Predicting outcomes of NFL games

Albert Shau

Dec 16, 2011

The purpose of my project is to learn how to predict the outcome of NFL (National Football League) games. Being able to do so accurately would be of interest to many NFL followers and could have implications in gambling if I am able to a good amount better than the average NFL fan. It could also be used in conjunction with a study on penalties called per game as a way to validate or invalidate claims of referees "fixing" games. If it is possible to show a pattern of favored teams losing due to an abnormal amount of penalties, that might be data in support of referees deliberately guiding the outcome of a game. To build the classifier, I took game data from 1970 to now, generated features from that data, and then ran Naive Bayes and SVMs over the data to predict the outcome of games. The evaluation metric used was, which is the number of games predicted correctly out of the total number of games. Success was judged by comparing classifier accuracy to human accuracy. Some measure of human accuracy is available on popular sports sites. For example, according to http://sports.yahoo.com/nfl/picks, so far in 2011, Yahoo! Sports users predicted correctly 68.42% of the time, whereas the accuracy of writers range from 63.64% to 71.29%. My goal was to beat them all by a good margin and hit 75% accuracy.

Game data from 1970 to 2011 was taken from http://www.pro-football-reference.com by scraping various pages available on the site. For each week, the boxscores of the game contained information such as the winner, loser, hometeam, score, passing statistics, rushing statistics, turnovers, penalties, and sacks. The site also contains information about pro bowlers (outstanding players) throught the years. From this data, for each team and year, I generated several statistics about each team's performace so far in the season. For example, a team's average points scored per game and average

points allowed per game. Using those statistics and using the outcomes of the games, I ended up with a training set of 33 features. These features were just statistics that I would use when considering what team would win in a matchups. They included things like the difference in average points scored, difference in win % between the teams, statistics comparing passing offense of team A versus passing defense of team B and vice versa, rushing offense of team A versus rushing defense of team B, tendency of each team to commit penalties, difference in turnover ratios, whether a team is playing at home, comparing first downs gained and given up, etc. For brevity I have not listed all the features, but they all followed the same vein, comparing the two teams in areas that most people think are impactful in winning or losing football games.

The first approach was to use a Naive Bayes classifier, where all the features were modeled as multinomial distributions, and were discretized to take on values from 1 to 10. 10 was chosen arbitrarily and no cross validation was done. Since each game must have a winner and a loser, the priors for winning and losing are the same and can be ignored. The conditional probabilities can be calculated simply and efficiently as $p(X_i = k|Y = 1) = \frac{1\{X_i = k\&\&Y = 1\}}{1\{Y = 1\}}$ and the analagous calulation for $p(X_i = k|Y = 0)$. Laplace smoothing was also used, so 1 was added to the numerator and the cardinality of each feature (10 for most of them in this case) added to the denominator. At the time of the first approach, I only had 8 features. Using this approach 62% accuracy was achieved. This accuracy is about as good as the worst Yahoo Sports writer. Not good, but definitely not terrible for such a simple approach. The most recent 30% of the data was used for testing and the rest for training.

The next approach was to use an SVM to classify. The same 8 features were used, except none of them were discretized for this, and the labels were changed to +1 and -1. The liblinear package was used to train and test the data. The table below shows accuracy as a function of the cost parameter. L2-regularized L2-loss support vector classification was used with a linear kernel.

| c | 10 | 1 | 0.1 | 0.01 | | 0.0001 | |
|----------|--------|--------|--------|--------|--------|--------|--------|
| accuracy | 47.25% | 57.81% | 59.86% | 63.31% | 63.41% | 63.51% | 62.44% |

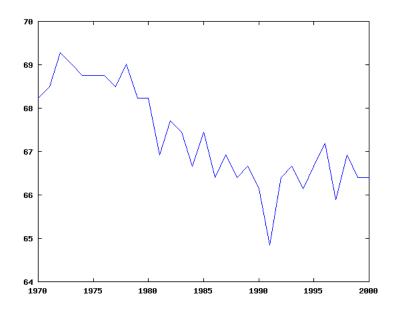
So this is slightly better than the Naive Bayes approach with the right cost

value, but still not better than humans. At this point, I decided to scrape more data and generate more than 8 features. The reasoning was that I had orders of magnitudes more training examples than features. Therefore it was unlikely getting data from before 1970 would help much, and that is not even considering the fact that the game was fairly different back then than now. Also, both algorithms performed similarly poorly, so the thought was that there was something fundamentally wrong about the modeling process; what was wrong was that I was not capturing all the different factors that are involved in who wins a football game. I also briefly considered doing feature selection, but that did not seem like it would be useful on a set of 8 features, all of which were standard metrics people look at when comparing teams. Also considered using different kernels for the SVM, and also normalizing the features to all be on the same scale (as suggested by a paper written by the libsym folks), but that seemed like it could only squeeze out tiny amounts of lift, whereas I needed a huge improvement. So after this point I went and got more data in order to generate the final set of 33 features, some of which were mentioned earlier in this report. Since the SVM performed better than Naive Bayes, I only used SVMs after this point.

At this point, I also decided that in order for the comparison to be fair, I should be using only the 2011 season as the test set because that is what I am comparing my classifier to. I would have used user and writer accuracies from previous years but could not find that data. The test set is much smaller, but it is still 209 examples so it is not too small. However, overfitting would still be a concern, so I would train the model using data from 1970 to 2010 with k-fold cross validation, but will only be reporting accuracy on the 2011 test set for comparison purposes. These numbers, however, are higher than using multiple years as the test data. However, it is possible 2011 is just an easy year to predict and that writers and average fans are having an easier time predicting as well.

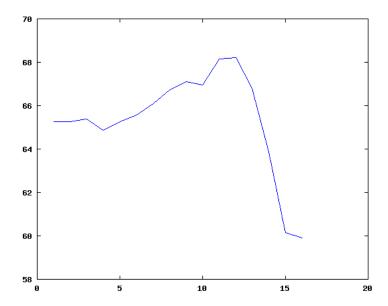
With the new features, SVM performance increased to 68.4%. This matches Yahoo Users, so it is a decent classifier. I also tried normalizing the data so it was all on the same scale, and using a radial basis kernal $\exp(-\gamma|u-v|^2)$ as suggested by the libsym folks. I tried a grid of values for c and gamma, exponentially scaling c and gamma up and down independently, but none of those outperformed the linear kernel. At this point, I did a little study on training data size to see if adding more data would help. The results are

shown in the figure below.



The x axis depicts the earliest year used in the training data (last year used was 2010 in all cases). The y axis depicts accuracy. From this graph, you can see that accuracy levels off before 1970, so adding more data would probably not be useful. There are not many more years to add anyway. One interesting thing to note is that the performance is best when using my full dataset. It is a pretty popular sentiment that the game is very different today than it was decades ago. The basic thought is that it is much easier to score today and passing and offense is much more important than it used to be, when running and defense would dominate. Based on that reasoning, one might expect performance to improve if the earlier years are left out. However, as evident in the graph, that is not the case. One could therefore argue that what determined the winner of a game 40 years ago still determines the winner of a game today. You could argue that the game has not really changed all that much.

One final thing I looked at was dividing the season up into chunks. The idea behind this is that early in the season, there is not much data on each team, so the statistics early in the season are less meaningful. So once we're at week 12, maybe it's better to ignore what happened in weeks 1-6. The figure below shows the results.



The x axis shows the earliest week of data used. Weeks before that were filtered out from training and test sets, and test sets consisted of 5 years of games minus the filtered out games. As you can see from the graph, there is a sizable jump in accuracy when earlier weeks are left out. Using this observation, we can create multiple models and use them in an intelligent way depending on the week we are trying to predict. This would boost overall accuracy.

I was not able to reach the original goal of 75% accuracy, but was able to create a classifier on par with regular sports fans. However, this is not useful at all, because a much simpler classifier than mine would simply go to the webpage and make a decision based on who the users and writers think will win. I would spend future work on getting even more features and seeing if performing feature selection would help. There are definitely important features that I have not included, such as injury information, that is vital in predicting the outcome of games. Other ideas for features include using the aggregate win percentage of all the teams a team has beaten in the past to account for some teams having an easier schedule than others. There are many many more features that could be used. Something I learned was that building a good, clean, dataset can be the bulk of the work in these sorts of machine learning problems.