Ping Pong Elo Rating System

Contents

1 Basic Rules

This project implements an Elo rating system adapted for a ping pong league, where players are ranked based on their performance in individual matches. The system uses a dynamic K-factor to adjust the sensitivity of rating changes based on the number of games played.

- Each game is played as a standalone match (first to eleven points), with no sets or multiple games considered.
- A player can only win or lose in each game; there are no draws.
- Ratings are updated after each game based on the match outcome, with higher rating changes for surprising results (e.g., a lower-rated player defeating a higher-rated player).

2 Initial Rating Setup

- All players start with a base Elo rating, typically set at **1000**. This starting point represents an average skill level for the league.
- New players receive this initial rating when they first join the ranking system.

3 Dynamic K-Factor Calculation

The K-factor controls the amount by which ratings change after each match. To make the system more dynamic and fair, K is adjusted based on the number of games each player has played. The formula for calculating K is:

$$K = K_{\min} + (K_{\max} - K_{\min}) \times \frac{1}{\log(n+1) + 1}$$

where:

- $K_{\max} = 40$ is the starting K-factor for players with minimal experience.
- $K_{\min} = 20$ is the minimum K-factor for experienced players.
- n is the number of games a player has played.

This logarithmic function ensures that K starts high for new players and gradually decreases as players gain experience, stabilizing their ratings over time.

4 Expected Score Calculation

The expected score for each player is calculated using the Elo formula:

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

$$E_B = 1 - E_A$$

where:

- R_A is the rating of Player A.
- R_B is the rating of Player B.
- E_A represents the probability that Player A will win the game.

5 Match Outcome and Rating Update

After each game, the actual score is determined:

- $S_A = 1$ if Player A wins.
- $S_A = 0$ if Player A loses.

Similarly, $S_B = 1$ if Player B wins and $S_B = 0$ if Player B loses.
The ratings are then updated using the dynamic K-factor:

$$R'_A = R_A + K_{\text{combined}} \times (S_A - E_A)$$

$$R'_B = R_B + K_{\text{combined}} \times (S_B - E_B)$$

The combined K-factor for the match is calculated as the average of the individual K-factors for Player A and Player B:

$$K_{\text{combined}} = \frac{K_A + K_B}{2}$$

This approach ensures that the rating change reflects the experience level of both players.

6 Google Sheets Integration

The system now supports reading match data from Google Sheets and writing the calculated rankings back to a Google Sheet.

6.1 Setup Instructions

- Create a Google Service Account and download the credentials JSON file (`google_cred.json`).
- Share your Google Sheets document with the service account email.
- Install the required Python libraries:

```
pip install pandas gspread
```

6.2 Loading Data

The script reads match data from a Google Sheet with columns:

`game_date, player_1, player_2, player_1_result, player_2_result`

6.3 Writing Data

After calculating the Elo ratings, the updated rankings are written back to a specified Google Sheet.

7 Examples

- **Example 1**: Player A (1 game played) vs. Player B (40 games played).
  - Player A's rating: 1000, $K_A \approx 33.1$.
  - Player B's rating: 1000, $K_B \approx 22.4$.
  - Combined K: $K_{\text{combined}} = \frac{33.1+22.4}{2} \approx 27.75$.
  - Expected scores:

$$E_A = \frac{1}{1 + 10^{(1000-1000)/400}} = 0.5$$

$$E_B = 1 - 0.5 = 0.5$$

  - If Player A wins:

$$R'_A = 1000 + 27.75 \times (1 - 0.5) = 1013.88$$

$$R'_B = 1000 + 27.75 \times (0 - 0.5) = 986.12$$

8 Advantages and Limitations

8.1 Advantages

- The system adjusts quickly to reflect new players' skill levels.
- Experienced players' ratings stabilize, preventing large fluctuations.
- Predictable rating changes make it easy to understand progress.
- Integration with Google Sheets allows for easy data management and sharing.

8.2 Limitations

- Players with few games may have volatile ratings.
- The system does not account for streaks or margin of victory.

9 Implementation Suggestions

- Use a database to track players' ratings, number of games, and outcomes.
- Update ratings immediately after each game using the formulas provided.

10 Usage

10.1 Requirements

- Python 3.x
- `pandas` library
- `gspread` library

10.2 Running the Code

1. Save the Google Sheets data as specified, or run the script to load and process it automatically.
2. Run the script:

```
python calculate_elo.py
```

11 Output Metrics

The script will output the following metrics for each player:

- **Rating**: The player's Elo rating.
- **Games Played**: The total number of games the player has participated in.
- **Wins**: The number of games won by the player.
- **Losses**: The number of games lost by the player.
- **Win Rate**: The percentage of games won out of the total games played.
- **Last Played Date**: The date of the most recent game.