Beating the NFL Football Point Spread

Kevin Gimpel
kgimpel@cs.cmu.edu

1 Introduction

Sports betting features a unique market structure that, while rather different from financial markets, still boasts high stakes and offers large amounts of publicly-available data to learn from. While using statistical models for forecasting in financial markets has been widely explored, exploiting the sports betting market has gone largely undiscussed in the machine learning community. We will describe efforts to attempt to win money by automatically choosing the winning teams (given the point spread) of National Football League (NFL) games. Most analyses of the professional football betting market have concluded that it is an efficient market, suggesting that it is very difficult to formulate a betting strategy that will succeed in the future.[3, 9, 1, 6] By pursuing this problem from a machine learning perspective, we will be challenging the market efficiency hypothesis empirically.

There are many kinds of bets for football games, including season-long bets regarding division winners or individual player statistics, but the most popular scenario revolves around the *point spread* for an individual game. The point spread is a positive or negative real number (where the fractional part is limited to either 0 or 0.5) that represents the predicted difference between the away team's score and the home team's score. For example, if the point spread is set at -3, the home team is effectively favored to win by 3 points. If one bets money on the home team and the home team wins by more than 3 points, the bettor wins, and the home team is said to have won *with the spread* (WTS). If the away team wins or if the home team wins by fewer than 3 points, the bettor loses, and the away team has won with the spread. If the home team wins by 3 points, no money is won nor lost; this situation is called a "push". The purpose of the point spread is to obtain near-equal amounts of money wagered on both teams for each game. We will assume that the pay-off odds are equal for both teams in each game. This is a realistic assumption and makes evaluation more intuitive: if we can consistently choose the winning team with the spread more than 50% of the time, we can win money. We will see that, while we can improve above baselines, it requires careful feature selection to get very far beyond 50%.

We explore the problem of using machine learning to automatically choose which team to bet on in an NFL football game. The problem reduces to a binary classification task in which the binary output determines which team to place money on given a pair of teams and a point spread. Therefore, positive training examples are those which the away team won with the spread and negative are those the home team won with the spread. The data consists of statistics from each regular season and postseason NFL football game between 1992 and 2001, totaling 2,487 games in all. Information for each game includes the date, the home and away team names, the final score, the point spread, and 37 additional fields, ranging from playing surface type (dome, artificial, or grass) to offensive and defensive statistical categories. In addition to the 43 data fields for each game, many sports bettors tout the utility of complex heuristics such as, "Since 2001, following a win against a division opponent, the Dallas Cowboys failed to cover the spread 6 out of 7 times in their next game." Given the infinite number of computable features such as this and the general lack of related work in this area, this paper will focus on the problem of feature selection for this task. We build upon intuitions about the domain and extract 230 features from the dataset and then use a randomized search algorithm to explore subsets of features to find the best-performing set. We use a logistic regression classifier for this approach due to its speed and ability to handle arbitrary features of the input data, and we experiment with a support vector machine classifier as well.

To evaluate, we will compute the accuracy of a classifier as follows:

$$accuracy = \frac{\#\ of\ times\ correct\ team\ chosen}{number\ of\ games - number\ of\ pushes}.$$

When a push occurs, bettors receive their money back and the result is as if no bets had been placed. Since we are viewing the task as a binary classification task, there will never be a correct prediction in such games, so none of the pushes will be included in the numerator of this quotient. Since they should not count for or against the accuracy, pushes should not be included in the denominator either, so we subtract them from the total number of games to compute the accuracy. Point spreads are chosen to be maximally difficult to beat, so even a success rate of 5% above a baseline would be considered a success. It has been said by certain professional sports bettors that one should not expect more than approximately 60% accuracy in betting, so this is the figure that we will strive towards.

The remainder of this paper is laid out as follows. In Section 2, we describe related work. In Section 3, we present our approach to feature selection and classification and in Section 4 we describe our randomized search algorithm for feature selection and give developmental results. Finally, we give our test results and discuss future work in Section 5.

2 Related Work

There has been a significant amount of theoretical work focusing on the notion of efficiency of the NFL football betting market. Zuber et al. (1985) present a negative finding, showing one study suggesting the possibility of inefficiency in the market. Golec and Tamarkin (1991) also arrive at the conclusion of inefficiency via statistical tests. More recently, Boulier et al. (2006) studied the NFL seasons from 1994 until 2000 and found that the betting market could not be shown to be inefficient during that time. Levitt (2004) rules in favor of efficiency and presents some arguments for why empirical studies have displayed inefficiency in the past. The recent market efficiency results are troubling when considering applying machine learning techniques to this problem; our efforts here can be seen as challenging the efficiency hypothesis by an empirical approach.

Irrespective of the efficiency verdict, there have been several statistical models developed to attempt to beat the spread. Harville (1980) used a linear model based on home-field advantage and a measure of team performance level over the course of a season and achieved moderate success. More recently, Glickman and Stern (1998) defined a state-space model to model a team's strength during the course of a season and used it to predict point differentials in NFL games. Related work includes models developed by Knorr-Held (2000) and Stern (1991), but in general there has not been a great deal of work in this area. Stern (1991) developed a model to estimate the probability of a team's success during a season based on the sequence of point spreads during the regular season, but did not attempt to pick the team to bet on. Knorr-Held (2000) defined a model to capture the strength of a team over the course of a season for use in rating of sports teams. Like Stern (1991), he did not apply his methods to sports betting, but there is nothing to prevent one from incorporating ideas from both authors for solving this problem. The problem of designing an algorithm to choose the team to bet on in a sports betting scenario has received very little attention

Wins in current season	Avg points scored in last k games
Wins WTS in current season	Avg points given up in last k games
Wins in last k games	Avg 1st quarter points in last k games
Wins with the spread in last k games	Avg 4th quarter points in last k games
Losses in current season	Avg points + 4th quarter points in last k games
Losses WTS in current season	Avg pass attempts in last k games
Away team won more than home team in curr season	Avg pass completions in last k games
Away team won WTS more than home team in curr season	Avg pass completion percentage in last k games
Away team lost more than home team in curr season	Avg yards per pass attempt in last k games
Away team lost WTS more than home team in curr season	Avg yards per pass completion in last k games
Away team record on the road in current season	Avg yards per rush attempt in last k games
Away team record WTS on the road in current season	Avg sacks in last k games
Home team home recordin current season	Avg fumbles lost in last k games
Home team home record WTS in current season	Avg interceptions thrown in last k games
Avg points beat spread by in current season	Avg turnovers in last k games
Avg points missed spread by in curr season	Point spread
Avg pts beat minus missed spread by in current season	Total points line

Figure 1: The full set of feature templates for each game. WTS stands for "with the spread". Letting k take values from the set $\{1, 2, 3, 4, 5, 15\}$ and duplicating features for both the home and away teams gives us a total of 230 features for each game.

by the machine learning community, so there are many possibilities for techniques that we can bring to bear on this problem.

3 Approach

A common strategy in football betting is to look at the recent history for each team in the game of interest. Statistics can be collected from these game histories, including overall win-loss record, win-loss record with the spread, points scored, and a variety of offensive and defensive statistics averaged over game sequences of varying lengths. It is customary to look at the season history for each team in a game, but not to stretch back into the previous season.[1] Due to the lack of published work in this area, it is unknown which features among the many are most useful for this task. So, we first focus on the problem of feature selection. To do so, we need a classifier that can handle arbitrary features of the data, including overlapping features, so we used a binary logistic regression classifier for developmental testing and feature selection. In order to find a feature set that will be effective for future NFL seasons, we do all of our developmental testing using cross-validation by season on the training data to get as accurate a measure as possible of the quality of a feature set. The training data consists of the 8 NFL seasons from the 1992 season up to and including the 1999 season. We reserved the last two seasons – 2000 and 2001 – for final testing, which we describe in Section 5.

We first extracted a set of 230 features from the data based on intuitions about the domain. This feature set is shown in Figure 1. To perform preliminary feature selection, we used a series of techniques. First, we computed the Pearson correlation coefficient for the features to see how well they correlated with the output variable (the team which won with the spread). The features with the highest squared Pearson correlation coefficient values are shown in Figure 2 below:

[1]In all experiments in this paper, the games from the first 4 weeks of each season, the final week, and the playoffs are excluded from training and testing. The early part of the season is difficult to predict given the lack of season data; the last week is often atypical as secure teams rest their star players; and the playoffs produce atypical game sequences, as all of the teams will be coming off a win or a bye. This still yields approximately 170 games per season.

Feature	r^2
Away team wins in current season	0.0044
Point spread line	0.0042
Home team won more than away team in season	0.0032
Away team lost more than home team in season	0.0029
Away team won WTS more than home team in season	0.0025
Home team average yards per completion in last 2 games	0.0025
Home team average yards per completion in last 3 games	0.0023
Away team wins WTS in current season	0.0022
Home team average yards per completion in last 4 games	0.0020
Away team losses in current season	0.0018
Home team average yards per pass attempt in last 4 games	0.0018
Away team lost WTS more than home team in current season	0.0017
Home team average yards per pass attempt in last 3 games	0.0017
Home team lost more than away team in season	0.0016
Home team lost WTS more than away team in current season	0.0016
Home team won WTS more than away team in season	0.0016

Figure 2: A partial listing of features in decreasing order of square of Pearson correlation coefficient. Each number represents the correlation between the feature and a boolean output variable indicating which team (away or home) won with the spread.

We also attempted to measure the performance of each feature individually and use the resulting accuracies to guide the grouping together of features. In preliminary experiments, the point spread feature was found to be essential in exceeding the 50% mark, so we included it with each of the individual features and then trained a logistic regression classifier on each of the feature pairs. We performed cross validation on the training data by season and obtained the resulting accuracies in Figure 3 below:

Feature	Acc.
Away team avg points + 4th quarter points in last 2 games	52.92%
Away team avg yards per rush attempt in last 15 games	52.86
Away team wins in current season	52.86
Away team wins WTS in current season	52.80
Away team won WTS more than home team in season	52.79
Home team avg pass completion percentage in last 15 games	52.79
Home team avg points given up in last game	52.73
Away team avg pass completion percentage in last 4 games	52.72
Away team wins in last 15 games	52.72
Home team avg pass completion percentage in last 5 games	52.72
Home team avg pass completion percentage in last 3 games	52.71
Away team avg pass completion percentage in last 15 games	52.65
Away team avg pass completion percentage in last 3 games	52.65
Away team wins in last 4 games	52.64
Home team wins in current season	52.63
Home team avg points scored in last 15 games	52.60

Figure 3: A partial listing of features in decreasing order of accuracy when paired only with the point spread feature and used to train a logistic regression classifier. Accuracies are averaged over all 8 seasons in cross-validation on training data.

We note that certain features have high rankings in both lists, while others are ranked highly in only one of the lists. This observation suggests that neither metric is alone sufficient for feature selection. We also tried using all features which exceeded a certain threshold of one of the categories above and obtained accuracies that were competitive with the accuracies shown in Figure 3, but no simple feature combination yielded significantly-higher results.

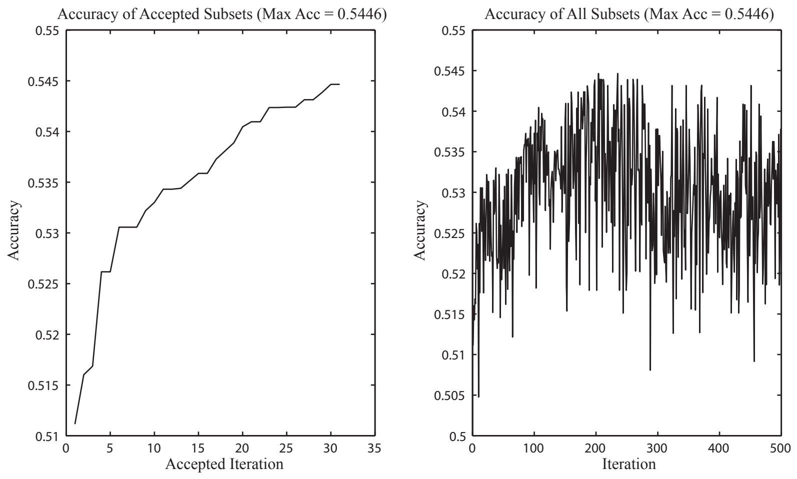

Figure 4: An example run of the FEATURE-SEARCH algorithm. (a) Plot of accuracies of the accepted feature sets (the S_i). An iteration is accepted if its feature set produces the highest accuracy seen so far. (b) Plot of accuracies of the proposed feature sets (the F_i) across all iterations. On each iteration, a new subset is generated from the last accepted subset by random perturbations.

To choose features, we need a way to automatically search through the set of possible feature sets and measure their performance through cross-validation on the training data. In the next section we will present an algorithm to do this.

4 Randomized Feature Search

The number of feature subsets is far too large for exhaustive search, so we use a simple randomized search through the space of feature subsets that we call the FEATURE-SEARCH algorithm. The inputs to the algorithm are a set of features F_{full}, a binary classifier, and parameters that determine how long to run the search and how to generate candidate feature sets on each iteration. The algorithm proceeds by generating a random proposal feature set by perturbing a base feature set on each iteration. The proposal is only accepted if it results in a new accuracy level that is higher than any previously seen. The accepted feature set then becomes the new base feature set for future iterations. An example run of the algorithm is shown in Figure 4 and an algorithm description is given below.[2] In the following description, the F_i denote proposed feature sets and the S_i are accepted feature sets:

- Generate an initial subset F_1 of features by walking along the full set F_{full} and randomly adding a feature with probability p. If $|F_1| > m$, generate a new F_1 repeatedly until $|F_1| < m$. F_1 is the first proposed feature set.
- $i \leftarrow 1$
- $S_1 \leftarrow F_1$ (accept the first proposed feature set)
- Repeat for $k = 1 \dots K$ iterations:
  - Train a classifier on feature set F_k and evaluate using cross-validation on the training set

[2] We used the following parameter values for our runs: $p = 0.06, m = 14, K = 500$.

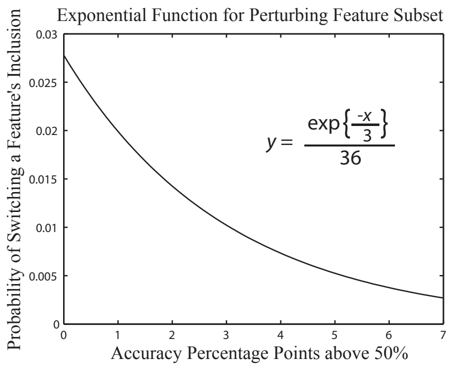

Figure 5: The exponential function is used to choose the probability of switching a feature's inclusion based on the accuracy of the current subset of features.

```
    – If accuracy is below 50%,
        ∗ Generate a new feature subset $F_{k+1}$ as in the initialization step
    – Else,
        ∗ If accuracy is the best seen so far,
            · $S_{i+1} \leftarrow F_k$ (accept the proposed feature set)
            · $i \leftarrow i + 1$
        ∗ $F_{k+1} \leftarrow perturb(S_i)$ (perturb the last accepted feature set to obtain a
          new feature set)
    – $k \leftarrow k + 1$
• Return $S_i$, the feature set corresponding to the last accepted iteration and the high-
  est accuracy.
```

Some details regarding the generation of new feature sets are left out in the description above. First, we impose a size limit of m whenever we generate a new feature set; this reduces the amount of computation required to test each feature set and speeds up the search.[3] Also, when we obtain a new feature set, either by generating a new one or by perturbing an existing one, we ensure that the new feature set is not identical to one already generated.

In perturbing a base feature set, the probability of changing the inclusion of a feature should be inversely proportional to the quality of the base feature set. That is, if a particular feature set results in high accuracy, we suspect that a better feature set may be nearby, so we only want to make small changes to it to attempt to explore its neighborhood. For our algorithm, we obtain the probability of changing a feature's inclusion from the exponential function as shown in Figure 5. Given this probability q, the $perturb(S_i)$ function referenced in the algorithm description above is performed by walking along the feature set and changing the inclusion of a feature with the given probability q. The algorithm will never get trapped, since there is nonzero probability of going to any feature set from any other feature set; similar sets simply have higher probability of being proposed than dissimilar sets.

The algorithm is helpful for finding high-scoring feature sets, but is too slow for running on the full set of 230 features for each game. We executed the algorithm several times using various feature sets as initial full sets, including the list of features above a certain squared-Pearson threshold and another set of features above an accuracy threshold. Then,

[3]We found that including many features often led to decreased accuracy and greatly increased time requirements for computation.

we took the union of the resulting best-performing feature sets from each run and used this union as the initial full set for another run of the algorithm. The best accuracy was reached by this last execution of the algorithm; the resulting feature set is shown in Figure 6.

Feature Set 1 (56.11% accuracy)
Point spread line
Away team wins with the spread in last 4 games
Away team won more than home team in curr season
Home team lost WTS more than away team in curr season
Away team avg yards per rush attempt in last 15 games
Home team avg yards per pass completion in last 5 games
Away team avg interceptions thrown in last 2 games
Away team avg sacks in last 2 games
Home team avg sacks in last 5 games

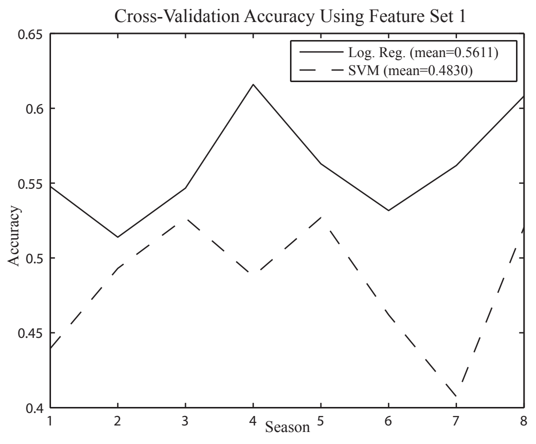

Figure 6: Best-performing feature set obtained from FEATURE-SEARCH algorithm and its accuracy on each fold of the training set. Logistic regression was used within FEATURE-SEARCH algorithm, so the logistic regression accuracy was optimized during the search. This is one reason for the performance discrepancy between classifiers.

The best-performing subset of features found is shown in Figure 6 along with its accuracy in each fold of the training data both for a logistic regression classifier and a support vector machine classifier.[4] The logistic regression classifier was chosen for use within the FEATURE-SEARCH algorithm due to its speed, but the SVM classifier could have been used in its place or in tandem. Though the accuracy varies by season, we note that the logistic regression classifier never performs below 50% accuracy. This gives confidence that we have found a feature set that may be effective on future seasons as well, and we test this in the next section.

5 Results and Conclusions

Classifier	2000 Season (175 games)	2001 Season (175 games)	Mean
Always Away	48.82%	48.49%	48.65%
Always Home	51.18	51.51	51.35
Random	49.41	52.73	51.07
Logistic Regression	51.18	**56.97**	**54.07**
SVM	**52.04**	49.47	50.76

Figure 7: Test set accuracies for baselines and classifiers trained on Feature Set 1.

To evaluate our feature set on new data, we used the logistic regression classifier and SVM classifier trained on Feature Set 1 to classify the held-out test data, which consisted of the 2000 and 2001 NFL seasons. The results are shown in Figure 7. The difficult part about market prediction is that the past can be misleading about the future, so we kept

[4]The SVM implementation uses the SimpleSVM Matlab toolbox available from http://asi.insa-rouen.fr/ gloosli/simpleSVM.html.[7]

two seasons for testing instead of one in order to lend more credibility to our results. The logistic regression classifier performed similarly on the test data as it did in cross-validation on the training data, suggesting that the feature set we obtained as a result of the FEATURE-SEARCH algorithm is an effective feature set for this problem. With additional feature extraction and runs of the FEATURE-SEARCH algorithm, we expect that we can obtain more accurate results. The results for the SVM classifier could be improved if we were to use the SVM within the FEATURE-SEARCH algorithm, and it would be interesting to see about jointly considering both classifier accuracies within the algorithm. Our results on the test set, along with the cross-validation results shown above, suggest that there is room to beat the NFL point spread. We have presented a set of effective features and a framework for searching through feature sets to attempt to find better feature sets. We believe that more feature extraction and computation will lead to better feature sets for this problem, and also that the techniques described here may be applicable to many other feature selection problems.

It may be an unreasonable goal to try to succeed in all games in the test data for a season. In reality, sports bettors typically bet only on games they are most sure about. Anecdotally, we found promising results by using a confidence measure from the logistic regression classifier and only betting on games that met a certain confidence threshold, but a comprehensive exploration of this area is left for future work. This same framework can be used for related tasks such as outright victory prediction and the prediction of which to choose given the "over/under" line of total points. Both of these are also binary classification tasks, and would likely require different feature sets. We have explored both of these problems using strategies similar to that used for the point spread problem described above. In doing so, we obtained an accuracy of nearly 70% for the overall winner problem and an accuracy for the over/under problem near that of the point spread results. An in-depth feature exploration has not yet been performed; it would be interesting to see how the feature sets differ among these different tasks.

References

[1] Bryan L. Boulier, H. O. Stekler, and Sarah Amundson. Testing the efficiency of the national football league betting market. *Applied Economics*, 38(3):279–284, February 2006.

[2] Mark E. Glickman and Hal S. Stern. A state-space model for National Football League scores. *Journal of the American Statistical Association*, 93(441):25–35, 1998.

[3] Joseph Golec and Maurry Tamarkin. The degree of inefficiency in the football betting market : Statistical tests. *Journal of Financial Economics*, 30(2):311–323, December 1991.

[4] D. A. Harville. Predictions for national football league games via linear-model methodology. *Journal of the American Statistical Association*, 75(371):516–524, 1980.

[5] Leonhard Knorr-Held. Dynamic rating of sports teams. *The Statistician*, 49(2):261–276, 2000.

[6] Steven D. Levitt. How do markets function? an empirical analysis of gambling on the national football league. *Economic Journal*, 114(495):2043–2066, 2004.

[7] Gaelle Loosli. SimpleSVM, available from http://asi.insa-rouen.fr/ gloosli/simpleSVM.html.

[8] Hal Stern. On the probability of winning a football game. *The American Statistician*, 45(3):179–183, 1991.

[9] Richard A. Zuber, John M. Gandar, and Benny D. Bowers. Beating the spread: Testing the efficiency of the gambling market for national football league games. *Journal of Political Economy*, 93(4):800–806, 1985.