Predicting NBA Shot Outcomes

Patrick Carron

Abstract

Play-by-play data from the 2009-2010 NBA regular season is analyzed in an attempt to classify which shots are made or missed. A sparse feature representation where each player on offense, defense, or shooting is given their own indicator variable is compared against a representation using k means clustering to group each player into one of five clusters based on their previous season's statistics. A degree-2 polynomial basis expansion is performed followed by Principle Component Analysis and i)Bernoulli Naive Bayes, ii)Stochastic Gradient Descent with a variety of loss functions and iii) Random Forest classifiers are implemented and compared for accuracy. A train validation test split is used for hyperparameter optimization ensuring that no future data is used in training. A Random Forest classifier with 40 estimators and a max tree depth of 10 is found to achieve the highest test accuracy observed at 59.44%.

1 Introduction

Roughly 180 field goals are attempted in a typical regular season NBA game [1]. Only 46% of these attempts are successful. During the 2009-2010 NBA season, 172 of 1,230 games were won by 3 or fewer points [1]. If a coaching staff were able to increase their team's shooting accuracy even slightly they may be able to win several more games per season. This research attempts to determine if any predictive information on the outcome of a specific field goal attempt can be extracted from historic play-by-play logs and historic shooting player statistics. A sparse vector representation where each player is represented by a set of offensive, defensive, and shooting indicator variables is compared against a vector representation where players are clustered into one of five player types based on their previous season's statistics. Both feature representations include the shooting player's previous season performance statistics. The clustered representation is found to be more flexible and just as predictive in baseline tests, so it is selected to be used in a data-pipeline featuring a degree-2 polynomial basis expansion, Principle Component Analysis, and hyperperameter tuning using a train validation test split and a wider set of classifying techniques. A Random Forest with 40 estimators and a max tree depth of 10 is found to be the best preforming classifier with a test accuracy of 59.44%

2 Related Work

Research on NBA datasets frequently focuses on player performance prediction or game outcome prediction. Game level data is different from play-by-play data in that game data is an aggregation of statistics across all of the plays in a game. At this level information about individual possessions is not available, but there are some similarities in how data needs to be processed for a classification task. In *NBA Oracle* [3] Beckler, Papamichael, and Wang describe a game outcome prediction system which implements k-means clustering for player position inference and uses up to date historic player performance while insuring that no future information is used in either training or cross-validation. They chose to use 3 clusters to represent player positions, which might be optimal at a game aggregation level, but may not be the case with possession level data. They also found that there was not much of an increase in accuracy between using the previous season player statistics and using the most recent player statistics. They evaluated SVM, Logistic Regression, and Neural

Network classifiers and found that Neural Networks produced the highest accuracy.

Shot prediction research does exist but it tends to use data from visual and spatial sensors which capture information on every player during every possession in an NBA game. This data is not widely available. In the paper *CourtVision: New Visual and Spatial Analytics for the NBA* Goldsberry [4] describes and analyzes the spatial information available at a player level. He creates two measures for analyzing a players shot performance, spread which is a measure of spatial spread across predefined scoring cells, and range which is a count of the predefined scoring cells where a given player averages at least one point per attempt. This would mean that for a given player, over 50% of their shots in a two point region would have to be successful to increase their range count, while only 33% of their three point attempts would need to be successful. Overall players with high spread metrics tend to be high preforming Power Forwards while players with higher range tended to be Point Guards and Shooting Guards. The only player in the top ten of both spread and range metrics is Ray Allen, who is often regarded as one of the best shooters to ever play in the NBA. Goldsberry's research strongly suggests that player level position information may be crucial for predicting the outcome of a shot, suggesting that a sparse representation where each shooting player is represented individually may be powerful. The only position information available in this research describes where the shooting player is on the court. No information is available on where a foul is committed or where defensive players are at any point during the possession.

In the paper *Simulating a basketball match with a homogeneous Markov model and forecasting the outcome* Štrumbelj and Vračar [7] analyze play-by-play logs to create a possession-based Markov model of an NBA game which included team level historic performance statistics. They used Logistic Regression and bookmakers odds to predict a games outcome. They also used their model to create simulations of NBA games and compared their results against previous predictive models and found that their model was comparable. Their best results were found using bookmaker odds. While this paper does use play-by-play logs, the aggregation results in a game forecast and the extra features included are at a team level. The shot success component of their research seems to be a random emission variable based on historical information.

3 Data Sets

A repository of comma separated game level Play-by-play data for the 2009-2010 NBA season exists at the website *www.basketballgeek.com* [2] Information about the date and teams is available in the file name which is important since the date of the game will be used to split the data into training, validation, and testing sets. Around half of the records are deleted, along with variables that describe events that are not included in possessions with shot attempts. The name of the shooting player is recorded, along with the names of all of the players on the floor. Offensive and defensive possessions are determined for each player by examining whether the home and away team is shooting. One important idiosyncrasy of this data is that players occur in multiple variables through out a game. For example, if on the first possession a player is in column a_1, they may leave the game and return at a later point in column a_2. This means that each record must be converted so that a set of players is recorded so indicator variables for each player at each position may be created in the sparse feature representation, or that player cluster assignments must be recorded for the clustered feature representation. The player level statistics are joined based on the year and shooting players name. $Time$ and $Period$ variables are converted to a single variable representing the number of remaining seconds in each game. Finally, each game needs to be processed twice, once for the sparse representation and once for the clustered representation.

Player level per game playing statistics for the 2008-2009 NBA season are available at *www.basketball-reference.com*. Since only historic information can be included for an individual record, and player data tends to be aggregated over the course of a season, Rookie players performance will need to be estimated. A list of rookie statistics from the 2008-2009 season is averaged for each count statistic and new ratios are calculated to create an 'Average Rookie' representation. This 'Average Rookie' greatly underestimates the statistics of a few high preforming rookie players, but rookie players tend to have far fewer possessions per game than veteran players. A list of players from the 2009-2010 NBA season is joined to their player statistics from 2008-2009 if the player is not a rookie, and if the player is a rookie the 'Average Rookie' statistics are used. One more interesting idiosyncrasy of NBA data is that player names change over time and sometimes nicknames are used in place of birth names. For example, the a single player is named Ron Artest in the play-by-play data while in the player statistics data he is called Metta World Peace. Instances like this are transformed manually to ensure that each record in each feature representation is populated with shooting statistics. A file is created to join to the sparse representation at this juncture. The player statistics are then analyzed using k-means clustering across a range of k from 1 to 20. The k-means algorithm minimizes the Sum of Squares within each cluster C^*

$$C^* = \arg\min_C \sum_{k=1}^{K} \sum_{\boldsymbol{x_i}, \in C_k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_i||^2$$

Where C_k are the set of records belonging to cluster k and $||\boldsymbol{x}_i - \boldsymbol{\mu}_i||^2$ is the squared distance from each point in cluster k to the centroid of cluster k. Using the elbow selection criteria and the fact that there are 5 positions in basketball, a k of 5 is selected. A new file containing the name of each player and their cluster type is created. For the clustered feature representation the number of each player from each cluster on the court on offense and defense is recorded.

Both the sparse and clustered feature representation are comprised of 307,305 records. The sparse representation, however, has 1,527 dimensions compared to 43 for the clustered representation. This is due to the fact that every player that played in the NBA during the 2009-2010 season is assigned three indicator variables, one for offense, one for defense, and one for shooting. Because no future information can be used in training, the datasets are split into training, validation, and test sets. The validation set is used to select optimal hyper-parameters. The training set is made up of 252,934 possessions from the beginning of the 2009-2010 season to March 18th 2010. [1] The validation set is comprised of 21,103 possessions from games between March 19th 2010 to March 31 2010. The test set is comprised of 33,308 possessions occurring between April 1st and the end of the regular season on April 15th. In an effort to be memory efficient, Scipy [5] sparse matrices are used with the sparse representation while Numpy [8] arrays are used with the clustered representation. A variable listing is included in the appendix.

4 Proposed Solution and Experiments

4.1 Clustered Representation Selection

In order to select which feature representation will be modeled, some baseline experiments are conducted with each feature representation and each method is evaluated on both accuracy and fexibility. Scikit-Learn [6] is the library which is used to build the data pipeline. A pipeline is constructed including a degree-2 polynomial basis expansion, a dimensionality reduction technique, and a baseline Naive Bayes Classifier with default parameters is run for each. The default of the Bernoulli Naive Bayes function sets the α smoothing parameter to 1. Since the sparse representation uses Scipy matrices the dimensionality reduction technique selected is TruncatedSVD from Scikit Learn because it supports Scipy sparse matrices. Even with the Scipy sparse matrix format, memory usage was an issue for the sparse representation. The degree-2 polynomial basis expansion results in over 1,000,000 terms and even with the TruncatedSVD function reducing dimensionality to only the top 100 components, the Naive Bayes function was unable to run. When the degree-2 polynomial basis expansion was removed, the classifier was able to run successfully and achieved an accuracy of 52.42% on the test dataset. This is worse than a classifier that assumed every shot is missed. The sparse baseline run completed in 25 minutes and 33 seconds. The clustered representation successfully executed the entire proposed test pipeline, including the degree-2 polynomial basis expansion. The clustered representation achieved 54.33% accuracy and completed in 4 minutes and 55 seconds. Besides having lower accuracy and the lack of the ability to hold the degree-2 basis expansion in memory efficiently, the sparse feature representation would also limit the types of classifiers that could be tested. The sparse representation was also 5 times slower than the clustered representation. Because of these initial results, the clustered representation is selected as the feature representation to be further modeled. Perhaps with access to more computing resources, the sparse representation would be more viable.

[1] 4 weeks after the NBA 'trade-deadline' wherein lineups are locked for the remainder of the season

4.2 Basis Expansion and Dimensionality Reduction

The degree-2 polynomial basis expansion remains in the pipeline for all remaining experiments. Instead of using the TruncatedSVD function, however, Principle Component Analysis is performed using the PCA function in Scikit Learn [6]. This allows for the basis expansion to be condensed to a lower dimensionality representation before hyper-parameter tuning takes place which will help shorten the runtime for each classification experiment.

4.3 Classifiers and Hyperparameters

As described above a train validation test split is predefined for the dataset. The algorithms that will be tested are Bernoulli Naive Bayes, Stochastic Gradient Descent, and Random Forest classification. Each classifier has its own set of hyperparameters which are selected based on their performance on the validation set. Once optimal hyperparameters are found, each classifier is run on the test data and accuracies are compared.

4.3.1 Bernoulli Naive Bayes Experiments

The equation for the Naive Bayes classifier is:

$$f_{NB}(x) = \arg\max_{c \in Y} \Pi_c \prod_{d=1}^{D} \Phi_{cd}(x_d)$$

Where Y is the set of classes c and Π_c is a given class's probability and $\Phi_{cd}(x_d)$ is the marginal class conditional distributions. Π_c and $\Phi_{cd}(x_d)$ are learned from the training data. Since Bernoulli Naive Bayes pertains to binary classification, $\phi_{cd}(x_d) = \theta_{dc}^{x_d}(1-\theta_{dc})^{(1-x_d)}$ where θ_{dc} is the probability of an observation being in a class and it is multiplied against the probability that is in the other available class.. The hyper-parameter for Bernoulli Naive Bayes include a smoothing term α to correct for out of data observations. A wide range of α differing in multiples of 10 from from .0001 to 1000 will are searched initially and then a specific range around the best preforming power of 10 is searched more exhaustively.

4.3.2 Stochastic Gradient Descent Experiments

The Stochastic Gradient Descent classifier in scikit learn allows for different loss functions to be passed as a hyperperameter, where gradient descent is performed to find the optimal local minima of the training error corresponding to highest accuracy. The scikit learn documentation [6] expresses this mathematically as

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

where L corresponds to a loss function, R is a regularization term and α is a regularization multiple. The selecting $Logistic\ Loss$ function $\log(1+e^{-y_i g(\boldsymbol{x})})$ where $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}+b$ effectively turns this classifier into Logistic Regression, while the $Hinge\ Loss$ function $\max(0, 1-y_i g(\boldsymbol{x}_i))$ implements a Linear Support Vector Machine classifier and the $Perceptron\ Loss$ implements the Perceptron algorithm. Each of these classifiers share a regularization hyperperameter that can be set to $l1$ corresponding to the l_1 norm $||x||_1 = \sum_{i=1}^{n}|x_i|$, $l2$ corresponding to the l_2 norm $||x||_2 = \frac{1}{2}\sum_{i=1}^{n} x_i^2$ or to $none$ for no regularization. Again wide range of α differing in multiples of 10 from from .0001 to 1000 will are searched initially and then a specific range around the best preforming power of 10 is searched more exhaustively.

4.3.3 Random Forest Experiments

Random Forest classification is an ensemble method in which multiple weak classifiers in the form of single decision trees are bagged together to increase accuracy and decrease variance. Because the temporal nature of this data creates inability to use future information at any point in training, only one validation set can be used which increases variance in model parameter selection compared to a cross validated approach. The bagging property of the Random Forest classifier may help combat this variance. A range of values will be searched for hyperperameters for the number of estimators and the maximum depth of each tree.

5 Experiments and Results

The same data preprocessing and dimensionality reduction were used for all classifier experiments to insure for controlled comparison of accuracy. The only portions of the pipeline that was varied were the hyperperameters to be considered which are listed in section four for each classifier. The optimal hyperperameter setting for Bernoulli Naive Bayes was α at 101 which resulted in a validation accuracy of 58.47% and a test accuracy of 57.58%. The optimal hyperperameter setting for the Stochastic Gradient Descent classifier was $Logistic\ Loss$ with an $l1$ regularization penalty and an α value of .004 as shown in Figure 1 which resulted in a validation accuracy of 59.34% and a test accuracy of 59.12%. The Random Forest classifier had optimal hyperperameters with a max tree depth set at 10 and with the number of estimators set at 40 as shown in Figure 5 and Figure 5 resulting in a validation accuracy of 59.47% and a test accuracy of 59.44%. Overall the Random Forest classifier narrowly outperformed the Stochastic Gradient Descent classifier with $Logistic\ Loss$.

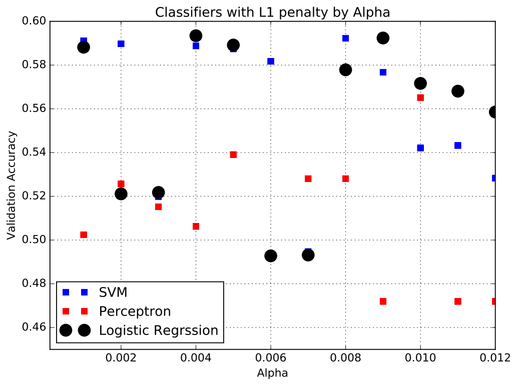

Figure 1: Validation Accuracy for Each Loss Function and L1 Penalty Over α Values

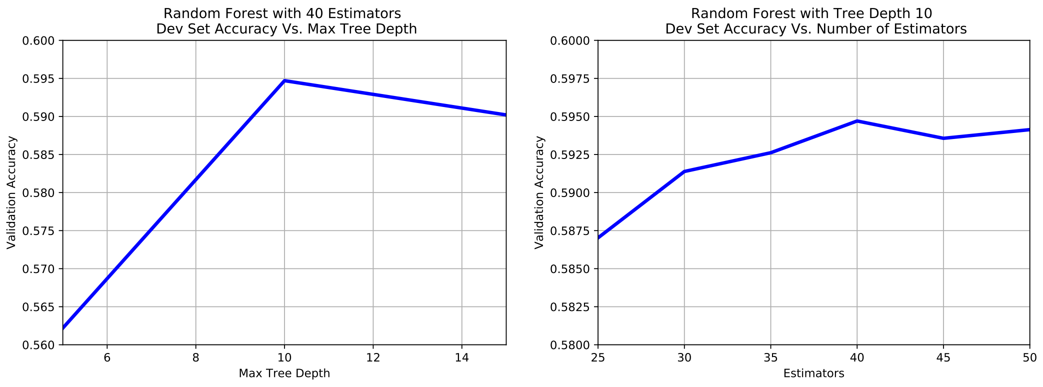

Figure 2: Validation Accuracy Random Forest Over Max Tree Depth Estimator Values

6 Conclusions

In conclusion while there is a small increase in accuracy over a baseline naive classifier that predicts every shot as missed, 59.44% may not be accurate enough to be useful for an NBA team. The the clustered feature representation allowed for increased flexibility for this research, but alternative clustering representations may in fact have higher accuracy. If more computational power were available, perhaps the sparse representation would give information about specific player interactions rather than a generic type of player interaction. One interesting fact about the NBA compared to other sports is that individual ‘Super Star’ players tend to have a disproportionate impact on game outcomes. Ideally a sparse vector representation might be able to encode this information. More player position information would also be interesting to incorporate. For example, if all defensive players are close to the basket perimeter shots would be uncontested. If the courtVision data described by Goldsberry [4] were made publicly available it would be interesting to analyze at a possession level similar to this research. Additionally, adding previous seasons of data to increase training observations, as well as implementing a system to update historic player statistics with in season performance might be useful for increasing accuracy.

7 Appendix

Table 1: Historic Player Statistics

Previous Season Statistics Included With All Records	
Name	Description
Age	Integer age of player
G	Games played
GS	Games started
MP	Minutes played per game
$FG, 3P, 2P$	Total/ 3-point/ 2-point shots made per game
$FGA, 3PA, 2PA$	Total/ 3-point/ 2-point shots attempted per game
$FG\%, 3P\%, 2P\%$	Ratio of type of shot makes to attempts per game
$eFG\%$	Effective shooting percentage
ORB, DRB, TRB	Offensive / Defensive/ Total rebounds per game
AST	Assists per game
STL	Steals per game
BLK	Blocks per game
TOV	Turnovers per game
PF	Fouls per game

Table 2: Play-by-play Features

Sparse and Clustered Differ Player Representation			
Name	Description	In Sparse	In Clustered
$time$	Integer representing seconds remaining in game	Yes	Yes
$result$	-1 for missed shot 1 for made shot	Yes	Yes
x	x coordinate of position on court when shot is attempted	Yes	Yes
y	y coordinate of position on court when shot is attempted	Yes	Yes
d_{0-4}	cluster assignment of defensive players on floor	No	Yes
o_{0-4}	cluster assignment of offensive players on floor	No	Yes
s_{0-4}	cluster assignment of shooting player	No	Yes
$d_{Playersname}$	Indicator if defensive players on floor	Yes	No
$o_{Playersname}$	Indicator if offensive players on floor	Yes	No
$s_{Playersname}$	Indicator if shooting players	Yes	No

References

[1] 12 2016. URL `www.basketball-reference.com`.

[2] 12 2016. URL `http://www.basketballgeek.com/data/`.

[3] Matthew Beckler, Hongfei Wang, and Michael Papamichael. Nba oracle. *Zuletzt besucht am*, 17: 2008–2009, 2013.

[4] Kirk Goldsberry. Courtvision: New visual and spatial analytics for the nba. In *2012 MIT Sloan Sports Analytics Conference*, 2012.

[5] Eric Jones, Travis Oliphant, P Peterson, et al. Open source scientific tools for python, 2001.

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[7] Erik Štrumbelj and Petar Vračar. Simulating a basketball match with a homogeneous markov model and forecasting the outcome. *International Journal of Forecasting*, 28(2):532–542, 2012.

[8] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science &amp; Engineering*, 13(2):22–30, 2011.