NBA LEAGUE PREDICTOR
based on
XGBoost

FINAL REPORT

Version 1.0
Prepared by Huang Qingwei
UID: 3035140407
Apr/08/2018
Abstract

The United States National Basketball (NBA) is one of the most famous basketball league in the world. Its betting market owns more than 10 millions of dollars in America. There are abundant historical statistical NBA competition data. Many websites leverage this historical data to help gambler predict the result of a competition. This project shows that this historical data can not reflect the contribution of NBA players directly and try to dig into this data through machine learning method. This project rearranges data season by season and generates training data set for XGBoost classifier. The test accuracy of the XGBoost classifier reaches 67%. This project builds a front end system based on Django framework for the convenience of the users.

Keywords: NBA league prediction, XGBoost, front end
Acknowledgment

Thanks for research guidance and advice from Dr. S. M. Yiu and Dr. T.W. Chim. Thanks for kindly reminding from my group mates. Thanks for kindly notification from Ms. Phoenix To. Thanks for NBA rules introduction from Mr. Zhang. Thanks for feature engineering advice from Mr. Wu.
# Table of Contents

1 Introduction: NBA League Predictor .................................................. 1
2 Background and Literature Review .................................................. 2
3 Methodology .................................................................................. 3
  3.1 Notations .................................................................................. 3
  3.2 Crawl NBA historical data from stat-nba.com ............................... 4
  3.3 Brief of XGBoost principle and method of feature engineering .... 6
  3.4 Brief principle and user manual of Django front end system ........ 9
4 Experiments and Result ................................................................. 12
5 Conclusion and Future works ......................................................... 18

References ...................................................................................... 20
1 Introduction

The United States National Basketball Association (NBA) is a famous basketball league in America. It is founded in 1946. It is one of the most famous sports league in the world. The NBA betting market owns more than 10 millions of dollars. NBA has abundant historical data. Each competition would generate new team data and individual data. It is the motivation for us to search the valuable information hidden behind this competition data. This statistical data has been used in the American betting market estimated in 10 billions of dollars. In 1999, the National Gambling Impact Study Commission shows that about 380 billions of dollars of legal sports wager is generated in America[1]. In 2006, the Nevada State Gaming Control Board reports existing 2.4 billions of dollars of legal wager. Many internet websites use the historical competition data to help gamblers, to accurately predict outcome of a upcoming competition.

However, there are diverse of factors affecting a competition outcome. The strategy of the coaches: whether a strategy is too conservative or extreme; The performance of starting players: hit rate, tackling, attempts and so on; Any players injury so can not contribute to the team? Whether play in the home field would affect the teamwork. The outcome of a competition is not only decided by any one of the factors. The process of a competition may be ups and downs. It implies simply providing the historical box score can not help prediction efficiently. However, it is exactly what most of prediction websites do. We try to combine diverse of factors together in some reasonable way, to get a model with high accuracy.

One possible way to mine this historical data is Machine Learning. Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed[2]. The tradition Machine Learning methods are Logistic Regression, Support Vector Machine, Decision Tree, Random Forest and so on. In this project, our goal is to build a Machine Learning model for NBA league to predict the outcome of upcoming competitions and find the important factors affecting the outcome. We want to abstract the name of players and name of teams through the scalars so that the
differences between Kobe and Jordon may be different hit rate and different number of steal. We write crawler in Python to gain competition data from stat-nba.com. We gain all competitions data from 1986 to now. We analyze this data and generate training data through feature engineering. Then, we study the characteristic of XGBoost and design a set of parameters. We use generated training data to train a XGBoost classifier and get a model with 67% accuracy and the feature importance in this task. Then we build a front end based on Django framework for the convenience of users. The users can query desired competition prediction result in our website.

The left part of the paper is organized as follows. Section 2 will introduce project background and review literature. In section 3, we introduce Python crawler, method of feature engineering, the principle of the XGBoost classifier and the user manual of Django front end. Section 4 we present our experiment method and experiment result. In section 5, we present the conclusion and future work.

2 Background and literature review

The NBA historical data used in this project comes from the site stat-nba[3]. Stat-nba is one of the biggest Chinese NBA information website. It provides all statistical competition data from 1986 to 2018. In these data, the statistical datas of 71 teams and 4647 players are included, season by season or by career. Our main purpose is to find out what is hidden behind the normal box score. Next we will show that why we need these implicit knowledge behind the box score.

In NBA, there are many indexes or box scores to evaluate the performance of distinct players. The three most significant box scores are points, assists and rebounds in a game[4]. Figure 1 shows the distribution of points, assists and rebounds the players gained in their NBA career. One observation is that the distributions are similar and follow exponential distribution. It means that in the past competitions, most of NBA players only contributed to a small fraction of total points, assists and rebounds. Besides, most of points, assists and rebounds are made by very small fraction of players. Another observation is that most of players box score is
In summary, most of NBA players have no excellent box score performance. Most of them even have box scores lower than the average. However, it doesn’t mean their value is low. Through the players transaction behavior, it is clear that related teams does not follow any rule directly positively correlated with these box scores. It shows that the performance of a team can not be directly evaluated by players box score. Hence, in next section, we will introduce how to find implicit knowledge behind box scores of players and teams through XGBoost classifier.

3 Methodology

In this section, the method to gain NBA historical data will be first introduced. Then the principle of XGBoost classifier and the method of feature engineering in this project will be presented. Last, the user manual of Django front end system will be shown.

3.1 Notations

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>The abstraction of the process part of a machine learning training.</td>
</tr>
<tr>
<td>Box score</td>
<td>including points, assists, rebounds and so on. A set of statistical index to estimate NBA players performance.</td>
</tr>
<tr>
<td>Accuracy</td>
<td>The division of the number of right predicted label by the total number of training samples. If possibility is given by a model. 0.5 is threshold for binary task.</td>
</tr>
<tr>
<td>Train accuracy</td>
<td>The accuracy on training data set. To estimate efficient of the training.</td>
</tr>
<tr>
<td>Validation accuracy</td>
<td>After each iteration, accuracy will be calculated on validation data set to check whether overfitting happens.</td>
</tr>
</tbody>
</table>
3.2 Crawl NBA historical data from stat-nba.com

Python is an convenient and legible script programming language. Most of work of this project is finished through Python. In this project, an crawler written in Python is used to gain NBA historical data from the internet. As mentioned above, all NBA historical data in this project comes from stat-nba. This site presents all competitions from 1986 to 2018 in the form of static html. Figure 2 shows an example of a competition data page and its page source in stat-nba.

To decode the web page and extract the valuable information from it, well analysis on the format of the page should be applied. From the example page in figure 2, it is clear that every competition has an unique ID so that it can be accessed through a url like ‘www.stat-nba/game/<ID>.html’.

Figure 2. A competition page in stat-nba
Besides, this ID is continuous. Based on this characteristic, we can use a Python library called as ‘request’ to send html request in the format of ‘www.stat-nba/game/<ID>.html’ to gain all page sources from stat-nba. Then, these page sources contain amount of information including the needed information for this project. The needed information is the teams name of both home field team and visiting team, their total scores in this competition, the number of team members, team members box score and whether they are start players. In this project, a Python regular expression library called as ‘re’ is used to extract the needed information from the page sources. Regular expression can search a pattern in a page source and extract what we need and save in a Python list. For simplicity of feature engineering in next subsection, the data of each competition is save as an independent file named by the ID of the competition. Figure 3 shows the result of this Python crawler and the structure of a competition data file.

From 1986 to 2018, there are 41211 competitions. So 41211 data files are created to save the information of each team. In a data file, the type of the competition, date, team name, score, team member and their box score has been extracted from the page source. In the next subsection,
the principle of XGBoost classifier and the method of feature engineering to generate training data for XGBoost classifier will be presented.

### 3.3 Brief of XGBoost principle and method of feature engineering

This subsection presents the brief principle of XGBoost classifier and advantage compared with other machine learning classifier. XGBoost is a tree ensemble classifier based on regression tree. The regression tree is similar to the decision tree. They have same decision rules. The difference is regression tree contains one score in each leaf value[5]. This leaf score is used in ensemble of multiple regression trees. Figure 4 shows the structure of a regression tree and the ensemble way of two similar regression trees.

![Figure 4. Structure of regression tree and ensemble way of two similar trees][5]

Figure 4 presents a family member classification task through regression tree. There are three features which are age, male or female and whether use computer daily. There are 5 classes which are boy, girl, mother, grandmother and grandfather. Each node of the trees represents a set of classes and each leaf have a score. The prediction score of a set of classes is the sum of leaf score in all regression trees. Compared with traditional boosted tree methods like gradient boosted regression tree (GBRT), XGBoost takes advantage of second Taylor Expansion of objective function. It means XGBoost leverages both first and second derivative. However the traditional boosted tree like GBRT only use first derivative. XGBoost adds regularization terms in objective function to control the complexity of the model. The regularization terms contain the number of

---

[5]: #![Figure 4. Structure of regression tree and ensemble way of two similar trees][5]
the tree nodes and the square sum of L2 leaf score. From the perspective of bias-variance tradeoff, the regularization terms decrease the variance of the model to simplify the model and to avoid overfitting. After each iteration, XGBoost will shrinkage leaf weight to weaken the effect of every tree to give more learning space to the following learning. XGBoost follows the column subsampling of Random forest. Column subsampling can avoid overfitting and reduce calculated amount. XGBoost can find out the split direction of the missing value. XGBoost also supports parallel based on the features. These advantages make XGBoost faster and better than traditional boosting tree classifier. In this project, 692 features in total will be generated in feature engineering. So XGBoost is decided to use considering the complexity of the model and training efficiency.

In last section, python crawler gain needed informations from stat-nba.com and save them into independent files. These files can not be fed into XGBoost classifier directly. The exact feature number and format should be decided first. In this project, 692 features are generated for a competition. These features can be classified as two categories — team feature and player feature. Team feature contains whether a team takes competition back to back which means taking two competitions in two consecutive days. Player features are based on 9 box scores and conducted season by season. The current season and the last season of a competition will be considered.

![Figure 5. Generated average, min and max value season by season](image)

From figure 5, each rectangle is a NBA competition. Assume season n is the current season, all competitions data is of season n-1 and season n is needed. For intuitive, the data of the later season should reflect better the current situation of a player. So the data of the current season is needed. However, at the beginning of a season, the information may be not
sufficient. So the data season n-1 is considered. As for season n-2, 2 years gap may not reflect the real situation of a player. For saving computational resource, season n-2 is not considered. For a specified season, the average, maximum and minimum values of 9 box scores for 5 start players of two seasons will be considered as independent features. Because each team has different number of substitution players and XGBoost classifier needs a fix number of features, the average values of all substitution players box scores will be calculated as features. The last feature is the result of the competition. 1 is first team wins and 0 is first team loses. The goal is to use XGBoost classifier to predict this label for an overcoming competition. A C++ program is written to read all data source files and generate training data set following the rules above. Figure 6 shows the generated training data set.

![Figure 6. Generated training data set after feature engineering](image)

In figure 6, first row is the label of the features. Then each row is a training sample corresponding to one competition. In next subsection, the Django front end system will be introduced to show that how to order a competition prediction.
3.4 Brief principle and user manual of Django front end system

In this project, a front end system is built based on the Django framework. Django is a free and open-source web framework written in Python, which follows the model-view-template (MVT) architectural pattern[6]. Model, view and template are three abstraction for different functions. When server receives a url request, it decodes the url and choose suitable ‘view’ to process the request. In ‘view’ process, ‘model’ may be used to contact with database. Finally ‘view’ pass the process result to ‘template’ to generate html response back to the user. Figure 7 shows the structure of MVT architectural pattern.

![Diagram of MVT architectural pattern](image)

Figure 7. The structure of MVT architectural pattern

Because this project apply the python implementation of XGBoost, Django framework based on Python becomes the best choice of this project. Url request from user is decoded into the information of team names and start players and is passed to ‘view’. Then ‘view’ initialize a XGBoost classifier and load a model we have trained in advance. A C++ program named ‘getTest’ will be called to generate test data set with the parameter of team names and start player names. This ‘getTest’ C++ program is similar to what is used to generate training data. The pseudo code of the ‘view’ is like:
Function predict(request):
    parameters was extracted from request
    load C++ program as getTest
    test data = getTest(parameters)
    bst = new xgb classifier
    bst.load('01.model')
    result = bst.predict(test data)
    return result

Then the user manual of this front end system will be shown.
1. Install Python3.5 from https://anaconda.org according to official manual
   and activate anaconda environment by export PATH=~/.anaconda2/bin:$PATH
   if it is installed successfully

2. Install Django by pip3 (pip3 install Django==1.10.4)

3. Install XGBoost by pip3 (pip3 install xgboost)

4. In the terminal, make working directory to nbet.

5. Compile getTest.cpp by ‘g++ -shared -Wl,-install_name,getTest.so -o
   getTest.so -fPIC getTest.cpp’

6. Open Django server at local host
7. Visit localhost:8000 through your browser

8. Full fill the form of team names and start player names.
9. Click ‘predict’ button and wait for result back.

The predict result shows that home field team has 80.68% probability to defeat visiting team. In next section, some experiments are designed to raise the accuracy of the model and the results are presented.

4 Experiment and result

In this section, some experiments are designed to find a set of the best super parameters for XGBoost classifier and search any ways can increase the accuracy of the model. Some common super parameters and their functions are listed below [5].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>definition and function</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_child_weight</td>
<td>it decides minimum weight sum of leaf nodes. Larger it is, less local special sample the model would learn. However, a too large min_child_weight causes under-fitting.</td>
</tr>
<tr>
<td>eta</td>
<td>it is similar to learning rate which used to shrinkage the weights after each boosting step. It can help avoiding over-fitting.</td>
</tr>
<tr>
<td>colsample_bytree</td>
<td>it controls the subsample ratio of columns for each regression tree.</td>
</tr>
</tbody>
</table>
After feature engineering, a data set with size of 36084 is generated. 10826 of them are used as test set. 17680 of them are used as training set. 7578 of them are used as validation set. Then different sets of parameters are applied to the model. The one with highest test accuracy would be chosen as the final parameter. First, a set \([1, 5, 10, 100]\) of min_child_weight parameter is checked by four experiment training. Other parameters are kept as default.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-depth</td>
<td>the maximum depth of a tree. It is also used to avoid over-fitting.</td>
</tr>
<tr>
<td>subsample</td>
<td>It controls the subsample ratio of training data for each tree. Less it is, more conservative the model is.</td>
</tr>
<tr>
<td>alpha</td>
<td>L1 regularization term on weights. Larger it is, more conservative the model is.</td>
</tr>
<tr>
<td>gamma</td>
<td>Leaf nodes only be partitioned when this would cause a smaller objective function value. Gamma defines the minimum decreasing value of objective function to allow to partition leaf node. Larger gamma is, more conservative the model is.</td>
</tr>
<tr>
<td>lambda</td>
<td>L2 regularization term on weights. Larger it is, more conservative the model is.</td>
</tr>
</tbody>
</table>

From figure 8, mid_child_weight is set to 5 which has the highest test accuracy.

Second, a set \([0.001, 0.01, 0.1, 1]\) of eta parameter is checked by four experiment training. Other parameters are kept as default.
From figure 9, eta is set to 0.01 which has the highest test accuracy.

Third, a set \([0.3, 0.5, 0.7, 0.9]\) of colsample_bytree is checked by four experiment training. Other parameters are kept as default.

From figure 10, colsample_bytree is set to 0.5 which has the highest test accuracy.

Fourth, a set \([5, 10, 15, 100]\) of max_depth is checked by four experiment training. Other parameters are kept as default.
From Figure 11, max_depth is set to 15 which has the highest test accuracy.

Fifth, a set [0.3, 0.5, 0.7, 0.9] of subsample is checked by four experiment training. Other parameters are kept as default.

From Figure 12, subsample is set to 0.5 which has the highest test accuracy.

Sixth, a set [0, 1, 10, 100] of alpha is checked by four experiment training. Other parameters are kept as default.
From figure 13, alpha is set to 1 which has the highest test accuracy.

Seventh, a set [0, 1, 10, 100] of gamma is checked by four experiment training. Other parameters are kept as default.

From figure 14, gamma is set to 1 which has the highest test accuracy.

After turning parameters, another experiment is applied to check whether deleting some useless features will increase the accuracy. After a training, XGBoost classifier will conclude a feature importance map. Feature importance of XGBoost means the frequency of a feature used to partition a node in regression trees. Figure 15 shows the feature importance after training.
From feature 15, there are some less important features. We plan to delete some less important features from training data set to decrease the bias. Different thresholds are used to delete features according to this feature importance map. A set [20%, 40%, 60%, 70%] of thresholds is checked by four experiment training. Figure 16 shows the result of four training.

From figure 16, deleting 20% features according to the feature importance map is suitable.

In summary, XGBoost classifier can make 67% test accuracy in this NBA league prediction. The parameters of the model are listed below.
5 Conclusion and future works

In this project, the competition result of NBA league is analyzed and predicted in an accuracy 67% by the help of XGBoost classifier. At the beginning, abundant historical competition data and huge gambling market of NBA become the motivation of this project. Then, the box scores like rebounds, assists and points of all players of NBA history are analyzed. We found the box scores can only reflect a small part of players performance. The contributions of most of players can not be evaluated by the box scores. The contributions of this project is summarized as below.

1. We crawl the historical competition data from stat-nba and generate training data set which is more suitable for NBA league prediction task.

2. We design a front end system based on Django framework for the convenience of the users.

3. We also design some experiments to turn the parameters and suitable number of training features to raise the accuracy of the model.

For future works, we plan to collect more historical NBA competition statistical data other than box scores. For example, body index of players and coach strategy. We want to try different methods of feature

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_child_weight</td>
<td>5</td>
</tr>
<tr>
<td>eta</td>
<td>0.01</td>
</tr>
<tr>
<td>(colsample_bytree)</td>
<td>0.5</td>
</tr>
<tr>
<td>max-depth</td>
<td>15</td>
</tr>
<tr>
<td>subsample</td>
<td>0.5</td>
</tr>
<tr>
<td>alpha</td>
<td>1</td>
</tr>
<tr>
<td>gamma</td>
<td>1</td>
</tr>
<tr>
<td>Deleted useless features</td>
<td>20%</td>
</tr>
</tbody>
</table>
engineering. For example, we can conclude average box scores of last 5 competitions for each players of a competition rather than average box scores of last season and current season to avoid using unstable current season features. We plan to deploy this project into internet server to provide stable and convenient server for the users.
References