Beating the MLB Moneyline

Leland Chen llxchen@stanford.edu
Andrew He andu@stanford.edu

1 Abstract

Sports forecasting is a challenging task that has similarities to stock market prediction, requiring time-series prediction using large datasets that are prone to variations and noise. Despite an abundance of studies performed in the realm of financial modeling, sports prediction receives relatively little attention in the machine learning community. We first describe a methodology for predicting the outcome of baseball games using machine learning algorithms, and discuss our attempts to generate returns using only this knowledge. We then explore a promising strategy in which historical betting odds are used to estimate team winrates, and go on to demonstrate that it can be profitable.

2 Background

The baseball betting market offers a wide variety of betting propositions, but the most typical bet involves fixed odds, commonly referred to as the *moneyline* bet. The moneyline consists of a pair of odds: one for the home team and one for the visiting team, expressed in either positive or negative figures. A positive figure represents how much money will be won on a $100 wager, and a negative figure represents how much money must be wagered to win $100. For example, if the home team has moneyline odds of -150, the bettor must wager $150 in order to win $100.

It is possible to profit in these scenarios by consistently choosing winning teams, though such a strategy is intuitively non-ideal. Nonetheless, we can use the problem of picking overall winners as a basis for applying machine learning, and use any initial findings to develop better strategies. For example, any algorithm will require an effective, relevant feature set in order to characterize games and matchups; training a binary classifier to identify winners is a natural approach towards developing this feature set. Therefore, our initial work will focus on a classification problem where the output represents whether a team should win or lose. We choose to classify home team wins as positive training examples, and visiting team wins as negative training examples.

3 Data

3.1 Game Data

We obtained detailed MLB game logs from Retrosheet, an online encyclopedia for baseball statistics. Data was retrieved for the most recent decade available (years 2000 through 2009), totaling 24,301 individual game records. Moneyline odds for the corresponding games were acquired from various odds-publishers, including Covers and DonBest.

We use a home-grown Ruby framework in order to parse and pre-process the data for use in our algorithms. Although each individual Retrosheet game log contains 161 data fields, our parser removes identifying information for individual batters, coaches, and umpires, leaving just 56 fields (partial listing shown in Figure 1). In doing this, we distill each game down to a record of the starting pitchers, accompanied by an array of team offensive, defensive, and pitching statistics in each game. These statistics are then aggregated for each season to produce

day-by-day sets of cumulative statistics for each team. Finally, our pre-processor reformulates the game records such that the data set (training example) for each game represents a history of statistics accumulated *prior* to the game, rather than the statistics from the game itself.

At-bats	Home runs	Strikeouts	Assists
Hits	Walks	Earned runs	Hit-by-pitch
Runs	SBs	Errors	Double plays
Doubles	Left on base	Pitchers used	Balks
Triples	Putouts	Wild pitches	Passed balls

Figure 1. Partial set of data included as part of one game log.

3.2 Features

Intuitively, any estimates of future performance for a baseball team should be predicated on the team's recent performance. So, we include in each training example a set of averages representing performance across variable-length sequences of games (e.g. statistics for last 5 games, last 10 games, etc). Consequently, we drop from our data set the first 20 games of each season for all teams, to ensure that each example includes these "recent past" features.

Our feature set also emphasizes matchups between starting pitchers, widely acknowledged to be the strongest influence on bookmaker odds. The feature set for a game thus includes averages of team performance for the set of games started by a certain pitcher. We can generate these statistics easily, since we map each game to the starting pitchers during pre-processing. Our final feature set consists of 344 fields.

For our initial binary classification problem, our target variable is {0 = home team loses, 1 = home team wins}. Later, we use an alternate problem formulation where the target variable is continuous between 0.0 and 1.0 (representing the chance that the home team will win).

4 Binary Classification

4.1 Feature Selection

Our generated feature set is fairly large and primarily based on human intuition, and it is not apparent which features are the most relevant for our classification task. In order to find a smaller, optimal feature subset, we first implement a filter selection algorithm, and select the most informative features by using the correlation coefficient as the scoring metric (Figure 2). Unfortunately, this metric reveals only minute separation of the features, and cross-validation using feature subsets from the top of this list show no increase in accuracy.

Feature	r^2
home pitcher's strikeouts/game	0.0054
opponent's assists/game against home pitcher	0.0049
away team's double plays/game	0.0046
home team's cumulative home runs/game	0.0044
home team's home runs/game in home stadium	0.0043
home team's caught stealing/game in home…	0.0042
away team's cumulative intent. walks/game	0.0041
home team's cumulative assists/game	0.0040
away team's assists/game in road stadiums	0.0038

Figure 2. Partial listing of features selected using filter selection with score $S(i) = r^2$.

4.2 Algorithms

For the classification problem of picking overall winners, we evaluate using the logistic regression classifier and SVM classifier (using the e1071-package for R). For the SVM, we use both linear and polynomial kernels (C = 1, d = 2). Testing results using 10-fold cross-validation are shown in Figure 3.

	2006	2007	2008	2009
Logistic regression	0.4055	0.4274	0.3974	0.3996
Linear SVM error	0.4276	0.4401	0.4297	0.4352
Polynomial SVM error	0.3935	0.3868	0.4020	0.4156

Figure 3. Error rates using 10-fold cross-validation.

The best predictor of winners is the SVM with polynomial kernel, with its accuracy rate hovering around 60%. Despite this, when evaluating against the betting market, this strategy produces large negative returns. This may be explained by comparing with the strategy of always picking favorites -- even if we select winners with relatively high accuracy, as such a strategy would do, it is unprofitable because it ignores the moneyline odds. Intuitively, we should prefer to bet only on games where we predict that a team is more likely to win (or lose) than indicated by the fixed odds.

5 Estimating Lines

As an alternative method, we propose using the moneyline as the target variable, rather than the binary value of whether a team wins/loses. We accomplish this by converting the home team odds to a fractional value, thus turning the target variable into a continuous one. As an example: if the home team has odds of -150, they can be considered a 60% favorite according to the line. In other words, our new approach is to approximate a team's chances of winning a certain game, rather than predicting whether they will win. The strategy naturally resulting from this formulation will be to place bets whenever we estimate a team's win percentage to be higher than that quoted by the moneyline. We can further filter bets based on "confidence", by only placing them if our estimate is beyond some threshold away from the moneyline value.

We evaluate two algorithms for this problem: regression trees and random forest.

5.1 Regression Trees

Based on the problem description (approximating a continuous output variable using continuous inputs), we can opt to use a regression tree. We use the rpart package in R to generate a regression tree for our data. One advantage of regression trees is that they are effective visualizations: from each node, we can see the input variables, and how different values force the tree down various paths. We used the default complexity parameter in rpart to determine when to prune the tree.

5.2 Random Forest

Random forest is an ensemble method specifically designed for tree classifiers. It combines the predictions made by multiple decision trees, with each tree generated based on randomly selecting input features to split at each node. As with most ensemble methods, we expect improved accuracy, because we are aggregating the predictions through taking votes amongst the base classifiers. We can observe the effect of balancing errors as the forest "grows" in the figure below.

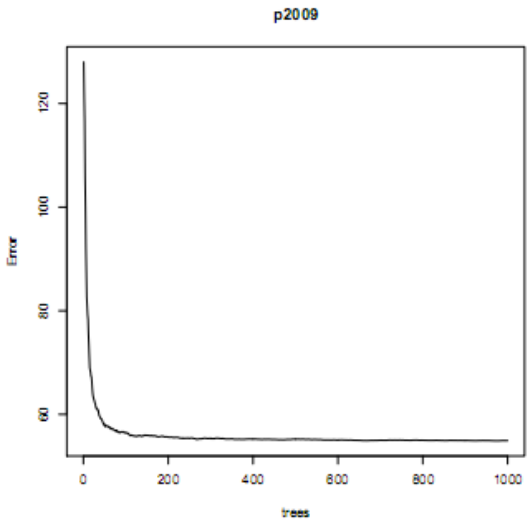

Figure 4. Mean squared error vs. number of trees generated in random forest for training data from the 2009 MLB season.

As long as the classifiers are independent, the random forest will make a wrong prediction only if more than half of the base

classifiers predict incorrectly, much lower than the error rate of the base classifier itself. This independence between classifiers is achieved through the aforementioned random selection of input features on which we perform splits. The advantages we see in using random forest are that it learns quickly even over the full feature set, and that it can also reveal the inputs that are most important.

5.3 Experimental Results

Having generated a separate regression tree for the first two-thirds of each season, we test on the remaining one-third. We evaluate the resulting betting strategy versus the open market, placing bets whenever our expected win (or loss) percentage is greater (lesser) than the converted moneyline odds by a threshold of 5%. The results below reveal inconsistency and large variation from year to year.

Year	Returns (units)
2000	**41.75**
2001	-8.52
2002	**9.72**
2003	-8.75
2004	**57.20**
2005	-37.62
2006	**8.20**
2007	**8.32**
2008	-4.51
2009	**42.69**

Figure 5. Returns from our betting strategy formulated from regression tree, 5% threshold. Gains emphasized in bold.

Likewise, Figure 6 shows the performance of the betting strategy based on random forest, which looks fairly promising across several seasons and a variety of thresholds. In particular, using a 5% threshold results in the largest return on investment.

Season	3%	4%	**5%**	6%
2000	62.90	53.30	**44.00**	30.10
2001	-17.07	-13.67	-17.62	-0.87
2002	3.50	3.85	**1.10**	-7.00
2003	16.35	5.00	**5.30**	9.25
2004	37.50	48.60	**59.50**	58.15
2005	-17.81	-15.61	-4.88	-6.26
2006	35.86	19.09	**34.98**	15.12
2007	14.89	16.66	**18.88**	10.46
2008	-3.50	-6.48	-9.12	-30.61
2009	64.11	59.33	**53.30**	50.09
Profit	196.73	170.07	**185.44**	128.43
Wagers	5964.22	5023.02	**4135.08**	3308.30
ROI	3.30%	3.39%	**4.48%**	3.88%

Figure 6. Returns from our betting strategy formulated from random forest, with variable threshold.

6 Summary

The goal of our project was to apply machine learning algorithms towards developing profitable betting strategies for baseball games. We evaluated two approaches: first, placing bets based on forecasted winners, and second, betting more intelligently, based on estimates of team winning percentage.

Our results suggest that it may be possible to generate returns in the baseball betting market based on the second of these prediction models, but the performance of our strategies is highly sensitive to user-specified parameters and varies greatly between years. A persistent problem with baseball prediction is that past results can provide the wrong impression of current and future.

7 Future Work

We can conceive of many further avenues of exploration for this project:

- Altering moneyline odds on the training set, as a form of amplifying reward and punishment.
- Evaluating other betting propositions, such as run-line bets or totals.
- Consolidating training data across multiple seasons, rather than limiting

training sets to portions of single seasons.
- Performing more careful feature selection, and exploring a wider array of statistics.

8 References

1. Lyle, Arlo. Baseball Prediction Using Ensemble Learning.

2. Maindonald, John and W. John Braun. Data Analysis and Graphics Using R.

3. Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to Data Mining.

4. Segal, Mark R. Machine Learning Benchmarks and Random Forest Regression.

5. Breiman, Leo and Adele Cutler. Random Forests.