6.867: Final Project

1. Introduction

The NBA is currently one of the most popular sports in the world. Just a few months ago, over 30.8 million viewers tuned into watch Game 7 of the NBA finals between the Cleveland Cavaliers and the Golden State Warriors making it the third most viewed event in the U.S in 2016 (just behind the Super Bowl and the Academy Awards). As a result of this popularity, the betting market for the sport is enormous. NBA commissioner Adam Silver predicted the market to be around 400 billion dollars.

This report is concerned with exploring a variety of supervised learning techniques in hopes of being able to make predictions of game outcomes that are more accurate than the predictions made by NBA experts who set the betting line for each game. Before diving too deep into the algorithms, models, and predictions, we want to provide a quick overview of the structure of an NBA season as well as the various types of bets you can make in the NBA.

First, it is important to understand that the NBA is comprised of 30 teams split into two conferences (East and West). These 30 teams will each play 82 games across the regular season meaning that 1230 NBA games are played in total each season. For each game of the season, there are multiple bets that can be made. Below, we will discuss the two most common bets:

a) Win/ Loss: This is the most basic bet that can be made. The gambler simply picks which team he believes will win the game and if he is correct then he'll win money.

b) The Spread: The spread is a bit more advanced than simply win/loss. When a gambler bets against the spread, he either bets that the favorite shall win by more than the spread or that the underdog will lose by less than the spread. Typically, the spread is expressed as a negative number, which signifies the expected margin of victory for the favorite. In order to place a bet against the spread, a gambler generally needs to bet \$110 for the chance to win a \$100 payoff. The amount required for the chance to win \$100 is generally expressed in parentheses.

e.g. Golden State Warriors -7.4 (-110)
Cleveland Cavaliers 7.4 (-110)

The gambling authority generally selects the spread with the goal of splitting the betting money down the line in order to essentially make arbitrage with 0 risk. Due to the fact that the spreads are created in this manner and not with accuracy in mind, we believe that there is definitely room for improvement

2. Data collection

For this project, we needed to gather a great deal of historical NBA game statistics in order to train our models. To collect season data, we primarily used publicly available data from `http://basketballvalue.com/downloads.php`.
These datasets contained time series data in the form of matchup logs for all NBA games ranging from 2005 to 2012. Matchup logs basically describe the interactions that occur between 5-person units. A new entry is created every time a substitution is made in a game. Through running a python script on these matchup logs, we were able to derive and construct a variety of statistics that would later serve as features. For seasons 2005-2008, the matchup logs were a bit simpler, so we were only able to derive basic statistics including points allowed and points scored. The matchup logs became a bit more advanced in seasons 2008-2011, which allowed us to derive more sophisticated statistics including offensive and defensive rebounds as well as possessions.

For all seasons, we also constructed a new feature called ELO that is basically a reflection of a teams performance taking into account strength of schedule. What this means is that if two teams possess identical records, one of the teams may actually have a higher ELO score if they had to face stronger competition. In our literature review, this was not a feature that had been incorporated before in machine learning so we were particularly excited about its implications.

Many of the features generated for training and testing our predictor were derived from this game data. In order to create a general proxy for how strong a team was, we calculated running averages of points allowed,

points scored, win rate, ELO, etc. and used these as input features to our classifier. These running averages were initialized to equal numbers for all teams at the beginning of the season and were subsequently updated as more data became available. Because these numbers were arbitrarily initialized to equal at the beginning of the season, however, running averages near the beginning of the season serve as less reliable data points, as outliers can significantly affect the running average.

These statistics were therefore plotted over time throughout the season in order to approximate when the running averages for each of these statistics stabilized to reliable averages. Example plots are given in Figures 1 and 2, in which the running average win rate and points scored for a given team is plotted over course of the season.

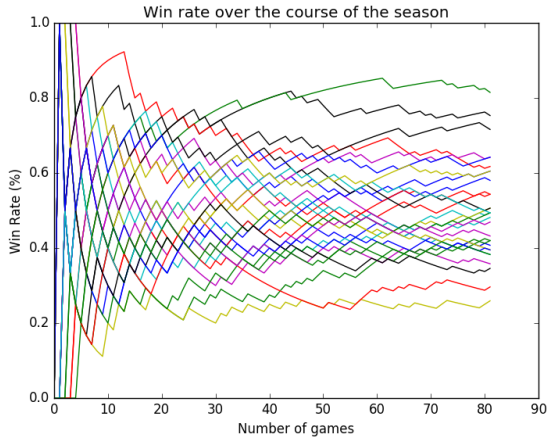

Figure 1. Win rate over time

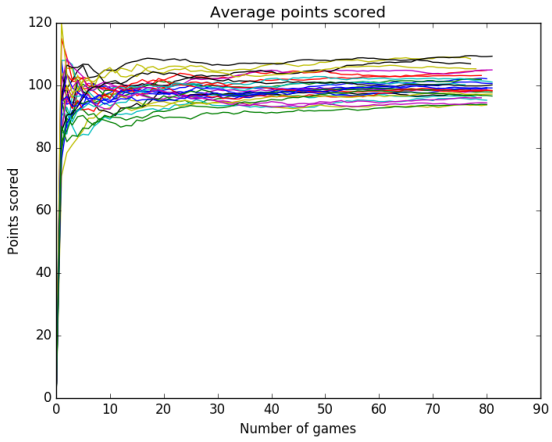

Figure 2. Points scored over time

In general, the running averages of the game data stabilized after approximately each team's twentieth game of the season (around the 1/4th mark of the season). When generating our training, validation, and test sets therefore, we only used data after each team's twentieth game in order to ensure that the features extracted from the data were representative of true team performance.

In order to compare our algorithm and our predicted spreads against Vegas and other betting authorities, we created a scraper to scrape relevant websites with both current and historical spreads. We primarily scraped this information from `http://www.sportsbookreview.com/betting-odds/nba-basketball/`.

3. Normalization

The need for feature normalization arises because our features are on different scales. For example, the average points a team scores every game is around 100 whereas the average number of offensive rebounds a team gets per game is around 15. In order to normalize the features we took a Z score approach that centered all the data at 0 and set the standard deviation to be 1.

4. Linear classifier

As a baseline, we used our own implementation of a linear classifier on basic features including running averages of home/away team points scored, points allowed, Elo score, games won, games played, and win rate. Based on the 2006-2007 season, in predicting win/loss (as opposed to the spread of a win/loss), the linear classifier had an accuracy of 64.3% on training data and 63.3% on testing data.

After adding further features, such as home court advantage, were added to the model, the linear classifier has an accuracy of 72.3% on training data and 66.9% on testing data based on the 2008-2009 season.

We additionally explored the ability of the classifier to predict spreads on games, in which the margin of victory is taken into account instead of a simple win/loss. In this case, the accuracy metric was the absolute value of the difference between the predicted spread and the actual spread. Using basic features on the 2006-2007 season, the average absolute value spread error was 9.31 points for training data and 10.07 points for testing data.

After adding further features, the average absolute value spread error decreased to 8.53 points for train-

	Win/Lose Accuracy	Spread Error
Basic features	63.3%	10.07
Extra features	66.9%	13.60

Table 1. Classifier performance on test dataset

M=1	**M=2**	**M=3**	**M=4**	**M=5**
10.15	11.35	11.013	12.40	11.32
M=6	**M=7**	**M=8**	**M=9**	**M=10**
12.05	33.26	33.56	17.11	150.23

Table 2. Average spread error for M values

ing data but increased to 13.60 points for testing data. This would imply that the classifier might be overfitting to the features that are provided as input to the algorithm, which would be expected for more complex architectures but was surprising in the context of the linear classifier explored here.

These results for the performance of the linear classifier are summarized in Table 1.

Similar results were found when we used a polynomial basis and allowed the degree of the polynomial fitting the data to grow. Out of values for $M \in [1, 10]$, we found that a regression model with $M = 1$ performed best, where $M = 10$ on the other side of the spectrum led to extreme overfitting. The average spread error is summarized in Table 2 for each of the different possible values for M, and graphs for representative values of M are illustrated in Figure 3. Note that for ease of illustration of data here, only the win rate feature is used as the feature vector

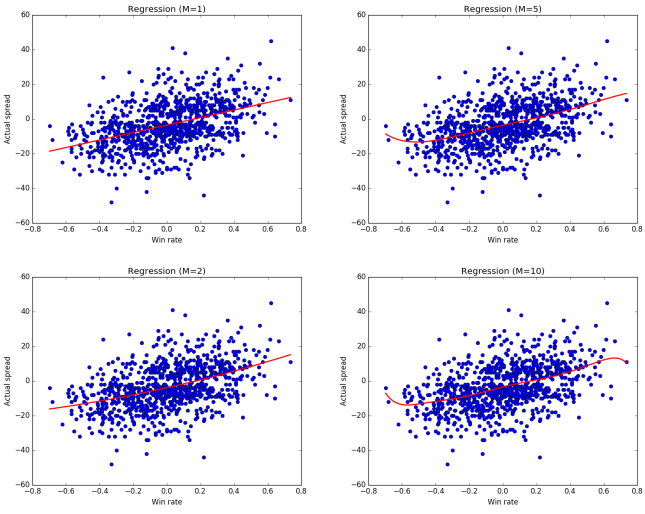

Figure 3. Win rate vs. actual spread

We additionally explored the use of generating training, validation, and testing datasets across multiple seasons of data. In this case, because we train the model against certain features of the teams playing (e.g. running average win rate, average points scored, average points allowed, etc.), and not the skill level associated with the team itself, we can aggregate data across multiple seasons. This holds true even if teams and their respective players have changed significantly between seasons, because the features we draw from the data are running averages calculated from the season so far.

When we run the linear classifier on three seasons worth of data, in which the training, validation, and testing datasets are each made up of a season of data (about 1300 games), the accuracy of the algorithm improves as expected. More specifically, when run on three seasons of data, the linear classifier is able to classify win/loss of matchups with an accuracy of 72.3% on training data and 68.6% on testing data. In terms of spreads, the linear classifier predicts spreads with an average spread error of 8.98 points on training data and 9.29 points on testing data. Thus, increasing the size of the data set has a fairly significant improvement in the accuracy of the linear classifier when compared with the 66.9% accuracy previously obtained for win/loss predictions and 13.60 spread error for spread predictions.

As previously mentioned, we explored classification of the linear classifier with a set of basic features and a set of more complex partially derived from these basic features. The basic features included each team's Elo score, average points scored, average points allowed, number of games played, number of games won, and win rate. The additional features we added included home court advantage (e.g. home court average points scored, home court average points allowed, home court win rate, etc.) as well as possessions, offensive rebounds, and defensive rebounds. In order to understand which features contributed most heavily to the win/loss and spreads predictions, we plotted the weights of each of the features. These relative weights are illustrated in Figure 4, in which the win rate features (away team win rate, home team win rate, away team home court win rate, and home team home court win rate) were in general the most important features. For three seasons of data however in win/loss predictions, the other features contributed much more relative to the classifiers for spreads predictions and one season of data.

Interestingly, when we train the linear classifier using only the win rate features, the classifier actually per-

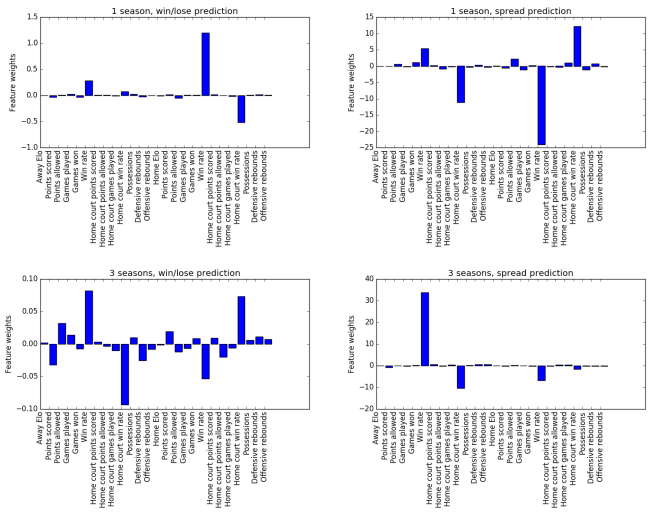

Figure 4. Linear classifier feature weights

	Win/Lose Accuracy	Spread Error
All features	66.9%	13.6
Only win rate	70.7%	8.95

Table 3. Classifier performance on 1 season

forms slightly better than with all the features. The numbers in Table 3 and 4 represent the accuracies and spread errors of the classifiers trained on all features and only win rate features when trained on one and three seasons of data, respectively. These findings seem to contradict the conclusions earlier in that adding more complex features such as home court advantage and rebounds improved the accuracy of the classifier. This could be due to overfitting to the number of features, especially in short periods of time in which averages might not necessarily be as representative of the overall skill level of a team.

5. Evaluation

In addition to calculating and analyzing the average spread error, we also compared our spreads predictions against other betting authorities' spreads. After creating a scaper which scraped the spreads posted by other betting authorities, we were able to determine the rate at which our spread would have beaten the betting authority's spread. This would occur in two scenarios: (1) if our predicted spread indicates that the winning team would have won by more than the other betting authority's predicted spread (e.g. if our spread is -8 and the other betting authority's spread is -5 on a game in which the favored team won by more than five points), or (2) if the predicted spread indicates that the winning team would have won by less or lost compared with theo ther begging authority's predicted spread (e.g. if our spread is +2 and the other betting authority's spread is -5 on a game in which the favored team lost or won by less than five points).

	Win/Lose Accuracy	Spread Error
All features	68.6%	9.29
Only win rate	68.7%	9.14

Table 4. Classifier performance on 3 seasons

6. Potential sources of failure

Some drawbacks to the algorithm presented here include the fact that we do not examine individual player data. This was a choice made largely due to the fact that individualized player data is quite difficult to obtain, especially in quantities large enough for accurate training, validation, and testing. Because we look at a team's performance overall so far in the season however, this means that our algorithm is susceptible to changes in starting player lineups, player trades made during the season, and player injuries. Thus, despite the acceptable performance of our algorithm, we do identify this source of weakness in the ability to predict spreads and classify win/loss in matchups. Future iterations of the algorithm would therefore primarily seek to address these potential sources of failure in order to further improve and refine the overall predictive power of the algorithm.

7. Breakdown of individual work

The breakdown of work between the two members of the group largely followed that given in the original project proposal. Andrea and Tyson jointly worked on preliminary review of the current techniques and the state of the art. Following this, Tyson was primarily involved in the collection of game data from past seasons, and Andrea created the scraper to collect data on historical spreads put out by various betting organizations on previous games. Andrea explored preliminary models such as linear regression, logistic regression, and regression with various basis functions and various degrees of regularization, and Tyson chiefly explored a neural network approach. As discussed during the project proposal meeting, we decided not to implement a convolutional neural network for this problem, as the architecture did not fit the structure of the problem as closely. We each tested the models we worked on against data test sets and historical spreads, and

we split the writing up of the report equally.