Elo Regression

Extending the Elo Rating System

Jonathan Dorsey

The University of Akron

April 16, 2019

Overview

- Introduce the Elo rating system
- Discuss prospective vs retrospective ratings and why Elo is not designed for retrospective ratings
- Understand Elo as a probabilistic/statistical model
- Use this understanding to correct its shortcomings (in terms of retrospective ratings): call the resulting model Elo regression
- Compare Elo and Elo regression on three data sets

Introduction

- Elo assigns ratings to players based only on outcomes of games
- Ratings are used primarily as a measure of the current strength of a player
- Invented by physicist and chess player Arpad Elo in the 1950s
- Adopted by FIDE in the 1970s, it's been the primary way of measuring chess strength ever since
- Chess ratings are very important to players: used for tournament invites and seeding, and titles like Grandmaster
- Elo has also been applied to other sports: Nate Silver's NBA Elo ratings, The World Football Elo Ratings

How Does Elo Work?

- The main idea of Elo is that ratings can be used to produce an expected outcome for a future game
- After the game is actually played, and the true outcome observed, ratings are updated based on the discrepancy between the actual outcome and expected outcome
- The ratings evolve over time and correct themselves based on the data

Why Elo?

- Elo's main strength is computational simplicity (for decent performance)
- With modest computing, chess federations can easily maintain ratings for hundreds of thousands of players
- Elo is simple enough that players can compute their own ratings
- Makes ratings feel transparent and fair

Retrospective Ratings via Elo

- Elo is also used to analyze players' abilities over time
- It's historically interesting for fans and players to compare chess players across time and try to find the strongest players
- Since a player's rating is updated after every game, by keeping track of their rating, you get some measure of how their skill is changing over time
- *skill*: 'true' underlying ability
- *rating*: estimated skill

Elo is not for Retrospective Ratings

- Elo is *prospective*: the ratings at each time period use only information from previous time periods
- In contrast, a rating system could instead retroactively change past ratings based on new information.
- There is a decision each system has to make between how much new information affects new ratings vs how much it retroactively fixes old ratings. Depends on how fast skill can change over time.
- The relationship between rating and time is the ad hoc result of the Elo updating procedure
- We should be able to do better than Elo at retrospective ratings if we fix these two deficiencies: use all available information, and explicitly model time as a covariate

Technical Details of Elo

- Every player i has a rating R_i
- When two players i and j compete, Elo produces an expected score for each player
- $$E_i = \frac{1}{1 + 10^{(R_j - R_i)/400}}$$
$$E_j = \frac{1}{1 + 10^{(R_i - R_j)/400}}$$
- $E_i + E_j = 1$

Technical Details of Elo

- From the actual game, players i and j get real scores S_i and S_j
- In chess scoring, win $= 1$, draw $= 1/2$, loss $= 0$, so $S_i + S_j = 1$
- Update ratings based on discrepancy between S_i and E_i in a linear fashion

$$R_i' = R_i + K_i(S_i - E_i)$$
$$R_j' = R_j + K_j(S_j - E_j)$$

K-Factors

- K is called K-factor and controls how much ratings can change
- When both players have the same K-factor, the updates sum to zero: rating points are conserved
- K-factor trade-off: too low and ratings stay inaccurate, too high and they fluctuate too much
- K-factors are higher for new players and lower for older or higher-rated players

A Probabilistic View of Elo

- Elo can be viewed a very particular way to fit a probabilistic model
- Let each player i have a skill θ_i, and assume for now no draws
- Say the probability of player i beating player j is $\sigma(c(\theta_i - \theta_j))$, where $\sigma(x) = \frac{1}{1+e^{-x}}$
- If $c = \ln(10)/400$, we get Elo's expected score function
- Without draws, probability of win is expected score
- Extend to multiple games by assuming independence

Fitting the Model

- Now we have a likelihood for a set of games and skills, so we can estimate the skills via maximum likelihood
- Choose ratings that maximize the probability of observing the games we saw, assuming the model from the previous slide

$$P(G|\theta) = \prod_{(i,j)\in G} \sigma(c(\theta_i - \theta_j))$$

Gradient Ascent

- We can maximize the log likelihood using gradient ascent: gradient points in the direction of steepest increase, so follow it to a (local) maximum

$$\frac{\partial}{\partial \theta_i} \ln \sigma(c(\theta_i - \theta_j)) = c(1 - \sigma(c(\theta_i - \theta_j))) = c(S_i - E_i)$$

$$\frac{\partial}{\partial \theta_j} \ln \sigma(c(\theta_i - \theta_j)) = -c(1 - \sigma(c(\theta_i - \theta_j))) = c(S_j - E_j)$$

Gradient Ascent

- Online gradient ascent is a crude, but sometimes useful approximation to gradient ascent
- Calculate the gradient of the likelihood using only a single data point, make a gradient ascent step using this gradient, then discard that data point and proceed to the next one.
- If we perform 'online gradient ascent' with step size ε, then updates look like

$$R_i' = R_i + \varepsilon c(S_i - E_i)$$
$$R_j' = R_j + \varepsilon c(S_j - E_j)$$

- This is exactly Elo!

Handling Draws

- Treat them as $1/2$ a win and $1/2$ a loss just like Elo
- Equivalent to replacing each game by two 'pseudo-games'
- If p is now probability of winning a pseudo-game instead of a whole game, the expected score is still p:

$$1 \times p^2 + \frac{1}{2} \times 2p(1-p) + 0 \times (1-p)^2 = p.$$

- So we can directly modify the likelihood, or we can just modify the data

Elo Regression

- Elo models the outcome of a game as depending on the logistic function applied to a difference in skills
- The MLE is then approximated via online gradient ascent
- So we have three avenues of experimentation, and we use all three:
  - Change model of game outcomes: we want to explicitly account for time
  - Change method of estimation: add regularization to improve predictions and ensure the objective function stays bounded
  - Change way of approximating those estimates: use more accurate but expensive optimization procedure

Modifying the Model

- To change model, replace fixed skill θ_i with function $\theta_i(t)$
- This way we directly incorporate time into the model in a sensible fashion
- Use linear regression with basis expansion:

$$\theta_i(t) = \sum_k \beta_{ik} f_k(t)$$

- By choosing non-linear basis functions f_k, we can model non-linear relationships

Basis Functions

- We use Gaussian radial basis functions since they are universal function approximators:

$$f_k(t) = \exp\left(-\frac{(t - C_k)^2}{L^2}\right)$$

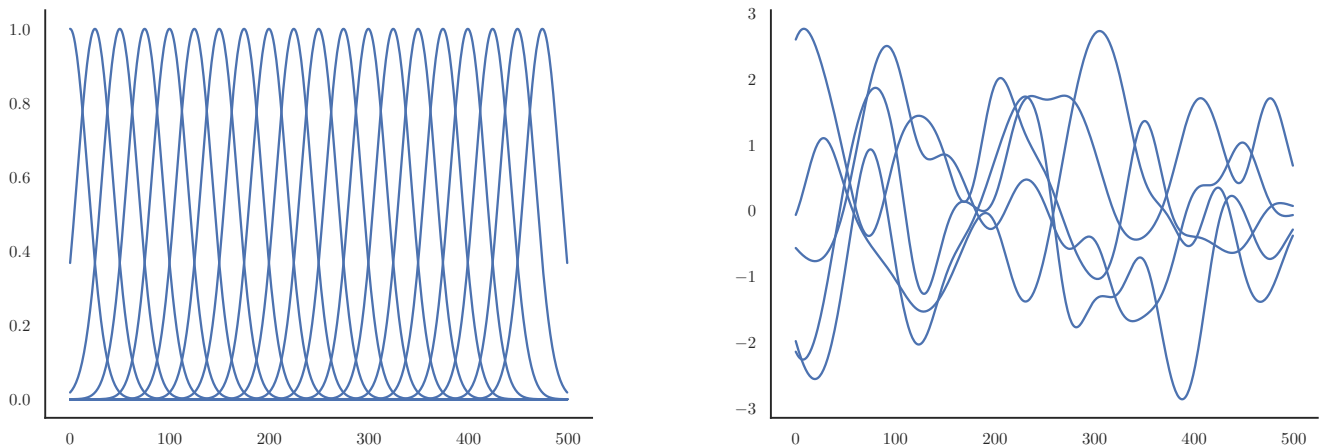

Figure: Left: Gaussian radial basis functions with uniformly spaced centers plotted individually. Right: Random linear combinations of the basis functions on the left.

Estimation and Approximation

- Change estimation only slightly: L_2 penalty
- This provides regularization which improves predictive performance and ensures the objective function remains bounded
- Change approximation from online gradient ascent to minibatch gradient ascent
- Use minibatches to approximate gradient: law of large numbers ensures high quality estimates of gradient for low computational cost
- Cycle through data until convergence
- This gives a higher quality approximation than online gradient ascent

Comparison to Elo

- Elo regression is not designed to be used as a general purpose rating system
- Much more computationally intensive
- Non-prospective nature would probably annoy players
- Elo regression is designed to use all possible information for retrospective analysis of skill over time
- As we will see soon, it succeeds at improving upon Elo in this regard

Data

- Three data sets for comparing Elo and Elo regression
- Kaggle data: real, large but sparse
- Synthetic data 1: quickly changing skills, but rich information
- Synthetic data 2: designed to display weaknesses of prospective systems

Kaggle Data

- Giant real dataset
- Comes from FIDE database
- $\sim$3 million games among $\sim$90k players over 135 months
- 0.2 games per player per month
- Built to be well-connected
- 70:15:15 train:val:test split
- Around 30% draws

Kaggle Data

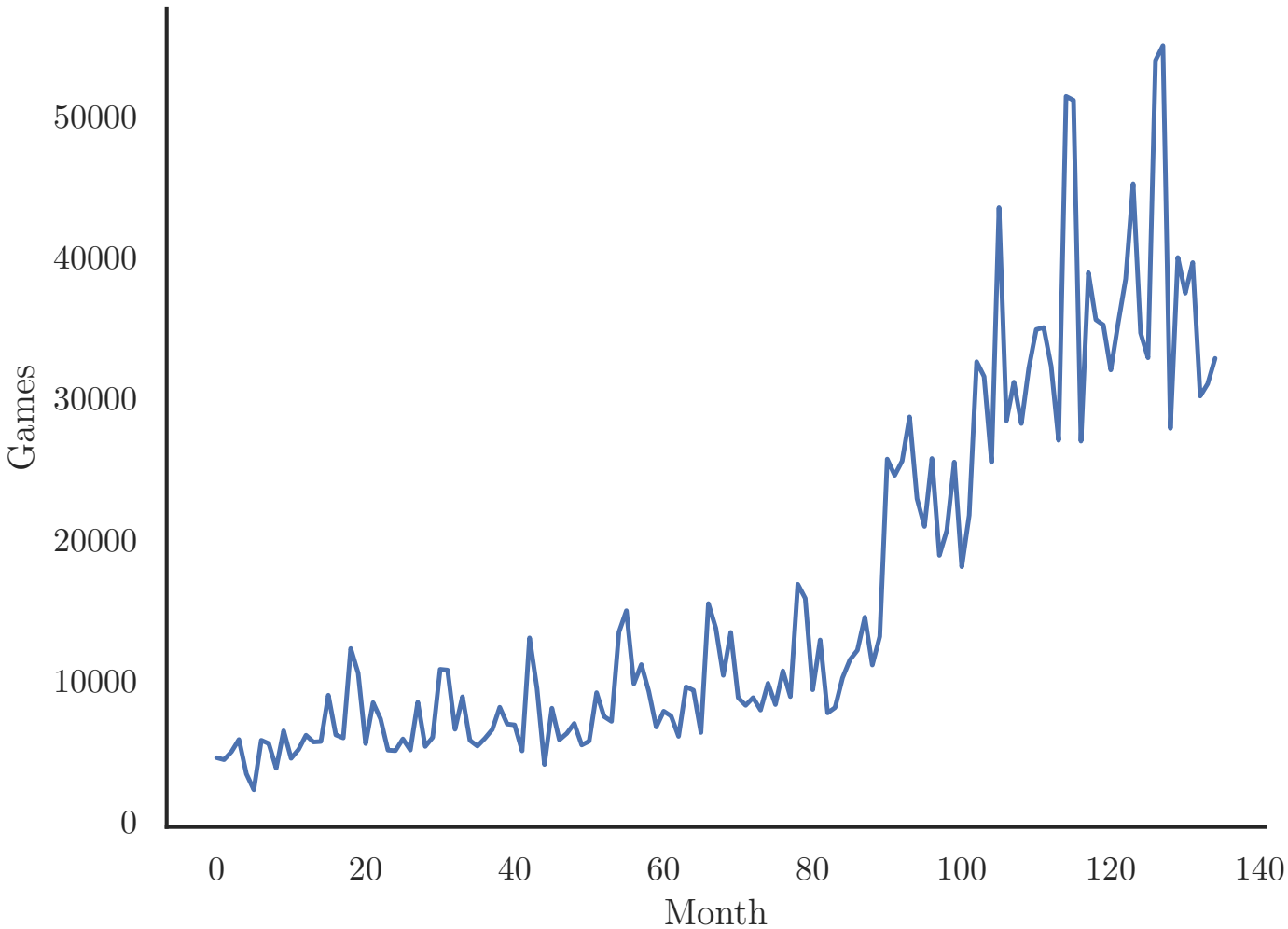

Figure: The number of games per month for the Kaggle data.

Kaggle Data

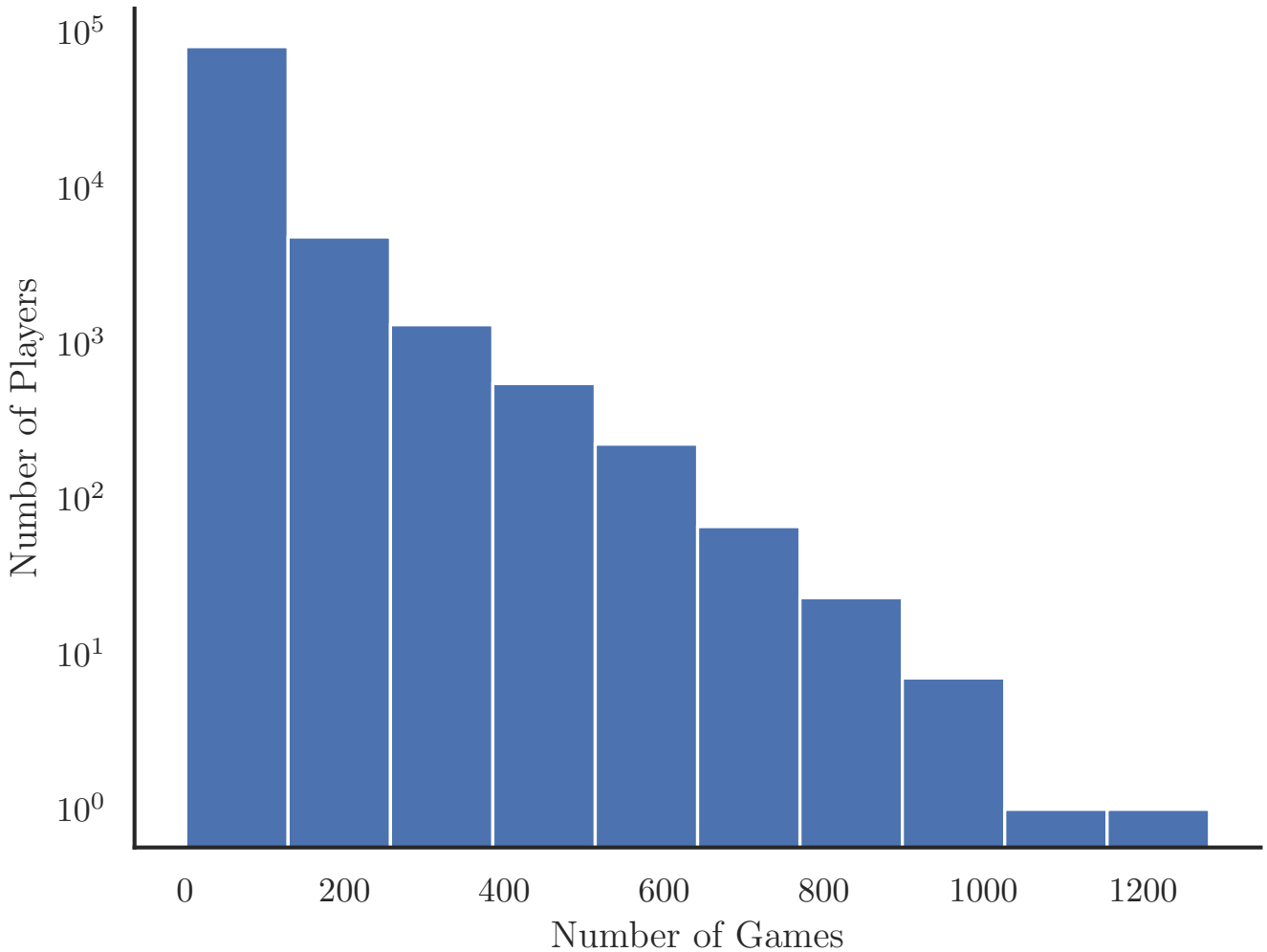

Figure: A histogram of the distribution of games per player.

Kaggle Data

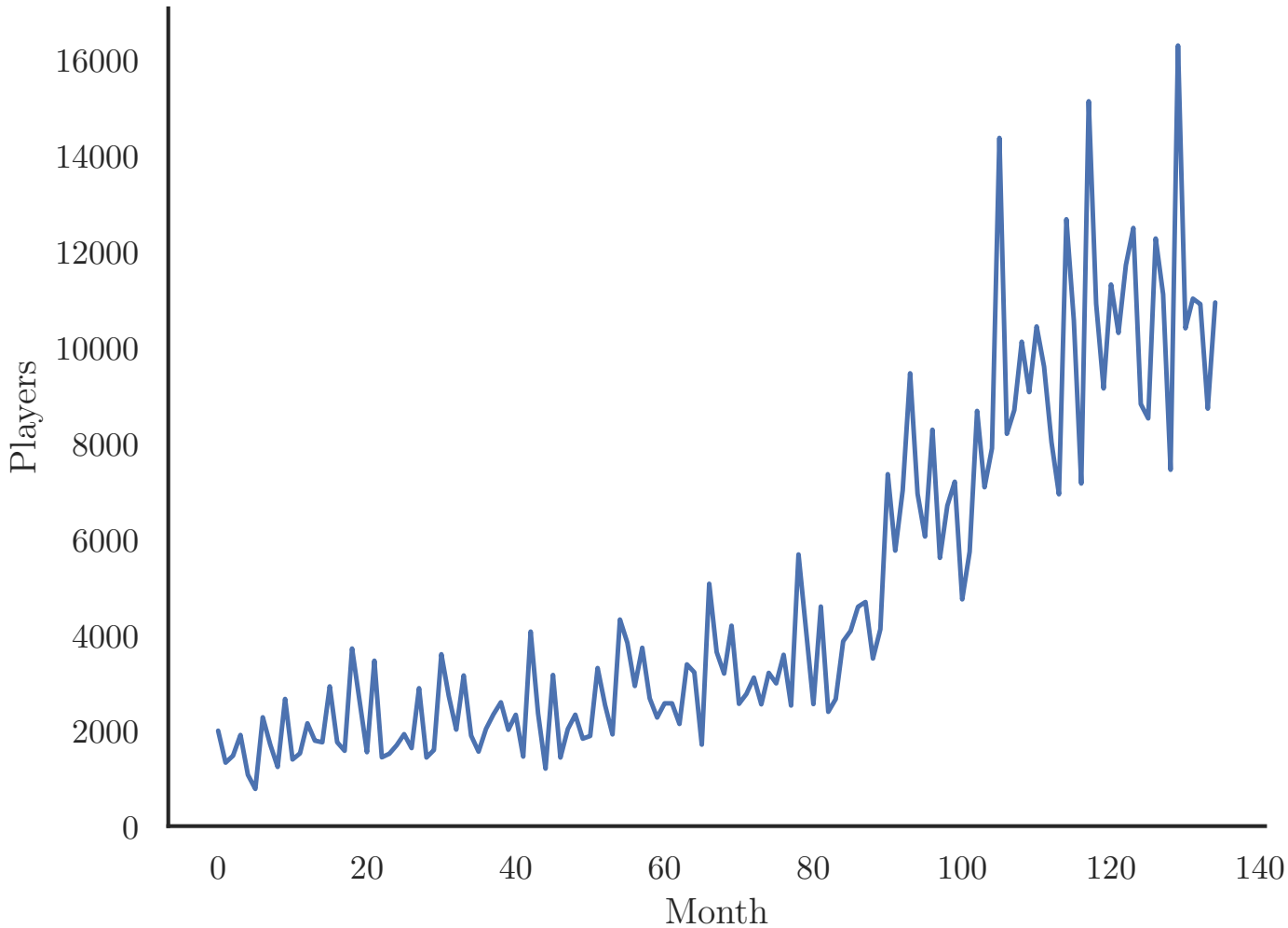

Figure: Number of players entering the player pool each month.

Synthetic Data 1

- Rich with information, unlike real chess data
- 100 players, 100 months
- 800k training games, 200k validation games
- Skill curves drawn from multivariate normal with zero mean and squared exponential covariance
- Players and months sampled uniformly, then game outcome randomly generated using logistic function and pseudo-games
- Tuned for 30% draw rate
- Skills vary quickly

Synthetic Data 1

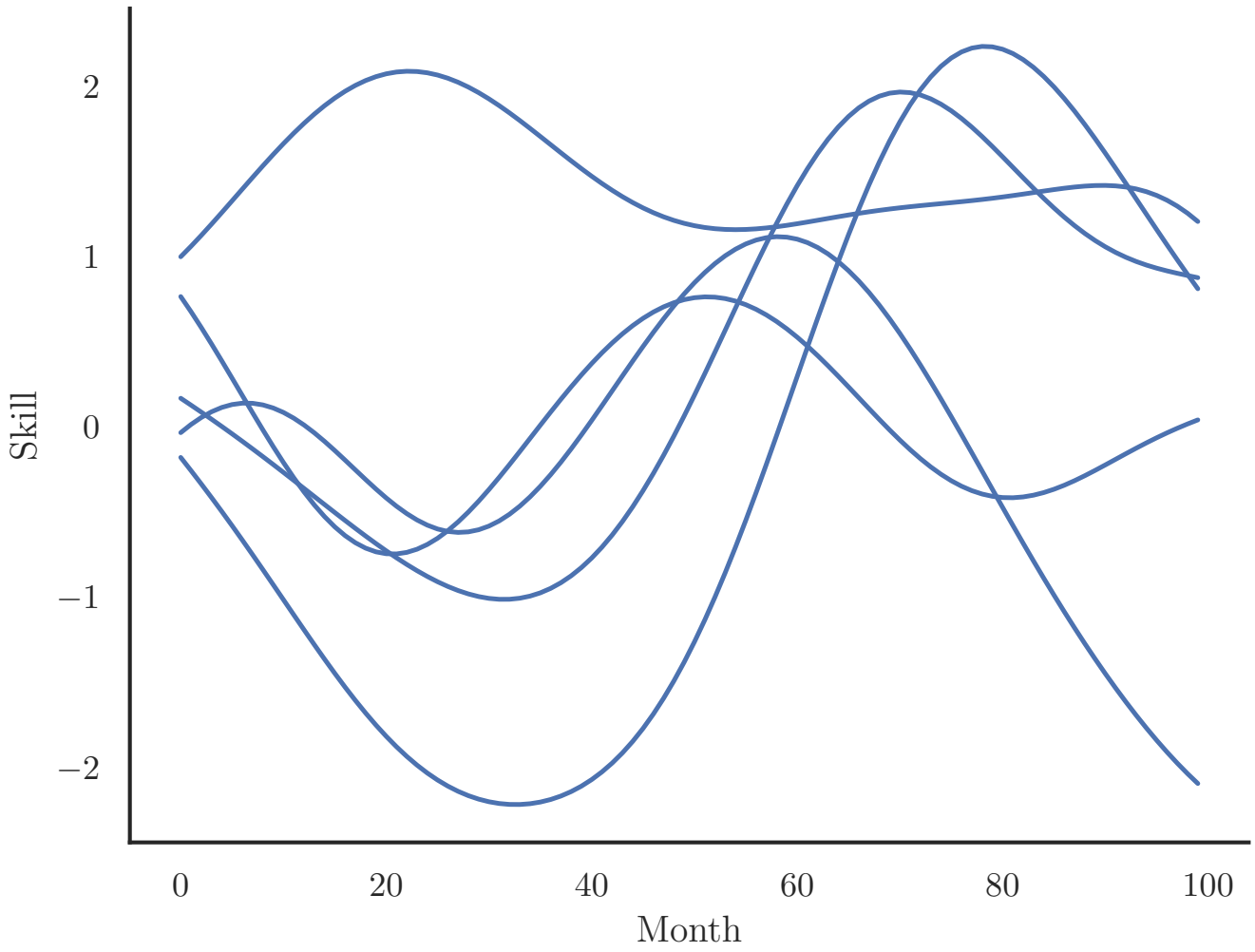

Figure: Example skill curves from Synthetic Data Set 1.

Synthetic Data 2

- Designed to display the weakness of purely prospective systems
- Similar to the first synthetic data set, but with 200 months, and two groups of players
- One group is stronger than the other (larger mean)
- For the first 100 time periods, players only play within their own groups
- For the last 100 time periods, players can play anyone
- Skills vary slowly: there is tradeoff between how much future information can affect the past and how quickly a model can adapt to new information
- May seem contrived, but (semi) closed-pool situations happen because of war and prison for example

Synthetic Data 2

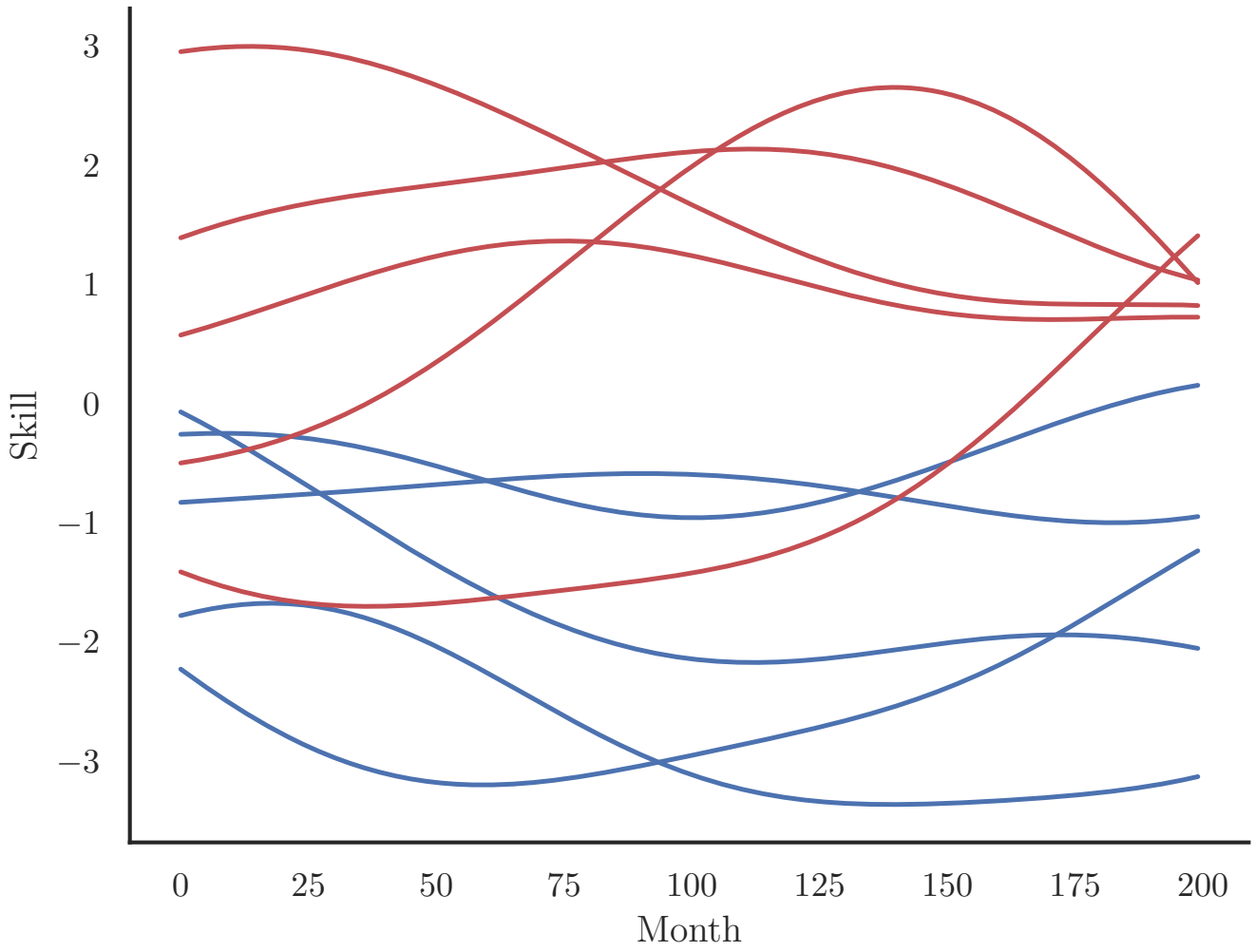

Figure: Example skill curves from Synthetic Data Set 2. Blue curves are from group 1, while red curves are from group 2.

Models Tested

- Tested Elo, Elo regression, and a constant model
- The constant model is Elo regression but with an intercept-only basis
- Equivalently, the constant model limit of Elo regression as $L \to \infty$
- For Elo, used a fix global K-factor optimized on validation data
- For Elo regression, optimized basis centers and length scale on validation data

Measuring Performance

- Use predictive performance on test set
- Used two metrics, binomial deviance and 3-way classification accuracy
- Deviance is proportional to negative log likelihood
- Deviance harshly penalizes confident but wrong predictions
- Accuracy is cruder, since it does not take into account confidence
- Used pseudo-game approach to convert expected score into win, loss, or tie
- Thresholds at 1/3 and 2/3

Results

Deviance

	Kaggle	Synthetic Data 1	Synthetic Data 2
Constant	0.618	0.592	0.449
Elo	0.632	0.455	0.490
Elo Regression	0.611	0.454	0.394

Accuracy

	Kaggle	Synthetic Data 1	Synthetic Data 2
Constant	0.475	0.505	0.663
Elo	0.432	0.670	0.639
Elo Regression	0.487	0.671	0.716

Kaggle Data

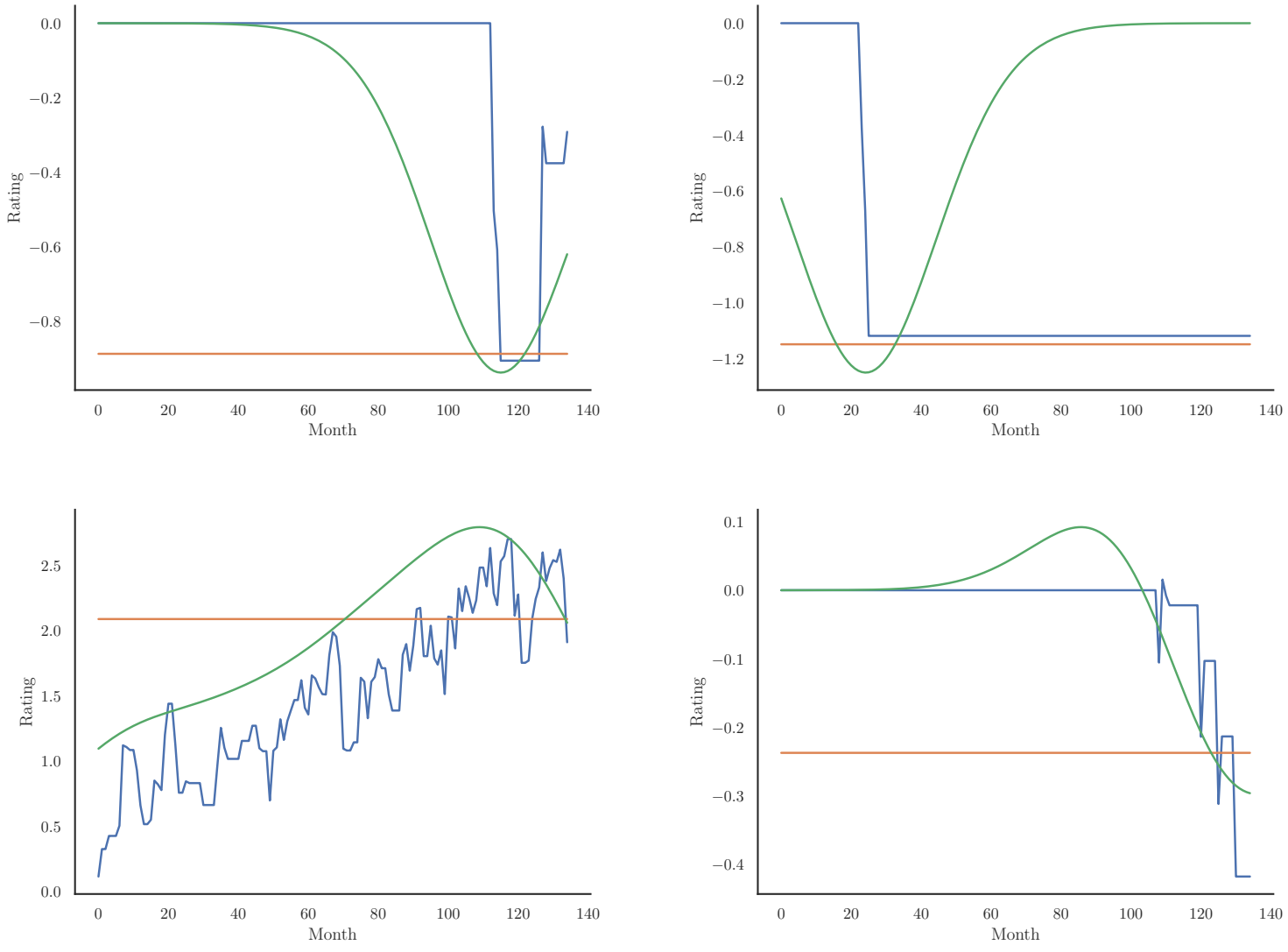

Figure: Some example rating curves for the Kaggle data. Blue is Elo, green is Elo regression, and yellow is constant.

Synthetic Data 1

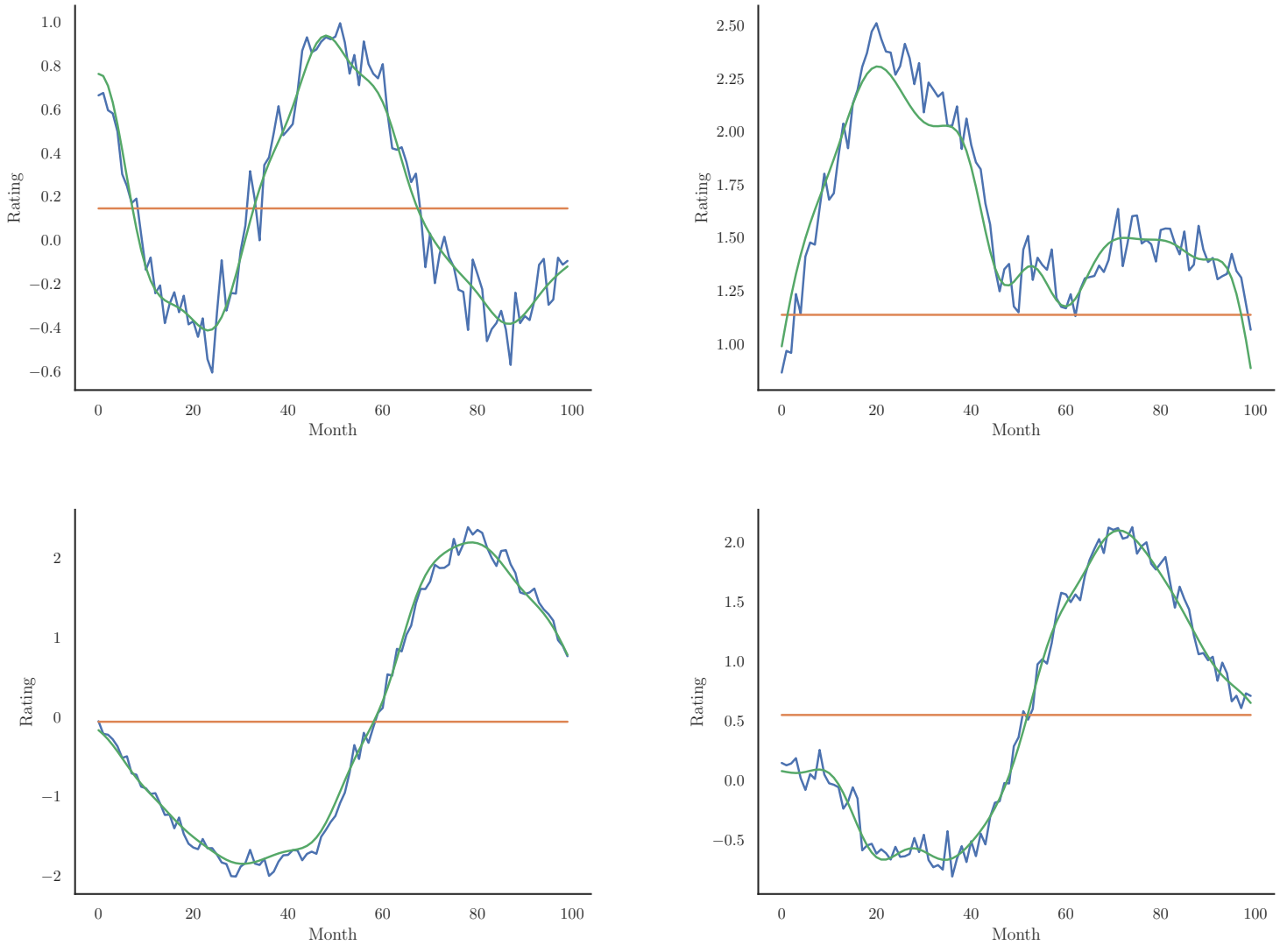

Figure: Some example rating curves for the first synthetic data set. Blue is Elo, green is Elo regression, and yellow is constant.

Synthetic Data 2

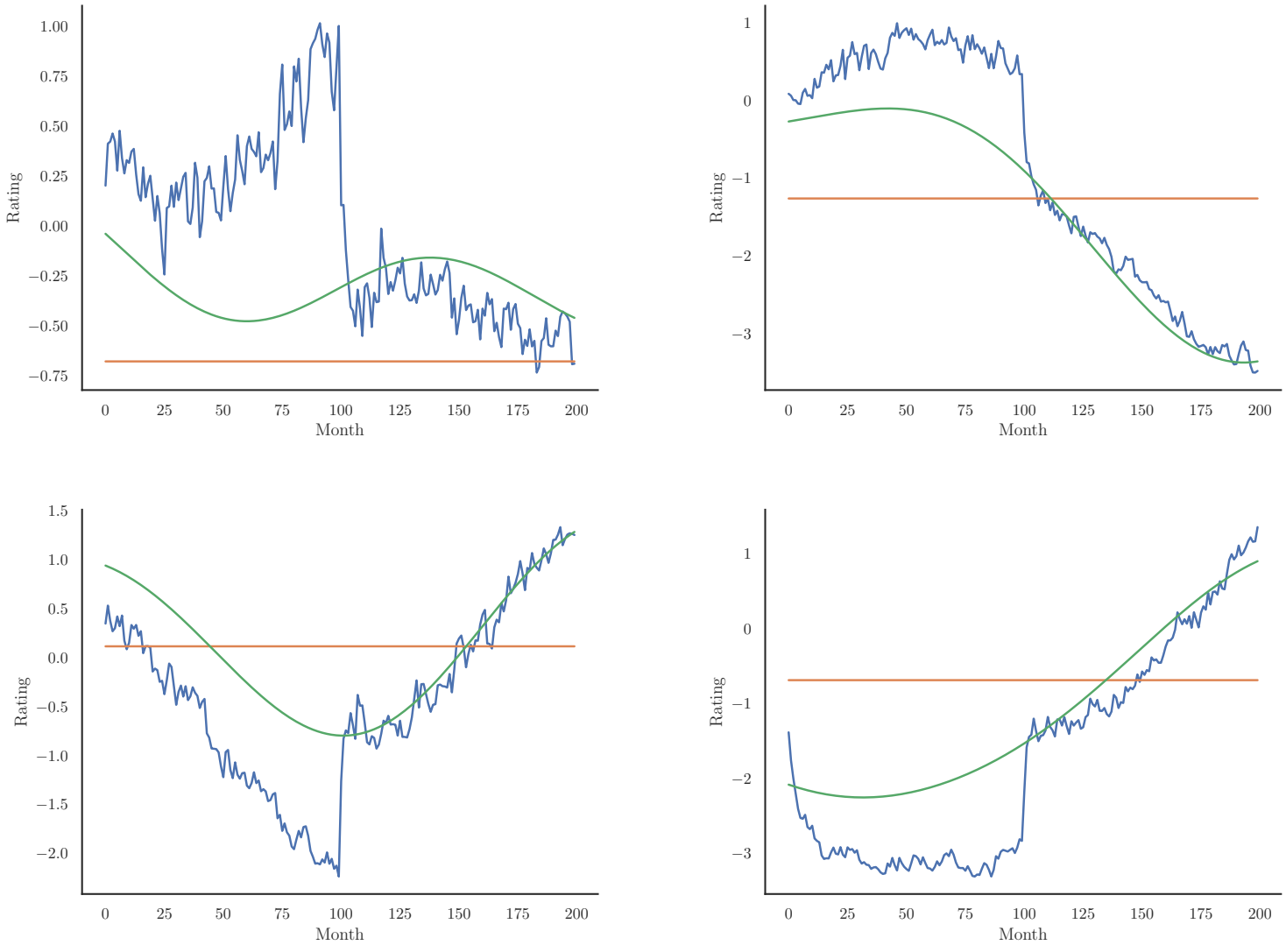

Figure: Some example rating curves for the second synthetic data set. Blue is Elo, green is Elo regression, and yellow is constant.

Conclusion

- Elo regression has superior predictive performance, but Elo scales better
- On the Kaggle data, Elo regression had over 4.5 million parameters and took hours to fit
- In ideal situations like synthetic data 1, virtually no downside to Elo, but real chess data is not ideal
- Again, Elo regression is not going to replace Elo anytime soon
- Neither model was pushed to its limit
- Extensions include: adding more covariates (color), using different bases, swapping out the logistic function for another CDF
- Possibilities are literally endless: once you've understood an algorithm in a probabilistic way, principled modification is easy