10707 Final Project: Report
Predicting Performance of NBA Basketball Players Using Deep Learning

Yugandhar Pavan Devarapalli, Grant Xu, Snigdha Rajeshk Purohit

1 Introduction

1.1 Problem Statement

Our objective is to predict the performance of NBA basketball players in an upcoming game using Deep Learning Techniques and apply it in fantasy sports.

1.2 Motivation

The general motivation is to apply Deep Learning methods to Sports Analytics. Much of sports data can be modeled by tracking temporal dependencies. Deep learning provides some efficient models to do it, provided enough data is available. More specifically, much of current basketball analytics is about player movements within a game at a very granular level. However, an algorithm which analyzes player performance from game to game using deep learning, hasn't been explored much.

Daily fantasy sports games' increasing popularity provide an excellent practical application to this motivation. The goal is to select the best team satisfying the given constraints, which is expected to perform well in a given day. Predicting individual player performance can be highly helpful in solving this constraint satisfaction problem (which is another problem in of itself).

2 Related Work

Most of the previous literature on ML in the NBA, focuses on predicting the outcomes of NBA games (i.e., which team wins). State-of-the-art linear and logistic regression models correctly predict the outcome of about 68 percent of games, while more complicated models such as artificial neural networks (ANNs) and deep belief networks (DBNs) have yielded similar success. However, there has been significantly less research related to predicting individual player stats, particularly as applied to fantasy basketball. [1]

Furthermore,the lesser research which has been done to predict individual player statistics is mostly based on applying traditional machine learning techniques that do not explicitly take into account dependencies from one game to the next. Techniques which have been attempted include Linear Regression, Support Vector Machines, Random Forest Regression, and Generalized Linear Models.

Paper[1] presents an approach to predict the individual game outcomes using Linear Regression, Support Vector Regression and Random Forest Regression, and also presents a comparison between the three models. The best accuracy obtained for prediction in this paper was 70 percent.

Paper[4] makes an attempt to apply machine learning to fantasy sports in order to gain an edge over the average player. This paper is related to the work we have done in this paper. The data was taken from DraftKings which is a fantasy sports gambling website. Individual basketball players' fantasy scores were predicted using a linear regression algorithm and stochastic gradient descent as well as a Naive Bayes classifier with discretized state space.

The linear regression model was based on box score statistics from a player in previous games. Whereas the Naive Bayes approach was implemented using a multinomial Naive Bayes classifier, with discretized parameters to prevent an infinite event space.

3 Proposed Approach

We propose the use of Recurrent Neural Networks(RNNs), specifically LSTMs, to solve this problem. The idea behind using RNNs is that they provide us an architecture to analyze sequential data and explicitly model temporal dependencies from one time step to the next. It helps us propagate important performance metrics from a player's previous games to help us predict the player's performance in the upcoming game.

LSTMs are useful in tracking long term dependencies.LSTMs help preserve the error that can be backpropagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps, thereby opening a channel to link causes and effects remotely [2].

4 Data Pre-Processing

Our dataset consists of all player game statistics from games played in the 2016-17 NBA season (25360 rows). First, the data was grouped by player and sorted by date. Then, we created our RNN with LSTM input dataset as follows :

1. Within each player, we used a sliding window approach.

2. Each row consists of [x_1, x_2, x_3, x_4, x_5, x_6,x_7,x_8,x_9,x_{10},x_{11}], x_1 is the oldest game and x_{11} is the most recent game.

3. x_1...x_{10} are our inputs and x_{11} is our target value.

4. All x_i are 18 dimensions. The 18 features include minutes played, field goals made, field goals attempted, two pointers made, two pointers attempted, three pointers made, three pointers attempted, free throws made, free throws attempted, offensive rebounds, defensive rebounds, total rebounds, assists, steals, blocks, turnovers, personal fouls, and points.

5. All columns were standardized by subtracting mean and dividing by standard deviation.

6. Resulting dataset is of 20824 X 11 X 18. Here 20824 are number of samples, 11 is the number of games, 18 is the number of features.

7. The dataset was split into train and test by random sampling with test ratio of 0.25 when used for LSTM.

5 Implementation

5.1 RNN with LSTM

The PyTorch package was used to implement the RNN with LSTM.

The 10 sequence RNN with LSTM model implemented for this project is as shown in Figure 1.

Model Diagram

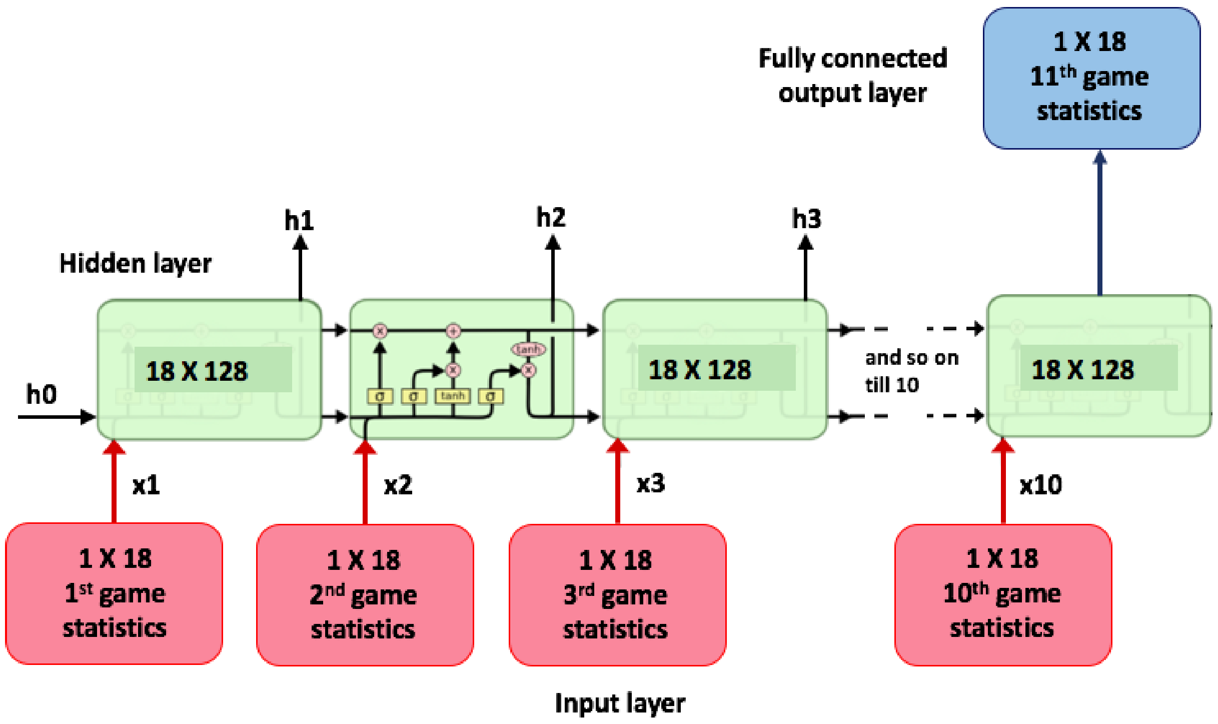

Figure 1: The 10 sequence RNN with LSTM model

We used the RNN with LSTM with the following :

1. The input is of batch size 128 with sequence 10 and 18 dimensions.

2. The output from hidden layer of the final step is then linked to fully connected output layer of dimensions same as the input i.e, 18.

3. We use a Mean Squared Error loss and output is linearly activated.

Once we get the 18 dimensional output layer we then rescale it to original magnitudes by multiplying standard deviation and adding respective means to each column/dimension.This vector is then used to calculate calculate an overall performance score using the formula is given by the fantasy sports website DraftKings [3].

Mean square error on the overall performance score was used for comparison.

5.2 Neural Network on top of LSTM output

The output of the LSTM (a 1 X 18 vector of predicted statistics for the 11th game) was then fed as part of an input to a neural network. This was done in order to incorporate some additional known variables for the 11th game that we do not have to predict. These include whether the player is playing at home or away, whether the player is starting, the opponent's overall defensive rating, the opponent's position-specific defensive rating, the Vegas projection of margin of victory, the Vegas projection of total points scored in the game, and the Draftkings salary of the player.

Thus, 25 attributes in total were used as an input to a neural network to predict the final Draftkings score in hope of achieving better accuracy than just using the 18 attributes from the LSTM. The neural network regressor was implemented using the sklearn package in Python. Various hyperparameters were tested, including the number of hidden layers (1 or 2), the size of the hidden layers (ranging from 2 all the way up to 128), the L2 regularization parameter, the learning rate, momentum, batch size, and the solver method. Performance improvements were minimal regardless of the hyperparameters used.

6 Results and Analysis

6.1 RNN

The best parameters are:
learning rate = 0.01; momentum = 0.9; weight decay = 0.001;
The train MSE output error decreases from 12.55 to 10.49 in 200 epochs
The test MSE output error converges to lowest value 10.76 in 66 epochs and then oscillates
The test score MSE error decreases from 93.38 to 88.6 (lowest at 28 epochs) and then increases slightly.

6.2 RNN with LSTM(10 sequence)

The best parameters are:
learning rate = 0.01; momentum = 0.9; weight decay = 0.001;
The train MSE output error decreases from 14.69 to 10.55 in 200 epochs
The test MSE output error converges from 12.19 to 10.337 in 72 epochs and then oscillates
The test MSE score error decreases from 91.60 to 82.8 (lowest at 27 epochs) and then increases slightly. The Figure 3, shows the plot of LSTM mean squared error for score.

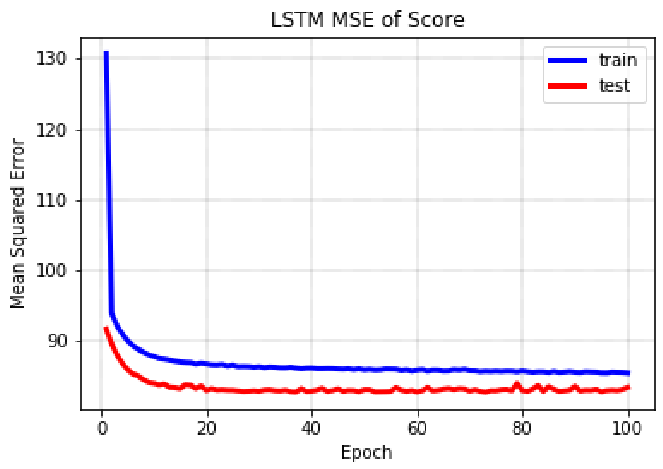

Figure 2: Error plot for the LSTM

6.3 Neural Network on Top of LSTM

The best performing networks on the training set were between 81 and 82 as observed from Figure 3, which is an insignificant improvement compared to the MSE of 82 achieved from the LSTM 10 sequence model. This suggests that the newly added attributes don't actually add much more predictive power, which was highly surprising, considering for example, the common perception that playing at home is better for the player. Either the effects of these newly added attributes are totally insignificant, or perhaps they predict the performance of an overall team much better than the performance of any individual player.

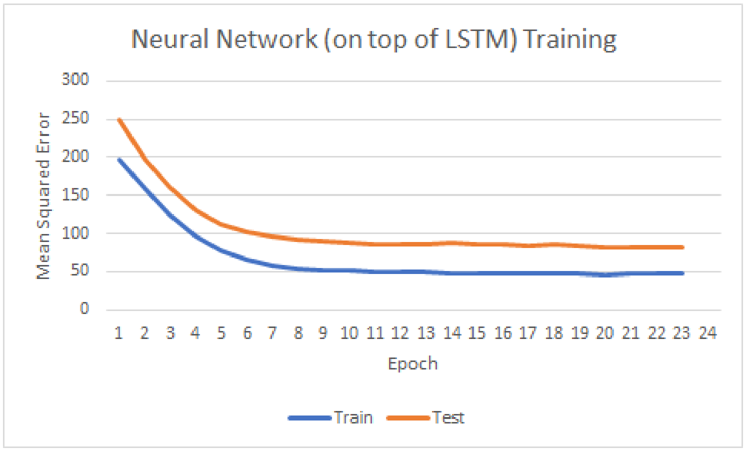

Figure 3: Error plot of Neural Network on top of LSTM

Alternative measures of error and model performance

Mean squared error is the metric that makes the most sense when training learning algorithms with no context. However, in the context of our problem, a more practical measure of error is needed to understand just how well the model is doing. We decided to define any prediction that was within 5 points of the actual score as "correct", and any prediction that was further than 5 points away from the ground truth to be "false". The 5 points is an arbitrary threshold, but a daily fantasy sports player would be decently comfortable adding a player to their roster if they can correctly predict that player's performance within such a range.

Classification accuracy on test data (LSTM only): 54.37%

Classification accuracy on test data (LSTM followed by NN): 75.49%

This is especially surprising, given that there was no discernible difference in MSE. With this measure of error, it seems that adding the neural network did improve performance. This could be possible if the neural network gives results that are on average moderately good, but the LSTM gives results that are both very good and very bad. In this case, MSE can be very similar, but the classification accuracy metric can have a wide disparity as seen above. Thus the NN likely has a smaller variance in its predictions, which can be helpful for a user or player looking to minimize their risk.

Another way to measure error is simply error with respect to the ground truth, which was used to compare our results with the results from a Stanford paper[4] in the following section.

7 Comparison with Previous work

There are two papers in the current literature on NBA Fantasy league Prediction which use traditional machine learning techniques as described in Related Work section. Our work is much similar to Stanford Paper [4] than the Paper [1] as later focuses on outcomes of games than explicit player performance in its metric evaluation.

To compare with the Paper[4] we made explicit changes to the performance metric evaluation. Instead of the MSE of the score, we calculated the error of predicted score to the ground truth as done in Paper [4]. The train and test error were plotted against the average player points per game ranging from 5 to 40. Figure4 shows the Player Points Vs Error Plot for our Model and Figure5 shows the Player Points Vs Error Plot for Paper[4] (Figure5).

The performance of our model far exceeded that of the previous model. For example, the test error for average player points of 5 of best model in Paper[4] was around 0.55 and our best model gives a performance of 0.30 which is quite significant. Similarly, the test error for average player points of 40 in Paper[4] was around 0.2 and our best model gives a performance of 0.15.

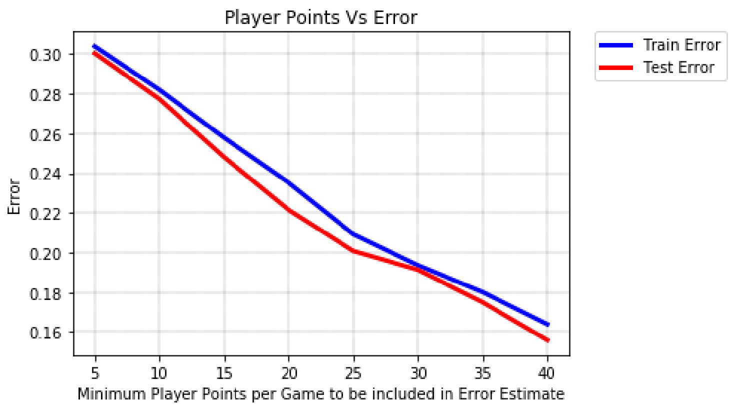

Figure 4: Player Points Vs Error Plot of our Best Model

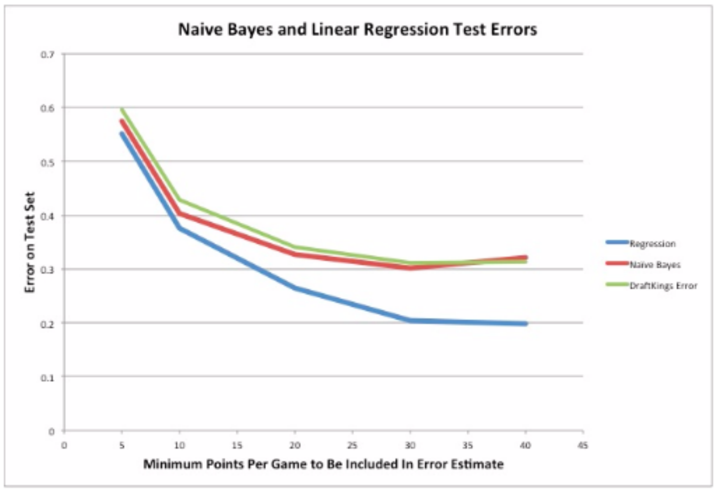

Figure 5: Player Points Vs Error Plot of Paper[4]

8 Future work

A Hidden Markov Model could be run on top of the LSTM model to try to increase accuracy of the predictions. In addition, different sequence lengths for the LSTM could be attempted (maybe longer sequence lengths). Another idea would be to add more training data from previous seasons or to exclude players from the dataset that tend to skew the model. These could be players that play minimally (less than a certain number of minutes), or who play incredibly erratically from game to game. Lastly, the model should be run on present day data (in addition to back testing), as that is the purest form of testing.

Beyond pure accuracy, it is also very important for the fantasy player to minimize their risk, or at least know what their risk is. To this end, it would be helpful if a model could be created that outputs some kind of variance measurement or confidence interval for the prediction.

Even further beyond this, the predictions and variance measurements can be used to solve the constraint satisfaction problem to create an optimal fantasy team. Much like assembling a stock portfolio, such an algorithm would have to optimize the right balance of bias and variance to help the user make as much money as possible.

9 References

[1] Chan, Hu, Shivakumar (2015) Learning to Turn Fantasy Basketball Into Real Money. `https://shreyasskandan.github.io/files/report-ChanHuShivakumar.pdf`

[2] A Beginner's Guide to Recurrent Networks and LSTMs. `https://deeplearning4j.org/lstm.html`

[3]`https://www.draftkings.com/help/rules/nba`

[4] Eric Hermann, Adebia Ntoso (2015) Machine Learning Applications in Fantasy Basketball. `http://cs229.stanford.edu/proj2015/104_report.pdf`