

Elo-R Draft

December 21, 2015

The Elo rating system sets out to assign a quantitative measure of the skills of players in a particular activity, such as chess, a video game, or an athletic sport. These ratings can be used to rank players, as well as forecasting the outcome of future matches.

To make things concrete, let's label the players $1, \dots, N$. The original Elo system deals with 1v1 zero-sum games with binary (win/lose) outcomes. From Player i 's history against other players, Elo derives a rating r_i , a real number capturing some measure of the player's skill. r_i increases when player i wins a match, and decreases when they lose. The magnitude of the increase or decrease depends on how "surprising" the result 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.

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 at the task, and the outcome is a total ranking of all the players in the match. Player i 's performance in the match can be viewed as a number p_i , drawn from a logistic distribution centered at s_i . We observe that i beat j in the match rankings, a result we denote as $i \succ j$, iff $p(i) > p(j)$. While we can't measure s_i directly, it's possible to get a fairly accurate reading on p_i by looking at the ratings r_j of players that win and lose against i . This suggests the following two-phase approach: upon a new match, estimate p_i for each player. Then, update the rating r_i to some sort of collective average of past p_i s.

1 Performance Averaging

Let's focus on the second phase for now: having computed the latest p_i for each player, we want to combine it with past values to get a more precise estimate. 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 k th match, and $r_{i,k}$ denotes their rating at this time. The sample mean is perhaps the most obvious statistic to take. That is, let

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

In addition to being simple, this method has the advantage of being a simple function of the previous rating $r_{i,M-1}$ and the new performance $p_{i,M}$, so there is no need to remember the performance history explicitly. This *memoryless* property also implies that players who enter a match with the same rating and same number of matches can expect to be treated similarly.

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. This is because all performances are weighted equally, even ones 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 wish to place less weight on the oldest performances. The second problem is that the mean is not robust to outliers. A single extreme performance, perhaps due to 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

$$\sum_k 1(p_{i,k} - r_{i,M}) = 0 \quad (2)$$

We say that $r_{i,M}$ is a *weighted mean* of the elements $\{p_{i,k}\}_{k=1}^M$, each of whom has equal weight, say 1. If the $p_{i,k}$ are thought of as independent measurements of the present true skill $s_{i,k}$, with Gaussian (normal) noise variances of τ_k^2 , treating $p_{i,1}$ as a prior, then the posterior belief over the true skill is a normal centered at $r_{i,M}$ given by the following equation:

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

All we did here is set the weight of $p_{i,k}$ to $1/\tau_k^2$. If the weights (or equivalently, the τ_k s) 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 its weight approach 0 as it 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. Thus, we opt for a weight of $\frac{\tanh \frac{p_{i,k} - r_{i,M}}{\tau_k}}{\tau_k(p_{i,k} - r_{i,M})}$. This choice will receive additional justification later: essentially, it has to do with assuming the heavy-tailed logistic distribution in place of the Gaussian noise discussed previously. The equation becomes:

$$\sum_k \frac{1}{\tau_k} \tanh \frac{p_{i,k} - r_{i,M}}{\tau_k} = 0 \quad (4)$$

It is monotonic in $r_{i,M}$, which can hence be solved for 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:

$$\frac{p_{i,0} - r_{i,M}}{\tau_0^2} + \sum_k \frac{1}{\tau_k} \tanh \frac{p_{i,k} - r_{i,M}}{\tau_k} = 0 \quad (5)$$

As a result, we only need to remember a small number of recent performances, and the size of the sum doesn't grow unboundedly with 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 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.

2 Approximate Bayesian Derivation

In general, the kernel of our Bayesian belief on the true skill s_i will be a product of normal and logistic pdfs, which can be thought of as a prior times a set of independent measurements centered on s_i with different variances. 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 . Let e be the evidence consisting of, for each j , whether $i \succ j$ or $i \prec j$. That is, we ignore relations like $j \succ k$ that affect the relative ordering of player pairs which don’t include i . Our goal is to approximate the posterior distribution of s_i given e :

$$f(s_i | e) \propto f(s_i) \Pr(e | s_i) = f(s_i) \int \Pr(e | p_i) f(p_i | s_i) dp_i \quad (6)$$

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

$$f(s_i | e) \propto f(s_i) f(p_i | s_i) \quad (7)$$

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 can be compactly summarized by finding the point r_i that achieves its maximum.

2.1 Performance Estimation

To compute the MAP of p_i , we must maximize

$$f(p_i | e) \propto f(p_i) \Pr(e | p_i) \quad (8)$$

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 $\tau_i^2 = \sigma_i^2 + \delta^2$,

$$f(s_i) \approx \frac{2e^{2(s_i-r_i)/\sigma_i}}{\sigma_i (1 + e^{2(s_i-r_i)/\sigma_i})^2} \quad (9)$$

$$f(p_i | s_i) = \frac{2e^{2(p_i-s_i)/\delta}}{\delta (1 + e^{2(p_i-s_i)/\delta})^2} \quad (10)$$

$$f(p_i) \approx \frac{2e^{2(p_i-r_i)/\tau_i}}{\tau_i (1 + e^{2(p_i-r_i)/\tau_i})^2} \quad (11)$$

Under these modeling assumptions,

$$\Pr(e | p_i) = \prod_{j \succ i} \Pr(p_j > p_i) \prod_{j \prec i} \Pr(p_j < p_i) \quad (12)$$

$$\approx \prod_{j \succ i} \frac{1}{1 + e^{2(p_i-r_j)/\tau_j}} \prod_{j \prec i} \frac{e^{2(p_i-r_j)/\tau_j}}{1 + e^{2(p_i-r_j)/\tau_j}} \quad (13)$$

$$\propto \frac{e^{2p_i \sum_{j \prec i} 1/\tau_j}}{\prod_{j \neq i} (1 + e^{2(p_i-r_j)/\tau_j})} \quad (14)$$

$$C_1 + \ln f(p_i | e) = C_2 + \ln f(p_i) + \ln \Pr(e | p_i) \quad (15)$$

$$\approx \ln \frac{2}{\tau_i} + (p_i - r_i) \frac{2}{\tau_i} - 2 \ln \left(1 + e^{2(p_i-r_i)/\tau_i} \right) + 2p_i \sum_{j \prec i} \frac{1}{\tau_j} - \sum_{j \neq i} \ln \left(1 + e^{2(p_i-r_j)/\tau_j} \right) \quad (16)$$

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

$$0 = \frac{1}{\tau_i} \left(1 - \frac{2e^{2(p_i-r_i)/\tau_i}}{1 + e^{2(p_i-r_i)/\tau_i}} \right) + \sum_{j \neq i} \frac{1}{\tau_j} \left(\mathbb{1}(j \prec i) - \frac{e^{2(p_i-r_j)/\tau_j}}{1 + e^{2(p_i-r_j)/\tau_j}} \right) \quad (17)$$

$$= \sum_{j \preceq i} \frac{1}{\tau_j} \left(\frac{1}{1 + e^{2(p_i-r_j)/\tau_j}} \right) - \sum_{j \succeq i} \frac{1}{\tau_j} \left(\frac{e^{2(p_i-r_j)/\tau_j}}{1 + e^{2(p_i-r_j)/\tau_j}} \right) \quad (18)$$

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_j s 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:

$$\frac{2}{\tau_i} \tanh \frac{p_i - r_i}{\tau_i} + \sum_{j \neq i} \frac{1}{\tau_j} \tanh \frac{p_i - r_j}{\tau_j} = \sum_{j \prec i} \frac{1}{\tau_j} - \sum_{j \succ i} \frac{1}{\tau_j} \quad (19)$$

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

2.2 Posterior Maximization

Recall our approximation $f(s_i | e) \propto f(s_i)f(p_i | s_i)$: the posterior $f(s_i | e)$ takes the prior $f(s_i)$ and multiplies it by a new logistic pdf $f(p_i | s_i)$. 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:

$$f(s_i | e) \propto e^{-(s_i - \mu_0)^2 / \tau_0^2} \prod_k \frac{e^{2(s_i - \mu_k) / \tau_k}}{(1 + e^{2(s_i - \mu_k) / \tau_k})^2} \quad (20)$$

$$C + \ln f(s_i | e) = -\frac{(s_i - \mu_0)^2}{\tau_0^2} + 2 \sum_k \left(\frac{s_i - \mu_k}{\tau_k} - \ln(1 + e^{2(s_i - \mu_k) / \tau_k}) \right) \quad (21)$$

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

$$0 = \frac{\mu_0 - s_i}{\tau_0^2} + \sum_k \frac{1}{\tau_k} \left(1 - \frac{2e^{2(s_i - \mu_k) / \tau_k}}{1 + e^{2(s_i - \mu_k) / \tau_k}} \right) \quad (22)$$

$$0 = \frac{\mu_0 - s_i}{\tau_0^2} + \sum_k \frac{1}{\tau_k} \tanh \frac{\mu_k - s_i}{\tau_k} \quad (23)$$

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.

3 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:

$$\frac{1}{\sigma_i^2} = \sum_k \frac{1}{\tau_k^2} \quad (24)$$

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:

$$\frac{1}{\sum_k 1/(K\tau_k^2)} = \frac{1}{\sum_k 1/\tau_k^2} + \eta^2 \quad (25)$$

$$\text{Therefore, } K = 1 + \eta^2 \sum_k 1/\tau_k^2 \quad (26)$$

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:

$$\frac{1}{\sigma^{*2}} = \frac{1}{\sigma^{*2} + \eta^2} + \frac{1}{\delta^2} \quad (27)$$

$$\text{Therefore, } \eta^2 = \frac{1}{1/\sigma^{*2} - 1/\delta^2} - \sigma^{*2} \quad (28)$$

4 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 performance will *not* have a large effect. More to be written later...