Fantasy NBA Predictions

Mid-Progress Report

Olivier DUTFOY : A20364492

May 7, 2016

Contents

1 Introduction

The goal of my project is to efficiently predict which NBA basketball players will have the most fantasy points on a given game. While this can be done with almost any sports, I focused on fantasy NBA due to the abundance of websites simulating it and the fact that I have some knowledge of the sport. While this kind of game is most of time compared to gambling, there exist a few numbers of *professional players* who have proven it is possible to win in the long term given the correct algorithms. In the future this project could be adapted to others team sports and with some adaption to individual sports.

The project is coded in **Python** and uses machine learning techniques to provide a prediction. In the future I will generate additional features using experts opinion on the topic.

2 Scoring Rules

There are different websites, each having their own rules in order to compute the fantasy score of a given player. I chose the ESPN fantasy league ([1]) scoring system which is as follows :

- Points = 1
- Blocks = 1
- Steals = 1
- Assists = 1
- Rebounds = 1
- Field Goals Made = 1
- Free Throws Made = 1
- Field Goals Attempted = -1
- Free Throws Attempted = -1
- Turnovers = -1

This is however customizable and the values can be changed quite easily in the code.

These basic values provide a fantasy score which ranges theoretically from $-\inf$ to $\inf$. However in practice the score is barely ever under 0 and almost never exceeds 60.

3 Data Gathering

I took my data from the NBA website itself ([2]). This data was very convenient because it provides the statistics for every player in every game for multiple seasons.

I considered different methods to fetch this data. I first considered using a web scraper (such as *Beautiful Soup* or *Scrapy*). However I had never used one before and realised it would be much easier to directly use a **cURL** request to gather the data used to fill the tables on the web page.

A first request is made to get the ID's of every player for a given season. This list is then stored in **pickle** file and used for the request of each individual player.

Two requests are necessary for each player. The first one to get information such as the name, age, experience, etc. The other to get the statistics for each game of the season. At least one second wait was added between each request. In the end it takes roughly 15 minutes to gather the data for an entire season (which contains 400-600 players).

Before storing it in a **pickle** file, some useless data fields are removed. Currently some fields are also removed due to lack of value for some players.

4 Data Structure

I used as features directly the data fetched for each game which are as follows:

- MIN : (minutes played)
- FGM (field goals made)
- FGA (field goals attempted)
- FG_PCT (field goals %)
- FG3M (3 pointers made)
- FG3A (3 pointers attempted)
- FG3_PCT (3 pointers %)
- FTM (free throws made)
- FTA (free throws attempted)
- FT_PCT (free throws percentage)
- OREB (offensive rebound)
- DREB (defensive rebound)
- REB (rebounds)
- AST (assists)
- STL (Steals)
- BLK (blocks)
- TOV (turnovers)
- PF (personal fouls)
- PTS (points)
- +/- (score difference while on the field)

To which I added the numbers of years of experience a player has as well as his age.

The fantasy score of game n of the season for a given player is then matched to the instance corresponding to the average of the previous features over the $n-1$ first games. The win rate is also added as a last feature.

5 Approaches

For the time being only a simple linear regression is used. The instances have then been used in different ways to generate the training matrix X and the score values vector y. Due to sometimes long computation times (around 15 minutes) generate those vectors are store in **pickle** files as well so they'll be usable on other models rapidly.

5.1 Raw

The first approach was to train using the averages of all the games of the season (except the last one). This was not a great idea because the testing would then occur in a different way then the training was done. Unless the prediction was done only on the last game which is then not very interesting.

5.2 Sliding

The first is the classic approach mentioned earlier. The number of points of a given game for a player are matched to the training vector containing the average of all previous games. This means that if a player has played N games in a season, there will be $N-1$ examples computed for the training data.

5.3 Weighted Sliding

The next idea was to differentiate whether the game is played at home or away. For that I weighted the average compute, meaning that if the next game is played at home, then the average used will have a user selected weight for the previous games played at home and vice-versa. The number of examples remains the same as above.

5.4 Doubling the Number of Instances

This time for each player we generate $2(N-1)$ examples by considering once again if the game is played at home or away. Two averages are then computed for each fantasy score value. The classic average, and the home (respectively away) average.

6 Evaluation

In order to evaluate the model I use the average error made as well as the maximum error. For now I assume the seasons are independent and use all but one k-folding (testing on one season, training on the rest, repeat for each season) to generate an error value.

As a baseline I predict a fantasy score for a given game to be equal to the score of the last game.

7 Results

When the training and testing sets are computed in the same way I obtain the following results using a dataset of 9 seasons (from 2005 to 2014):

Method	Mean Error	Maximal Error
Baseline	5.23	36.22
Raw	5.14	25.00
Sliding	4.27	30.70
Weighted_Sliding (w = 2)	4.29	30.85
Double_Instances	4.32	31.26

For every case I tried normalizing the features but the results are extremely similar.

8 Current Issues

I have noticed there are already a few issues with my code so far. The most important one being how I compute my averages. Some of the features are percentage, which means that just averaging them actually provides a wrong value.

The choice of linear regression may not be perfectly adapted for the task. In the future, the use of a more complex model could be useful considering the data.

Some parts could already use some re-factoring to make it more user-friendly and erase some hard-coded values who might be a problem later on.

Currently I have not included the position of a player due to incomplete data. I believe this is a hue mistake and could quite importantly affect the results. I plan on fetching new data using only players for whom this information is available if there is enough of them.

Finally some features in my model are redundant (such as total number of rebounds when I already have the offensive and defensive ones as a feature). While this is not a big issue, getting rid of these values could improve efficiency.

9 Future Tasks

The next thing I'll work on is fix the issues mentioned above. I'll then try to change the model to a Ridge Regression first to see if any improvements are made. I'll then hope to try on more complicated models.

I also would like to implement a model in which the seasons are not considered as independent and neither are the players. Right now the examples are computed using every player, but the ID of a player as a testing instances is irrelevant. I would like to improve my prediction by considering the ID of a player as a feature of some sort and look at the data from his previous seasons.

I will then try to find new features, by using combinations of already existing ones.

Finally the final idea would be to use experts opinions. I have already found some articles but archives for old seasons are not always easily reachable. From these articles I'll extract some additional features.

10 Conclusion

While I expected to have better results at this point in time, I believe I have a better idea on how to approach the next steps of the project. I'll need some time to re fetch the data and improve my current code. However if this is done properly, progress should be much smoother from then on.

11 Links

1. http://games.espn.go.com/fba/resources/help/content?name=scoring-settings-custom
2. http://stats.nba.com/