Determining Baseball Performance Quality with an Elo Rating

Mac Bagwell

June 12, 2019

Abstract

Recent advances in data collection for MLB games has made quantitative analysis of individual player performance simultaneously easier and richer. However, these advances have been limited to MLB players; in particular, analysis of minor-league players lags behind that of major-leaguers in sophistication, despite the fact that franchises comparatively place greater value on the individual performance of their prospects than on the performance of the teams they happen to be playing on. In this project, we attempt to develop a elo-style rating system, similar to that employed by FIDE for chess players, to function as a point-in-time indicator of skill. In contrast to the FIDE system, we employ a modified version of L_2 regularization, inspired by the winning submission of a Kaggle competition. We also apply this elo system to the task of predicting Baseball America's top prospects for the following season. We see significant improvement over baseline models with the incorporation of an elo rating and other evidence that our rating system contains information about point-in-time performance.

1 Introduction

Professional baseball franchises that form the MLB also own and operate teams in many smaller leagues which are collectively known as the "minor leagues", or MiLB. Usually, newly drafted players are not added directly to MLB rosters but are instead sent to teams in one of these minor leagues. If a player performs well enough during their time in these leagues, the franchise will recall them to their respective MLB club's roster. The MiLB is generally not as visible to baseball fans as the MLB, but they remain important sources of future talent for baseball franchises. Naturally, franchises and analysts often want to consider which MiLB players are performing well and which are not.

It is generally difficult to systematically evaluate MiLB prospects in the same way that we might evaluate major league players. This is because the conditions under which MiLB players are playing are highly variable. More specifically, MiLB players will often develop rapidly or be moved between leagues multiple times over the course of the season. This means that the relative skills of a MiLB player and his opposition are in constant flux. For this reason, common summary statistics like on-base percentage or earned-run average are less reliable indicators of performance; any comparison of two such indicators between different MiLB players is inevitably an apples-to-oranges comparison.

The most common way to ascertain which MiLB players are performing the best are prospect lists published by several different sources, such as Baseball America, FanGraphs, or the MLB.[1, 2] While these do give a fair amount of information regarding which MiLB prospects are likely to appear as successful players in the MLB, they still leave something to be desired in the form of a comprehensive system for player evaluation. The development of such a system is the motivating impetus for this project.

2 Methods

Motivation

In order to rate MiLB players across such a variable landscape, we will use a modification of the Elo rating system. This rating system is popular for its use as a chess rating system, but many variants exist across different competitions.[3] Its use as a tool for analyzing and predicting the outcomes of sporting contests has been popularized by FiveThirtyEight editor-in-chief Nate Silver.[4] Here, we hope to leverage the Elo system as a way of systematically comparing minor-league prospects.

At a high level, the idea behind an Elo ratings system is to keep track of a number called a "rating" for every player we are observing. When two players play a game, we estimate the expected outcome of the game based on the ratings of the players; players with higher ratings are more likely to win over players with lower ratings. We then observe the real outcome of the game and update the ratings of the players. Players who win see an increase in their ratings, and players who lose see a decrease. The size of this increase/decrease will naturally depend on how different the outcome was from the expectation. That is, if a player beats another player of a much higher rating, we should apply a larger update in this case than in the case where a player beats another player of a much lower rating.

We also seek to incorporation information contained in the abilities of a player's opponents. In particular, we expect similarly skilled opponents to be competing together most often. This will be implemented in the form of L_2 regularization that will be discussed later.

Definitions

Here we define terminology and notation to be used for the remainder of the paper.

- Let P be the set of all players in our universe.
- For every $i \in P$, let r_i denote the "rating" of player i.
- For every $i \in P$, let a_i denote the exponentially weighted moving average of all of i's opponents.
- Whenever a batter i makes a plate appearance against pitcher j, let o_{ij} be an indicator for a ball-in-play (BIP), or a hit or walk; that is, $o_{ij} = 1$ if i gets a hit or walkds and 0 otherwise.

Update Rule

The process for maintaining and our modified Elo rating system is as follows. It is largely modeled after and inspired by the winning submission from the "Chess Ratings — Elo vs the Rest of the World" Kaggle contest.[5]

1. For every player $i \in P$, $r_i, a_i \leftarrow b$, where b is some base rating chosen arbitrarily.
2. For every plate appearance,

(a) Calculate the probability that the batter i gets on base against the pitcher j as

$$p = \frac{1}{1 + \exp\left(-(r_i - r_j) + \gamma\right)}$$

where γ is a bias term which is set according to any prior asymmetries in each player's win probability. In this case, we set γ so that when $r_i = r_j$, p is the average on base percentage. On-base percentages across all leagues have historically hovered around 0.333[6], so we set $\gamma \approx \log 2$ for ease of calculation.

(b) Update r_i, r_j, a_i, a_j as

$$\begin{aligned} r_i &\leftarrow r_i + \alpha\left((o - p) - \lambda(r_i - a_i)\right) \\ r_j &\leftarrow r_j + \alpha\left(-(o - p) - \lambda(r_j - a_j)\right) \\ a_i &\leftarrow \beta r_j + (1 - \beta)a_i \\ a_j &\leftarrow \beta r_i + (1 - \beta)a_j \end{aligned}$$

where $o = 1$ if i gets on base and o otherwise λ, β are hyperparameters to be tuned.

The update rule here bears a striking similarity to stochastic gradient descent. Indeed, one can view the Elo ratings system as a very primitive classifier which predicts o_{ij} given i and j, in which the hypothesis is given by

$$h(i, j) = \frac{1}{1 + \exp(-(r_i - r_j) + \gamma)}$$

and the loss for a single training example is given by the log-loss

$$\begin{aligned} \mathcal{L} = (o_{ij} - \log h(i, j)) + (1 - o_{ij}) \log(1 - h(i, j)) \\ + \frac{\lambda}{2}[(r_i - a_i)^2 + (r_j - a_j)^2] \end{aligned}$$

for some regularization constant λ. Here, we now see the effect of the regularization. Since we expect players to be playing other players of a similar skill level, when we update the rating of a player, we regularize the update towards the EWMA of all of his previous opponents.

The difference between the stochastic gradient descent update rule and the ratings system described here is that instead of shuffling the training examples (in this case, plate appearances) randomly, we perform updates based on the temporal order of the plate appearances. This way, the resulting system will more easily capture variations in the skill of a player over time.

Data

We scraped play-by-play data from the MLB's GameDay servers from all games played in the MLB, AAA, AA, High-A, A, Short-season A, and Rookie leagues since January 1, 2007. In particular, we determined the following data from every plate appearance:

- The date of the game, `date`.
- An ID number for the batter and pitcher
- The league the game was played in, `league`.
- A 0-1 variable corresponding to a batter success, `event_cd`.

We also used data containing player names as well as professional and MLB debut dates taken from the Chadwick Baseball Bureau. [7]

For evalutation purposes, we are using a dataset of 40977 pitchers who pitched in the MiLB during the 2007 season or later. The dataset contains data about the pitchers' draft round and historical prospect status. This data was taken from The Baseball Cube and provided to us graciously by Pando Pooling.

Each row in this dataset is an observation of a pitcher at the end of the year. In each row, we observe the draft round that the pitcher was drafted in and whether or not the pitcher made Baseball America's Top 10 Prospects list for their organization or the league-wide Top 100 Prospects list.

Experiment

We will calculate the Elo ratings of all American professional baseball players starting at the beginning of 2007. We will also consider initializing players in different levels at different ratings. More specifically, we will define a hyperparameter O so that new players observed i levels below the AAA level will be initialized at rating $100 - iO$.

We will select the optimal values of K, β, and λ by optimizing for pre-update log-loss (that is, the log-loss of $h(i, j)$ immediately preceding the updates to r_i and r_j). The hyperparameter sweep will employ an "early-out" grid search; after training on a season, we will halt training on the worst 50% of grid-searched rating systems as measured by log-loss. This will continue until we are left with only one set of hyperparameters, which we will consider optimal.

To test the efficacy of our ratings system, we will then construct two logistic regression classifier that predicts whether or not an MiLB player will be listed as one of their organizations "Top 10 Prospects" by Baseball America before the following season. We will consider the following features:

- t_i: equal to 1 if player i was in the "Top 10 Prospects" list in the previous year for their team and 0 otherwise.
- l_i: equal to 1 if player i was in the "Top 100 Prospects" list in the previous year for the league and 0 otherwise.
- $a_{i,k}$ for $1 \leq k \leq 5$: equal to 1 if player i was a k-th round draft pick and 0 otherwise.
- y_i: equal to the number of seasons player i has played
- r_i: the Elo rating of player i at the end of the season.

and we will attempt to predict o_i, equal to 1 if player i makes his organizations Top 10 Prospects list released at the beginning of the next year.

The features other than r_i listed above are intended to encompass as much information about a player as possible that is unrelated to their current season's performance. For example, draft round information gives lots of information about how well that player is expected to play, but is not necessarily related to his performance during a season. This selection of features will allow us to examine the usefulness of the Elo rating in this context in as much of a vacuum as possible.

The baseline model will be of the form

$$h_0(i \mid t_i, a_{i,1}, \ldots, a_{i,5}, y_i, r_i; \theta) = g(\theta_0 + \theta_t t_i + \theta_{a_1} a_{i,1} + \ldots + \theta_{a_5} a_{i,5} + \theta_y y_i)$$

where $g(z)$ is the sigmoid function $1/(1+e^{-z})$. We will compare this to the model

$$h(i \mid t_i, a_{i,1}, \ldots, a_{i,5}, y_i, r_i; \theta) = g(\theta_0 + \theta_t t_i + \theta_l l_i \theta_{a_0} a_{i,0} + \ldots + \theta_{a_5} a_{i,5} + \theta_y y_i + \theta_r r_i)$$

where r_i denotes the rating of player i as of December 1 on the year that the observation of player i is made.

We will divide the dataset into a training set, a dev set, and a test set according to a 70-10-20 split. For both models, since only about 3% of the training examples have positive labels, we will also weight the classes inversely proportional to their frequencies (i.e. positive samples will have weight $\approx 1/0.03$ and negative samples will ahve weight $\approx 1/0.97$). Both models will be constructed and trained using the `sklearn` module of Python 3.6. Due to class imbalance, we will score the models on area under the precision-recall curve (AUPRC).

Table 1: Grid-Searched Hyperparameters

K	λ	β	O
0.1	0.1	0.95	0
0.01	0.01	0.9	0.125
0.001	0.001	0.85	0.25
0.0001	0.0001	0.8	0.5

Table 2: Hyperparameters and Loss of Optimal Model

K	λ	β	O	log-loss
0.01	0.01	0.9	0	0.623341

3 Results

Ratings System

Table 1 shows the values that were grid-searched for each hyperparameter. Table 2 shows the values of the optimal hyperparameters and the associated average log-loss. It is surprising that the optimal value for O was 0. This seems to suggest that the level that a new player appears in does not contain any significant information about how well that player performs. However, considering the scale of the elo ratings, it appears that the values of O that were searched could simply have been too large to be an improvement over $O = 0$.

The elo ratings obtained seem generally to correspond to better players. Figure 1 shows historical elos for an established MLB pitcher, Kyle Hendricks, a pitcher who recently debuted in the MLB, Yancy Almonte, and a short-season A regular, Jeferson Mejia. Figure 2 shows the distribution of elo scores in the training data by class label. Figure 3 shows the mean elo rating by level on Jan. 1, 2018. Al figures illustrate separation between high qualiity players and low quality players, which indicates that the elo rating contains some useful information about a player's performance.

Models

Table 3 shows the AUPRC scores of both logistic regression models on the dev set and the test set. Figure 4 shows the PRC curves for both classifiers on the test set. A cursory examination of these scores suggests that the addition of an elo term modestly increases AUPRC scores, but not by much.

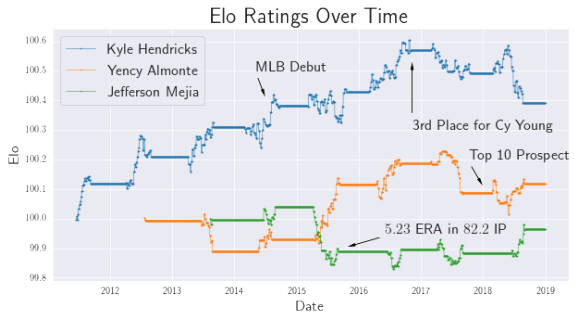

Figure 1: Sample Elo Ratings for Three Profesional Baseball Players: Kyle Hendricks (Blue), Yancy Almonte (Orange), and Jeferson Mejia (Green)

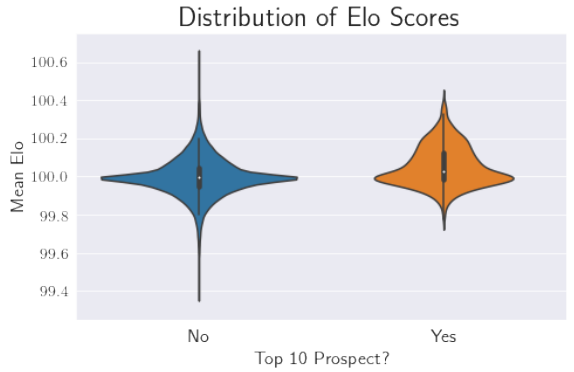

Figure 2: Distribution of Elo Ratings by Class Label

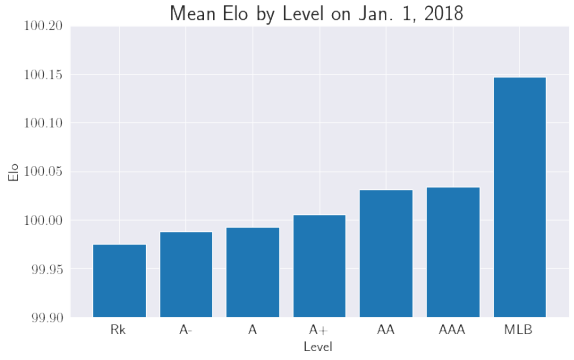

Figure 3: Mean Elo Rating by Level

Table 3: AUPRC Scores for Logistic Regression Classifiers

Model	Dev AUPRC	Test AUPRC
h_0 (baseline)	0.3941	0.4136
h	0.4141	0.4234

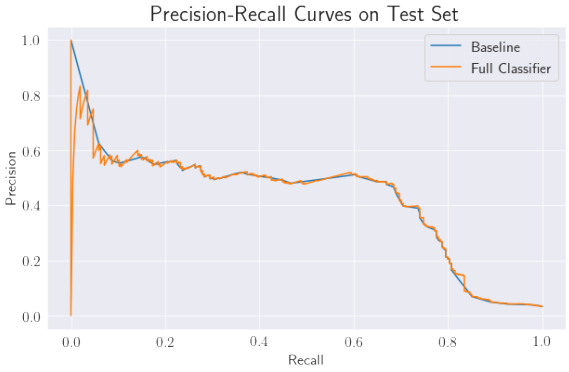

Figure 4: PRC Curves on Test Set

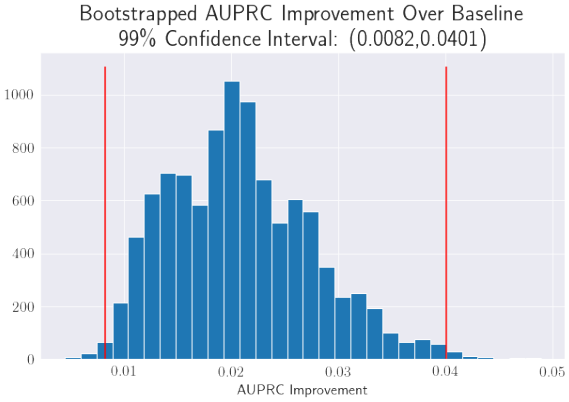

Figure 5: Improvements of Bootstrapped Classifiers over Baseline

To test the significance of this increase, we generated 100,000 bootstrapped samples of the training data and observed the increase in AUPRC of h over h_0 on the dev set after training on each bootstrapped sample. The histogram of those improvements is shown in figure 3. The 99% confidence interval does not contain 0, suggesting that this increase is significant with fairly high certainty.

4 Conclusion

Summary

In order to develop a general method for evaluating baseball player quality, we developed a modified elo rating system that minimized the log-loss of its predicted probability of a ball-in-play. The resulting statistic appeared superficially to agree with subjective measures of player quality; for example, average elo ratings tended to increase with minor league level, and players listed as top prospects tended to have higher elo ratings than other MiLB players.

Our specific application of elo to classifying Top 10 Prospects was met with limited success. While a modest increase in AUPRC over our baseline classifier was observed after adding an elo term to the model, the size of that increase was small, and did not change the classifiactions of the model at all. However, this still further suggests that the elo rating developed here contains potentially actionable information about the point-in-time quality of a baseball player's performance.

Areas for Improvement

We did not necessarily expect dramatic improvements over our baseline model. Our method of incorporating elo into our logistic regression model was certainly *ad hoc*, and there is significant room for improvement with regards to this specific application.

The most limiting aspect of this experiment was the hyperparameter sweep. Even when parallelizing on a 16-core virtual machine and implementing the early-out approach, conducting a sweep of 4 values for each of 4 hyperparameters was a 10-hour task. This leaves the hyperparameter optimization wanting. Our recommendation for someone who wants to move forward with implementing this elo rating system is to conduct a more thorough hyperparameter search.

Related to hyperparametrs, the value of γ in our elo rating system was fixed at $\log 2$ and was not optimized. We did this because historical on-base-percentages across all levels of professional baseball have hovered around 0.333 for the past decade. However, there may be some other optimal value, or it might be the case that varying γ accoridng to past observations yields improvements.

Finally, our use of BIP as a binary indicator for success/failure of the batter does not allow us to fully capture the value of events such as extra-base-hits or home runs. It may be possible to smartly weight plate appearance outcomes so that home runs and extra-base-hits lead to more drastic changes in elo than singles or walks.

References

1. Baseball America. *Baseball America Home Page* 2019. `https://www.baseballamerica.com/rankings/` (2019).
2. FanGraphs. *FanGraphs Prospect Coverage* 2019. `https://www.fangraphs.com/prospects/`.
3. Wikipedia. *Elo rating system* 2019. `https://en.wikipedia.org/wiki/Elo_rating_system` (2019).
4. Silver, N. *2019 MLB Predictions* 2019. `https://fivethirtyeight.com/features/how-our-mlb-predictions-work/`.
5. Sismanis, Y. How I won the "Chess Ratings - Elo vs the Rest of the World" Competition, 1–8 (2010).
6. BaseballReference. *2016 Minor League Encyclopedia* 2019. `https://www.baseball-reference.com/register/league.cgi?group=Minors&year=2016` (2019).
7. Bureau, C. *Baseball Data Bank* 2019. `http://chadwick-bureau.com/open-data/`.

Source Code

All python scripts for this project can be found at `https://github.com/bagalaster/cs229-project.git`.

Acknowledgements

Thanks go out to Eric Lax, Jonathan Armitage, and Pando Pooling for supporting this project and for providing the MiLB pitcher prospect dataset used to train and evaluate our models.