Predicting the outcome of NFL games using machine learning

Babak Hamadani
bhamadan-AT-stanford.edu
cs229 - Stanford University

1. Introduction:

Professional football is a multi-billion industry. NFL is by far the most successful sports league in America. The games are very exciting because outcomes of the games are very unpredictable. The injuries to players, psychological factors and leagues rules to create parity by allowing weaker teams to draft first make it even harder to guess the winner.

The goal of this project is to predict the winner of an NFL game. The outcome of no game can be predicted deterministically. Therefore the goal is to come up with a system that is comparable or better than human prediction.

2. Data:

Data is based on NFL seasons 2003, 2004, and 2005 scores and statistics. I crawled the web for the data. Different statistics are available on different web pages. I wrote some code in java and perl to extract required links and parse the html pages. I also wrote a fairly big java program to combine and process the different statistics for each team and output them in the required format.

3. Training and testing methodology:

I used a scheme similar to leave-one-out-classification with an additional constraint: You can not use the data from the games that happen in the same week or future weeks. We can technically use the statistics from games that happen in the earlier hours of the same day, but it would make the implementation much harder.

I did not classify the first two weeks of each season, although those weeks are used as training data. I didn't classify the first two weeks because some of my features depend on averages in past 3 weeks.

4. Algorithms:

At the beginning I was not sure if machine learning is applicable to this subject. Therefore I started with the simplest models to gain an intuition about the problem. I chose to try logistic regression and svm with different kernels.

4.1 Logistic Regression

I used my own implementation of logistic regression in matlab, which used Newton-Raphson method. It was by far the fastest algorithm in all my experiments.

4.2 SVM with Different Kernels

I used svm-light with linear, polynomial and tangent kernels. I also tried the sigmoid kernel for one or two experiments; however the results were not very good. I tried different parameters for each type of kernel with two different basic feature sets and settled on the following parameters for the rest of the experiments:

kernel	**formula**	**params**
linear		C:1
poly	(s a*b+c)^d	s:default, c:1, d:2, C:1

Tan	tanh(s a*b + c)	s:0.1, c:1, C:1

Table 1.(C: trade-off between training error and margin)

4.3 Feature Sets

I started with two groups of features and experimented with different feature sets from each group. For each feature I calculated that feature's average. I calculated the average in two different ways (see 5.2 for an explanation of averaging methods)

4.3.1 Win-ratio Features

The first group was what I call Win-ratio features. Here are all the features in the Win-ratio group. For different experiments I omitted some of the features.
{weekNo, VT_TeamNo VT_BothWin-ratio VT_HomeWinRatio VT_AwayWinRatio VT_3weekWinRatio VT_AvgPointsScored VT_AvgPointsAllowed HT_TeamNo HT_BothWinRatio HT_HomeWinRatio HT_AwayWinRatio HT_3weekWinRatio HT_AvgPtsScored HT_AvsPtsAllowed }

4.3.2 Game Statistics Features

These were mainly composed of statistics for each game averaged over different games. There are too many game statistic features to mention all of them here (78 for each team plus weakNo). Here is a sample of these features.
{weekNo VT_TeamNo , final_score, AVG_YARDS_BY PASSING, AVG_YARDS_BY PENALTY[1] AVG_THIRD DOWN EFFICIENCY, AVG_FOURTH DOWN EFFICIENCY EFFICIENCY[2],TOTAL NET YARDS, TOTAL OFFENSIVE PLAYS AVERAGE GAIN PER OFFENSIVE PLAY, NET YARDS RUSHING, TOTAL RUSHING PLAYS, AVERAGE GAIN PER RUSH, TACKLES FOR A LOSS-NUMBER,...}

4.4 Backward/Forward Selection

I implemented backward/forward selection mechanisms in matlab. Since these two mechanisms are wrappers and non-linear svm kernels take a long time to train I only tried them with logistic regression (also logistic regression almost always performed the best).
1- In each season try feature selection for some games and apply it to the whole season.
2- Use feature selection on a specific season (2003 for example) and use the best features on different seasons.

The logic behind #2 is that if a feature is important in one season it should be important in another season because the game doesn't change much. This is a not a fool-proof assumption. I used the second approach with the game statistics sets. However in practice it didn't give very good results. For each season I got a different set of features and when I tried classification with feature sets chosen based on feature selection of another season, the result was not consistently better.

I also tried the selection algorithms with the win-ratio features. Backward selection eliminated Week-No as the first feature for each season and forward-selection never chose it as a good feature. After removing the Week No, the results immediately improved for logistic regression (and almost all linear-kernel tests) on every experiment that I had done. It even improved the results for feature sets that were derived from game-statistics.

5. New Techniques

5.1 Momentum Features:

There are many variables that are hard to include in our models. One example would be a team's recent adjustment in strategy or personnel. To account for such changes I introduced the idea of momentum. The idea is that a team is more likely to have a closer performance to its recent performance (similar to locally weighting). I chose to add two features for each game VT_3weekWinRatio (visitor's past 3 week win ratio), HT_3weekWinRatio (host's past 3 week win ratio). Initially the result seemed positive almost in every model. However after I eliminated the week Number it turned out that the best feature set didn't have any 3-week averages. I believe Week Number was a bad feature and initially when I added 3-week averages only lessened week-number's negative effect. However I don't rule out that it can be useful with different feature sets.

5.2 Using averages up to the test week:

I started with an initial set of win-ratio features. For each sample the averages were computed up to the week of the game. For example to classify games in week 16, I would need to create training samples for all the games that happened between week 1 and week 15 including a game between team A and B in week 2. To compute the average winning ratios of that game I would calculate the average winning ratio of team A and B up to week 2 and the training sample would look like:

<result:1, A, visitor, win-ratio(up to game's week 2): 0, B, host, win-ratio(up to w 2): .5>

(where 1 means visitor won). This increased the number of training samples where teams with lower winning ratios actually won.

Instead of using the averages up to week 2 for a game that happened in the 2nd week, I used averages up to week 15. It is completely fair because we want to classify week 16's games and by then we have all the outcomes of week 1-15. Using this approach the training sample for the previous game may look like:

<result:1, A, visitor, win-ratio (up to test week 16): .75, B, host, win-ratio (up to w 16): 0.2>

This technique increased the accuracy for all models. Intuitively we use teams' true strength (win-loss ratio). Assume team A had a tough schedule at the beginning of the season and team B had an easy one, therefore B had a better win-loss ratio. However in reality team A is a much stronger team and that shows up in its results as the season progresses.

5.3 Using week 17 training data of a different year:

The idea is to train a model based on all 17 weeks of a prior season for classification. I tried this approach with svm (linear kernel) and it gave very good results. That experiment included the team numbers as features. It would be interesting to know the results after removing the team numbers. The intuition behind removing team numbers is that teams may be very different from one season to another (given player and personal change) although the counter argument is I haven't had time to completely explore this option.

6. Results and Analysis:

6.1 Results:

set	Feature types	avg-method	weekNo a feature?
1	win-ratio	t	Yes
2	win-ratio	t	Yes
3	game-stat	t	Yes

4	game-stat	t	No
5	win-ratio	t	No
6	win-ratio	t	No
7	win-ratio	g	No
8	win-ratio	g	No
9	game-stat	g	No

Table 2 (t: average up to test week, g: average up to game week)

set	sl_05	sp_05	lr_05	st_05	sl_04	sp_04	lr_04	st_04	sl_03	sp_03	lr_03	st_03
1	65.03	59.31	63.63		60.03	55.2	60		65.02	56.7	66.25	
2	64.55	59.42	63.63		59.61	53.17	60.06		63.65	60.27	64.94	57.16
3	60.38	60.5	62.32	61.33	57.83	53.71	59.29	54.49	61.98	66.81	61.01	61.87
4	60.8	61.93		62.76	59.07	54.91		54.9	54.9	64.49		65.44
5			65				61.73				66.61	
6	63.66	54.8	65.83	57.34	60.09	52.71	61.37	57	66.27	58.54	67.08	58.36
7	63.53	52.59	62.14	47.7	56.15	47.7	57.74	51.33	60.81	55.99	62.14	53.47
8	61.93	53.84		58.41	54.6	50.14		47.94	57.59	49.49		50.03
9	54.73				53.9	52.23						

Table 3 (sl: svm linear kernel, sp: svm poly kernel, lr: log regress, st: svm tan kernel)

6.2 Analysis:

6.2.1 Best Results:

Best results were obtained using set 6 (win-ratio features without week number) by logistic regression. (2005: 65.83% - 2004: 61.37% - 2003: 67.08%) This is another reason why only performed feature selection with logistic regression.

6.2.1 Best Algorithm:

Over all logistic regression is the best method, although svm with linear kernel has a similar performance. An interesting observation is if any of the methods performs better on set of features A than it does on B, it is almost guaranteed that other methods will perform better on A as well. So one can pick the best set of features using the fastest method and try it with other algorithms.

6.2.3 Comparison of averaging methods:

In the left column table 4 you can see feature sets calculated using averages up to the game's week, and on the right column you can see feature sets whose averages were calculated up to the test's week. If you refer back to table 3 you will see that in every case the set whose average was calculated up to the test week had better results.

avg_to_game_week	avg_to_test_week	approximate improvements
7	6	3-5 %
8	1	1-8%
9	4	2-6%

Table 4

6.2.4 Impact of removing week Number from the features:

Again, on the left are the sets with week-number, and on the right are the exact same sets without the week-number. In this case there is always an improvement on the best case (logistic regression).

With weekNo	Without
3	4
5	1
6	2

Table 5

6.3 Expert Picks:

Unfortunately I don't have the results of expert-picks for 2003-2005 seasons. Here is their accuracy for their picks for 2006 season through week 14 (including 14) [1]. Bear in mind that in their picks they have the luxury of not picking a winner for some games hence for games that are too close they may not pick.

expert	Right [1-14]	Wrong [1-14]	Right [1-2]	Wrong [1-2]	Accuracy [2-14]
Theisman	117	77	19	11	0.5976
Salisbury	123	85	20	12	0.5852
Hoge	121	87	23	9	0.5568
Jaworski	124	84	20	12	0.5909
Schlereth	126	82	19	13	0.608
Allen	127	81	19	13	0.6136
Mortensen	114	94	21	11	0.5284
Golic	128	80	20	12	0.6136
Accusoure	133	75	24	8	0.6193

Table 6

7. Conclusion:

When I started this project I was not sure if machine learning can be applied to this problem. Given my current results (2005: 65.83% - 2004: 61.37% - 2003: 67.08%) I believe machine learning can be a reasonable solution specially compared to humans. However any solution will be able to reach very high accuracy.

Given the time constraints I have not explored all the possible algorithms and features. BCS like formulas can be used to account for unbalanced schedules. It would also be nice to compute a confidence score to improve the results by not predicting close games. I think a couple of new things that I tried such as using averages up to the test game can be useful for similar experiments.

[1] http://sports.espn.go.com/nfl/features/talent