Nikhil Bhargava, Andy Fang, Peter Tseng
CS 229 Paper

Machine Learning an American Pastime

I. Introduction

Baseball has been a popular American sport that has steadily gained worldwide appreciation in the past few decades. Over the years, sports statisticians have become very interested in analyzing data from baseball games, coming up with various metrics ranging from the simple, RBIs (runs batted in), to the more complex, OPS (on-base plus slugging percentage), to get a numerical sense of player performance. More recently, the application of statistics to baseball has reached pop culture fame in the movie, *Moneyball*. We wanted to predict the outcome of an upcoming baseball game based on prior information on the teams' respective performances as well as general trends in baseball using a predictive machine learning model.

II. Analyzing the Problem; Our Initial Hypotheses

Baseball as a sport has many interesting characteristics that we tried to account for in our attempts at building an effective predictive model. One issue that consistently arises with baseball is that the sport itself is highly variable. If the best and worst team play each other back-to-back, it would not be unreasonable to predict that they split their games. As a result, we predicted that it is of key importance to include factors that measure a team's recent success. For example, if we looked at a player's batting average over the past 162 games, it is likely that the batting average over this time interval would not change much from one game to another. However, if we were to also examine that player's performance over the last five games, the result would change much more dramatically from game to game. Furthermore, we originally hypothesized that short-term metrics could help with factors like momentum and streakiness, which are highly imprecise yet still considered important to assessing the strength of a team.

This streakiness also means that a short-term view should not be the only data we include about a player. Thus, we hypothesized that it is also important to account for players' performances over the course of previous games, spanning months and even years past.

III. Obtaining Data

While there exist several databases loaded with baseball statistics, we were only able to find ones that aggregated player statistics for an entire season. For the purposes of our project, it was imperative that at any given moment in time, we had the most up to date player statistics and metrics of their recent performances. Consequently we collected game data in the MLB dating back to 2001 but excluding the 2012 season via retrosheet.org. This data provided us with more than 26,000 games and was arranged in the following fashion: for each year, there were plain text files delineating the roster of each team, and there were CSVs for each game scrupulously detailing the play-by-play events. We then converted this unstructured data into relational data. After obtaining our data, we decided to hold out the entire 2011 season from our data and to reserve it for a final run where we better estimate the true accuracy of our classifier. We hold these data points out to prevent any sort of fitting to the data. For our normal development cycles, we elected to test on games in 2010 and to train on only the 2008 and 2009 seasons to increase the throughput of our learning algorithms. This decision was justified as even in two seasons we had 4860 training examples, which was roughly ten times larger than our largest feature set.

IV. Machine Learning Strategy

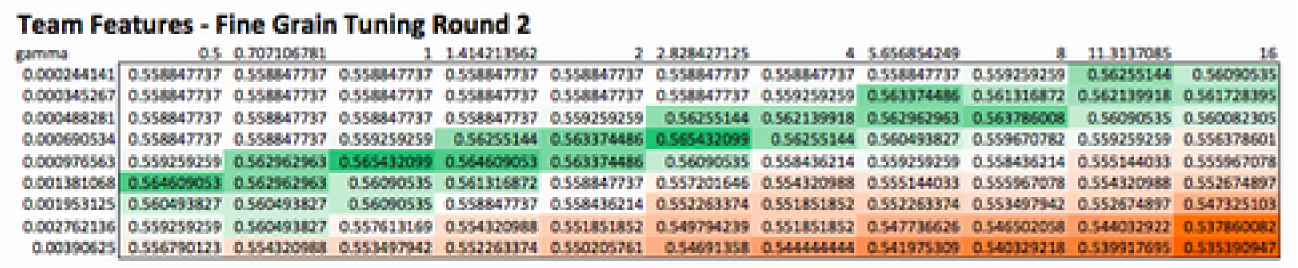
We looked at many different aspects of the machine learning process to determine what algorithm would produce the best output for us. The two elements we varied from run to run were (1) the feature set and (2) the learning model.

Initially, we decided that a support vector machine (SVM) would provide a sufficient model for predicting baseball game outcomes and that it would be a good model for us to test our feature set on, since we assumed that the learning model was sufficient. For a given match-up between two teams, our initial feature set simply looked at team-related aspects such as the head-to-head performance of the two teams, the overall win/loss record of a team over the last *n* games, the overall run differential[1] of a team over the last *n* games, etc. We called this initial feature set the "team features." We then decided to append features related to individual player's career statistics for each of the starting nine batters in each team's lineup; these stats included but were not limited to each player's career batting average and on-base plus slugging percentage.

[1] Run differentials differ from win/loss record because run differentials factor in how much a team won by.

We called this new feature set the “team data and career players data features” feature set. Lastly, we decided to add on information about individual player momentum, examining features like a player’s batting average over the past *n* games and so on; this feature set contained all the features we looked at.

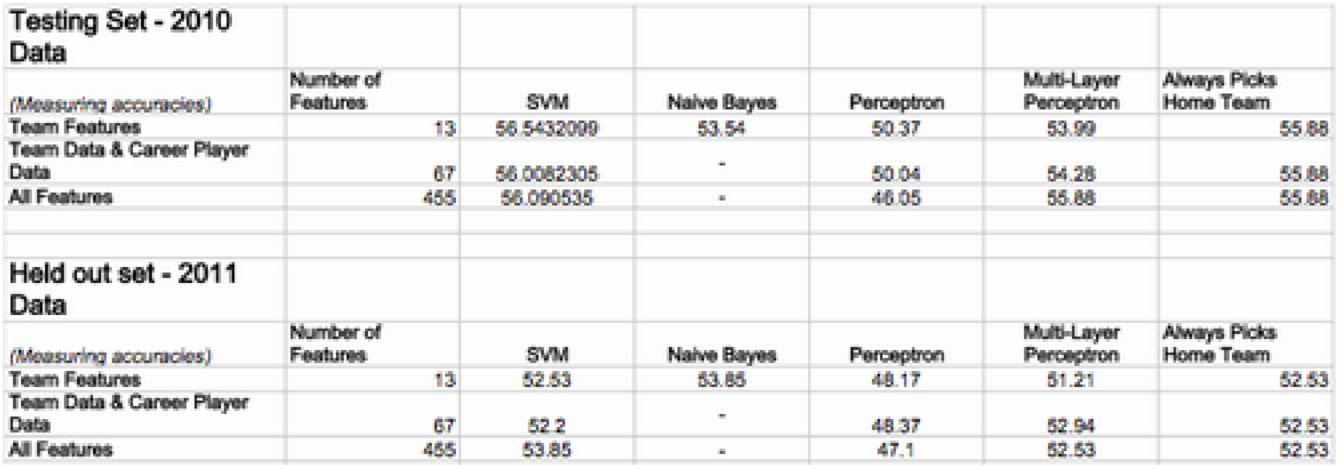
Optimizing the SVM itself proved to be a nontrivial task. We first chose to look at different kinds of kernels we could apply to the model, and in the end, we decided that a radial-basis function (RBF) was the most appropriate kernel to use for our problem since we found that other kernels, such as linear, polynomial, and sigmoid functions, performed consistently worse than the RBF kernel. We used grid search over the two RBF variables C and γ to fine-tune our SVM. We found for many values that the SVM ended up predicting that the home team would win for every game, and we noticed that there was a boundary between value combinations of C and γ that caused the SVM to always guess the home team winning and combinations that caused the SVM to perform worse than guessing that the home team would win. However, along this boundary, we found that the SVM actually out-performed the model that always predicted the home team winning. A heatmap of a part of the grid search results below illustrates this boundary phenomenon, where white space represents the area where the SVM always roots for the home team, red space is where the SVM performs worse than rooting for the home team, and green space is where the SVM outperforms always rooting for the home team:



Team Features - Fine Grain Tuning Round 2

gamma	0.5	0.707106781	1	1.414213562	2	2.828427125	4	5.656854249	8	11.3137085	16
0.000244141	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.559259259	0.56255144	0.56090535
0.000345267	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.558847737	0.559259259	0.563374486	0.561316872	0.562139918	0.561728395
0.000488281	0.558847737	0.558847737	0.558847737	0.558847737	0.559259259	0.56255144	0.562139918	0.562962963	0.563786008	0.56090535	0.560082305
0.000690534	0.558847737	0.558847737	0.559259259	0.56255144	0.563374486	0.565432099	0.56255144	0.560493827	0.559670782	0.559259259	0.556378601
0.000976563	0.559259259	0.562962963	0.565432099	0.564609053	0.563374486	0.56090535	0.558436214	0.559259259	0.558436214	0.555144033	0.555967078
0.001381068	0.564609053	0.562962963	0.56090535	0.561316872	0.558847737	0.557201646	0.554320988	0.555144033	0.555967078	0.554320988	0.552674897
0.001953125	0.560493827	0.560493827	0.56090535	0.558847737	0.558436214	0.552263374	0.551851852	0.552263374	0.553497942	0.552674897	0.547325103
0.002762136	0.559259259	0.560493827	0.557613169	0.554320988	0.551851852	0.549794239	0.551851852	0.547736626	0.546502058	0.544032922	0.537860082
0.00390625	0.556790123	0.554320988	0.553497942	0.552263374	0.550205761	0.54691358	0.544444444	0.541975309	0.540329218	0.539917695	0.535390947

After testing the performance of the three feature sets on the support vector machine, we decided to experiment with different learning models to see whether or not they could outperform the SVM. We looked at naive bayes, perceptron, multi-layer perceptron models and compared their performances to that of the RBF SVM and came up with the following results, which validated our prediction that the SVM was a good model to start out with for predicting the baseball game outcomes:



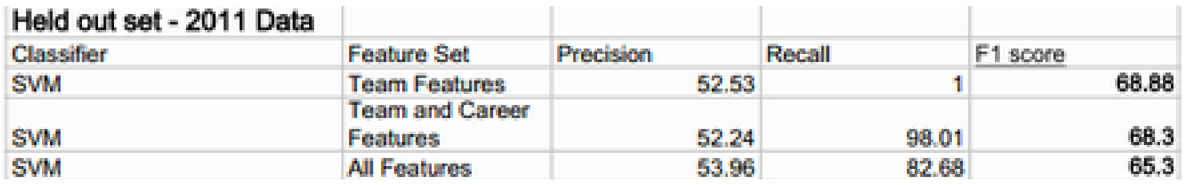
Testing Set - 2010 Data						
(Measuring accuracies)	Number of Features	SVM	Naive Bayes	Perceptron	Multi-Layer Perceptron	Always Picks Home Team
Team Features	13	56.5432099	53.54	50.37	53.99	55.88
Team Data & Career Player Data	67	56.0082305	-	50.04	54.28	55.88
All Features	455	56.090535	-	46.05	55.88	55.88
Held out set - 2011 Data						
(Measuring accuracies)	Number of Features	SVM	Naive Bayes	Perceptron	Multi-Layer Perceptron	Always Picks Home Team
Team Features	13	52.53	53.85	48.17	51.21	52.53
Team Data & Career Player Data	67	52.2	-	48.37	52.94	52.53
All Features	455	53.85	-	47.1	52.53	52.53

V. Analyzing Results

Analyzing our results, we see that to a large extent, our guesses are affected by the high variability of the problem, and in fact the strategy of always picking the home team seems to outperform many of our classifiers, notably the perceptron and multi-layer perceptron. In terms of our results generally, we saw a large drop-off between our training results and our testing results. We hypothesize that this could be due to overfitting of our data. In a few select cases, certain feature sets outperform the always-pick-the-home-team strategy, but we feel like they do not exceed the strategy by a large enough margin for us to consider it to a superior classifier.

One noteworthy point was that our Naive Bayes classifier did roughly as well on the testing set as it did on the training set, and additionally it seemed to outperform the strategy of always guessing that the home team wins. Future steps could involve expanding on our Naive Bayes classifier to see whether additional features would help in its prediction. At least initially, we decided not to include individual player features in our Naive Bayes classifier as we had no good standard for discretizing a continuous-valued range (such as batting average) into individual buckets.

Since winning and losing a baseball game, the two possible classifications for the SVM, happen at the same rate, using the accuracy of the model's predictions as a measuring stick is reasonable. However, we also tried using the F1 scoring mechanism to see if they tell us anything more meaningful about our features than simple accuracy:



Held out set - 2011 Data				
Classifier	Feature Set	Precision	Recall	F1 score
SVM	Team Features	52.53	1	68.88
SVM	Team and Career Features	52.24	98.01	68.3
SVM	All Features	53.96	82.68	65.3

It is interesting to observe that though measuring by accuracy seems to indicate that having a learning model that accounts for features about the team, individual career statistics, and individual momentum statistics is better than one that does not, but measuring by F1 score actually gives the opposite result. With the research we conducted, we believe it is inconclusive which measurement system gives us a clearer picture of what the right feature set is for predicting baseball game outcomes.

VI. Conclusion

In conclusion, the feature set and classifier combinations that we explored performed only marginally better than using the simple heuristic of always selecting the home team to win even after optimizations like grid search. We highly doubt that performing a more extensive grid search or trying another classifier will yield better results. However, we suspect that the feature sets that we chose hindered our performance. The natural follow up question is: what does this mean in the context of baseball? With moderate confidence, we assert that:

- head-to-head team statistics
- recent win-loss and run differentials versus other teams
- metrics[2] of individual players' batting performances in recent games[3]
- metrics of individual player's batting performance for their entire careers

do not have a large impact on the result of an individual game. We conjecture that adding features to capture more information about pitchers will increase performance of the various classifiers. We recommend that future baseball statisticians pursue models which account heavily for pitcher data. Ultimately, we will not be taking our machine learning project to Vegas.

[2] metrics include OPS, batting average, strikeout rate, and walks per game
[3] recent games include the last 1, 2, 5, 10, and 20 games