Parity and Predictability in the National Football League
Final Write-up
Siddharth Patel

Abstract

Predicting the outcome of National Football League games is a staple of American sports culture and a significant component of the multi-billion dollar sports betting industry. The NFL prides itself on the perception of parity, the idea that any team can beat any other on "any given Sunday." In this paper, I apply machine learning techniques to team performance statistics in order to predict the outcome of games. I seek to determine with what accuracy the outcome can be predicted and how those predictions would perform against sports book betting odds. After tuning a random forest classification tree on the last ten year's worth of NFL games, I find that I can predict the outcomes of games in the 2013 season with 60% accuracy. I cannot consistently make money against Las Vegas sports book odds.

1. Introduction

A sports league is said to have parity when most teams in the league have similar levels of talent and are equally likely to do well in a season or to beat each other in head-to-head matchups. Parity is important to the reputation of the NFL, and this commitment manifests itself where it counts - in the money. The NFL has a hard salary cap, so no one team can just buy up a bunch of talented players. In addition, the NFL pools and redistributes the massive television revenue, ticket sales, and merchandise sales relatively evenly to all 32 teams. This shared revenue system is meant to hinder any one team from establishing a wealthy and winning dynasty. The NFL does not want the New York Yankees model of sports teams. The perception of parity has significant advantages for the NFL. Fans pay attention for most of the regular season because they believe that their team, with the right breaks of luck here and there, may win enough games to get into the playoffs, or may defeat the best team in the league in a key matchup. This sense of unpredictability, the belief that one standout performance from one player or one big break on one key play can make all the difference, is the bedrock of the excitement generated by NFL games.

That said, some teams are obviously better than others, as demonstrated through their win-loss records, team statistics, and playoff performance. This information is readily available and remarkably complete. My hypothesis is that machine learning techniques can be applied to this dataset to accurately predict the winner of a given football game. Other CS229 students have already given this a try. (Hamadani; Sierra) Specifically, to predict the outcome of a regular season game, I will use the team performance statistics from all of the games in the prior weeks as a training set. As the season progresses, the machine learning algorithm should be able to more accurately predict who will win any particular game. I do not believe that any given game is truly a 50-50 coin toss, so I want to compare my prediction accuracy to some other measure - and sports betting odds provide a great reference point.

The National Gambling Impact Study Commission estimates that total annual sports wagering reaches into the hundreds of billions of dollars internationally. Legally operated sports books in Nevada handle over $1 billion annually on football. (Karp, 2009) Given the large sums involved, one would expect that these sports books define betting odds that are very hard to beat consistently. Therefore, I will consider my machine learning algorithm accurate if it can consistently make money against Las Vegas sports book odds, even if it's just for some period of the regular season. One important note - sports books will adjust the odds they offer based on the volume of wagers coming in on either side of the bet. Sports books want to keep even money on both sides of the bet so that once the game finishes, the losers' money will pay the winners, and the sports book keeps a cut. This method has an important im-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

plication - the way that human fans bet will influence the money line odds faced by my machine learning algorithm. Human bettors may be influenced by many factors, but they probably pay some attention to what sports broadcasters and analysts say.

If my algorithm can predict the outcome of games accurately, it undermines to an extent the claim of parity in the NFL. In particular, I intend to examine how often the algorithm gets "very confident" predictions wrong. Furthermore, if my algorithm can consistently win against Vegas sports book odds, then perhaps it will have found the true signal underneath the noise of sports analysts, hunches, and loyalties.

2. Data

The NFL keeps track of an enormous volume of team and individual performance statistics. I decided to focus on team game statistics alone because using individual player statistics would overcomplicate this project. In addition, I believe that football is fundamentally a team sport, moreso than baseball or basketball.

I use historical data from the 2003 through 2012 regular seasons because the set of teams has remained the same over that period (i.e., no team moved, no expansion team was created). I focus exclusively on regular season games because playoff games are played very differently, in terms of strategies and schemes, so I don't want to mix the two up when trying to learn game outcome prediction.

At the end of each game, the team statistics for each team are tallied up and reported. Here are some examples of team statistics:

- Yards gained per rushing play
- Total passing yards
- Number of interceptions thrown
- Total touchdowns

The NFL's website reports approximately 40 such statistics per team per game.

I learned how to web scrape using Python's BeautifulSoup package, and I am very glad to have acquired that skill. I wrote a script that extracts team game statistics for each game of the 2003 to 2013 seasons from the NFL's website. (To be frank, for me the most exciting part of this project was watching my web scraping script start executing successfully!) I created a dataset of money-line odds for each game in a similar manner, scraping from FootballLocks.com. I use the 2003-2012 seasons to compare and train the machine learning algorithm, and I will test the algorithm on the 2013 season.

2.1. Feature definition

For each game, I construct a feature vector that can be used to predict that game's outcome. The basic idea is that the average performance of a given team over the course of the season to date should be useful in predicting whether or not they will win the current game. Let $\mathbf{p}_i^w$ be a vector of the 40 game statistics achieved by team i in its game during week w of the current season. Suppose we are predicting the outcome of game G in season S, week W, between team $T1$ and $T2$, where $T2$ is always set as the home team in order to consistently capture home field advantage. The feature vector for this game is defined as $\mathbf{x} = [\mathbf{p}_{T1}\ \mathbf{p}_{T2}]$, where:

$$\mathbf{p}_{Ti} = \frac{1}{W-1}\sum_{w=1}^{W-1}\mathbf{p}_{Ti}^w \tag{1}$$

For example, if the San Francisco 49ers are playing at home against the Seattle Seahawks, then to compute $\mathbf{p}_{T1}$, I average together all of the Seahawks' game statistics from their prior games of the current season, do the same thing for 49esr to compute $\mathbf{p}_{T2}$, and then concatenate the two vectors to form the feature vector for the current game. Note that I do not include performance in games from prior seasons when constructing this feature vector because I have a sense that team performance can change quite a bit from season to season, due to player trades, injuries, coaching changes, and intangibles like morale and momentum.

The feature vector as defined above is rather naive, with two major deficiencies. For one, the NFL game statistics for a particular team focus almost entirely on offensive performance, but defense is definitely important to game outcomes. In addition, taking the simple mean of the last several weeks doesn't take into account that more recent performances probably have more information than more distant ones, so some type of time-weighting would probably be useful. Complicating the matter, the feature vector has dimension $n \approx 80$, which is relatively large compared to the number of games in each season, 256. For example, when making predictions in the early weeks of the 2007 season (the fifth season I am considering), the training data set would have dimension $\approx 1000 \times 80$, a setting that may be susceptible to high variance and overfitting. I did intitially reduce the dimension of the variable space by weeding out a handful of statistics that

were linearly dependent on others, but before investing much more time into fine-tuning the feature vector, I started by testing a few different classification methods against each other.

3. Initial trials

In order to get some sense of where to put my time, I ran initial trials comparing a naive Bayes classifier, a random forest, logistic regression, and an SVM. Each of these classifiers was run using the simple feature vector $\mathbf{x}$ as defined above. For predicting whether team $T2$ would win a game in week W of a given season, the classifiers used the games from prior seasons and from the past $W-1$ weeks of the current season as a training set. Thus, with each passing season, the size of the training set grows, so the classifiers should get better at prediction with time.

3.1. Naive Bayes classifier

I took Professor Ng's suggestion to start with a simple-to-implement naive Bayes classifier in order to build an end-to-end system quickly. The initial results using naive Bayes were promising from the standpoint of predicting wins and losses. (For simplicity, I disregard the possibility of ties, though there have been four over the last decade). Figure 1 plots the average classification error (0-1 loss) for each season for each of the classifiers. The naive Bayes classifier predicts the outcome of games correctly more often than not. A misclassification rate of 0.4 is meaningfully better than random guessing.

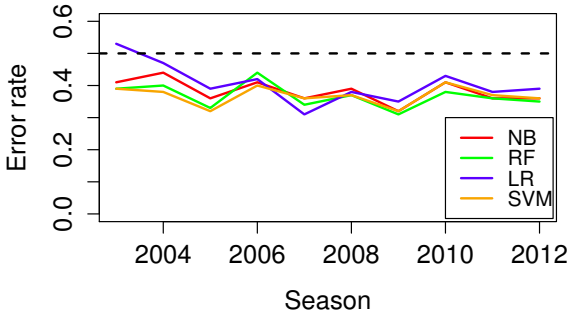

Figure 1. Classification error rates (0-1 loss) by season for each of the classifiers initially tested. They are all comparable, and all are better than random guessing. The random forest is the most accurate in the last few seasons, but only by a thin margin.

For each prediction, the naive Bayes classifier outputs the posterior probability of each class - the probability that team 2 loses (which is the same as the probability that team 1 wins), and the probability that team 2 wins. I used the maximum of the two probabilities in order to predict the winner of the game. I kept track of this maximum posterior probability as a measure of the classifier's confidence in its prediction. Figure 2 is a histogram - it shows what percent of the predictions were made for a given prediction probability. In short, more than 50% of the time, the classifier made its prediction with almost 100% confidence. It would be useful if the classifier's most confident predictions were also consistently more correct, but this is not the case; there was no decrease in the error rate with higher-confidence predictions.

Taken together, these two conditions are serious limitations for use in betting. The naive Bayes classifier makes most of its predictions with probability close to 1, so this makes it impossible to compute a meaningful expected payoff of a bet. In addition, the high-confidence predictions are no more accurate than the low-confidence predictions, so that makes it difficult to set some threshold on confidence above which bets should be placed. These deficiencies drove me to look for other methods.

3.2. Random forest

I learned about random forests in Statistics 202 this quarter. I varied the main tuning parameter, the number of trees, to find the best performer with reasonable computation time, settling on 500 trees. The random forest generates probability estimates for its predictions, based on the fraction of votes that each class received from the 500 trees. For example, if 300 of the trees predicted that $T1$ would win, then the prediction probability would be 0.6. Figures 3 and 4 show that the random forest has nice behavior in terms of its prediction probabilities. It makes predictions over all ranges of confidence, and the higher confidence predictions tend to be more correct.

3.3. Logistic regression and SVM

I implemented logistic regression and got very similar results to the naive Bayes classifier, so I will not describe them in detail for the sake of concision. As for the SVM, I spent a lot of time trying to tune the SVM, eventually settling on the radial basis function kernel and a very low penalty for margin violations. As seen in Fig. 1, the SVM performed well in terms of classification error. In addition, Figures 3 and 4 demonstrate that its prediction probabilities behaved nicely, similar to the random forest. I'm not certain how the 'e1071'

package in R generates probabilities for the SVM predictions, but I took them as a black box output that I used for evaluating the algorithm and placing bets.

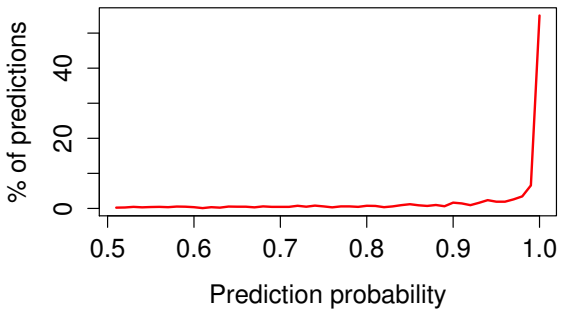

Figure 2. The naive Bayes classifier made most of its predictions with extremely high confidence; that is, the posterior probability for the predicted winner was almost always near 1. This undermines the ability to compute a meaningful expected value of betting on the predicted winner.

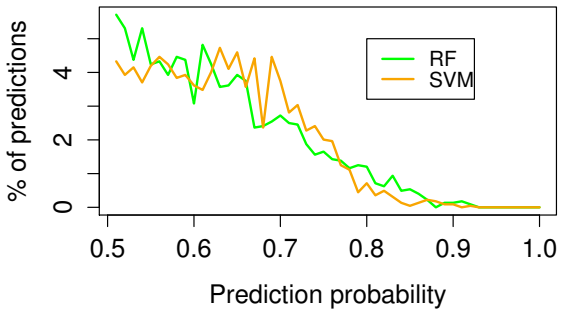

Figure 3. Both the random forest and the SVM make predictions over a spectrum of confidence levels. This suggests that the probabilities output by the classifiers may contain useful information that can be used in evaluating whether to place a bet or not (e.g., expected payoff).

3.4. Choosing just one

I decided to choose the random forest for four main reasons. The first is that it was one of the most accurate classifiers, and it seemed to be getting better in the later seasons. The second is that its prediction probabilities behaved in a way that could make them useful for evaluating bets. The third is that the random forest can evaluate the importance of the various variables it uses for prediction, producing a summary of which variables contributed the most to the accuracy of the classification trees. This interpretability advantage could be useful in variable selection or in evaluating the stability of the prediction algorithm. Lastly, random forests tend to have less problems with high variance and overfitting because each of the trees created has a certain randomization in which variables it chooses when growing leaves; therefore, the trees tend to be weak learners that are not strongly correlated with each other.

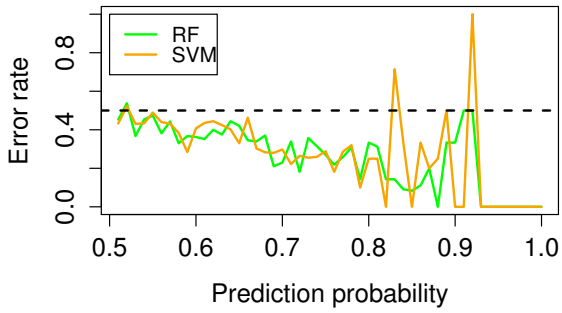

Figure 4. The more confident the random forest or the SVM is in its prediction, the more likely the prediction is to be correct. The spikes on the right hand side are likely noisy side-effects of the fact that very few predictions were made with probability greater than 0.9, as seen in Fig. 3.

I also evaluated the amount of money that I would have won or lost if I had made bets based upon the classifiers' predictions. The betting strategy was to place a bet if the classifer made a prediction with confidence greater than 75% and if the expected payoff of the bet was positive. This strategy led to the following over the last three seasons of the training data, 2010 to the 2012: $4,000 lost using naive Bayes; $1,000 won using random forest (on a total wager of $4765; $1,200 lost using logistic regression; and $400 lost using the SVM. These results were mostly in line with the observations about the probability outputs from each classifier. The naive Bayes and logistic regression classifiers were too confident and encouraged too many bets, whereas the random forest and SVM were more tentative about their predictions and therefore placed fewer bets.

4. Improving the random forest

My efforts to improve the random forest focused on the feature vector itself. The first thing I did was to add defensive statistics to the feature vector - for exam-

ple, how many net yards of offense did the *other* team achieve, how many rushing touchdowns, etc. Now the random forest could take into account defensive performance. This generated a slight improvement the prediction accuracy, though increasing the dimension of the feature vector increases the risk of overfitting. Next, I implemented a very simple weighted mean, applying a geometrically decaying weight for performance statistics from more distant weeks, but that did not improve accuracy. Lastly, I aggressively stripped down the feature vector by removing variables that I did not think were the most important to game outcomes. This variable selection did not noticeably improve prediction accuracy, but it didn't make it all that worse either - and for the purposes of predicting accurately on 2013, a more parsimonious model may face less risk of overfitting on the 2002-2012 training set, and therefore may be more accurate.

5. Validation and Conclusions

For validation testing, I ran the random forest classifer and betting algorithm on data and odds from the 2013 season through Week 14. None of this data was used to tune the random forest a-priori. Figure 5 is a visualization of the prediction accuracy for the random forest with defensive statistics added. The map shows that in later seasons and later weeks the algorithm tends to be right more often, which is a good indication that it's learning something meaningful. With a relatively simple approach, I was able to predict games in 2013 with an error rate of .395, which is just mediocre. For comparison, I looked at the prediction accuracy of thirteen experts designated by espn.com for the 2013 season. The most accurate expert has a .324 error rate to date, and the least accurate, .401. My algorithm did a little better than the worst expert.

Placing bets with this algorithm was surprising - it decided to bet on just one game in 2013! It won that bet, so the net balance for 2013 is a gain of $175. The accuracy for 2013 is noticeably worse than what it was for 2010-2012, so perhaps the fact that the random forest was not predicting well was reflected in lower prediction probabilities, meaning that the algorithm almost never thought it was confident enough to bet.

Upon reflection, I believe that tuning for prediction accuracy implies a different objective function that tuning for winning bets. During the training on the 2003-2012 data, I focused on minimizing the 0-1 loss on the assumption that this would also minimize betting losses. For raw prediction accuracy, it only matters that the probability be over 50% for the team most likely to win. For betting, in particular for evaluating the expected payoff of a wager, it matters that the probability actually be close to the 'true' probability of the team winning. Therefore, if my priority had been to choose and tune algorithms to perform well in the gambling setting, I would have needed to use the monetary winnings and losses as the key criterion instead of the 0-1 classification loss.

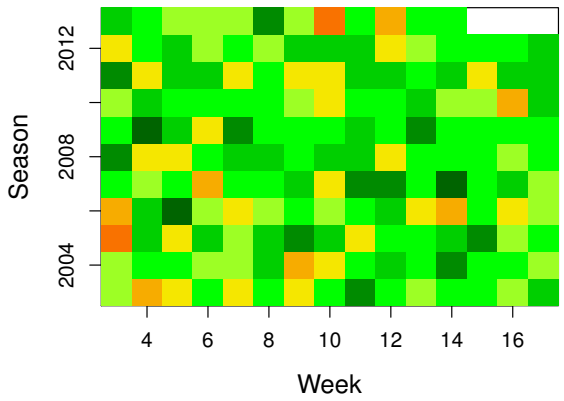

Figure 5. Color map of prediction accuracy for the final random forest algorithm. Green hues indicate that for a given week in a given season, most of the predictions were correct (the darker the better). Yellow to red hues indicate that most of the predictions were wrong (the darker the worse). The map is greener in the upper right half.

It's not terribly surprising that my project didn't uncover the secrets of the NFL and Las Vegas simultaneously, but it is surprising that the algorithm didn't do a little better than 60% accuracy. In my opinion, this reinforces the notion that the outcome of NFL games is pretty hard to predict. Based on the results of this project, I cannot substantively argue against the claim of parity in the NFL. And that's fine by me.

References

Hamadani, Babak. Predicting the outcome of nfl games using machine learning. URL http://cs229.stanford.edu/proj2006/BabakHamadani-PredictingNFLGames.pdf.

Karp, H. The nfl doesn't want your bets. *The Wall Street Journal*, 2009. URL http://online.wsj.com/news/articles/SB124511421029417367.

Sierra, A, et. al. Football futures. URL http://cs229.stanford.edu/proj2011/SierraFoscoFierro-FootballFutures.pdf.