Prediction of Future Offensive Performance of MLB Position Players

Susana Benavidez susana@cs.stanford.edu
Stephanie Brito sbrito@stanford.edu
Derek McCreight dmccreig@stanford.edu
Peter McEvoy pmcevoy@stanford.edu

December 13, 2019

1 Introduction

Major League Baseball (MLB) is comprised of thirty professional teams, each of which is valued between $1 billion and $4.6 billion USD. The annual revenue for any given team is in the hundreds of millions of dollars, around half of which is paid to players in the form of salaries. The minimum annual salary in 2019 was $550,000 [1], and the top players earn well over $30 million per year. Because of the market value of MLB players, a team's decision over which players to sign to a contract and which to let go is crucial to the success of the team both athletically and financially [2]. Unfortunately for teams, the prediction of a player's future value is difficult[3], and there are many instances of players earning much more than their performance warrants [4].

In our literature review we did not come upon any work using LSTM/RNN to predict individual athlete performance. In light of this, we set out to predict the value of an MLB position player–a player who does not pitch–in the next year given the player's performance over the previous n years.

We trained multiple models on data from the 2005 - 2015 seasons and validating on data from 2016 - 2018[5]. We also built individualized LSTM models structuring it as a time series forecasting problem.

2 Literature Review

Barnes et al [6] developed two classes of models, performance and market, to assess the market value and wins above replacement (WAR[1]), respectively, of a player entering free agency. They trained models on both hitters and pitchers. For hitters, they discovered that 30 of their features displayed high correlation, which prompted them to remove those features. They additionally filtered out hitters with insufficient playing time by imposing a 130 at-bat minimum per season. After doing so, their number of training examples was small enough for them to use 5-fold cross-validation. The algorithms that they used include linear regression, linear regression with feature selection, regression trees, and gradient boosted trees. Their feature selection methods included forward selection, backward elimination, correlation feature selection, feature selection motivated by random forest variable importance, and regularized regression with elastic nets. To evaluate performance, they used cross-validated root mean squared error.

Bierig et al [7] selected players who played in at least 7 seasons since 1970. They used per-season and aggregate data from each player's first 6 seasons along with handedness, debut age, and position to predict the player's (Baseball Reference) WAR. Each training example had 200+ features, but many were highly correlated. Near-optimal model performance was achieved with per-season and cumulative WAR from a player's first 6 seasons. The four models that they trained were linear regression with L2 regularization, multi-layer perceptron regression, random forest regression, and ϵ-support regression. For multi-layer perceptron regression, neural networks of 1 and 2 layers were used along with the ReLU activation function. The loss function used was sum of squared error. For random forest regression, 100 decision trees were created via bootstrapping and then averaged together. The Gaussian kernel was used in ϵ-support vector regression.

Previous research by Panda [8] relied on methods such as principal component analysis **(PCA)** with penalized linear regression to determine which offensive metrics were most indicative for predicting performance and overall ability throughout a player's career. Panda opted to use 45 unique offensive metrics–a subset of the statistics used were at-bats (AB), Batting average on balls in play (BABIP), and base on balls (BB), among others. However, Panda found that many of the selected features were highly correlated and instead relied on PCA to generate a new set of uncorrelated variables that could then be used to capture both the variance of the data while simultaneously being uncorrelated amongst one another. Each principal component Y_i is simply a weighted sum as follows:

[1] See section 3.1 for a description of WAR.

$$Y_i = \begin{bmatrix} c_{i1} & c_{i2} & c_{i3} & \dots & c_{ik} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_k \end{bmatrix}$$

$$Y_i = c_{i1}x_1 * c_{i2}x_2 * c_{i3}x_3 * \dots * c_{ik}x_k$$

where Y_1 is a linear combination of the original data and each variable weight c_{ij}. For example, a weight that is greater in magnitude indicates the respective variable provides more information and predictive power for that particular component.

Of the k principal components that were generated, only a subset was selected as being good indicators of performance. Panda used scree plots, which represent the cumulative total variance of a component, with a variance proportion threshold of approximately 0.80 in order to decide which of the principal components were the least helpful for this particular task.

Using PCA and penalized linear regression, Panda was able to reduce the dimensionality of the features from 45 distinct features to just 7 uncorrelated components with higher predictive potential.

3 Methodology

3.1 Data

We used data pulled from baseball-reference.com, which contains over 40 statistics for players dating from 1871 to present. The following statistics, among others, are in the data[2]:

- Plate appearances (PA)
- Games played (G)
- Innings played (Inn)
- Defensive wins above average (WAA_def)
- Offensive wins above average (WAA_off)
- Wins above average (WAA)
- Defensive wins above replacement (WAR_def)
- Offensive wins above replacement (WAR_off)
- Wins above replacement (WAR)
- Normalized and adjusted on-base plus slugging (OPS_plus)
- Salary (salary)

Many of the features in the Baseball Reference dataset are highly correlated. Take plate appearances (PA)–the number of a times that a player comes to bat over the course of a season–and games played (GP)–the number of games that a player appears in–as an example. Clearly, the more games that a player appears in, the more times he will come to bat. Similarly, defensive wins above replacement and offensive wins above replacement are correlated with wins above replacement since the former two are subsets of the latter.

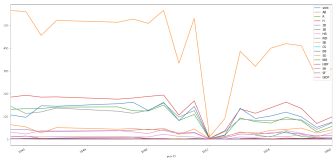

Figure 1: Plot of Ted Williams Offensive Metrics

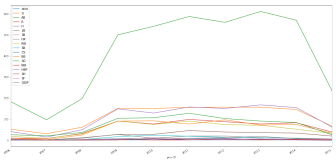

Figure 2: Plot of Ben Zobrist Offensive Metrics

Another aspect of the dataset worth commenting on is the inconsistency in player data. Some players appear in only one or two seasons and record a handful of plate appearances. Others appear in twenty consecutive seasons and amass thousands of plate appearances. Others still drop out of the MLB at times and return, which creates gaps in these players' records. There were many zeroes to handle (and many which cropped up in unexpected places) in the data, which required significant data cleaning effort.

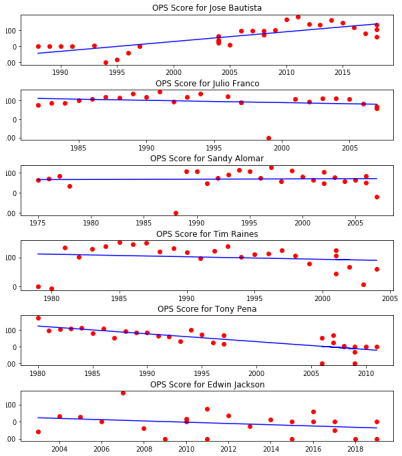

Figure 3: Linear Regression fit to OPS+ metric

3.1.1 Data preparation

We first cleaned our data to remove NaN values, which were the result of players with gaps in their careers (i.e. because of an injury or serving in World War II). We filtered out players with less than 16 years of playing experience because we wanted to train on 15 years of playing data. We also filtered out players who weren't hitters. After selecting players that fit the criteria, we had approximately 450 players with correctly formatted metrics. We learned a model for each player.

[2] See Appendix 1 for a description of all statistics present in the dataset

3.2 Metrics

In terms of player performance, the metrics deemed most relevant by the baseball community are WAR and OPS+. WAR indicates how many more games the player's team won because of that player's performance. A WAR of 3.0, for example, would indicate that the team would lose three more games if the player was replaced for a replacement level player[9]. WAR can be thought of as the metric that encapsulates a player's total performance–hitting, base-running, and defense. OPS+ is the league-normalized and adjusted version of OPS–on-base percentage plus slugging percentage [10]. OPS is the sum of on-base percentage–the percentage of plate appearances in which a player reaches base–and slugging percentage–the average number of bases that a player records per at-bat. OPS+ can be thought of as the metric that encapsulates a player's pure hitting performance.

We formulated this problem as a regression task where the output of our model is the input player's OPS+.

3.3 Linear Regression

3.3.1 Unregularized

We used multiple, unregularized linear regression in order to create a line of best fit for the OPS+ data over a player's career. This is equivalent to solving the Least Squares Objective Function for our data which is described by the following linear equation:

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{1}$$

where $\hat{\beta}$ is the desired linear approximation for our dataset and where $(X^T X)^{-1} X$ is the Moore–Penrose pseudoinverse. The Least Squares Approximation requires this pseudoinverse, as the design matrix X may not actually be an invertible matrix. In other words, a closed solution for the linear system may not exist.

3.3.2 Regularized

Similarly to the unregularized case, regularized Linear Regression also attempts to create a line of best fit to the data. In the regularized case, however, the cost function is altered in order to reduce the likelihood of over-fitting to the dataset as follows:

$$J(\theta) = \frac{1}{2n}[\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{m} \theta_j^2] \tag{2}$$

where λ is the regularization parameter. Setting the regularization term to be larger has the effect of penalizing the loss function by the squared sum of the weights.

3.4 Support Vector Regression

As a baseline experiment with our data, we used support vector regression (SVR). SVR is similar to support vector machines in that it maps the data into a higher dimension; for SVRs, instead of classification, data is mapped to a higher dimension to construct a hyperplane that contains the most points to then perform a linear regression. We used the **Gaussian Radial Basis** kernel for the SVR as it did the best job in mapping the data. The formula is described below:

$$\varphi(r) = e^{-(\epsilon r)^2}, \quad where \;\; r = ||x - x_i||_2 \tag{3}$$

Also note, that in our case ϵ is a shape parameter that allows the data to be scaled.

We also compared Gaussian Radial Basis kernel with a Linear kernel and a Poly kernel; we found that the Gaussian kernel was the best of the three kernels.

3.4.1 SVR Hyperparameters

There are several hyperparameters for SVR. The first is γ, which is the kernel coefficient for RBF.

$$K(x, x') = exp(\gamma \left\|x - x'\right\|^2) \tag{4}$$

The second is ϵ, which defines the size of the epsilon-tube (the hyperplane to constrain the data to). Finally, there is C, which is the regularization parameter.
To handle hyperparameter tuning for the SVR, we simply used sklearn's GridSearchCV which tests across a set of hyperparameters and chooses the best combination.

C	{1,10,10,100,150,1000, 10e5}
γ	{1e-8, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e2, 1e8}
ϵ	{0.1, 1e-4, 1e2}

Table 1: Hyperparameters for SVR

Average Mean Squared Error	0.088
Average Absolute Error	0.232

Table 2: Average error for support vector regression on all data

3.4.2 Results

The SVR model performed well on players with reasonable variations in OPS+. Players who varied significantly year-to-year or who peaked early or late in their careers followed by a sudden dip produced high absolute errors. The most important aspect of SVR that is lacking in linear regression is the ability to produce a non-linear prediction curve, which is the

result of mapping the data into a higher dimension. Since a player's career path is not linear, the ability to produce such non-linear prediction curves is imperative. Figure 4 illustrates this phenomenon.

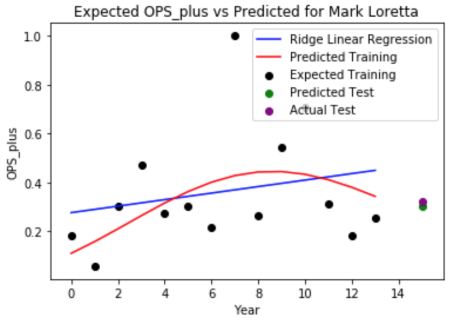

Figure 4: SVR with low error that can properly extrapolate future data. Notice the general increase to the peak in year 7 followed by the general decrease to year 15.

3.5 Recurrent Neural Network with Long Short-Term Memory

We approached the question: predicting a player X's performance based on player X's past performance by modeling it as a Time Series Forecasting with the Long Short-Term Memory problem using root squared mean error [11]. In order to maintain the features for each players, we implemented a **Multivariate LSTM** model. We then applied Multivariate LSTM model to all players to be able to predict player X's performance–measured in terms of OPS+–in year $n+1$ given the previous n years of performance.

3.6 Results & Analysis [3]

When we first defined our problem statement, we expected to learn a model that would be useful in evaluating the future performance of a player. We envisioned a model that would accept the previous n years of offensive hitting statistics of player x and give us the x's predicted OPS+ in year $n+1$, even if the year $n+1$ was in the future. In other words, we wanted to make future predictions on players. If accurate, a model with such capabilities would be of immediate use to MLB teams as they go about projecting player performance into the uncertain future to build the best roster possible.

Unfortunately, our model did not end up being so robust. We required the OPS+ for year $n+1$ in order to train our model. Figure 6 depicts the LSTM inputs and outputs: we fed in players in the training set, each of which had hitting metrics for n years, and outputted OPS+ for each player in year $n+1$. As seen in Figures 7 and 8, the model managed to successfully predict OPS+ for many of the players. That said, our measures of statistical success are lacking. We suspect

[3]Link to Github repo: https://github.com/LACoderDeBH/CS229-Final-Project.

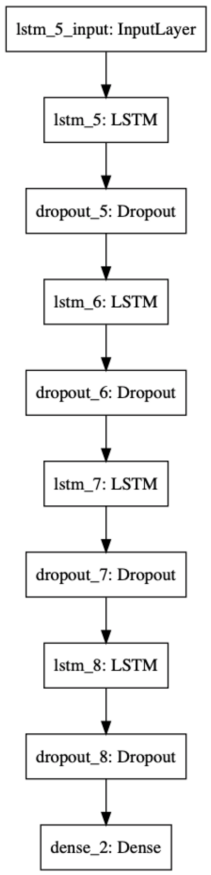

Figure 5: Stacked LSTM architecture

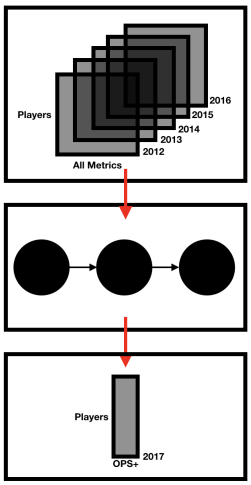

Figure 6: LSTM work flow

that the reason for the poor performance is due to a combination of the variability in player performance and insufficient features. A player's performance in a given year can't be explained solely by previous years but instead by a combination of on-the-field performance, physical health, mental health,

and the like.

	Training Set	Test Set
R^2	0.49	-0.51
Mean Squared Error	0.0035	0.0031
Mean absolute error	0.036	0.041

Table 3: LSTM results

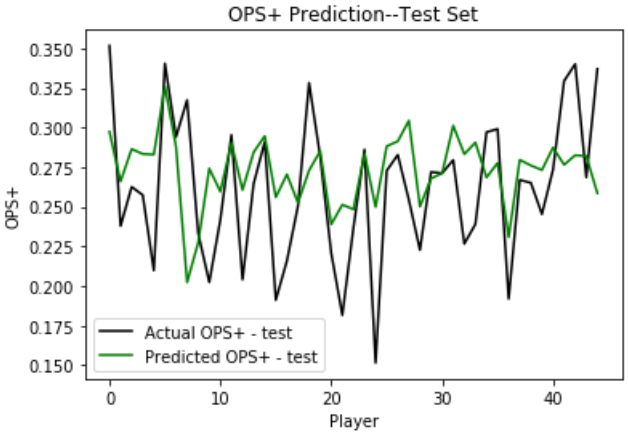

Figure 7: LSTM OPS+ Prediction Vs. Actual

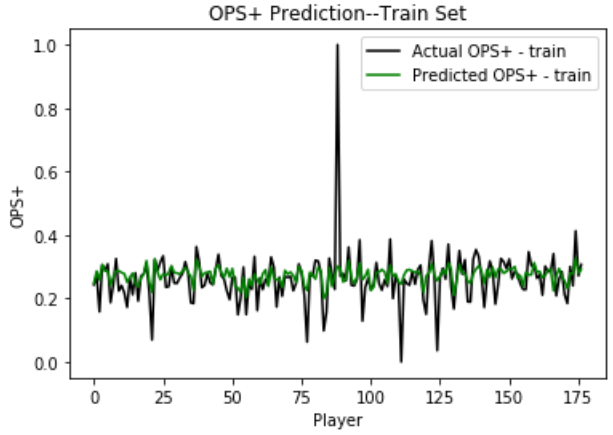

Figure 8: LSTM OPS+ Prediction Vs. Actual

3.7 Next Steps

There are three big categories in which we'll be looking to further our work: iterative model design and architecture, data engineering, and error analysis. We recently discovered a database of play-by-play outcomes for all MLB games at Retrosheet. We're interested in incorporating a more finely-grained set of features into our existing feature set and observing the impact. Leveraging Retrosheet will require scraping the data and formatting it appropriately; each year of play-by-play logs is split into multiple hybrid .csv files, not all of which contain relevant information to our task. Additionally, we will work towards a comprehensive error analysis in order to discern the underlying patterns between different player data.

3.7.1 News Analysis

The SVR model performed worst when a player dramatically increased or decreased their performance. Analyzing players where our model failed shows that injuries (as expected) can greatly affect a player's future performance. When these external factors come into play, our model extrapolates poorly. A potential solution to this would be to use reported injuries to weigh "down" previous OPS+ scores. For example, if we knew data from 2012-17 and wanted to predict OPS+ for 2018 and knew that a player was injured in 2017, we could weigh down OPS+ in 2017 to account for that.

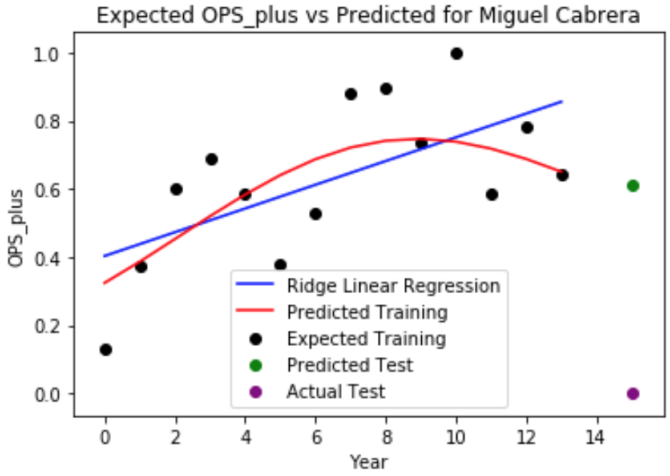

Figure 9: Poor performing SVR prediction on Miguel Cabrera

4 Conclusion

As a first application of RNN with LSTM in the area of baseball player performance prediction, we made significant headway on a challenging problem. To build upon what we accomplished, more diverse and more granular data needs to be incorporated into the feature set. Furthermore, the ability to handle the variety of player career trends will need to be developed–perhaps clustering of career trends with an unsupervised algorithm like k-means can be of use in this domain. This problem interests itself in knowing the future before it happens, which is impossible. However, with further engineering efforts, we believe that a solution exists which can predict future player performance more accurately than existing methods.

References

[1] Jahn K. Hakes and Raymond D. Sauer. An economic evaluation of the moneyball hypothesis. *arxiv.org*, 2006.

[2] Ray C. Fair. Journal of quantitative analysis in sports. *http://fairmodel.econ.yale.edu*, 2008.

[3] Rory P. Bunker and Fadi Fayez. A machine learning framework for sport result prediction, Sep 2017.

[4] Zachary D. Rymer. Chris davis' $161m megadeal has spiraled into mlb's worst contract bust, May 2018.

[5] Alan McCabe and Jarrod Trevathan. Artificial intelligence in sports prediction. *Research Gate*, 2008.

[6] Bjarnadottir Margret V Barnes, Sean. Great expectations: An analysis of major league baseball free agent performance. *Wiley Online Library*, Jun 2016.

[7] Brian Bierig, Jonathan Hollenbeck, and Alexander Stroud. Understanding career progression in baseball through ... *arxiv.org*, Dec 2017.

[8] Mushimie Lona Panda and Cheoloo Park. Penalized regression models for major league baseball metrics. *https://athenaeum.libs.uga.edu*, May 2015.

[9] Steve Slowinski. What is war?

[10] What is a on-base plus slugging plus (ops)?: Glossary.

[11] Kaan Koseler and Matthew Stephan. Machine learning applications in baseball: A systematic ... *Miami University*, Feb 2018.

A Feature Descriptions

- age : Player age
- mlb_ID : Player ID number
- PA : Plate appearances
- Inn : Innings played in field
- runs_bat : Batting runs from modified wRAA
- runs_br : Baserunning runs (includes SB and CS runs)
- runs_dp : Runs from avoiding GIDP's in DP situation GB's
- runs_field : Total fielding runs from TZR and DRS
- runs_infield : Defensive DP-turn runs
- runs_outfield : Outfield arm ratings
- runs_catcher : Catcher defense
- runs_good_plays : Superlative defensive plays
- runs_defense : Overall defensive ability
- runs_position : Runs from the position played (non-pitching)
- runs_position_p : Position runs after zeroing negative contribution of pitcher offense
- runs_replacement : Replacement level adjustment for the league
- runs_above_rep : Sum of the other runs-related statistics (excludes double-counting)
- runs_above_avg : Sum of the other runs-related statistics without replacement level runs
- runs_above_avg_off : Sum of the other runs-related statistics without defense
- runs_above_avg_def : Runs from defense plus position
- WAA : Wins above average
- WAA_off : Offensive wins above average
- WAA_def : Defensive wins above average
- WAR : Wins above replacement
- WAR_def : Defensive wins above replacement
- WAR_off : Offensive wins above replacement
- WAR_rep : WAR attributable to replacement level
- salary : Annual player salary
- Pitcher : Whether the player is a pitcher or position player
- teamRpG : Average team runs scored with this player
- oppRpG : Average team runs allowed with this player
- oppRpPA_rep : Average team runs allowed per PA by a replacement player
- oppRpG_rep : Average team runs allowed per game by a replacement player
- pyth_exponent : Pythagenpat exponent based on a run environment for average teams
- pyth_exponent_rep : Pythagenpat exponent based on a run environment for a replacement player
- waa_win_perc : Win percentage using pythagenpat exponent and team and opp R/G
- waa_win_perc_off : Offensive win percentage using pythagenpat exponent and team and opp R/G
- waa_win_perc_def : Defensive win percentage using pythagenpat exponent and team and opp R/G

- waa_win_perc_rep : How a replacement player would do
- OPS_plus : Adjusted on-base percentage plus slugging percentage
- TOB_lg : Cumulative total number of times a player has reached base
- TB_lg : Total number of bases that a player has gained through hit