CS 221 Final Project Progress Report

cmayer, hughec

November 14, 2014

1 Problem Description

Fantasy Football (FF) is one of the most popular hobbies in the country. FF participants (called owners) make weekly decisions about which football players to put in their starting lineups with the goal of maximizing their team's overall performance, measured in FF points. Even though massive amounts of detailed data is available for every player, owners usually make decisions based on a few indicators and gut instinct. For our project we plan on building a Fantasy Football lineup predictor that, given a set of features about the past performance of a football player, will try to predict his performance in a given game.

We initially aimed to build a regression model to predict the FF output of any given player in any given week. However, due to the distribution of FF points across NFL players, we have decided to limit the scope of our model to a subset of players. We will limit our model to players occupying "skill positions" – offensive positions that regularly handle the ball (QB, RB, WR, and TE). This excludes two positions that typically figure into FF lineups: kicker and defense/special teams. Our reasons for choosing to exclude these positions include domain-specific factors (e.g., FF owners often "stream" kickers and defenses by selecting a different one each week based on the kicker/defense's opponent) as well as factors specific to the data (e.g., FF scoring for kickers depends on the distances of made field goals – a statistic not present in our dataset).

Among players in "skill positions", we will also limit our model to only those players whose average FF production exceeds a certain threshold (currently set at 5 points/game). For a great deal of NFL players (backups), the average FF score is near 0. In order to build the best model possible for players who regularly see game action in the NFL it is important that we exclude data from players who do not play or do not perform. It is very unlikely that any of these players would be part of a FF team and thus there is little benefit to modeling their performance. As we attempt to develop our regression model, we may see fit to limit the pool of players even further – to the top 100-200 FF players, for example.

2 Prior Work

Most of the prior work on this problem has been informal – FF participants privately employing machine learning methods to better the performance of their own teams. Of those such attempts with available documentation, most have used unsupervised methods and ensemble methods on expert projections.

One notable exception uses (bayesff.com) uses Bayesian inference to make predictions on player performance in each game. The historical performance of each player is used to fit a prior Gamma distribution for that player. The prior distribution is then updated by the player's performance each week. We have yet to scrape the data from this source to evaluate their predictions, but it is an interesting and influential approach. Indeed, a later approach to FF projection expands on the model at bayesff.com by first classifying each player as belonging to one of three different distributions ("weak", "average", or "elite") and predicting performance based on the parameters of that distribution. We hope to improve on models such as these by incorporating features that capture information about the player's recent performance, opponents, and teammates.

3 Data

We use the nflgame python library to download game statistics from NFL.com. The dataset gathered through these means consists of all game statistics available at NFL.com for every game during the 2010, 2011, 2012, and 2013 seasons. Data for each week of each of these NFL seasons is stored as a CSV file where each row is a single NFL player's statistics for that week. We utilize data from each week (1-17) of these seasons (2010-2013) as our training set, and hold out data from the current 2014 season for testing purposes. As discussed in the problem description, we may limit our training and evaluation to the top 100-200 FF players, since players with limited playing time are unlikely to be on a fantasy roster and have limited signal to learn from. Our input data would be a feature vector based of various player statistics:

Player	Yards Last Game	Opponent	Quarterback Rating	Age
Randy Moss	89	NYJ	89.2	31

FF uses a scoring function to combine different statistical categories into a weighted total score. Prior to training we chose a commonly used scoring function to produce our expected output. For instance, the FF score for a QB is calculated as

$$FFScore = 0.04 * PassingYards + 4 * PassingTDs - 1 * Ints$$

Our regression model then directly predicts the FF score for a player in a given week.

4 Baseline and Oracle

Historical averages provide a simple and intuitive baseline model for predicting a player's performance in a given week. Given a player's statistics over a number of games played, the historical average baseline model simply outputs the average of the player's performance as a prediction for future games. The closest thing to an oracle is the service provided by major FF leagues, like Yahoo, ESPN and CBS, which similarly projects player performance on a game by game basis. The system used to develop these projections is a black box, but appears to be a combination of machine learned features and expert opinions. Our goal will be to try and produce similar performance using only machine learning.

The historical average we implement for our baseline model is a decaying weighted average over all of a player's previous games in the dataset. The most recent game is given weight 1 with weights decreasing at a constant rate of $\frac{1}{numberofgames}$ for earlier games. To provide a concrete example, consider the use of the baseline model to predict the performance of Pittsburgh rookie WR Martavis Bryant in this week's game against Tennessee:

$$\hat{y} = \tfrac{1}{4}\left(20.30 + 0.75 * 16.70 + 0.5 * 20.30 + 0.25 * 10.00\right)$$

Thus the baseline model predicts that Bryant will achieve a FF score of 11.37 (compared to the Oracle projection of 8.55).

5 Approach

We will explore the use of different regression models, starting with linear regression and possibly moving to neural networks if we need more expressive power. The majority of the work and performance gains is expected to come from feature engineering: finding which pieces of data, and combinations of pieces of data, provide the best signal for predicting player performance.

We have implemented a baseline model for linear regression that uses a player's recent performance (FF scores in previous three games) as features. As a concrete example, if we were to attempt to predict Peyton Manning's FF score for this weekend's game against St. Louis, the feature vector would take the form $x^{(P.Manning)} = [31.60, 23.82, 23.44]$. Prediction would then be made according to the product of the feature vector with the learned weight vector θ:

$$\hat{y} = \theta^\top x^{(P.Manning)}$$

This feature representation provides a very low baseline and does not include most of the information that we hope to capture in our model. Our next steps with this approach center around generating features to capture the strength of the opponent, strength of the player's teammates, injury history of the player, usage trends and offensive playcall share of the player, etc. We plan on carrying out a great deal of experimentation with regard to the features to include in the model.

6 Evaluation

With such a large corpus of available data, evaluation is straightforward. We train on a collection of historical data and previous games from the current year, then predict the performance for a player each week and use the actual statistics after the game to evaluate our predictor using mean squared error as our evaluation metric.
We include some preliminary/baseline results below:

Model	Mean Squared Error
Decaying Weighted Average	43.91
Linear Regression with Baseline Features	47.78

The mean squared error reported for the decaying weighted average above is the average error over the entire dataset, since the model requires no training. For Linear Regression, the error reported is the average leave one out cross validation error over the training set. As is clear from the table, the decaying weighted average outperforms the linear regression model that uses baseline features. This result is not unexpected, because each prediction in the decaying weighted average model incorporates a great deal more data (as many as 83 games) than do the features in the linear regression model (only 3 games).