

Applying Machine Learning to MLB Prediction & Analysis

Gregory Donaker

gdonaker@cs.stanford.edu

December 16, 2005

CS229 – Stanford University

Introduction

Major League Baseball (MLB) is a multi-billion dollar statistics filled industry. Individual players are chosen based on their raw statistics such as batting average, on-base percentage, or slugging percentage. Furthermore, there are communities of fans who closely follow the statistics of the game to analyze the contributions of individual players and including them in fantasy leagues. There is no definitive formula for what factors will lead a team to victory, yet by analyzing many years of historical records many trends may emerge. I am applying machine learning techniques to predict the outcomes of individual Major League Baseball games and demonstrate the relative importance of different elements of a team. I believe that applying these techniques to a broad set of features will provide not only a basis for prediction but also reveal potential strengths and weaknesses of individual teams by quantifying the significance of a specific asset and its relevance to victory. Such a tool could prove invaluable to MLB managers, sports broadcasters and fantasy sports enthusiasts.

Data Sources

I have harvested my data from two sources for the two levels of granularity they offer. “The Baseball Archive” (<http://www.baseball1.com>) offers downloadable historical statistics for teams, batting and pitching on an annual basis. While this data is very useful for incorporating in broad statistics, it does not provide the game by game granularity that my research needs. Unfortunately, I could not find a source of ready to process boxscores so I have had to compile them via web-scraping. “Baseball Boxscores” (<http://www.baseball-boxscores.com/>)¹ has thousands of complete boxscores for a subset of MLB teams, including about two hundred complete team-seasons. I used Perl scripts to harvest a total of 120 complete team-seasons (18,717 individual games). I selected the complete seasons for the Boston Red Sox, Baltimore Orioles, Chicago Cubs and Atlanta Braves and for all years between 1971 and 2000. My team-selection was based on the limited number of box-scores available and the desire to have two teams from each the National (Atlanta and Chicago) and American (Boston and Baltimore) Leagues. I compiled the individual game data in a form where it can be relatively easily cross-referenced to the broader dataset for feature extraction.

Features

I have chosen a set of twenty-five features which form a 25-dimensional vector representing a single game. Each feature must be extracted/computed from the different datasets and includes only information available before the game officially started. Most of my selected features are not readily available but must be computed based on the boxscores of the game and the proceeding games. My features include:

¹ My apologies to baseball-boxscores.com for putting an abnormally large load on their servers while harvesting the data.

- Opposing team record of previous season {0.000 – 1.000}
 - Win in previous game {0, 1} (0.5 for first game of season)
 - Wining percentage over the previous 10 games {0.000 – 1.000}
 - Wining percentage over the previous 20 games {0.000 – 1.000}
 - Starting pitcher ERA² in previous game {0.0+} (divided by 4)
 - Starting pitcher ERA in previous season {0.0+} (divided by 4)
 - Opposing starting pitcher ERA in previous season {0.0+} (divided by 4)
 - Head to Head matchup between the two teams {0.000 – 1.000}
 - Home game {0, 1} (1 for a home game)
 - Hits by starters in previous game {0.0+} (divided by 14)
 - Hits by starters in previous 10 games {0.0+} (divided by 140)
 - Left-handed starting pitcher {0, 1}
 - Left-handed opponnet starting pitcher {0, 1}
 - Percentage of batters opposite handed from opponnet starting pitcher {0.0 – 1.0}
 - Percentage of opponnet batters opposite handed from starting pitcher {0.0 – 1.0}
 - Average age of starters {0.0+} (divided by 30)
 - Average age of opponnet starters {0.0+} (divided by 30)
 - Average height (inches) of starters {0.0+} (divided by 72)
 - Average height (inches) of opponnet starters {0.0+} (divided by 72)
 - Average years since starters' MLB debut {0.0+} (divided by 10)
 - Average years since opponnet starters' MLB debut {0.0+} (divided by 10)
 - Average batting average of starters in previous season {0.000 – 1.000}
 - Average batting average of opponnet starters in previous season {0.000 – 1.000}
 - Avg num of at-bats by starters in prev season {0.0+} (divided by 250)
 - Avg num of at-bats by opponnet starters in prev season {0.0+} (divided by 250)
- and
- Win {0, 1} (for supervised learning, testing)

In the case that no previous ERA for a pitcher can be found, I am assigning a value of 4.0 (a relatively average ERA). Winning percentages for the first games of the season do not reference the previous season, rather I calculate them as if the team was 20-20 (.500) before the start of the season. Head to head matchups are smoothed by initializing with one win and one loss. I acknowledge that some of these assumptions/smoothings may not be ideal, yet throwing out all games with an imprecise feature set would produce a model that would not apply to roughly 20% of games played.

While the divisors on some of these statistics may seem arbitrary, they have been chosen to scale all features to a range between roughly zero and one. As defined, no features can be negative, yet many features have the ability to exceed one. This scaling was done so upon examining the trained weights, one can get a good sense of the relative importance of different features. Many more features would be available if performing real-time gathering, yet many desirable statistics are extremely difficult to compute based on the difficulty of obtaining complete historical data.

Training and Test Sets

I designed my training and test sets to be as close a representation as possible to applying a classifier on a future season. I built both generic and team-specific classifiers. From here on, I will define a 'horizontal' dataset as one which includes data for many teams for the same time period and a 'vertical' dataset as one composed entirely of a single team's games over a longer time period.

² Earned Run Average = number of earned runs allowed in 9 innings.

My primary ‘horizontal’ dataset consists of the 1991-1999 seasons for the Red Sox, Braves, Orioles and Cubs (36 team-seasons, 5,556 games). The corresponding test set contains the 2000 season for the same four teams (4 team-seasons, 646 games). I initially implemented and tuned my classifiers on Red Sox examples, so I chose to evaluate my ‘vertical’ classifiers on an arbitrary different team to avoid bias. My primary ‘vertical’ training set is composed of all Atlanta Braves games for the 1971-1979, 1981-1989, and 1991-1999 seasons (27 seasons, 4,106 games). The corresponding test contains all Braves for the 1980, 1990 and 2000 seasons (3 seasons, 485 games). While some classifiers performed notably better on different datasets, I originally assigned these for my training and test sets and will report the errors accordingly.

Applied Algorithms

I implemented a range of machine learning models for the two datasets defined above including Logistic Regression, Naïve Bayes, Support Vector Machine (SVM) and an ensemble classifier over the other three. I initially implemented the Perceptron algorithm but decided to limit my focus to the others in order to maintain an odd number of models for the ensemble classifiers. As I had been gathering and preprocessing my features in Perl, I chose to use Perl for my Logistic Regression implementation.

I used multinomial Naïve Bayes both with and without including the priors of the training sets. While my datasets are close to 50% wins, I wanted to see the effect of any imbalance on the resulting classifiers. Both these implementations were performed in Matlab.

I attempted to train my SVMs using my Matlab SMO implementation, yet my running time was excessively long so I switched to an optimized off-the-shelf product for these models. SVM-light³ provided an efficient SVM implementation allowing me to easily train multiple models without further trimming my datasets. I built models with both linear and polynomial kernels, trying numerous parameterizations of the polynomial kernel.

Results and Discussion

My result accuracies are summarized in the following two tables.

Classifier (‘Vertical’)	Accuracy
Logistic Regression	56.3 %
Naïve Bayes	53.4 %
Naïve Bayes (no prior)	53.4 %
SVM (linear kernel)	53.4 %
SVM (polynomial kernel)	(all win)
Ensemble – majority	54.8 %
Ensemble – unanimous	57.9 %

Classifier (‘Horizontal’)	Accuracy
Logistic Regression	52.1 %
Naïve Bayes	50.5 %
Naïve Bayes (no prior)	50.5 %
SVM (linear kernel)	51.9 %
SVM (polynomial kernel)	(all win)
Ensemble - majority	52.3 %
Ensemble - unanimous	53.7 %

³ T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods – Support Vector Learning, B. Schoelkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

The confusion matrices corresponding to the above results demonstrate that the performance of these models is better than just selecting the outcome with the higher prior. For all models, the number of true positives exceeded the number of false positives and similarly the number of true negatives exceeded false negatives. This trend can be easily observed in the Logistic Regression confusion matrix for the ‘Vertical’ test set:

	Actual win	Actual loss
Predicted win	124	95
Predicted loss	117	149

Logistic regression outperformed all non-ensemble classifiers. One of the most useful attributes of this model is that the converged weights indicate a relative importance of different features. As expected, the weights for the vertical and horizontal datasets differ in their ordering and in some cases even their sign. The weights on the horizontal dataset could be used when assembling a baseball team to help choose one player over another. The two most significant weights were the sum of starter at-bats in the previous season and the starting pitcher’s ERA in the previous season. Both of these match common intuitions of what makes a successful team. I am not implying that these features cause the win, but rather noticing a strong correlation. The least significant weight, on the other hand, was the number of hits by starters in the previous 10 games. The value of this feature was likely subsumed by the team’s record over the same period. The interplay between these weights would be useful for a baseball owner, sports gambler or fantasy sports enthusiast. It is worth noting that my accuracies of Logistic Regression over the four possible ‘vertical’ datasets ranged significantly – with accuracy over 60% for the Chicago Cubs vertical.

While the accuracies for the different Naïve Bayes models (with and without priors) are identical, the models produce different confusion matrices. The results had the same margins, but were shifted ‘up’ or ‘down’ based on the prior. As I am interested in the most general and effective models possible, I used the model without the prior probability for my ensemble classifier.

Even though the accuracies were not as high as Logistic Regression, SVMs with a linear kernel provided tangible results. For both datasets, SVMs with polynomial kernels produced models that predicted that all elements of the test sets were of the positive class (wins). I trained multiple models – varying parameterizations of the kernels – all giving the same result. While I was unsuccessful in finding a useful parameterization, this does not mean that SVMs with polynomial kernels cannot be used for baseball classification.

The ensemble classifiers improved accuracies over the datasets. The standard classifier made a prediction based on the majority vote of the three sub-classifiers, while the unanimous classifier returned ‘unknown’ for any game where the three classifiers did not completely agree. The higher accuracy of the unanimous ensemble classifier can help assign an effective confidence value for a given prediction.

When looking at the above results, it is important to note that the records of baseball teams do not exhibit uniform distributions. In the 2005 season, for example, the team with the best record won 61.7% of its games while the worst won 34.6%.⁴ Even the

⁴ Statistics as reported on MLB.com

best teams lose with a certain amount of underlying randomness; hence one cannot expect a baseball classifier to ever have near perfect accuracy. While my classifier accuracies do not reach my initial hopes, they do demonstrate a practical application of machine learning techniques and offer insight into the dynamics of the game.

Conclusion

Baseball prediction is a challenging and uncertain endeavor. While my classifiers were not able to achieve the accuracies I had hoped for, they did reinforce that there is a strong correlation between these quantifiable features and victory. With the underlying randomness of baseball, I imagine that any classifier – human or machine – would be unable to demonstrate the high accuracies achieved by some common classification applications. Regardless, my experiments demonstrate the feasibility of applying machine learning to baseball and suggest potential improvement with a more extensive feature set.