

Elo-like rating system for asymmetric games

Michal Horanský

September 19, 2025

1 Introduction

Suppose we are designing a zero-sum game which allows players to play competitive one-on-one matches on custom boards (by board we here refer to a specific starting configuration of the game). We wish to implement a rating system to serve a purpose akin to that of Elo rating for chess players. This rating system has to satisfy the same principal property of Elo rating, which is as follows: *the rating difference has to reflect the expected outcome of the game.* That is to say, we assume that each player has their "true strength", and the difference of true strengths of two players facing each other governs the expected outcome of the match. We then adjust player ratings based on their games so that *if a player would always perform according to his true strength, his rating would converge to a constant value regardless on the true strength of his opponents.*

The Elo rating in chess achieves this by identifying the difference between the measured outcome of a match and the expected outcome as the measured fault of the player's rating, and the post-match rating adjustment is technically a gradient descent towards the equilibrium. If player A scores S_A points in a match against player B , his new rating R'_A will be calculated as such:

$$R'_A = R_A + K \cdot (S_A - P_A(R_A - R_B)) \quad (1)$$

where $P_A(R_A - R_B)$ is the probability player A wins the game¹ given the rating difference, and K is the speed of the descent towards equilibrium (high K means volatile system, low K means rigid system). We see that *any* choice of rating system—i.e. the specific function P_A , which links true strength and rating—satisfies the principal condition if Eq. (1) is used. An important consequence is that Eq. (1) automatically conserves the total number of Elo points in circulation, since $S_B = 1 - S_A$ (as we assume a zero-sum game) and $P_A(R_A - R_B) = 1 - P_B(R_B - R_A)$, since P as a probability value must sum up to 1 over all participants.

¹Equivalently: the expected value of player A 's score.

2 Asymmetric boards and handicap

In our case, the problem is complicated by the asymmetry of the board. Suppose we parametrise this asymmetry with a real parameter h , which quantifies the advantage player A has over player B . Then, we need to devise a rating system such that Player A 's win probability in a match is given by the function $P_A^{(a)}(R_A - R_B, h)$ (the superscript serves to distinguish the asymmetric probability from the standard Elo-like win probability). We have a freedom of choice, but let us take the most straightforward way and interpret h as a handicap applied to the rating difference like so:

$$P_A^{(a)}(R_A - R_B, h) = P_A(R_A - R_B + h) \quad (2)$$

In other words, the "true" rating difference is context-dependent, and is calculated as the sum of the difference of player ratings plus the handicap.

2.1 Inferring the handicap from prior data

The problem is that we do not know the value of h for a given board. Suppose, however, that we keep track of all games ever played for every board in the following format:

board i	
ΔR_1^i	S_1^i
ΔR_2^i	S_2^i
⋮	
ΔR_j^i	S_j^i
⋮	

Table 1: Data about games played on a specific board.

where ΔR_j^i is the rating difference $R_A - R_B$ between the players who played the j -th game on board i , and S_j^i is player A 's achieved score in that game (standardly 1 for win, 0 for loss, 1/2 for draw).

What we now want to find out when players A and B sit down to play a game on board i whose historical games have been recorded in 1 is the probability of player A winning given the rating difference *and* the previous games. In other words, we are looking for the conditional probability

$$P_A^{(a)}(A \text{ wins} \mid \text{data})$$

We start by considering every possible value of h and identifying its contribution to the conditional probability as the product of the probability given the value of h multiplied by

the probability of that value being correct given the data:

$$P_A^{(a)}(A \text{ wins} \mid \text{data}) = \int P_A(R_A - R_B + h)P(h \mid \text{data})dh \quad (3)$$

where we used Eq. (2) to express the probability conditioned by a specific handicap value. As for $P(h \mid \text{data})$, we can now turn to Bayes' theorem:

$$P(h \mid \text{data}) = \frac{P(\text{data} \mid h)P(h)}{P(D)} \quad (4)$$

Here, $P(\text{data} \mid h)$ is the likelihood function which gives us a probability of measuring the dataset in Tab. 1 given a specific value of h . As such, it is a function of h and can be evaluated easily if we assume that every game played is independent of every other game:

$$P(\text{data} \mid h) = \prod_j^{\text{not draws}} P(S_j^i \mid \Delta R_j^i, h) \quad (5)$$

$$P(S_j^i \mid \Delta R_j^i, h) = \begin{cases} P_A(\Delta R_j^i + h) & S_j^i = 1 \text{ (Player A wins)} \\ 1 - P_A(\Delta R_j^i + h) & S_j^i = 0 \text{ (Player B wins)} \end{cases} \quad (6)$$

Notice how we ignored datapoints which ended in a draw. This is because in our underlying symmetric model, we have no way of assigning probability to the draw output. This requires a new free parameter, as per Elo-Davidson rating systems (see further sections). Here we just assume that drawn games have nothing to say about the handicap value.

2.2 Handicap prior

Coming back to Eq. (4), we see two more unexplained terms. $P(D)$ is simply a normalization factor for the likelihood function (which, as you can see, is not normalized with respect to h), and as such is simply equal to

$$P(D) = \int P(\text{data} \mid h)P(h)dh \quad (7)$$

so that $P(h \mid \text{data})$ has integral norm 1 w.r.t. h . Finally $P(h)$ is the prior probability distribution of the handicap—in other words, the probability that a random board has handicap h . We could use a non-informative prior and rely on a large dataset to accurately retrieve the posterior handicap distribution, but there is a better way. Since we are already storing information about all games across *all boards*, we can simply inspect all the other boards to estimate a prior standard deviation on h and assume e.g. a normal distribution around

zero². We do not actually need to re-calculate the prior standard deviation every time from all of the games for all the boards; we can just keep a list of mean values of h measured for existing boards and estimate σ_h from this aggregate dataset.

2.3 Expected score

We can now write down player A 's expected score as a function of the rating difference and the data for this board and handicaps for previous boards:

$$P_A^{(a)}(A \text{ wins} \mid \text{data}) = \int P_A(R_A - R_B + h) \frac{P(\text{data} \mid h)P(h)}{\int P(\text{data} \mid h')P(h')dh'} dh \quad (8)$$

$$P(\text{data} \mid h) = \prod_j^{\text{not draws}} P(S_j^i \mid \Delta R_j^i, h)$$

$$P(S_j^i \mid \Delta R_j^i, h) = \begin{cases} P_A(\Delta R_j^i + h) & S_j^i = 1 \text{ (Player A wins)} \\ 1 - P_A(\Delta R_j^i + h) & S_j^i = 0 \text{ (Player B wins)} \end{cases}$$

$$P(h) = \frac{1}{\sigma_h \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{h}{\sigma_h}\right)^2}$$

The new expected value of h to be added to the list which is used to calculate σ_h is simply

$$E[h] = E[P(h \mid \text{data})] = \frac{\int h P(\text{data} \mid h)P(h)dh}{\int P(\text{data} \mid h)P(h)dh} \quad (9)$$

and the update to the rating based on the match outcome is obtainable by substituting Eq. (8) into Eq. (1).

2.4 Dynamical step-size and retroactive rating adjustment

Note that the history of games played on a specific board retroactively adjusts the rating gain from each of those games, as the likelihood function becomes more and more accurate in its description of the true handicap value. Consequently, the rating of a player gets retroactively adjusted after each game played *after* his match on the same board. This sounds like a bug, but it is actually a feature, since early players should *not* suffer from lack of information about the board's fairness. Following this philosophy, there is one more improvement ready to be made to incentivize players to use boards with small datasets: retrospectively adjusting the K parameter with regard to our certainty about the board's fairness. To keep things

²For the first few boards, we will indeed need to use a non-informative prior, e.g. an initial guess on the standard deviation σ_h , which then iteratively gets better as we update this value with more and more boards and more games on each board, and converges to the true, equilibrium value.

simple, this could be estimated purely by the size of our dataset about the given board, i.e. N^i = number of non-drawn games recorded on board i . Then, we would choose a function $K(N^i)$ such that as $N^i \rightarrow 0$, $K \rightarrow 0$, and as $N^i \rightarrow \infty$, $K \rightarrow K_{\max}$. I propose a simple hyperbolic curve:

$$K(N^i) = 32 \frac{N^i}{10 + N^i} \quad (10)$$

This function is designed so that K converges to 32, a value used for low-rated players in chess, when played on boards about whose unfairness we are certain about, and it reaches half of that value after 10 games played on the board. These parameters can, of course, be changed to reflect your game's needs.

It is customary to note the specific function for expected score which is used for Elo rating in chess:

$$P_A(R_A - R_B) = \frac{1}{1 + 10^{(R_B - R_A)/s}}$$

where s is typically chosen to be 400, so that a difference in rating of 400 points means player A is ten times as likely to win as player B .

3 Including drawn games

In the previous section, we have sketched the idea for inferring information about the handicap inherent to an asymmetric board using data about the games played previously (on that and all other boards). However, we have deliberately omitted drawn games when estimating the likelihood of a given handicap value, as we had no way of stating the probability that a game between two players with a specific handicap will be drawn.

3.1 Parametrising the probability of a draw

There is a way to expand our model to include the probability of drawing and then go through an analogous process to determine the expected score for player A . To do this, we begin with the Elo-Davidson model, which introduces a free parameter κ , which is related to the probability a game between two players will be drawn. The probabilities of the three outcomes in the Elo-Davidson model are as follows:

$$P(A \text{ wins} | \Delta R, \kappa) = \sigma(\Delta R, \kappa) \quad (11)$$

$$P(B \text{ wins} | \Delta R, \kappa) = \sigma(-\Delta R, \kappa) \quad (12)$$

$$P(\text{draw} | \Delta R, \kappa) = \kappa \sqrt{\sigma(\Delta R, \kappa)\sigma(-\Delta R, \kappa)} \quad (13)$$

$$\text{where } \sigma(\Delta R, \kappa) = \frac{10^{\Delta R/s}}{10^{-\Delta R/s} + \kappa + 10^{\Delta R/s}} \quad (14)$$

Now, suppose we once again have a dataset about all the previous games in the form $(\Delta R_j^i, \text{outcome})$ for the j -th game played on the i -th board. We can first write two equations analogous to Eq. (3):

$$P(A \text{ wins} \mid \text{data}) = \iint P(A \text{ wins} \mid \Delta R + h, \kappa) P(h, \kappa \mid \text{data}) dh d\kappa \quad (15)$$

$$P(\text{draw} \mid \text{data}) = \iint P(\text{draw} \mid \Delta R + h, \kappa) P(h, \kappa \mid \text{data}) dh d\kappa \quad (16)$$

where, naturally,

$$P(B \text{ wins} \mid \text{data}) = 1 - P(A \text{ wins} \mid \text{data}) - P(\text{draw} \mid \text{data})$$

and thus we do not need to calculate the probability of player B winning.

As for the posterior probability distribution for h, κ , we once again turn to Bayes' theorem:

$$P(h, \kappa \mid \text{data}) = \frac{P(\text{data} \mid h, \kappa) P(h, \kappa)}{\iint P(\text{data} \mid h, \kappa) P(h, \kappa) dh d\kappa} \quad (17)$$

The likelihood function is now straightforward to write down and uses all datapoints, including drawn games:

$$P(\text{data} \mid h, \kappa) = \prod_j^{\text{board } i} P(\Delta R_j^i, \text{outcome} \mid h, \kappa) \quad (18)$$

$$P(\Delta R_j^i, \text{outcome} \mid h, \kappa) = \begin{cases} \sigma(\Delta R + h, \kappa) & \text{if player } A \text{ won} \\ \sigma(-\Delta R - h, \kappa) & \text{if player } B \text{ won} \\ \kappa \sqrt{\sigma(\Delta R + h, \kappa) \sigma(-\Delta R - h, \kappa)} & \text{if draw} \end{cases}$$

3.2 Prior for κ

3.2.1 Assumptions for separability and lack of correlations

As for the prior probability distribution for h, κ , a seasoned statistician with a large dataset could go to town. We here propose the simplest solution, once again aggregating data about all the other boards. Let us assume that the probability that a random board has a specific handicap and that it has a specific probability of a draw occurring are uncorrelated. Then, we can separate the prior like so:

$$P(h, \kappa) = P(h)P(\kappa) \quad (19)$$

As for the prior $P(h)$, we use just the same approach as in the previous sections. For $P(\kappa)$, we can use a similar approach, given one more assumption. For the sake of numerical simplicity,

let us assume that the average game played on a random board has a zero rating difference. Then, the probability of a draw is simply

$$P(\text{draw} \mid \kappa) = \frac{\kappa}{2 + \kappa} \quad (20)$$

For the i -th board, we can estimate its probability of a draw as the ratio of the number of draws and the number of all games played on this board:

$$P^i(\text{draw}) = \frac{N^i(\text{draws})}{N^i(\text{all games})}$$

and then

$$\kappa^i = 2 \frac{N^i(\text{draws})/N^i(\text{all games})}{1 - N^i(\text{draws})/N^i(\text{all games})}$$

This is purely a simple estimate for the value of κ for a given board, and it could be improved by including the rating differences in Eq. (20) and then performing either a numerical fit, or another Bayesian iteration. However, for the sake of numerical simplicity, we shall make the zero-difference assumption in this calculation, which is justified by the fact the matchmaking algorithm tries to minimize rating difference, and so for most matches the probability of a draw will be roughly constant. Naturally, this means we are slightly underestimating the value of κ , but as this estimate is only used for a prior, it matters little.

After collecting the values of κ^i for all the other boards, we can quickly calculate their mean value $\langle \kappa \rangle = N(\text{draws})/N(\text{all games})$. As will be demonstrated below, the mean value is all we need to find the prior $P(\kappa)$ in this simplified model.

3.2.2 Probability distribution of p and κ

As mentioned above, *in the prior* we assume that the probability of a game being drawn is correlated to neither its handicap nor its rating difference. Since all the games are played independently of each other, if the true probability of a game being drawn is p , then the probability of N_d games out of N being drawn follows the binomial distribution. As the number of N goes to infinity, the probability distribution of the ratio $\hat{p} = N_d/N$ approaches a continuous distribution, which can be identified as the Beta distribution:

$$P(\hat{p}, p) = \frac{\Gamma(N+2)}{\Gamma(N\hat{p}+1)\Gamma(N(1-\hat{p})+1)} p^{N\hat{p}} (1-p)^{N(1-\hat{p})} \quad (21)$$

We have measured the value of \hat{p} and are interested in the likelihood of a given value of the true probability p . Since Eq. (21) is already normalized with respect to p for a fixed \hat{p} (as $N \rightarrow \infty$), we can interpret it directly as the probability density function of p .

We are interested in the probability distribution P_κ of the $\kappa = \frac{2\hat{p}}{1-\hat{p}}$ parameter. For this, we employ the rule for transforming probability distributions:

$$\text{If } Y = G(X) \text{ and } X \text{ follows } f_X(x), \text{ then } Y \text{ follows } f_Y(y) = f_X(G^{-1}(y)) \left| \frac{d}{dy} G^{-1}(y) \right| \quad (22)$$

Substituting Eq. (21) into Eq. (22) yields the probability density function of κ given a measured mean probability of drawing a game $\langle P(\text{draw}) \rangle$:

$$P_\kappa(\hat{p}, \kappa) = \frac{\Gamma(N+2)}{\Gamma(N\hat{p}+1)\Gamma(N(1-\hat{p})+1)} \frac{2^{N(1-\hat{p})+1}\kappa^{N\hat{p}}}{(2+\kappa)^{N+2}} \quad (23)$$

This probability density function has all we need: it is undefined for $\kappa < 0$ and normalized on the non-negative reals, and its expected value is $\frac{2\hat{p}}{1-\hat{p}}$, agreeing with the initial measurement of drawing probability. As such, it is a useful prior for the κ parameter.

The total prior then becomes

$$P(h, \kappa) = \frac{1}{\sigma_h \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{h}{\sigma_h} \right)^2 \right] \frac{\Gamma(N+2)}{\Gamma(N\hat{p}+1)\Gamma(N(1-\hat{p})+1)} \frac{2^{N(1-\hat{p})+1}\kappa^{N\hat{p}}}{(2+\kappa)^{N+2}} \quad (24)$$

3.2.3 Approximate version of the κ prior

For very large N , Eq. (21) is subject to the Central Limit Theorem, since it describes the distribution of the sum of a large number of independent measurements. Therefore

$$\lim_{N \rightarrow \infty} P(\hat{p}, p) = \mathcal{N} \left(\hat{p}, \frac{\hat{p}(1-\hat{p})}{N} \right) \quad (25)$$

We observe the mean $\langle p \rangle = \hat{p}$ and standard deviation $\sigma_p = \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$ agree with the binomial distribution for N_d .

Applying Eq. (22) once again, we see that κ does not follow a normal distribution even in the large N limit:

$$\lim_{N \rightarrow \infty} P(\hat{p}, \kappa) = \frac{2}{(2+\kappa)^2 \sigma_p \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{\kappa - \frac{2\langle p \rangle}{1-\langle p \rangle}}{\frac{2+\kappa}{1-\langle p \rangle} \sigma_p} \right)^2 \right] \quad (26)$$

However, if we assume $\kappa \rightarrow 0$, which is reasonable if and only if our game has a low probability of a draw, this approximates a normal distribution with the properties

$$\langle \kappa \rangle = \frac{2\langle p \rangle}{1-\langle p \rangle} \quad (27)$$

$$\sigma_\kappa = \frac{2}{1-\langle p \rangle} \sigma_p \quad (28)$$

This final approximation allows us to write the total prior $P(h, \kappa)$ as a two-dimensional normal distribution.

3.3 Score calculation from win and draw probability

As for the final step, we evaluate player A 's expected score

$$E_A^{(A)}(\Delta R \mid \text{data}) = P(A \text{ wins} \mid \text{data}) + \frac{1}{2}P(\text{draw} \mid \text{data}) \quad (29)$$

Using this expected score value and the dynamic value of $K(N^i(\text{games}))$, we obtain the Elo-Davidson-Horanský rating system, characterised by the rating adjustment to player A after the game:

$$R'_A = R_A + K(N^i)(S_A - E_A^{(A)}(\Delta R \mid \text{data})) \quad (30)$$

4 Algorithm summary for the Elo-Davidson-Horanský rating system

The following algorithm can be readily applied when building any asymmetric one-vs-one zero-sum game which permits wins, losses, and draws:

1. We first choose the value of s , which simply sets the scale of the rating. If every new player starts with a rating of 1000, the chess value of $s = 400$ is reasonable. This value's interpretation is as follows: for every s points of rating difference, the ratio of the two players' win probabilities multiplies tenfold.
2. Before any boards are created, we need to estimate a starting prior for the handicaps and draw probabilities. This requires some knowledge of the specific game we are designing, but the following values should be good benchmarks for any application:
 - For the handicap, we estimate that a carefully-made board will rarely make one player more than twice as likely to win as their equally-rated opponent. Hence a good standard deviation estimate for h will be $\sigma_h \approx s \log_{10}(2) \approx 120$.
 - For the draw parameter, first estimate the probability of a draw on a symmetric board for two equally-rated players. If, for example, this estimate is $1/10$, the corresponding expected value of $\langle \kappa \rangle$ is $2 \cdot \frac{1/10}{1-1/10} \approx 0.2$. As per Sec. 3.2, the mean fully specifies the prior distribution from Eq. (24). Since we have no data, we can set $N = 1$ in the prior for the first calculation, maximizing the uncertainty.

We will be using these low-information priors until our dataset grows large enough (let's say $N(\text{all games}) > 30$ for $P(\kappa)$ and $N(\text{boards with at least 5 games played}) > 5$ for $P(h)$)

3. If two players rated R_A and R_B , respectively, play a game on board m , we follow this procedure:

- (a) If this was the first game played on this board, the step-size K is zero, and the ratings do not change. Record the game and exit algorithm.
 - (b) Otherwise, we calculate the likelihood function $P(\text{data} \mid h, \kappa)$ from all previous games played on this board, *including the one that was just played*. The reason for that is we will be retrospectively adjusting the rating changes from all of the games played on this board, and thus they should all use the same statistic. The uniformness of this outweighs the trouble of one badly-behaved datapoint for each game to be re-rated.
 - (c) We choose the priors for $P(h)$ and $P(\kappa)$, either based on the data already collected, or, if the datasets are too small, just opting in for the initial approximations prepared in earlier steps, and combine them to form the two-dimensional prior $P(h, \kappa)$. For this, we use the aggregate data for all the boards *excluding the one that was just played on*. The reason for this is that the priors are meant to estimate the probability this board has a specific handicap and drawing probability, and thus using its own data would be ill-formed.
 - (d) We calculate the normalization factor $\iint P(h, \kappa)P(\text{data} \mid h, \kappa)dhdk\kappa$, and by dividing the integrand by the integral we obtain the posterior $P(h, \kappa \mid \text{data})$.
 - (e) We calculate the new step-size for the board, K .
 - (f) Now, we will readjust the rating for every game played on this board. For every game, we calculate the probability of player A winning and the game being drawn, respectively, from the posterior and the rating difference. From this, we obtain player A 's expected score. The new, updated value of the rating adjustment for player A will be the new step-size multiplied by the difference between player A 's obtained score and his expected score. Player B 's updated rating adjustment will be the of equal value and opposite sign. We retroactively update each player's rating as the sum of the initial value of 1000 and all the adjustments across all the games they played (but this can be simplified as only the rating adjustments from games played on this specific board have changed).
4. We update the expected handicap value for the board in the aggregate dataset using the new posterior: $\langle h \rangle = \iint hP(h, \kappa \mid \text{data})dhdk\kappa$. This value will be used to calculate the priors for games played only on other boards, not this one.

5 Implementing retroactive rating on a large-scale dataset

The aforementioned algorithm produces a "correct" statistic in the sense that its repeated application converges to the true values of player rating and board handicap. However, it is massively inefficient on a large scale (in terms of player/board/game count), because it is

applied retroactively to all the games played on the affected board and, implicitly, to all the other games played at a later date than any of the recalculated games, since retroactively changing the rating of a game affects the ratings going into all subsequent games for the affected players. Hence, in this short section, we propose a "good-enough" algorithm which is designed with these features in mind as the top priorities:

- **Correctness:** Applying this algorithm will lead to the same results as applying the full-scale recalculation with the correct statistic.
- **Efficiency:** This algorithm is largely inexpensive, and only requires one expensive calculation per a fixed time interval.
- **Responsiveness:** Using this algorithm means that, after submitting a new concluded game, the rating of the engaged players and the handicap of the used board are updated immediately.
- **Symmetry:** When recalculating player rating adjustments for games on a single board, they all take the same step size and values of h, κ , which means their time-ordering does not matter in the approximating limit of small step size, where the rating adjustments can be taken to commute.

The algorithm is sectioned into two parallel processes:

1. **Immediate updates:** On submitting a new concluded game, the rating and handicap update is calculated according to the full statistic and applied to the "hot" values of these parameters for the players and the board. However, no retrospective recalculation is applied across the dataset. The date of the earliest game played on the affected board is stored; next time this happens, only the earlier of the two dates is stored.
2. **Housekeeping:** At regular intervals (e.g. nightly), a recalculation for all games which have occurred later than the date stored is performed, and the "hot" values of ratings and handicaps are adjusted to match the fully correct statistic.

The downside of this algorithm is that the "hot" values only approximate the correct statistic; however, they are constantly adjusted by the housekeeping process, and so they remain largely correct. The upside is that large-scale recalculation, which has to happen at least sometimes if we require our model to be correct with regard to the statistic, does not occur based on user input, and thus its overall demand on computing power does not scale with the user base.

Another method to limit the time complexity of the housekeeping process, which in theory makes it independent on the number of boards, is to stop updating the handicap value on any given board once a specific condition is satisfied (accurate: if the handicap has not changed by a significant amount for the last N new datapoints; approximate: once the

number of datapoints exceeds N , with N being arbitrary for both approaches). This means that, in theory, as new boards are added, old boards either "solidify" and no longer warrant retroactive recalculation (so that playing on such a board simply updates the rating but does not store its earliest game date), or stop being used for games, and thus no longer enlarge the scale of housekeeping recalculation. This second approximation is useful to be implemented on large-scale userbases, but before doing that, ensure you have enough data to estimate reasonable values of the parameter N for the handicap-calculation-freezing condition.

5.1 Implementation

The following is a mockup on how to implement the Elo-Davidson-Horanský rating system in your game, assuming your database has relational tables USER, BOARD, and GAME.

5.1.1 Approaches to forming priors

If we formed the prior on κ based on all the games played, we would need to recalculate every single game ever played for every housekeeping procedure. It would be better and more accurate to only form the prior from the data for the specific board, as different boards will have different values of $kappa$. For small dataset values, the prior should be informed mostly by the initial estimate κ_0 (and the corresponding initial estimate of the drawing probability \hat{p}_0), and as the number of games played on the board (N) grows, the prior will be more and more informed by the measured value of $\hat{p} = N_d/N$. This can be achieved e.g. with the following formula:

$$\hat{p}^{\text{prior}} = \frac{\hat{p}_0 + \frac{N}{\nu_p} \hat{p}}{1 + \frac{N}{\nu_p}} = \frac{\nu_p \hat{p}_0 + N_d}{\nu_p + N} \quad (31)$$

where ν_p is an arbitrary parameter which determines how quickly the measured value of \hat{p} overtakes the initial estimate as the dataset on the board grows. Specifically, for $N = \nu_p$ games measured on the board, the value \hat{p}^{prior} is exactly the arithmetic average of the initial and the measured value.

The same issue holds for estimating the prior on h , which is determined by its expected value $\langle h \rangle$ and its standard deviation σ_h , as it is modelled as a normal distribution. We do not want to assume that even the prior for the handicap depends on the handicaps of other boards (which would also make the model less numerically stable and more complex), so we use similar techniques for estimating these parameters for the prior depending only on the

initial estimates and the hot values:

$$\langle h \rangle^{\text{prior}} = \frac{\frac{N}{\nu_h} \langle h \rangle^{\text{hot}}}{1 + \frac{N}{\nu_h}} = \frac{N \langle h \rangle^{\text{hot}}}{\nu_h + N} \quad (32)$$

$$\sigma_h^{\text{prior}} = \frac{\sigma_{h,0} + \frac{N}{\nu_h} \sigma_h^{\text{hot}}}{1 + \frac{N}{\nu_h}} \quad (33)$$

with ν_h having an analogous meaning to ν_p .

There is no reason why we should two different scales for the priors on h and p , so we can take $\nu_p = \nu_h = \nu$, where ν is the rigidity of the model.

5.1.2 Database requirements

Table USER has a column RATING_HOT, which is the hot value of the user's rating. Similarly, table BOARD has columns HANDICAP_HOT, STD_HANDICAP_HOT, which are the hot values of the board's handicap and its standard deviation, column KAPPA_HOT, which is the hot value of the draw parameter κ , and column STEP_SIZE, which is the hot value of the step size K . Finally, table GAME has column R_A_ADJUSTMENT, which is the adjustment to player A 's rating due to that game.

As the model has a few arbitrary parameters, it is useful to have a table PARAMETERS: KEY, VALUE, in which we store the following values:

- Initial rating of a new player, R_0 (in my application 1000)
- The rating difference scale, s (in my application 400)
- Initial estimate of \hat{p}_0 (in my application 1/10)
- Initial estimate of $\sigma_{h,0}$ (in my application 120)
- Rating adjustment step scale K_0 (in my application 32)
- Game count scale for estimating \hat{p}_{prior} , h_{prior} , $\sigma_{h,\text{prior}}$, i.e. the model rigidity ν (in my application 10)
- Condition on dataset size for a board to stop updating its handicap, N_{freeze} . (Can be set to NULL to prevent handicap freezing.)
- The volatile value of the earliest datapoint scheduled for recalculation. By default, this is NULL (i.e. unknown).

To oversee the housekeeping process, it is also useful to have a table HOUSEKEEPING_LOGS: D_INITIATED, TIME_TAKEN, GAMES_AFFECTED, USERS_AFFECTED, BOARDS_AFFECTED, GAMES_TOTAL, USERS_TOTAL, BOARDS_TOTAL, which records the state of the total dataset and the dataset's fraction subject to recalculation, and allows the administrator to monitor trends in demand on the server's computation time.

5.1.3 Realisation of efficient algorithm

The algorithm constitutes two processes: immediate updates and housekeeping. Immediate updates are triggered on every submission of a new concluded game, where a particular game is characterised by the two engaged players, the outcome (winning player or draw), and the board utilised. Housekeeping is triggered at specific intervals, or manually by administrator, and does *not* adjust handicap and draw-parameter values, only player ratings.

- **Immediate updates.** On new submission of a concluded game:

1. Consider the hot handicap for the board, h^{hot} , its hot standard deviation, σ_h^{hot} , the hot draw factor for the board, κ^{hot} , and the hot ratings for each of the two players, $R_{A/B}^{\text{hot}}$, as well as the total number of games played on this board N and total number of drawn games on this board N_d (both excluding the newly submitted datapoint).
2. Find the prior estimate for the probability of drawing a game on this board, \hat{p}^{prior} , and the prior estimate for the handicap and its standard deviation for this board, σ_h^{prior} , using the model parameters and considered values like so:

$$\hat{p}^{\text{prior}} = \frac{\hat{p}_0 + \frac{N}{\nu} \hat{p}^{\text{hot}}}{1 + \frac{N}{\nu}} \quad \text{where} \quad \hat{p}^{\text{hot}} = \frac{\kappa^{\text{hot}}}{\kappa^{\text{hot}} + 2} \quad (34)$$

$$\langle h \rangle^{\text{prior}} = \frac{N \langle h \rangle^{\text{hot}}}{\nu_h + N} \quad (35)$$

$$\sigma_h^{\text{prior}} = \frac{\sigma_{h,0} + \frac{N}{\nu_h} \sigma_h^{\text{hot}}}{1 + \frac{N}{\nu_h}} \quad (36)$$

3. Then, the prior $P(h, \kappa)$ is like so:

$$P(h, \kappa) = \frac{1}{\sigma_h^{\text{prior}} \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{h - h^{\text{prior}}}{\sigma_h^{\text{prior}}} \right)^2 \right] \cdot \frac{\Gamma(N+2)}{\Gamma(N\hat{p}^{\text{prior}} + 1)\Gamma(N(1 - \hat{p}^{\text{prior}}) + 1)} \frac{2^{N(1-\hat{p}^{\text{prior}})+1} \kappa^{N\hat{p}^{\text{prior}}}}{(2 + \kappa)^{N+2}}$$

We can omit the normalisation factors, as the posterior will be renormalised explicitly, and thus the effective log prior is

$$\log P(h, \kappa) = \text{cons.} - \frac{1}{2} \left(\frac{h - h^{\text{prior}}}{\sigma_h^{\text{prior}}} \right)^2 + N \hat{p}^{\text{prior}} \log \kappa - (N + 2) \log(\kappa + 2)$$

4. The log-likelihood function for the board (evaluated on all games including the one just submitted) is like so:

$$\log L(\text{data} \mid h, \kappa) = \sum_{i \text{th game}} \log P_G(\Delta R_i, \text{ outcome} \mid h, \kappa) \quad (37)$$

where $\Delta R_i = R_{A,i} - R_{B,i}$ and the single game log-probability function is

$$\log P_G(\Delta R_i, \text{ outcome} \mid h, \kappa) = \begin{cases} \log \sigma(\Delta R + h, \kappa) & \text{if player } A \text{ won} \\ \log \sigma(-\Delta R - h, \kappa) & \text{if player } B \text{ won} \\ \log \kappa + \frac{1}{2} (\log \sigma(\Delta R + h, \kappa) + \log \sigma(-\Delta R - h, \kappa)) & \text{if draw} \end{cases} \quad (38)$$

which can be explicitly expressed as

$$P_G(\dots) = -\log \left(10^{\frac{\Delta R + h}{s}} + 10^{-\frac{\Delta R + h}{s}} + \kappa \right) + \begin{cases} \frac{\Delta R + h}{s} \log 10 & \text{if player } A \text{ won} \\ -\frac{\Delta R + h}{s} \log 10 & \text{if player } B \text{ won} \\ +0 & \text{if draw} \end{cases} \quad (39)$$

5. Then the unnormalised posterior becomes

$$P(h, \kappa \mid \text{data}) \propto \exp(\log P(h, \kappa) + \log L(\text{data} \mid h, \kappa)) \quad (40)$$

Evaluate this explicitly without cancelling the logarithms, because numerical evaluation of $\log P(h, \kappa), \log L(\text{data} \mid h, \kappa)$ does not suffer from the possibility of numerical underflow.

6. Normalise the posterior with $\frac{1}{Z}$ by numerically integrating

$$Z = \int_{h=-\infty}^{\infty} \int_{\kappa=0}^{\infty} \exp(\log P(h, \kappa) + \log L(\text{data} \mid h, \kappa)) dh d\kappa \quad (41)$$

7. Calculate the step-size for ratings, $K = K_0 \frac{N}{\nu + N}$.

8. Calculate the probability of player A winning and probability of a draw:

$$\begin{aligned} P(A \text{ wins}) &= \frac{1}{Z} \int_{h=-\infty}^{\infty} \int_{\kappa=0}^{\infty} \sigma(\Delta R + h, \kappa) \exp(\log P(h, \kappa) + \log L(\text{data} | h, \kappa)) dh d\kappa \\ P(\text{draw}) &= \iint \kappa \sqrt{\sigma(\Delta R + h, \kappa) \sigma(-\Delta R - h, \kappa)} P(h, \kappa | \text{data}) dh d\kappa \end{aligned}$$

Then, calculate player A 's expected score:

$$\langle S_A \rangle = P(A \text{ wins}) + \frac{1}{2} P(\text{draw})$$

Then, calculate the rating adjustment for both players:

$$\begin{aligned} \delta R_A &= K(S_A - \langle S_A \rangle) \quad \text{where } S_A = \begin{cases} 1 & \text{if player } A \text{ won} \\ 0 & \text{if player } B \text{ won} \\ \frac{1}{2} & \text{if draw} \end{cases} \\ \delta R_B &= -\delta R_A \end{aligned}$$

Update the hot values of player ratings with these values. Also, save the value of δR_A to the column R_A_ADJUSTMENT in the recorded game's datarow.

9. Calculate the new expected value of the handicap and its standard deviation:

$$\begin{aligned} \langle h \rangle &= \iint h P(h, \kappa | \text{data}) dh d\kappa \\ \langle h^2 \rangle &= \iint h^2 P(h, \kappa | \text{data}) dh d\kappa \\ \sigma_h &= \sqrt{\langle h^2 \rangle - \langle h \rangle^2} \end{aligned}$$

Calculate the new expected value of κ :

$$\langle \kappa \rangle = \iint \kappa P(h, \kappa | \text{data}) dh d\kappa$$

Update the hot values to be equal to these newly calculated values. This includes the new step size K .

10. Unless the freezing condition is satisfied ($N > N_{\text{freeze}}$), note down the date of the earliest game played on this board, and overwrite the global earliest date scheduled for recalculation.

- **Housekeeping.** Regularly:

1. Reset the values of RATING_HOT for every user to the default value R_0 .
2. Consider all games recorded *earlier* than the date scheduled for readjustment. By summing up their values of R_A_ADJUSTMENT (with signs flipped according to the player's role) for each player, we calculate the hot values of player ratings for each player as they were right before the earliest game scheduled for readjustment.
3. Then, for each game, ordered by time of submission, we recalculate the rating adjustment for both players while taking the hot handicap, κ , and step size value as the *true* values, like so:
 - (a) Take the hot handicap for this board, h^{hot} , and interpret it as the true value h . Take the hot draw-parameter for this board, κ^{hot} , and also interpret it as the true value κ . Take the hot step size of the board, K^{hot} , and set the step size K equal to it. Consider the hot ratings of the two players $R_{A/B}^{\text{hot}}$.
 - (b) Calculate the effective rating difference r , which takes into account the handicap value:

$$r = R_A - R_B + h$$

- (c) Calculate the expected score of player A like so:

$$\langle S_A \rangle = P(A \text{ wins}) + \frac{1}{2}P(\text{draw}) = \frac{10^{r/s} + \kappa/2}{10^{r/s} + 10^{-r/s} + \kappa}$$

- (d) Calculate the rating adjustment for the players like so:

$$\begin{aligned} \delta R_A &= K(S_A - \langle S_A \rangle) \quad \text{where} \quad S_A = \begin{cases} 1 & \text{if player } A \text{ won} \\ 0 & \text{if player } B \text{ won} \\ \frac{1}{2} & \text{if draw} \end{cases} \\ \delta R_B &= -\delta R_A \end{aligned}$$

Store the value of δR_A into R_A_ADJUSTMENT and update the hot ratings of both players for the next game's recalculation.

4. After recalculating the rating adjustment for every game, we terminate the program without adjusting handicap or draw-parameter values.