{j=1}\Big[w_{ij}\log\gamma_i - (w_{ij} + w_{ji})\frac{\gamma_i}{\lambda_i + \lambda_j}\Big] - \frac{1}{2\sigma^2}(\log\gamma_i - \mu_i)^2,$$

so that $Q_\lambda = \sum q_{i,\lambda} + \text{const.}$

Maximizing $q_{i,\lambda}$ is an exercise in calculus. Define

$$A = \sum_{j=1}^{n} w_{ij} \qquad B = \sum_{j=1}^{n} \frac{w_{ij} + w_{ji}}{\lambda_i + \lambda_j} \qquad C = \frac{1}{\sigma^2},$$

and expand out

$$\begin{aligned} q_{i,\lambda}(x) &= A \log x - Bx - \tfrac{C}{2}(\log x - \mu_i)^2 \\ &= -\tfrac{C}{2}\log^2 x + (A + \mu_i C)\log x - Bx \end{aligned}$$

Taking the derivative and setting it equal to zero, one obtains (after some algebra)

$$\big(\tfrac{B}{C}x\big)e^{\left(\frac{B}{C}x\right)} = \tfrac{B}{C}e^{\frac{A+\mu_i C}{C}}.$$

Letting W be the Lambert W-function (the inverse of xe^x) we find

$$x = \frac{C}{B} W\Big(\frac{B}{C}e^{\frac{A+\mu_i C}{C}}\Big) \tag{5}$$

Note that the argument of W here is positive, so W is single-valued and real.

This, then, gives the final iterative algorithm: we begin with some ratings vector $\gamma^{(1)}$ (which we choose to be given by $\gamma_i = \mu_i$), and we iteratively define $\gamma_i^{(k)}$ by the formula (5). Equation (4) guarantees that the sequence $\ell(\gamma^{(k)})$ is monotonically increasing.

The implementation of this process is done in python with the mpmath module. In the implementation, we test convergence by inspecting the gradient of $\ell(\gamma)$ and the change in γ, stopping the iteration when both are sufficiently small (less than 10^{-12}).