Elo-R Draft

January 3, 2016

1 Introduction

Competition, in the form of sports, tests and games, have always been a part of our history. Whether the aim is pure entertainment, skill improvement, or selection of the best people for a given role, contestants and spectators alike are interested in estimating the relative skills of contestants. Various competition formats, such as tournaments, exist to declare a champion. However, we may be interested in comparing players who have never played directly against one another. Furthermore, there is a great deal of variance in a player's performance, and we may be interested in an aggregate estimate of skill that's based on a player's entire history of games.

Skills are easiest to compare when there is a standard quantifiable objective, such as a score on a standardized test or a finishing time in a race. However, most team sports, as well as games such as chess, have no such measure. Instead, a good player is simply one who can win against other good players. The Elo rating system sets out to assign a quantitative measure of skill in such a scenario. These ratings have little objective meaning, but can be used to rank players relative to one another, as well as forecast the probable outcome of future matches.

To make things concrete, let's label the players $1, \ldots, N$. The original Elo system deals with 1v1 zero-sum games with binary (win/lose) outcomes. Player i has a rating r_i, a real number that should be higher the more skilled they are. After two players compete, the loser gives some number of points to the winner, so that the sum of ratings is conserved. The number of points transferred depends on how "surprising" the outcome was. To determine the level of surprise, we need some model of the likelihood of match outcomes between players of known ratings. Typically, the logistic function is used to model a probability that decays exponentially with the rating difference between the losing and winning player.

Now we describe the setting for the new Elo-R system. Instead of 1v1 matches with win/lose outcomes, we assume a large number of players compete simultaneously at the task, and the outcome is a ranking: a total order among all players in the match.

To model how the outcome came about, suppose player i's true (unknown) skill is s_i. Player i's performance in the match can be viewed as a number p_i, drawn from a logistic distribution centered at s_i. If i beat j in the match rankings, a result we denote as $i \succ j$, then we know that $p_i > p_j$. While we can't measure s_i directly, it's possible to get a fairly accurate reading on p_i by looking at i's position in the ranking. This suggests the following two-phase approach: from the match ranking, estimate p_i for each player. Then, update the rating r_i to a weighted average of past p_is. As we'll soon see, less weight will be put on performances that are either outdated or unusual for the player.

2 Performance Averaging

Let's focus on the second phase for now: having computed the latest p_i, we want to combine it with past performances of the same player to get a more precise estimate of their skill. To clarify the notation in this section, we add an extra subscript k, ranging from 1 to the number of matches M in which player i has competed, so that $p_{i,k}$ denotes the performance of player i on their kth match, and $r_{i,k}$ denotes their rating upon completion of the kth match. The sample mean is perhaps the most obvious statistic to take. That is, let

$$r_{i,M} = \frac{1}{M} \sum_k p_{i,k} = \left(1 - \frac{1}{M}\right) r_{i,M-1} + \frac{1}{M} p_{i,M} \tag{1}$$

In addition to being simple, this method has the advantage of being a function of only the number of matches M, the previous rating $r_{i,M-1}$ and the new performance $p_{i,M}$, so there is no need to remember all of the $p_{i,k}$. We say such a method is *memoryless.*

However, there are two key problems here. The first is that, as a player accumulates more experience, the relative weight of a new performance approaches 0; if an established player improves their skill, it will take a long time for their rating to reflect it. This is because all performances are weighted equally, even those which are too outdated to be informative about a player's current skill. It corresponds to a model in which a player's true skill is an eternal constant, never changing, and each $p_{i,k}$ is a measurement of this constant. To properly account for a player's lifelong evolution, we should put less weight on old performances.

The second problem with the sample mean is lack of robustness against outliers. A single extreme performance, perhaps due to illness or a power failure if the competition takes place over a web server, will drastically affect a player's rating. Thus, we wish to reduce the weight of outliers.

Before we proceed, note that the above equation can be rewritten as

$$\sum_k w_k (p_{i,k} - r_{i,M}) = 0 \tag{2}$$

where $w_k = 1$ for all k. We say that $r_{i,M}$ is a *weighted mean* of the elements $\{p_{i,k}\}_{k=1}^M$ with corresponding weights $\{w_k\}_{k=1}^M$. As a simplifying assumption, let's imagine the $p_{i,k}$ are independent measurements of the present true skill $s_{i,k}$, with Gaussian (normal) noise variances of τ_k^2. If we treat $p_{i,1}$ as the prior, then our posterior belief on the true skill is a normal distribution centered at the weighted mean $r_{i,M}$ given by the following equation:

$$\sum_k \frac{1}{\tau_k^2} (p_{i,k} - r_{i,M}) = 0 \tag{3}$$

All we did here is set $w_k = 1/\tau_k^2$. If the w_ks (or equivalently, the τ_ks) form a geometric sequence, we maintain the memoryless property. We'll derive the weights later; for now, it's simplest to imagine them decreasing by a constant ratio for each match backward in time.

In order to bound the effect of any individual outlier, we also require that $w_k |p_{i,k}|$ approach a positive constant as $p_{i,k}$ approaches $\pm\infty$. We still want to treat non-outliers in essentially the same way as before, so their weight should be close to $1/\tau_k^2$ for performances close to the weighted mean, which is $r_{i,M}$ by definition. To meet these criteria, we opt for

$$w_k = \frac{1}{\tau_k (p_{i,k} - r_{i,M})} \tanh \frac{p_{i,k} - r_{i,M}}{\tau_k} \tag{4}$$

This choice will receive additional justification later: essentially, it comes from modeling performances according to the heavy-tailed logistic distribution in place of the Gaussian noise discussed previously. The equation defining $r_{i,M}$ becomes:

$$\sum_k \frac{1}{\tau_k} \tanh \frac{p_{i,k} - r_{i,M}}{\tau_k} = 0 \tag{5}$$

It is monotonic in $r_{i,M}$, so it can be solved using binary search. A straightforward linear approximation shows that in the limit as the performances vary much less than any of the τ_k, the rating behaves like a mean with weights $1/\tau_k^2$. In the limit as the performances vary much more than any of the τ_k, the rating behaves like a median with weights $1/\tau_k$. While this equation is not memoryless, all τ_k corresponding to old performances will be very large and behave like the linear approximation. The sum of these linear terms is itself a linear term where $p_{i,0}$ here is the weighted sum of the old performances, and the new weight $1/\tau_0^2$ is the sum of the old weights:

$$\frac{p_{i,0} - r_{i,M}}{\tau_0^2} + \sum_k \frac{1}{\tau_k} \tanh \frac{p_{i,k} - r_{i,M}}{\tau_k} = 0 \tag{6}$$

As a result, we only need to remember a small number of recent performances. Hence, the number of terms in the sum doesn't grow unboundedly with the size of the history. If we decide to remember zero performances, it reverts to the linear memoryless method. Hence, this general formula encapsulates everything we've covered thus far. We will see that our formula is equivalent to maximizing the product of a collection of normal and/or logistic probability density functions (pdfs). τ_k^2 corresponds roughly to the pdf's variance, although it differs by slightly different constant factors for the normal and the logistic; we choose to stick to this parametrization in order to maintain the equivalence involved in the linear approximation.

3 Approximate Bayesian Derivation

Now we derive our rating algorithm in full, by treating the rating r_i of player i as a summary of our approximate Bayesian belief over their true skill s_i, given all the evidence from past matches. In general, our belief on s_i will be proportional to a product of normal and logistic pdfs. We may think of one of these pdfs as a prior belief, and the others are likelihoods of independent measurements of s_i. Given this belief distribution, the rating r_i is defined as the *maximum a posteriori* (MAP, a.k.a. posterior "mode") estimate of s_i; i.e., the point of maximum density in the belief kernel.

Recall the competition model: each player i performs at level p_i, drawn independently from a logistic distribution centered at s_i. The ranking we observe is a total ordering on the players based on their performances: i outranks j, written $i \succ j$, iff $p_i > p_j$. In order to sidestep the issue of modeling ties as a separate event and determining their probability, we will simply treat them as half a win and half a loss; such division will be very straightforward with the formulas we get.

Fix i, the player whose rating we presently compute. Let e be the evidence consisting of, for each j, whether $i \succ j$ or $i \prec j$. That is, we ignore relations like $j \succ k$ which affect the relative ordering of player pairs that don't include i. Our goal is to approximate the posterior distribution of s_i given e:

$$f(s_i \mid e) \propto f(s_i) \Pr(e \mid s_i) = f(s_i) \int \Pr(e \mid p_i) f(p_i \mid s_i) dp_i \tag{7}$$

Since the integral is hard to evaluate, we take a leap of faith here and treat $\Pr(e \mid p_i)$ as a (possibly scaled) delta-function that spikes at the MAP estimate of p_i. This is justified as $N \to \infty$, because the evidence e would overwhelmingly concentrate p_i into a narrow range. Having fixed p_i, the previous equation simplifies to

$$f(s_i \mid e) \propto f(s_i) f(p_i \mid s_i) \tag{8}$$

Our update algorithm thus divides into two phases: first, it must determine the player's performance p_i in the match. In the second phase, p_i is used to update the belief distribution on s_i, which is then compactly summarized by finding the point r_i that achieves its maximum.

3.1 Performance Estimation

To compute the MAP of p_i, we must maximize

$$f(p_i \mid e) \propto f(p_i) \Pr(e \mid p_i) \tag{9}$$

Clearly, $p_i = r_i + (s_i - r_i) + (p_i - s_i)$. The first term is a known constant; the second is a random variable whose variance is approximately (perhaps a bit more than) that of the prior belief on s_i; the third is a logistic distribution prescribed by the model. If we replace these by normal random variables with the same mean and variance, then their sum is conveniently also normal. We then approximate the sum by a logistic with the same mean and variance, justified by the fact that most of the variance comes from the logistic term. The model I implemented for Codeforces has $\sigma_i = 100$ (by default; see discussion of new players), $\delta = 250$, and $\tau_i^2 = \sigma_i^2 + \delta^2$,

$$f(s_i) \approx \frac{2e^{2(s_i - r_i)/\sigma_i}}{\sigma_i \left(1 + e^{2(s_i - r_i)/\sigma_i}\right)^2} \tag{10}$$

$$f(p_i \mid s_i) = \frac{2e^{2(p_i - s_i)/\delta}}{\delta \left(1 + e^{2(p_i - s_i)/\delta}\right)^2} \tag{11}$$

$$f(p_i) \approx \frac{2e^{2(p_i - r_i)/\tau_i}}{\tau_i \left(1 + e^{2(p_i - r_i)/\tau_i}\right)^2} \tag{12}$$

Under these modeling assumptions,

$$\Pr(e \mid p_i) = \prod_{j \succ i} \Pr(p_j > p_i) \prod_{j \prec i} \Pr(p_j < p_i) \tag{13}$$

$$\approx \prod_{j \succ i} \frac{1}{1 + e^{2(p_i - r_j)/\tau_j}} \prod_{j \prec i} \frac{e^{2(p_i - r_j)/\tau_j}}{1 + e^{2(p_i - r_j)/\tau_j}} \tag{14}$$

$$\propto \frac{e^{2p_i \sum_{j \prec i} 1/\tau_j}}{\prod_{j \neq i} 1 + e^{2(p_i - r_j)/\tau_j}} \tag{15}$$

$$C_1 + \ln f(p_i \mid e) = C_2 + \ln f(p_i) + \ln \Pr(e \mid p_i) \tag{16}$$

$$\approx \ln \frac{2}{\tau_i} + (p_i - r_i)\frac{2}{\tau_i} - 2\ln\left(1 + e^{2(p_i - r_i)/\tau_i}\right) + 2p_i \sum_{j \prec i} \frac{1}{\tau_j} - \sum_{j \neq i} \ln\left(1 + e^{2(p_i - r_j)/\tau_j}\right) \tag{17}$$

To maximize this expression, differentiate it w.r.t. p_i, divide by 2 and set to zero:

$$0 = \frac{1}{\tau_i}\left(1 - \frac{2e^{2(p_i-r_i)/\tau_i}}{1+e^{2(p_i-r_i)/\tau_i}}\right) + \sum_{j\neq i}\frac{1}{\tau_j}\left(\mathbb{1}(j \prec i) - \frac{e^{2(p_i-r_j)/\tau_j}}{1+e^{2(p_i-r_j)/\tau_j}}\right) \tag{18}$$

$$= \sum_{j\preceq i}\frac{1}{\tau_j}\left(\frac{1}{1+e^{2(p_i-r_j)/\tau_j}}\right) - \sum_{j\succeq i}\frac{1}{\tau_j}\left(\frac{e^{2(p_i-r_j)/\tau_j}}{1+e^{2(p_i-r_j)/\tau_j}}\right) \tag{19}$$

The terms in parentheses can be thought of as a measure of surprise at the outcomes between i and j: they are the probability of the outcomes opposite to what actually occurred, when the performance of player i is fixed to p_i, with the insertion of two artificial outcomes in which player i wins once and loses once against clones of itself. If the τ_js were all equal, this equation is equivalent to finding the performance level p_i at which one's expected rank would match player i's actual rank, clones included. Another equivalent presentation, achieved by rearranging terms, is as follows:

$$\frac{2}{\tau_i}\tanh\frac{p_i-r_i}{\tau_i} + \sum_{j\neq i}\frac{1}{\tau_j}\tanh\frac{p_i-r_j}{\tau_j} = \sum_{j\prec i}\frac{1}{\tau_j} - \sum_{j\succ i}\frac{1}{\tau_j} \tag{20}$$

Use binary search to solve for p_i. This defines the *performance* of player i in the match.

3.2 Posterior Maximization

Recall our approximation $f(s_i \mid e) \propto f(s_i)f(p_i \mid s_i)$: the posterior $f(s_i \mid e)$ takes the prior $f(s_i)$ and multiplies it by a new logistic pdf $f(p_i \mid s_i)$. The general form for our posterior will be proportional to a product of normal and logistic pdfs. Since a product of normals is another normal kernel, wlog we assume there is only one normal in the product:

$$f(s_i \mid e) \propto e^{-(s_i-\mu_0)^2/\tau_0^2}\prod_k \frac{e^{2(s_i-\mu_k)/\tau_k}}{\left(1+e^{2(s_i-\mu_k)/\tau_k}\right)^2} \tag{21}$$

$$C + \ln f(s_i \mid e) = -\frac{(s_i-\mu_0)^2}{\tau_0^2} + 2\sum_k\left(\frac{s_i-\mu_k}{\tau_k} - \ln(1+e^{2(s_i-\mu_k)/\tau_k})\right) \tag{22}$$

To maximize this expression, differentiate it w.r.t. s_i, divide by 2 and set to zero:

$$0 = \frac{\mu_0 - s_i}{\tau_0^2} + \sum_k \frac{1}{\tau_k}\left(1 - \frac{2e^{2(s_i-\mu_k)/\tau_k}}{1+e^{2(s_i-\mu_k)/\tau_k}}\right) \tag{23}$$

$$0 = \frac{\mu_0 - s_i}{\tau_0^2} + \sum_k \frac{1}{\tau_k}\tanh\frac{\mu_k - s_i}{\tau_k} \tag{24}$$

Solve for s_i using binary search, and use its value as the new rating r_i. Notice the resemblance with the final equation in the first section.

4 Noise, Uncertainty, and Evolution

Now we may ask: where do the μ_k, τ_k come from? In our parametrization, τ_k is proportional but not equal to the variance of a normal or logistic distribution centered at μ_k. It can be shown that a product of normal pdfs with variances $\{\sigma_k^2\}$ corresponds to a normal with variance $1/(\sum_k 1/\sigma_k^2)$. While general products of probability distributions don't have this property, let's define by analogy the *uncertainty* of our product distribution to be $1/(\sum_k 1/\tau_k^2)$. Recall that σ_i also represents the uncertainty involved in our belief. For consistency's sake, we would like the following relation to hold:

$$\frac{1}{\sigma_i^2} = \sum_k \frac{1}{\tau_k^2} \tag{25}$$

By manipulating the τ_k appropriately, σ_i will be defined by this equation instead of being a global constant. Each match multiplies the posterior by a logistic distribution centered at p_i, corresponding to new information about s_i. Here, the term *information* refers to the inverse of uncertainty. As we add information, the uncertainty tends toward zero. As discussed in the first section, we must prevent this in order to respond to a player's evolution over time. To this end, let's say s_i changes between matches by a noise term centered at 0 with variance η^2. We could easily model noise as being added continuously over time, as in Brownian motion, but instead let's add this constant lump term right before each match.

Thus, s_i after this match is equal to s_i before the match, plus a noise term. The prior belief is not identical to the posterior from the previous match: we must take this noise into account. A crude way to approximate its effect would be to increase all of the τ_k. If we want the weight of old matches to decay exponentially in some natural limiting cases, then it makes sense to multiply all of the τ_k by the same constant K. Thus, adding a noise η^2 corresponds to setting the variances in component k of the belief to $K\tau_k^2$ where:

$$\frac{1}{\sum_k 1/(K\tau_k^2)} = \frac{1}{\sum_k 1/\tau_k^2} + \eta^2 \tag{26}$$

$$\text{Therefore, } K = 1 + \eta^2 \sum_k 1/\tau_k^2 \tag{27}$$

In summary, we perform the K-correction before a match to get the prior, then compute p_i during the match, and then get the posterior from the prior by appending a logistic term with $\mu_k = p_i$ and $\tau_k = \delta^2$.

The noise level η is chosen such that σ_i approaches the limit $\sigma^* = 100$ as the number of matches goes to infinity. Setting $1/\sigma_i^2$ to $1/\sigma^*$ both before and after the match yields the fixpoint equation:

$$\frac{1}{\sigma^{*2}} = \frac{1}{\sigma^{*2} + \eta^2} + \frac{1}{\delta^2} \tag{28}$$

$$\text{Therefore, } \eta^2 = \frac{1}{1/\sigma^{*2} - 1/\delta^2} - \sigma^{*2} \tag{29}$$

5 Properties, Comparisons and Conclusions

The outlier-reduction behavior can be likened to a presumed purpose of TopCoder's volatility measure: one very strong performance won't change the rating much, but the second or third consecutive strong performance will have a larger effect. Unlike on TopCoder, a very weak performance following a very strong perforance will *not* have a large effect. More to be written later...