Predicting the Outcomes of Baseball Games

Richard Harris, Allan Joshua, Justin Sirignano

Stanford University

Abstract: In this paper we investigate using machine learning algorithms to predict the outcomes of baseball games. Baseball has a large amount of raw data, including pitching, batting, and defensive statistics for each game. In addition, baseball has the largest total number of games per season of any sport. This combination of readily available data and the large number of games make baseball a great prospect for machine learning. We use past data to predict the outcomes of baseball games with the goal of discerning any patterns or shedding light on what characteristics produce a winning baseball team. We use logistic boost and SVMS.

0.1 Introduction

Inherently, certain statistics a baseball team produces represent its capability to win. For example, a team's record gives good indication of how well a team will do in a game. More times than not, the team with the better record wins. Another example is head to head record. Usually, the team that has won most of the previous matchups wins, either because the team is better as its record might indicate, or because the team matches up well against the opposing team.

In addition to including features that account for the entire team's performance, it is also plausible that certain players account for a large amount of the team's success, whether it be power hitters on offense, or pitchers on defense. To represent this, we included features that reflect how good home team hitters are vs. away team hitters, and how well hitters on each team have done against their opponent's starting pitcher. We also created similar features for pitchers.

It is also common knowledge that both teams and players go on streaks and slumps. To provide an accurate account of existing knowledge of these streaks and slumps to the learners we used, data reflecting past performances in recent games is also included in the feature set. Given features summarizing the opposing teams' past and recent performances, opposing hitters' past and recent performances, and opposing pitchers' past and recent performances, we felt our learners were ready to give us killer results!

0.2 Data and Choice of Features

We created a large database detailing pitch-by-pitch events in each game for the last thirty years. With this raw data, we wrote a series of queries to calculate the features we were interested in. The features we looked at are:

- Team statistics Head-to-head win differential
- Win count differential in last *x* amount of games
- Error count differential
- Win count differential of home team wins at home in last *x* games vs. away team wins away in last *x* games
- Comparatively how well each team does against a starting pitcher of a certain hand (left vs. right)

- Offensive statistics (recent performances[1], performance by year to date[2], and career performance against opposing starting pitcher[3])
  - Player-wise on base percentage
  - Player-wise slugging percentage
  - Total number of bases

- Pitching statistics (recent performances and performance by year to date)
  - At bats
  - Strike Outs
  - Walks allowed
  - Singles allowed
  - Doubles allowed
  - Triples allowed
  - Homeruns allowed

We believe that recent performances of players are essential because they reveal important information about the "momentum" of those players entering the game. All players go through streaks and slumps. Therefore, just using statistics for their careers would be wildly inaccurate. It is better to take into account recent performance levels.

0.3 Methodology

0.3.1 Features

We are dealing with an inherently noisy data set (i.e., the best team may not win all the time). Therefore, we desired an algorithm that is robust to noise. That is, it is less likely to overfit noise. We choose to use logistic Boost for this reason. The weak classifiers are trees. We also use SVM and compare the results.

---

[1] Recent performances refer to player performance in the past 5, 10, 15, 20, and 25 games

[2] Performance by year to date refers to player performance in the past 1, 2, 3, 4, and 5 years

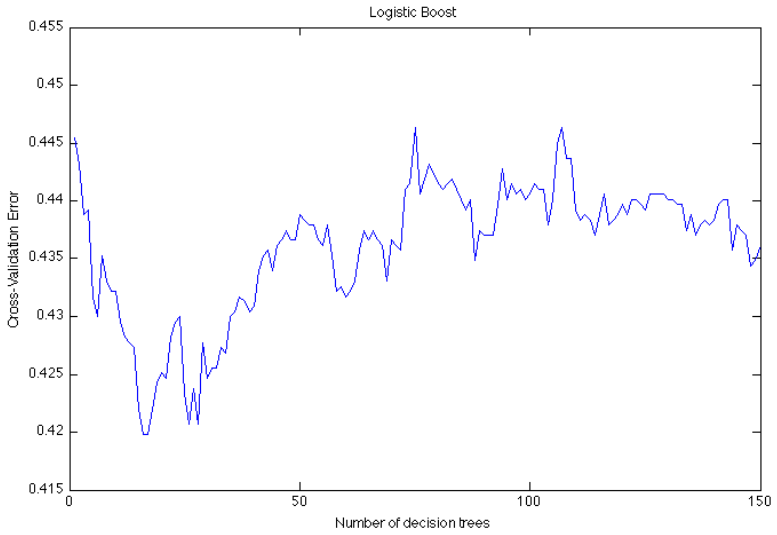
[3] Career performance against the opposing starting pitcher refers to a hitter's past performance against the opposing starting pitcher in the past 1, 2, 3, 4, and 5 years.

We investigate several ways in which to improve our feature set. First, we tried normalizing the features by their mean and variance. In addition, we attempted to use polynomial features with a hope that the SVM would provide a superior fit given the nonlinear data. Then, we tried to reduce the feature set by taking the features most correlated with the outcome of the game. Finally, we also did a principal component analysis to reduce the feature set into its most important principal components. We found that these methods helped marginally, if at all.

To determine the optimal number of iterations for the Logistic Boost, we did a k-fold cross validation with k = 10. To gauge the out-of-sample accuracy of the SVM and Logistic Boost algorithms, we tested their fitted models on a separate validation set.

0.3.2 Choosing the Optimal Number of Weak Classifiers for Boost

Boost uses a number of weak classifiers to make predictions. The optimal number of weak classifiers (i.e., the number of iterations Boost takes) must be determined. Too many iterations is prone to overfit the data while too few may underfit the data. We find the optimal number of iterations by dividing the training set into a smaller training set and a cross-validation set. Below we show the cross-validated accuracy for Logistic Boost for different numbers of iterations. We use these results to pick the number of iterations for each Boost algorithm which gives the highest cross-validated accuracy as the "optimal" number of iterations. Given the optimal number of iterations, we retrain the Boost algorithm on the entire training set and then test it on a separate test set.



0.3.2 Choosing the Optimal C for SVM

We fully expect our data from baseball games to not be linearly separable. To compensate for this, we used included a "C" parameter for the SVM. The final value of

"C" was chosen by empirically running through values of "C" from 2^{-5} through 2^{5}. Ultimately, we chose the value that produced the best results from these empirical tests.

0.4 Results and Discussion

First, we will provide a brief summary of our best results. In the paragraphs following the summary, we outline some steps we took to attempt to improve our learner's performance. (BEST RESULTS HERE)

Our immediate thoughts were to try three learners: Logistic Boost, AdaBoost, and an SVM. The results for each of these three learners is given below:

Learner	Description	Accuracy	True Pos	False Pos	False Neg	True Neg
Logit Boost	25 iterations	57.60%	53.60%	46.40%	40.28%	59.72%
AdaBoost	50 iterations	58.60%	54.81%	45.19%	39.25%	60.75%
SVM	C=10	58.4836%	53.85%	46.15%	37.86%	62.14%

Both of the implementations of the learners we used provide a probability associated with each prediction. In thresholding the probability of the result, we can filter out games that are difficult for the learner to decide. This trick usually provides about 5 to 10% better results, but at the cost of making three quarters of the data unusable. Of course, this translates to not being able to bet on a quarter of all games.

Learner	Accuracy	% examples used
Logit Boost	65.52%	26.30%
SVM, C=10	67.41%	23.75%

We next tried to reduce the dimensionality of our feature vector by using the features most correlated with the result. This produced mixed results, but overall, the performance of the learner did not improve. We also tried using principal components analysis where we took the first 75 principal components. This also yielded mixed results.

Learner	# Princ Comp	Accuracy	True Pos	False Pos	False Neg	True Neg
Logit Boost	75	55.94%	51.35%	48.65%	41.96%	58.04%
SVM	75	54.83%	49.72%	50.28%	40.68%	62.82%

While we understand sports are not deterministic, we were hoping to obtain better results. To try to improve our results, we tried higher order polynomial features. This did not help, however.

On analyzing the learning curve and plotting Jtrain and Jcv,as part of the model selection process, we realized that we were under fitting the data and the training error was high as well which indicated a high bias problem. We then decided to use a neural network and learn the parameters using back propagation. We used one hidden layer with 250 logistic units on it. Using an advanced optimization algorithm (fmincg) we were able to train the network within a reasonable timeframe. On increasing the number of iterations close to 500, the network started over-fitting the data and our training accuracy went up to 100 % on the training set. The network however, did poorly on the test set and the accuracy was only 53.124%

We then regularized the cost function and minimized that and by trying various values of lambda, the network started generalizing well and gave good results on the test set. With 50 iterations we were able to get to an accuracy of 64%

We also tried SVM with an RBF kernel and tried to pick gamma and c. We did not have time to implement this but we are planning to continue working on it. Our feature set size and the number of training data points make an RBF kernel a prime candidate to pursue. The cost function of the neural network was still decreasing on every iteration which indicated that we would still be able to get better results. This is an avenue which we plan to pursue as well.

0.5 Conclusion and Future Work

In conclusion, we tried several machine learning algorithms to predict the outcomes of baseball games. The results were not spectacular, although we must recognize that the outcomes of baseball games are very noisy and therefore difficult to predict. Predictions made when the algorithm is confident (i.e., thresholding) are much more accurate. Future work might attempt to develop a better feature set or a better algorithm. Something that might be interesting to try would be Logistic Boost with logistic regression or SVMs as weak classifiers.