Football Futures

Adrian Sierra
Vigilante

James Fosco
TinMan

Carlos Fierro
Sleeping Tiger

1 Introduction

The emergence of online fantasy sports since the mid 1990s and a continued growing interest in sports handicaps for betting purposes have shown the significance of accurately predicting the outcomes of sports games. Traditional methods for predicting sports games uses several features to predict future games, assigning a point spread to provide input on the outcome. Popular free oddsmakers set the line to keep an equal number of bets on both sides with limited features. Private methods with more features that are given different weights would prove more effective and more accurately predict game outcomes without any adjustments by oddsmakers.

The difficulty of predicting game outcomes is the assembly and analysis of game statistics, player statistics, and other features in a meaningful way. Also, this data must consistently be analyzed and updated after each new game to more accurately predict upcoming games. A system that analyzes freely available game data and can be updated efficiently to make predictions would prove useful for those interested on the outcome of an upcoming game.

In this paper, we explore the use of machine learning to process, weigh, and interpret game statistics to correctly predict game outcomes. Our work will focus on only one sport, NFL football, but a similar approach for interpreting game data can be used for other similar sports. We will attempt for predictions to be competitive with professional sports bettors and test our data on several algorithms: SVM (linear, polynomial, and radial basis function) and logistic regression (including locally weighted).

2 Game Predictions

2.1 Training Data

Given an upcoming game, we need to be able to predict the winning team and losing team. To do this, we needed training data related to both teams' past performance on individual games. This data was obtained freely from ESPN's game stats website, http://espn.go.com/. Box Scores provided valuable game-by-game data and individual statistics for each game.

Manual input into correct matrix format for MatLab is something we wanted to avoid. To do this we scraped each game's Box Score web page and stored the features in a nicer table format using MySQL. This was automated using shell script, gawk, and python to do some initial parsing. Our script allows us to specify a year and receive all of ESPN's game data for every game in that year. ESPN has game data dating back to 2009

in a convenient format, which gave us a sufficiently large data set to train and test our algorithms on. Additionally, the automation of scraping online data proved valuable for updating our learning algorithm and in saving time gathering data. Fast data collection is as necessary as a strong algorithm so that it can be used quickly and easily.

Once we had the Box Score site for a game, we then created an HTML parser in python to parse and write the data into an easier format for our database. This database will allow us to manipulate our NFL data quickly and efficiently, and allows for us to make individual queries over the data to try to find additional patterns. Taking the difference, average, and other important values is much easier by using database queries wrapped in python.

2.2 Features

Training samples collected include the following features and facts on a per game basis:

Team One	Team Two
Team Identification • Team ID • Name • City	**Team Identification** • Team ID • Name • City
Game Identification • Game ID • Season • Week	**Game Identification** • Game ID • Season • Week
Turnovers • Fumbles • Interceptions	**Turnovers** • Fumbles • Interceptions
Red Zone • Made • Attempted	**Red Zone** • Made • Attempted
Penalties • Number • Yards	**Penalties** • Number • Yards
1st Downs • Total • Passing • Rushing • Penalties	**1st Downs** • Total • Passing • Rushing • Penalties
Game Statistics • Total Plays • Total Yards • Non-Offensive TDs • Possession time	**Game Statistics** • Total Plays • Total Yards • Non-Offensive TDs • Possession time

3 Machine Learning Algorithms

Initially, we ran linear support vector machine (SVM) on the data from the 2011 season. To test the algorithm on our data we trained the algorithm on the games from weeks 1 to 9, and then put each teams average performance over the first 9 weeks into the match-ups that occurred during the 10th week. This basic case gave us 62.5% accuracy, correctly guessing the winner of 10 / 16 games on average. This accuracy shows that even our initial algorithm has a statistical advantage over random guessing. However, while it's better than guessing, we need to be able to beat typical calculated odds in order to turn a profit. This means we will have to beat about a 68% accuracy rating.

To improve accuracy, we tried to manipulate our features in a number of ways. For one, we found that our initial way of training on specific game data and testing with a team's average game data gave results with extremely high variance, often going down below a 30% prediction rate on some weeks. To fix this, we tried normalizing our data so that every feature had values between 0 and 1, which helped marginally. However, the major change came with a larger overhaul of our features.

First, we collected more data, so that in total we had games from years 2009, 2010, and 2011. Then we made each training sample contain the average of a team's stats over their past 5 and 10 weeks, to account for the most recent performance of that team. This also enabled us to actually test on the same type of data that we trained on, giving more consistent results.

With this method we achieved an average prediction rate with linear SVM of about 66.34% (138/208) on each week of 2011. Interestingly, using the 5 and 10 week game histories did not make much of a difference at all. Additionally, using 2nd and 3rd order polynomial SVM actually lowered our test results to 56.25% (117/208) and 57.21% (119/208), due to over fitting (there was 0 training error).

To improve our prediction rates, we also decided to manually remove major upsets from our training data. In order to get an overall better success rate, we had to accept that it's often nearly impossible to predict certain outcomes, so these outliers were removed. This managed to boost our linear SVM predictions to an average of 68.27% (142/208).

	Training Error	Test Accuracy	Best Week	Worst Week
Random	-	51.92%	-	-
Logistic Regression	-	31.73%	62.5%	25%
Linear SVM	34.86%	68.27%	93.75%	46.15%

Polynomial SVM (2 & 3)	0%	57.21%	87.50%	42.86%
RBF SVM	0%	55.77%	76.92%	28.57%

Analysis and Conclusions

ESPN Expert Results of 2011 Season (through week 14)

Name	Accuracy	Games
Allen	65.53%	135/206
Golic	64.90%	135/208
Hoge	64.73%	134/207
Jaworski	65.82%	129/196
Mortensen	61.54%	128/208
Schefter	62.02%	129/208
Schlereth	67.31%	140/208
Wickersham	66.35%	138/208
Accuscore Algorithm	68.75%	143/208

In comparison to ESPN's experts and ESPN's Accuscore algorithm, our best solution (using linear SVM) performs at as least well. Specifically, it performs on par with even the best of the expert picks, as well as with the Accuscore algorithm. Given that ESPN's "experts" are paid to analyze and predict football games, we consider this a success.

Our linear SVM performed fairly well, with an accuracy of 68.27%, and did better than our 2nd and 3rd degree polynomial SVM. Over fitting the training set was, therefore, a significant problem that our linear model avoided. Furthermore, omitting game statistics from major upsets that were certainly outliers improved our accuracy. Careful data and feature selection is therefore necessary to avoid over fitting and improve accuracy. Other

algorithms that performed worse used a large number of features and included all past season data to make their predictions.

Possible areas of improvement include using more features, including defensive statistics for each team. Our current feature set only includes non-offensive touchdowns, and therefore cannot get a comprehensive understanding of a team's overall performance. This resulted in a sub-optimal evaluation of teams with a better defense, weighing our algorithm toward offense heavy teams.

References

1) Beating the NCAA Football Point Spread. http://cs229/proj2010/LiuLai-BeatingTheNCAAFootballPointSpread.pdf

2) http://www.mathworks.com/help/toolbox/bioinfo/ref/svmtrain.html

3) ESPN NFL Expert Picks
http://espn.go.com/nfl/picks

4) ESPN Football Scores
http://scores.espn.go.com/nfl/scoreboard