Predicting Baseball Postseason Results from Regular Season Data

Category: Athletics & Sensing Devices

Ruiqi Chen[1]
Aeronautics and Astronautics
Stanford University
Stanford, California
rchensix@stanford.edu

Alex Hobbs[2]
Aeronautics and Astronautics
Stanford University
Stanford, California
ashobbs@stanford.edu

Wally Maier[3]
Aeronautics and Astronautics
Stanford University
Stanford, California
wmaier@stanford.edu

Abstract— **Two supervised learning methods, one employing logistic classification and another employing an artificial neural network, are used to predict the outcome of baseball postseason series, given team performance statistics from the regular season. Various refinement methods are investigated to improve model accuracy. The results from the logistic model are promising, with training and test dataset accuracies of 73.6% and 62.6%, respectively, while the results from the artificial neural network are relatively poor and on the same order as random guessing.**

Keywords—Machine learning, supervised learning, classification, logistic model, feature selection, deep learning, artificial neural network

I. INTRODUCTION

Accurately predicting postseason results of sporting events is (sometimes) a billion dollar undertaking [1]. More commonly, it is a multi-million dollar industry, with casino sportsbooks, gamblers, and diehard fans eager for more accurate predictions and probabilities. In this report, we discuss our attempts to predict the outcome of baseball playoff series given regular season statistics.

Baseball was chosen not only because it is America's pastime, but because it is a statistically driven game, with a plethora of data available. This project was conducted in conjunction with CS238 and can roughly be divided into two portions:

A. CS229 Component

We use a variety of supervised learning techniques on historical baseball statistics to develop a model for predicting win-lose probabilities for postseason series given regular season data. Models used include a logistic classifier and a two layer neural network.

B. AA228/CS238 Component

We find the optimal betting policy for baseball postseason series using model-based reinforcement learning techniques. The predicted win-lose probabilities from the CS229 Component are incorporated into our model as the state transition probabilities. We compare our results with a random policy, a conservative highest-seed policy and a greed/risky lowest-seed policy.

This paper will primarily focus on the CS229 Component of our work, but the interested reader is invited to read the AA228/CS238 Component [2].

II. DATASET

Our dataset was taken from Sean Lahman's baseball database [3], which includes complete batting and pitching statistics from 1871 to 2016. The data primarily consists of team data, such as batting, pitching, and fielding statistics. Supplementing this data are statistics on standings, post-season series, and various other tables. We decided to primarily focus this research on features based on team statistics, such as hits, runs, and ERA, with 24 features chosen. The full list of features is given below.

Team: Batter Park Factor (BPF), Double Plays (DP), Errors (E), Fielding Percentage (FP), Pitcher Park Factor (PPF)

Running: Caught Stealing (CS), Runs Scored (R), Stolen Bases (SB)

Batting: At Bats (AB), Doubles (2B), Hits by Batter (H), Triples (3B), Home Runs (HR), Strikeouts (SO), Walks by Batters (BB)

Pitching: Earned Runs Allowed (ER), Earned Run Average (ERA), Hits Allowed (HA), Home Runs Allowed (HRA), Opponents Runs Scored (RA), Saves (SV), Shutouts (SHO), Strikeouts by Pitchers (SOA), Walks Allowed (BBA)

[1] SUID 05726448; enrolled in CS229 and CS238

[2] SUID 05806162; enrolled in CS229 and CS238

[3] SUID 06045130; enrolled in CS238

Each example came from a postseason series between two teams. For example, the 2016 World Series is considered one example. The two teams in each series were designated as “Team 1” or “Team 2” randomly. Two types of input features were used: differential and concatenated. In differential input, Team 2’s features are subtracted from Team 1’s features. This preserves the number of features originally selected. In concatenated input, Team 1 and Team 2’s features were concatenated together, doubling the feature size. The output feature was {-1,1} depending on whether Team 1 won or lost the series.

III. METHODOLOGY

Two models were used for this project: binary classification using a logistic model, and a two-layer artificial neural network.

A. Logistic Classification

We used four variations on logistic classification methods:

- Head-to-head statistic differentials
    - Differential input features
    - 1975-2016 seasons
    - 80/20 training/test split
- Head-to-head statistic concatenation
    - Concatenated input features
    - 1975-2016 seasons
    - 80/20 training/test split
- Additional seasons
    - Both input features
    - 1930-2016 seasons
    - 80/20 training/test split
- Higher order features
    - Concatenated input features
    - 1974-2016 seasons
    - 60/20/20 training/dev/test split

The first three methods involved running linear classification via logistic regression with Newton’s Method on a training set, then checking the results on a test set. The training/test set split was generally kept to a roughly 80/20 split. Initial experiments used data from 1975-2016 seasons of baseball. This was done because in 1973, the American League introduced the designated hitter [4], which we believe is a significant rule change. For additional seasons, this lower limit was dropped to 1930 to see how including older data affected prediction results.

For the higher order features method, we took concatenated features and squared certain features that we believed would play a more significant role, such as earned runs allowed (ER), earned run average (ERA), opponent’s runs allowed (RA), hits (H), and hits allowed (HA). To prevent overfitting, we utilized training/dev/test datasets for this analysis using a 60/20/20 split. This had the effect of significantly reducing our training dataset size.

After these methods were carried out with logistic regression, each feature in the featureset was analyzed by being systematically removed from the logistic regression algorithm. After removal, the training set error and test set error were both calculated. The features that increased the error the most were the most important features in the feature set, as the algorithm became incredibly inaccurate without them. The features that were the least important resulted in either no change or a decrease in training or test set error upon removal.

B. Artificial Neural Network

A two-layer artificial neural network (ANN) was developed using sigmoid activation functions. The number of neurons in the 1st layer (hidden layer) was variable, while a single neuron served as the output layer. The loss function used was the standard negative log likelihood function. A schematic of the ANN is shown in Figure 1.

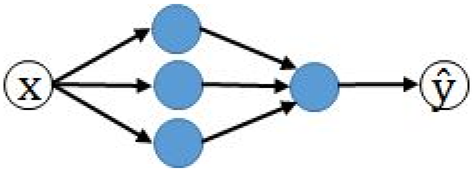

Fig. 1. Two-layer ANN with three hidden layer neurons. The actual number of neurons in the hidden layer was left as an experimental parameter.

Various experimental parameters were adjusted in an attempt to improve the ANN performance:

- Number of hidden layer neurons *n*
- Learning rate α
- Regularization λ

The input data for the ANN was the 80/20 training/test split using concatenated features from 1975-2016 seasons.

All data processing and machine learning algorithms were written in MATLAB.

IV. RESULTS

A. *Logistic Classification*

The results from this set of work are very interesting. The head to head statistic differentials for a training and test set from 1975 to 2016 were found to have a training set error of .3371 and a test set error of .5250. These results are not preferable to a coin flip, unfortunately. In an attempt to fix this data, more years were added onto the training set and test set. In this set of trials, the training and test sets were composed of years from 1930 to 2016. The training set error increased to .3600 and the test set error increased to .6600. Obviously these results were not an improvement, seeming to indicate that the statistics surrounding baseball have changed significantly since the 1930s. It is worth noting that the stat related to a player being caught while stealing a base was excluded from the 1930 to 2016 group, due to a lack of data in some years. The authors do not believe this to have caused a significant change.

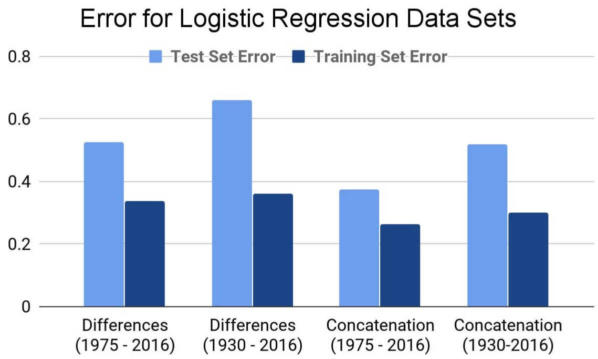

Fig. 2. A comparison of error between concatenation and statistical differentials for different decades.

The second set of results comes from the concatenation of statistics from 1975 to 2016. The results of these tests were notably improved compared to results from the head to head differential statistics. The training set error was only .2640 and the test set error was .3750. Not only is this a notable improvement when compared to previous results, but it is a significant improvement when compared to the results of previous sports groups [5][6]. In an attempt to improve results and see if including more years in the past would help with predictions, another trial was carried out for years from 1930 to 2016. The training set error rose to .3022 and the test set error rose to .5200, further reinforcing the idea that years far into the past do not accurately represent the game of baseball today. All of the results discussed to this point can be seen compared in Figure 2.

Next, to see which parameters fit the data better, we selectively added higher order features. The data was split into a 60/20/20 training/development/test set. Each experiment tried squaring a different group of stats. For example, one group was related to runs for and against the team throughout the regular season, another group was composed of hits for and against a team, and another group was composed of strikeouts for and against a team. These stats were compared to the baseline stat combination, which was also split in a 60/20/20 manner. The results can be seen clearly in Figure 3.

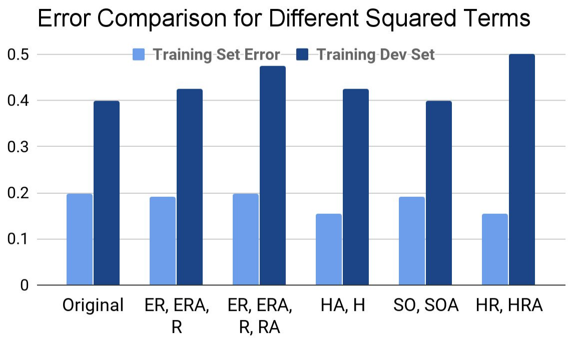

Fig. 3. Above, the error when different stats are squared and added as features is compared.

Figure 3 shows that the potential "best" groups were the baseline and the stat group that squared SO and SOA (strikeouts by batters and strikeouts by pitchers). Due to the increase in error from the training set to the development set in the experiment that included SO and SOA, we concluded that the baseline was preferable in this case. The error in the test case for the base case in this calculation is .4889. Although this is significantly higher than in the results with an 80/20 split, it is still preferable to random guessing, and almost certainly had increased error due to a significant drop in data available. The actual algorithm to be used would still be the original 80/20 split; this set of work simply confirmed that it was the most favorable outcome. For future predictions, that last 20% could theoretically even become part of the training set, since the data has already been confirmed to work.

For the final portion of the logistic regression analysis, each feature was removed from consideration in the algorithm and the training set error and test set error were compared in order to see which statistics were the most influential in predicting the winner of a postseason series. Results can be seen in Figure 4. It is interesting to note that removing the number of times each team has been at bat (AB) actually reduced the error. This makes sense, as the number of times a team has batted should not influence the outcome of a postseason series. Furthermore, two of the features that increased the error the most upon removal were saves (SV) and hits against (HA). This indicates that pitching is incredible important when trying to predict who will be the victor in a postseason series.

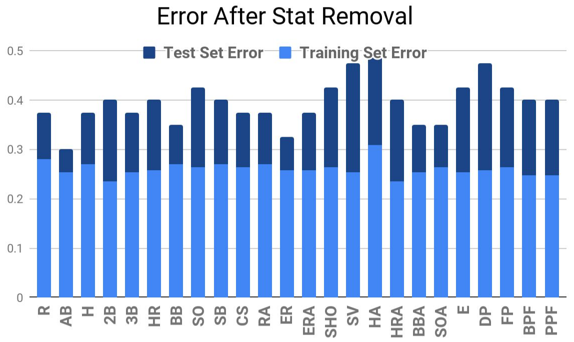

Fig. 4. The resulting training set and test set error upon removing each feature can be seen above. It should be noted that the test set error bars are behind the training set errors. For example, the first set of results, for runs (R) , reads as having a training set error around .28 and a test set error around .37.

B. Artificial Neural Network

The following algorithm was used to fine tune the parameters of the ANN.

```
Initialize two of the following parameters:
        {n, α, λ}
Repeat through range of parameter
        Initialize weights W and B randomly
        Vary parameter of interest
        Perform stochastic gradient descent
        Calculate accuracy and loss
```

The results of parameter tuning are shown in Figures 5, 6, and 7.

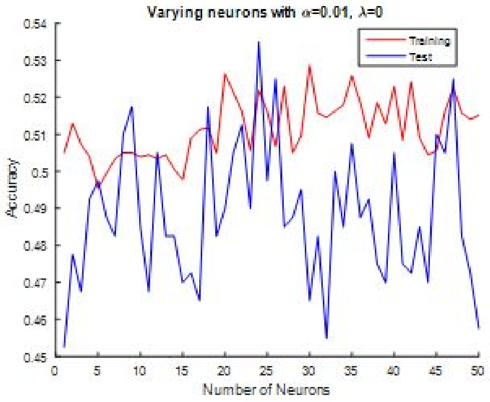

Fig. 5. Variable number of neurons n with learning rate held constant and no regularization.

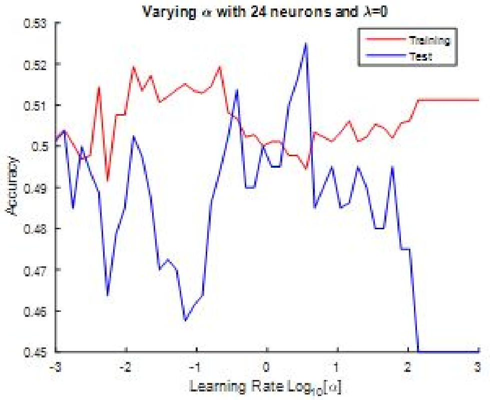

Fig. 6. Variable learning rate α with constant 24 hidden layer neurons and no regularization.

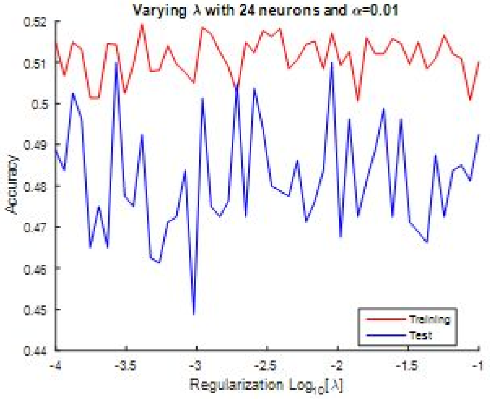

Fig. 7. Variable amount of regularization with constant 24 hidden layer neurons and fixed learning rate.

As shown in Figure 5, the accuracy of the ANN peaked at $n = 24$ and $n = 48$, which correspond to exactly half and exactly the number of features. Varying the learning rate yielded interesting results, as the training and test accuracies tended to stray away from each other, i.e. as one increased, the other decreased, as shown in Figure 6. However, they both peak at a value of 0.01. Finally, adding regularization did not seem to improve the accuracy at all, as seen in Figure 7.

The following set of parameters were considered the best performing:

$$n = 24$$
$$\alpha = 0.01$$
$$\lambda = 0$$

Due to random initializations of the weights, multiple trials using these parameters were run and the results were averaged. This is shown in Figure 8.

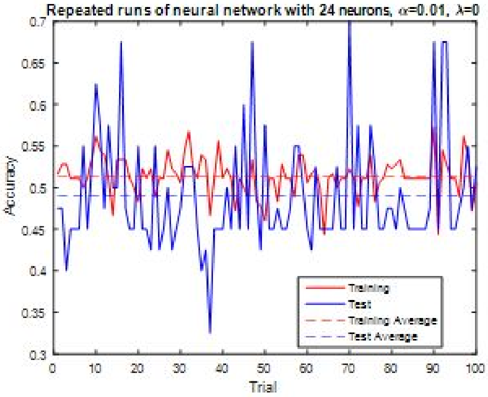

Fig. 8. Running the neural network with random weight initializations.

As shown in Figure 8, the average training and test accuracies are 0.52 and 0.49 respectively. Unfortunately, this is no better than a coin flip.

Additional experiments were conducted on known separable data (like the ones in the homework) in an attempt to root out the problem. The ANN could successfully classify those datasets, leading us to believe the issue was not related to ANN implementation.

We believe the biggest issue with the ANN is the lack of data available, as only seasons 1975-2016 were relevant. Without enough data, the algorithm could not find any meaningful relationships between the features.

V. CONCLUSIONS

Overall, this project was very successful in achieving its goals. The accuracy of the logistic regression algorithm exceeded algorithms from previous years of CS 229 in other sports. It was discovered that data from more recent decades is more relevant than data from far into the past. Additionally, it was discovered that squaring some features tends to result in overfitting. Furthermore, it was discovered that pitching is one of the most useful metrics when trying to determine who will win a postseason baseball series. The artificial neural network yielded less than helpful results, showing that it is difficult to find complex relations between features with such little data available.

VI. FUTURE WORK

Currently, the results of each playoff series is assumed to be independent. This is not the case in real life, as the winners of the League Division Series matchups move on to play each other in the League Championship Series, and finally those winners play each other in the World Series. Also, the outcome if each individual game may have an effect on future games within a series. For example, a pitcher who receives an injury in Game 1 may not be able to play again in Game 5. Future models could look at modeling the baseball playoff series as a Bayesian network to capture these conditional events.

Future models may also incorporate additional statistics, especially ones related to the offseason (drafts, trades, player salaries, etc.). In addition, the effect of home field advantage, which is not only influenced by regular season performance, but also performance within the playoffs themselves, may have a huge impact on a team's performance.

To predict the winner of the World Series before any playoff games are played, we must define belief states on which teams are believed most likely to advance to the World Series. This can be modeled using a partially observable Markov decision process (POMDP) and be extended to a betting decision problem.

Finally, due to the general lack of data (due to years before 1975 tending to not be representative of modern day baseball), cross-validation may be a worthwhile endeavor.

ACKNOWLEDGMENTS

The authors would like to thank Professor Andrew Ng, Professor Dan Boneh, and the class TAs for their valuable insight on this project.

CONTRIBUTIONS

Both members of the team in the CS 229 group contributed equally. Alex Hobbs worked more on the data organization and applying the logistic regression algorithm, while Ruiqi Chen created the algorithms and error test functions required for this work.

REFERENCES

[1] N. Rayman, “No One Won Warren Buffett's $1 Billion Bracket Challenge,” Time Magazine, accessed online Oct. 19, 2017

[2] R. Chen, A. Hobbs, and W. Maier, “Determining the Optimal Betting Policy,” *AA228 2017 Projects*

[3] S. Lahman, “Lahman Baseball Database,” SeanLahman.com, accessed online Nov. 20, 2017, http://www.seanlahman.com/baseball-archive/

[4] “Baseball Rule Change Timeline,” Baseball Almanac, accessed online Nov. 20, 2017, http://www.baseball-almanac.com/rulechng.shtml

[5] M. Painter, S. Hemmati, and B. Beigi, “Beating the Odds: Learning to Bet on Soccer Matches using Historical Data,” CS229 2016 Projects, accessed online Nov. 20, 2017, http://cs229.stanford.edu/proj2016/report/PainterHemmatiBeigi-BeatingTheOddsLearningToBetOnSoccerMatchesUsingHistoricalData-report.pdf

[6] K. Bishop, “Data-Driven Insights into Football Match Results,” CS229 2016 Projects, accessed online Nov. 20, 2017, http://cs229.stanford.edu/proj2016/report/Bishop-DataDrivenInsightsIntoFootballMatchResults-report.pdfieeexplore.ieee.org/xpl/bkabstractplus.jsp?bkn=7288640.