Daily Fantasy Basketball Picker

Automated Decision Systems
Dec 20 2015
William Cai, David Hatch, Evan Green

1 Introduction

Daily Fantasy Sports, or DFS, is a new type of sports betting which has recently come into vogue. Because there is some element of skill, prosecutors have been unable to shut it down in the majority of the country. In this paper, we will detail an automated decision system that we have constructed, using various paradigms of decision making, to play Daily Fantasy sports. Section 2 gives a detailed explanation of the fantasy sports problem along with a mathematical problem statement. Section 3 breaks down our various data sources and methods for scraping them. It also explains the database we use to hold the data. Section 4 contains our methodology for cleaning the data into the final features we base the model on. Section 5 explains how are linear regression algorithm of choice works, and gives a human-friendly explanation of the model. In section 6 we talk about how we generate our lineups using our projections. Finally, in section 7 we put our money where our mouths are and bet on our lineups.

2 Problem Statement

blah blah double up, high expected value low variance blah blah

3 Getting the Data

yo

4 Cleaning the Data

hi

5 Leaning the models

The code related to this section is in the regression folder in the regress.py and predict.py files.

$$\beta_{elasticnet} = \arg\min_{\beta}(||y - X\beta||^2 + \lambda_2||\beta||^2 + \lambda_1||\beta||_1)$$

$$\beta_{LASSO} = \arg\min_{\beta}(\frac{1}{N}||y - X\beta||_2^2 + \lambda||\beta||_1)$$

$$\beta_{ridge} = \arg\min_{\beta}(||y - X\beta||^2 + \lambda||\beta||^2)$$

Figure 1: The cost functions of various regression methods in the regularized linear regression family. Note how elastic net contains the terms from both the LASSO and ridge regression.

Our linear regression algorithm of choice was the elastic net, a form of regularized linear regression. It was important to choose from the family of regularized linear regression because we have a pretty small ratio of data points to features, and we wanted the ability to add as many features as we wanted. Doing linear regression on a dataset with comparable amounts of features to data points quickly becomes an exercise in creating ridiculous models because of the danger of overfitting. Regularization combats this trend by adding in a term based on the coefficients of the model to the loss function of linear regression, which is simply the residual sum of squares.

There are several regression methods in the regularized linear regression family. The most popular of these are the LASSO, ridge regression, and elastic net. Our choice, elastic net, is an interpolation between the other two and has several properties which lend themselves to our project. For one, given a group of correlated variables, of which we have many, the elastic net will give all of them a coefficient. This is in contrast to the LASSO, which will arbitrarily select one and give it a coefficient and set the rest to 0. Using the elastic net allows us to better understand our program's explanation of it's decisions. The advantage that the elastic net has over ridge regression is that it will zero out some variables, which will allow us to say which of our features give no signal - another useful way for us to understand which features are important in predicting fantasy scores. On the other hand, ridge regression will almost never set a coefficient to zero, making interpretation more difficult.

In our code, we read in each player-game for each position from a csv file which was produced from our data cleaning. Each line of the csv is of form
$x_1, x_2, ..., x_n, y$,player-name We then load that into numpy matrices X and Y, where X is $num_{player-games}$by n and Y is $num_{player-games}$ by 1, and use numpy's elastic net, ElasticNetCV, to find a linear model β which solves

$$\beta = \arg\min_{\beta}(||y - X\beta||^2 + \lambda_2||\beta||^2 + \lambda_1||\beta||_1)$$

The λs are learned through three-fold cross validation, which splits the data into thirds and learns it on two thirds at a time, using the last third for validation. The elastic nets are made for a variety of λs, and the λs which have the lowest error are used. Furthermore, note that all features are regularized before using the elastic net to allow for the coefficients to be penalized equally. Mathematically, this means that for each feature x_i we subtract the

mean of the x_i such that

$$\sum_{j=1}^{n} x_j(i) = 0$$

Where $x_j(i)$ denotes the ith feature of the jth player-game. We then divide each $x_j(i)$ by the standard deviation of the x_is.
The result of this is a linear model for each position, which predicts how many daily fantasy points. The models, along with an interpretation, are given in files TODO TODO. They are not included here because there are over 100 features so it would take the majority of the paper.
Given these models, we can generate the expected value of fantasy points for each player by plugging their scraped data into the model. We can further generate their variance by going through their projections for previous games and seeing how much they deviate from the model. This allows us to generate our expectedvalues and residPOSITION csv files. Each row of expectedvalues is of form

Name, position, expected daily fantasy points, salary

and each row of residPOSITION is of form

Name, standard deviation

Our lineup generator will read in these files and use them to generate lineups with high expected value and low variance.

6 Generating the lineups

Now that we have the projections, we need to generate some lineups to submit that we hope will make money!
A formalization of this problem is as follows:
Given a list P of players, we would like to generate lineups L s.t. the sum of the expected values of the players of L is maximized under the constraint that the the sum of the salaries of the players of L do not exceed a maximum salary.

7 Taking it out for a spin!

hi