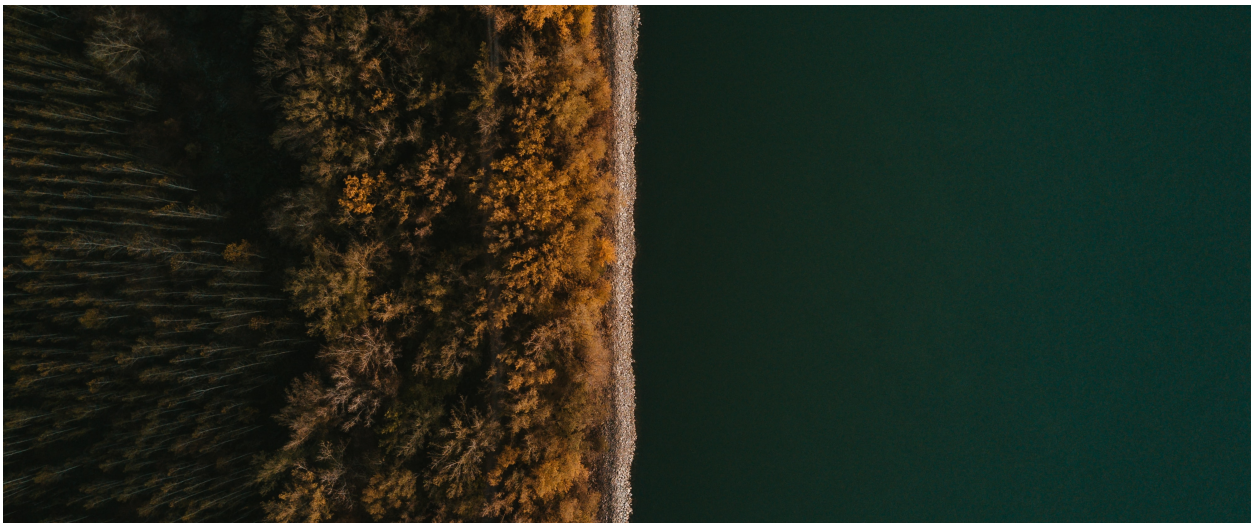


Rating Algorithms

Introduction

Humans surround themselves with competition. These competitions can be one-to-one like chess, tennis or single player like competitive coding contests.

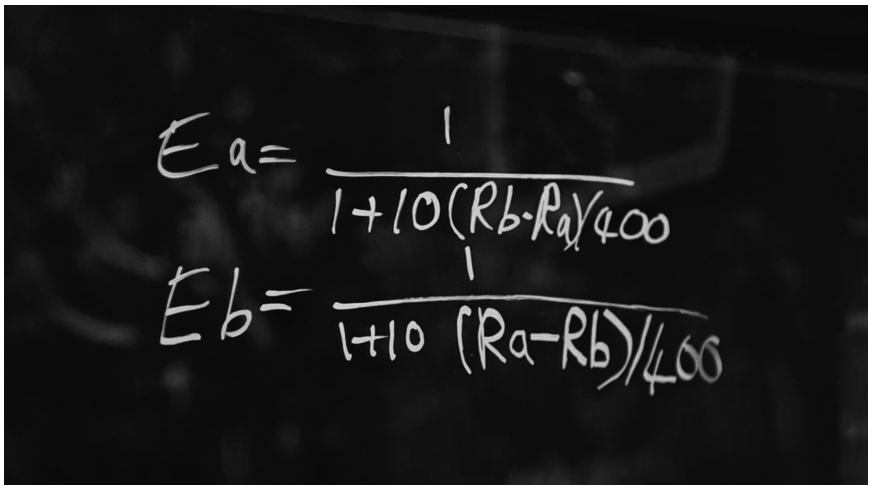
Performance ranking and rating is the process of describing the skill of a player in relation to that player's competitors.

What we need is a method of estimating the skill difference between any two players and although we might assume that the player with higher rank will dominate, we need a way of actually evaluating the chances.

Performance **rating** arranges players along an interval scale representative of that player's skill. Given a rating we can develop a ranking that accurately reflects the current state of the game. However we cannot construct a rating from a ranking, which makes performance rating systems much more powerful than ranking systems.

Instead of wondering what position a player holds in a competitive system
that player can now ask, *"what is the skill difference between me and first place ?"*
The methods used to generate ratings/rankings range from the simple to the extremely complex.

Elo Rating Algorithm is one of the rating algorithms that is widely used to rank players in many competitive games.



Elo Rating Algorithm

The algorithm is named after it's creator Arpad Elo, a Hungarian-American physics professor born in 1903.

> Players with higher ELO rating have a higher probability of winning a game than a player with lower ELO rating. After each game, ELO rating of players is updated. If a player with higher ELO rating wins, only a few points are transferred from the lower rated player. However if lower rated player wins, then transferred points from a higher rated player are far greater.

The *performance* in the ELO system is **not** measured in *absolute* terms. It is *inferred* from wins, losses, and draws against other players. Players' ratings depend on the ratings of their opponents and the results scored against them. After every game, the winning player takes points from the losing one, and the number of points is determined by the difference in the two player's rating.

- If the higher-rated player wins, a few points are taken from the lower-rated player.
- If the lower-rated player wins, a lot of points are taken from the higher-rated player.
- If it’s a draw, the lower-rated player gains a few points from the higher rated player.

One of the important assumptions in the elo rating system is that the mean value of their performances (a reflection of their true skill) only changes *slowly* over time.

The difference in the ratings between two players serves as a predictor of the outcome of a match. If players A and B have ratings R^A and R^B, then the expected scores are given by :

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} \qquad E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}$$

If two players have equal ratings ($R^A = R^B$), then the expected scores of A and B evaluate to 1/2 each. That makes sense — if both players are equally good, then both are expected to score an equal number of wins.

When a player scored differently from the expected score, the rating of the player is updated using a simple linear adjustment proportional to the amount by which a player over-performed or under-performed. If Player A was expected to score E_A points but actually scored S_A points, the player’s rating is updated using the formula:

Here K is the K-factor, the maximum possible adjustment per game.

$$R'_A = R_A + K(S_A - E_A)$$

In the long-run, the Elo rating system is **self-correcting**. If a player’s rating is too high, they will perform worse than what the rating system predicts, and if a player’s rating is too low, they will perform better than what the rating system predicts. Thus, their rating eventually settles to the correct value in the long run.

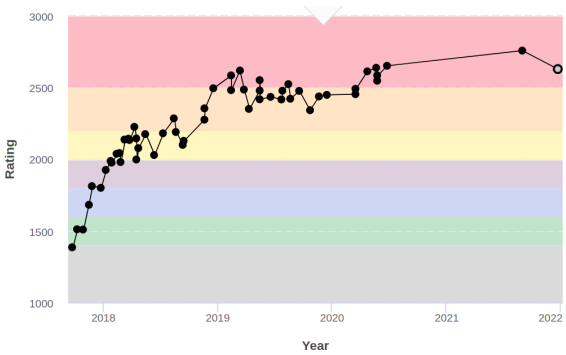
```
//Pseudo code for the Elo rating system
import math;
class elo_core:
  def getExpectation(rating_1, rating_2):
    calc = (1.0 / (1.0 + pow(10, ((rating_2 – rating_1) / 400))));
```

```
    return calc;
def modifyRating(rating, expected, actual, kfactor):
    calc = (rating + kfactor * (actual – expected));
    return calc;
```

Rating mechanism used by Codechef

> Whenever you participate in a contest, you compete in a duel with all other participating players. Based on your existing rating and the other participating players' rating, your probability of winning against each of them is calculated. Your actual performance is compared against this expected performance, and correspondingly, you gain or lose some points.

Codechef uses a modification of the ELO rating algorithm, tailored for group contests for rating participants in individual competitions, and to calculate the Long Rating, Short Rating, LunchTime Rating and combined CodeChef Rating (which includes all rated contests).



In this system, the *skill* of the player is described by a probability distribution specified by two variables, namely *Rating* ***R*** and *Volatility* ***V***. The value of Rating **R** describes the most probable level of the player, and the Volatility **V** describes how volatile the player is i.e how confidently we can tell the level of a player.

In more mathematical terms, the values **R** and **V** behave similar to the *mean* and *variance* of the distribution respectively. The higher the value of **R**, the better the player is on an average day. The lesser the value of **V**, more is the probability that the player performs closer to the expected level, **R**. The probability that a player A with rating R_a and volatility V_a performs worse than an player B with rating R_b and volatility V_b is given by the formula :

$$E_{ab} = \frac{1}{1 + 4^{\frac{R_a - R_b}{\sqrt{V_a^2 + V_b^2}}}}$$

The sum of E_{ab} for player A over *all* B, gives the expected rank, **ERank**, of player A in the contest. Let number of contestants in the given contest be **N**. And let average rating be R_{avg}.

The performance is calculated as :

$$Perf = \frac{\log(\frac{N}{rank-1})}{\log(4)}$$

Using this formula, we calculate both expected and actual performance factor of the player, as below:

If the actual rank of a player is **ARank**, then Actual Perf will be

$$APerf = \frac{\log(\frac{N}{ARank-1})}{\log(4)}$$

and Expected Perf will be

$$EPerf = \frac{\log(\frac{N}{ERank-1})}{\log(4)}$$

For each competition, we compute a competition factor as

$$Cf = \sqrt{\frac{\sum\limits_{\text{over all a}} V_a^2}{N} + \frac{\sum\limits_{\text{over all a}} (R_a - R_{avg})^2}{N-1}}$$

We define rating weight of a player as

$$RW_a = \frac{0.4 * timesPlayed + 0.2}{0.7 * timesPlayed + 0.6}$$

And volatility weight of player is defined as

$$VW_a = \frac{0.5 * timesPlayed + 0.8}{timesPlayed + 0.6}$$

, where **timesPlayed** is the number of competitions the player has participated so far.

If the old rating is R_a , then, new rating of the player will be

$$NR_a = R_a + (APerf - EPerf) * Cf * RW_a$$

New volatility of the player will be

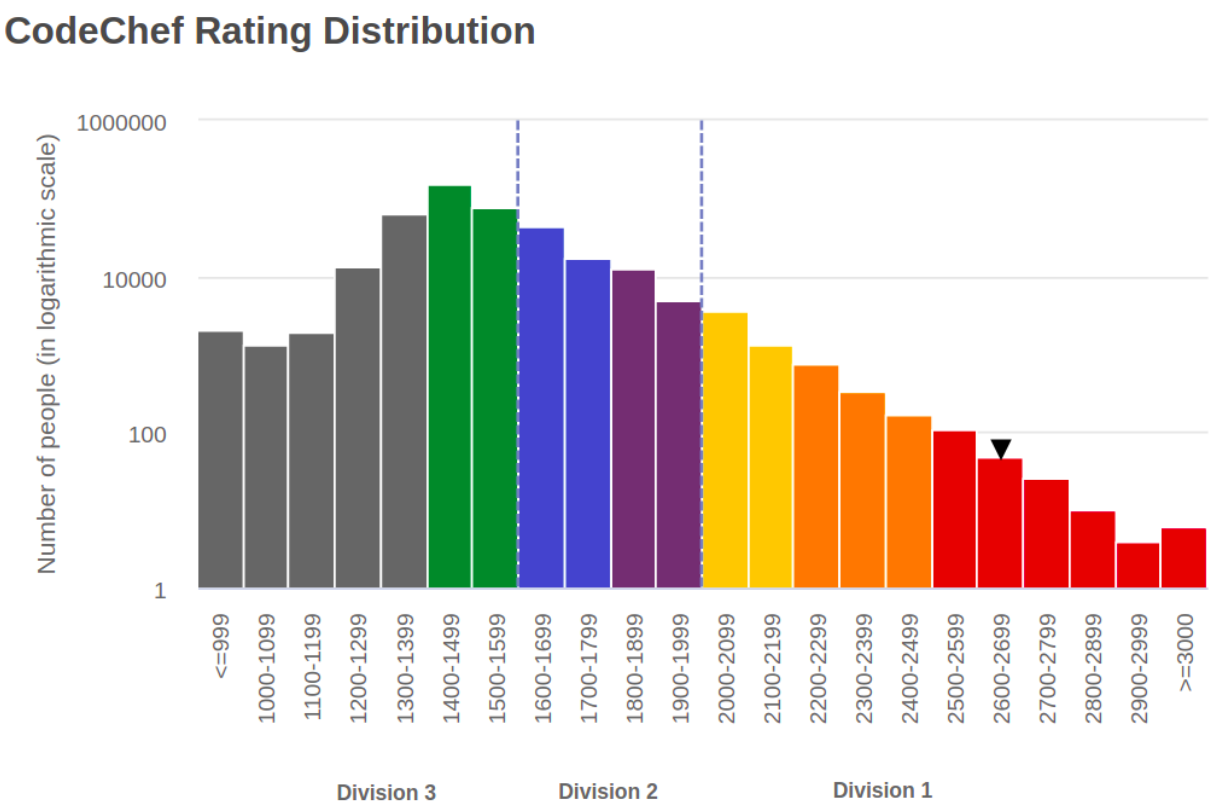
$$NV_a = \sqrt{\frac{VW_a * (NR_a - R_a)^2 + V_a^2}{VW_a + 1.1}}$$

Each player in his first contest would have an initial rating of 1500 and initial volatility of 125.

The rating changes are capped with a max rating change computed as

$$100 + \frac{75}{timesPlayed + 1} + \frac{100 * 500}{|R_a - 1500| + 500}$$

The volatility value is moderated to lie in the range (75, 200).



A basic implemention of the Elo rating algorithm

1. Intial Ratings - At starting every players are assigned with a default score. This default score definitely has influence on the ratings. The intials ratings are some sort of baseline . At any given point in time if a player's rating is below the intials ratings that means that player is performing badly. At the same time if a player's rating is eventually higher than the baseline rating then we can say this player is performing pretty well.

2. Probability of Winning - In a match between two players the probability of winning will be same for both the player if the rating of the players are same.

```
r1 = Rating of player A.
r2 = Rating of player B.
P1 = Probability of winning for player A.
P2 = Probability of winning for player B.
P1 = r1 / (r1+r2)
P2 = r2 / (r1+r2)
```

3. Rating Shifting Factor - This is basically a parameter or factor by which rating's from losing player to be shifted to wining player. It is denoted as 'K'. After every game the rating is updated

for both team using the below formula..

```
CASE-1 : Suppose player A wins:
  r1 = r1 + k*(1 – P1)
  r2 = r2 + k*(0 – P2)
CASE-2 : Suppose player B wins:
  r1 = r1 + k*(0 – P1)
  r2 = r2 + k*(1 – P2)
```

K is a constant. If K is of a lower value, then the rating is changed by a small fraction but if K is of a higher value, then the changes in the rating are significant.

```
//Example
k = factor of shifting of rating = 30
r1 = Rating of player A before match starts = 1100
r2 = Rating of player B before match starts = 900
P1 = Probability of winning for player A
= 1100 / (1100 + 900)
= 0.55

P2 = Probability of winning for player A
= 900 / (1100 + 900)
= 0.45
```

Let's say player B win the match. The ratings will be updated as :

```
r1_updated = r1 + k*(0-p1)
= 1100 + 30*(0-0.55)
=  1083.5

r2_updated = r2 + k*(1-p2)
= 900 + 30*(1-0.45)
= 916.5
```


NEWBIES/FAKE ACCOUNTS: ATCODER BEGINNER CONTESTS ARE A JOKE ! SO EASY THEY ARE !
ATCODER : THEN WHY IS YOUR RATING 352 ON ATCODER ?
imgflip.com