NBA Predictions with Neural Networks

Prathik Sannecy (prathik@pdx.edu)

I. Introduction

Avid basketball fans are always interested in making predictions before an NBA game about who would win. There are numerous popular sports shows that debate who would win an upcoming match-up, and there are many Fantasy NBA leagues dedicated to fans predicting the records of teams. With so much popularity about choosing who would win in an upcoming NBA game, it becomes imperative to find methods to accurately make predictions besides just having a gut-feeling. This paper investigates the use of Neural Networks to accurately predict NBA games, and compares it to several other techniques. In the end, I hope to create a model that an ordinary person can use to confidently predict an NBA game outcome without having an intimate knowledge of the teams' statistics.

II. Background and Related Work

Neural networks have already been used for predicting NCAA basketball and football games, as described in "NFL & NCAA Football Prediction using Artificial Neural Networks" [1]. Here, they used statistics for each team (e.g. rushing yards, turnovers, etc) to decide which team would be likely to win, and their results showed that they were consistently in the upper bracket. Similarly, a neural network was used for filling up a March Madness bracket [2]; this was done based off of using individual team statistics.

Mean Absolute Error					
Model	2007	2008	2009	2010	Mean
On the Field	11.00	11.31	11.68	10.99	11.24
Efficiency	11.00	11.70	11.69	11.25	11.41
PCA	11.21	11.12	11.48	11.20	11.25
Every Statistics	11.26	11.24	11.68	11.18	11.34
LRCO	11.20	11.10	11.51	11.16	11.24

Legend
Upper Quartile
Top Half
Bottom Half
Lower Quartile

Fig. 1: Results of NFL prediction with Neural Networks [1]

Although it wasn't a published paper, a student at Stanford had done a project similar to this one for predicting NBA games [3]. This paper uses the same framework (keras), but the dataset used differed. Like the other papers described above, Uudmae's dataset only included features and statistics about the actual individual teams themselves, not the games they had played against one another. Using this approach, he had achieved a 65% accuracy rate. The dataset and approach I have taken will be in contrast to this paper, and I use this paper as a comparison to gauge the level of success of my own approach.

	Model 1:SVM	Model 2: LR	Model 3: NNR
TEST: Average Accuracy	62.07%	63.75%	**64.95%**
TEST: Home Score Distance	N/A	**9.8203**	11.7816
TEST: Away Score Distance	N/A	**9.9472**	12.1915
TRAIN: Average Accuracy	**70.99%**	66.85%	70.34%
TRAIN: Home Score Distance	N/A	8.7656	8.8918
TRAIN: Away Score Distance	N/A	8.8426	8.8631

Fig. 2: Results of NBA prediction with Neural Networks [3]

III. Training and Testing Data

Rather than use pure statistical data from different teams to decide which one is more likely to win an upcoming game, this paper focuses on using prior matches instead. This list of matches, along with where they were played and the result of the game, was fed into the neural network to train it. That

way, to predict who would win an upcoming game, the model relies solely on the results of previous games.

Initially, I had planned to used the nba_py Python API to gather the NBA data for training and testing the Neural Network. However, the API had become outdated, and wouldn't correctly interface with the NBA website. As such I looked for an alternative source of gathering data, and found a complete dataset on Kaggle [4]. This data included information for all of the NBA games played between 2012-2018. In total, there were 7380 rows of data. To extract the data that I wanted to use, I created a Python script to remove all information except for the teams that played in each game, and which team was the home team in a particular game. The dataset was then reformatted as follows:

Home Team 1, Home Team 1 Result (win/loss), Away Team 1

Home Team 2, Home Team 2 Result (win/loss), Away Team 2

...

From this dataset, 80% was randomly chosen to be the training dataset, and the remaining 20% became the testing dataset, resulting in ~5900 games being used for training and ~1500 games being used for testing.

IV. Baseline Used for Comparison

To accurately gauge how well the neural network performs at making NBA predictions, two different baselines were used. They were both written in Python, and they both rely solely on prior results for making a prediction on a future game.

The first baseline looked solely at what teams were playing against each other, and used data from prior matchups. The team that had won the most matchups against the opponent previously was

predicted to win again. So for example, if team A had played Team B 3 games prior, and Team B had won 2 of these games, Team B was predicted to win again in the following game. In this model, 80% of games (5900) were used to set up the prior likelihoods; this algorithm was run on on the remaining 20% of games (1500). This model was able to correctly predict the winner ~59.7% of the time in the test dataset.

The second baseline looked solely at which team was playing at home. Generally, teams, that play at home have an advantage over the opponent. This model just predicted that the team playing at home would win, and using this strategy, this model was able to correctly predict the winner ~57.0% of the time in the complete dataset (7400 games).

V. RESULTS

The code to create the Neural Network was written in Python, and uses the keras library. The steps and procedures used for creating and training the neural network were taken from the keras tutorial [5] as a starting point.

To find the best hyperparameters for the neural network, I experimented with the number of epochs, the learning rate, the depth of the network, the height of the hidden layers in the network, and the dropout rate. Below are figures showing the results of experimenting with each one:

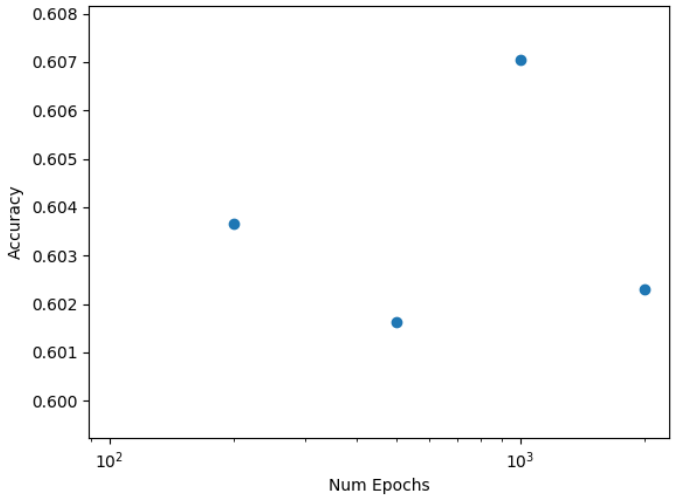

Fig. 3: Number of epochs vs accuracy on testing data

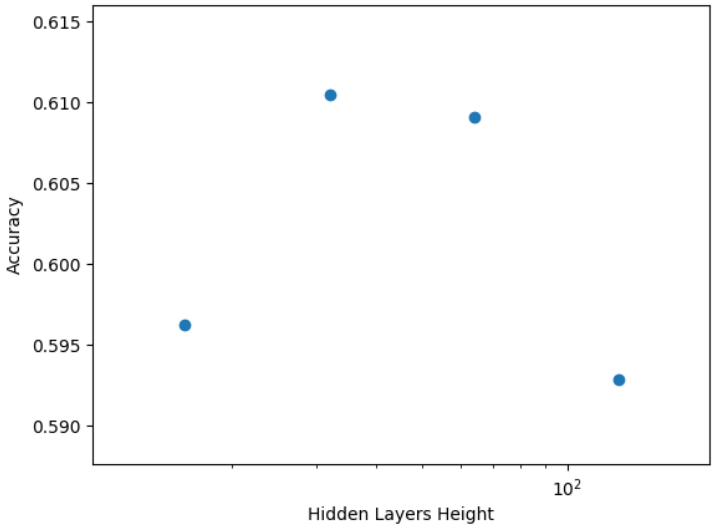

Fig. 4: Height (number of neurons) of hidden layers vs accuracy on testing data

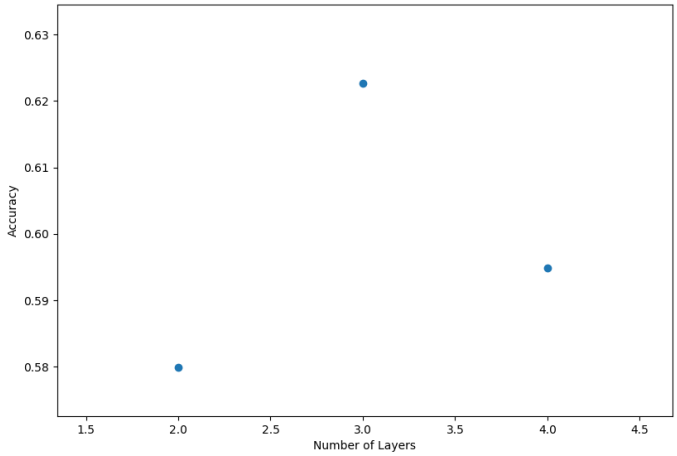

Fig. 5: Total number of layers vs accuracy on testing data

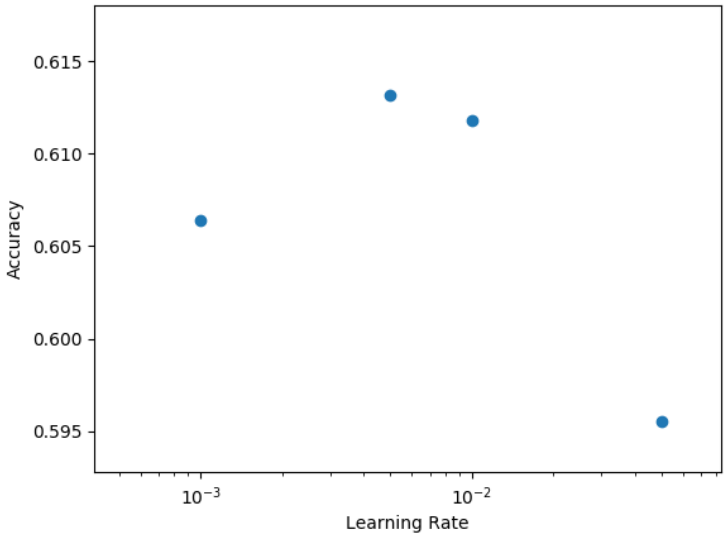

Fig. 6: Learning rate vs accuracy on testing data

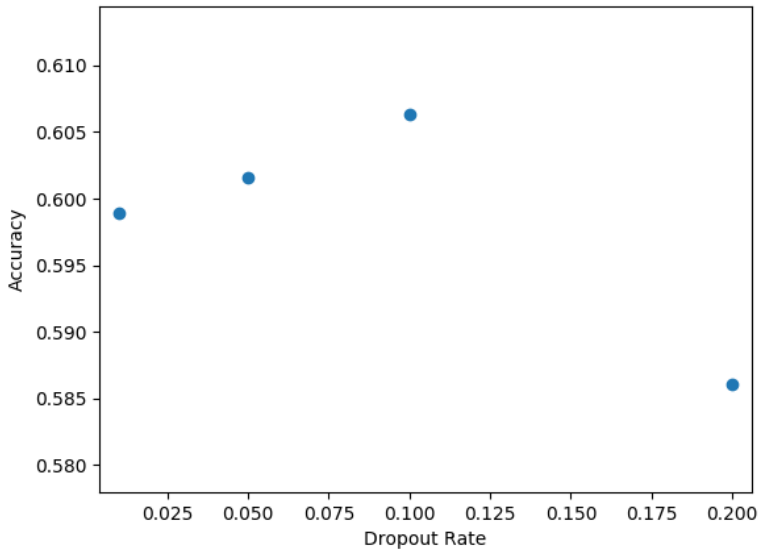

Fig. 7: Dropout rate vs accuracy on testing data

After a lot of trial and error the hyperparameters were tuned to result in the neural network below:

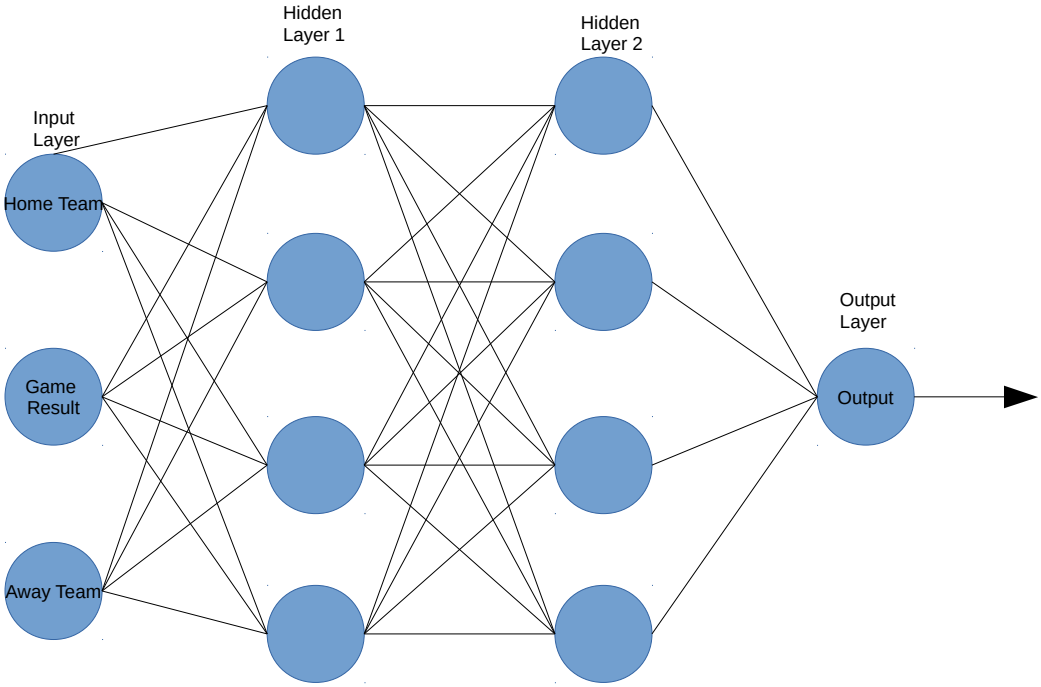

Fig. 8: Neural network used for NBA prediction

The final neural network has two hidden layers, each of which had 32 neurons. Each hidden layer uses the ReLU activation function, whereas the output layer uses the sigmoid function. The dropout rate is 0.01, and the learning rate is 0.005. To decide the winner of a game, if the output is greater than 0.5, the home team had won; else, the away team had won.

Using these hyperparameters to model the neural network, I compared it to the simple baseline models (see Baseline Used for Comparison section above), as well as the Uudmae's neural network model for predicting NBA games [3]:

Model	Uudmae Neural Network	Home Team	Prior Likelihood	Prior+Home Team Neural Network
Accuracy (%)	65	57	59.7	**61.5**

Table 1: Prediction results of different models

VI. Conclusions and Final Thoughts

Based on the results, we can see that a neural network in general performed better than the simple, baseline models. However, we can also see that optimizations to a model, or even using a drastically different model, only results in a slight increase in prediction accuracy. Therefore, if time is a factor, it may be worth it to sacrifice a few percentage points of accuracy with one of the simple, baseline models to save time optimizing and retraining a neural network.

In terms of accuracy, we can see that the model that had the greatest accuracy was Uudmae's neural network [3]. However, the drawback of using this model was that it takes 30 different features as an input (each of the competing team's statistics). Getting this feature set for making a prediction is not an easy task, and may not be a feasible expectation for an everyday fan or sports pundit. Therefore, while this model may be the most accurate among the four compared, it is certainly the most difficult to use.

Usually, casual fans that participate in Fantasy Leagues don't research the minute statistics when making a prediction. They usually just know who's playing, and where. With this limited information, the neural network that this paper presents is likely the best source for them, given their limited knowledge. Although it performs 3.5% worse than Uudmae's neural network, it performs at least 1.8% better than other methods would using the same information.

In conclusion, while there are better method and models out there for predicting NBA games, including other neural networks, the neural network presented in this paper performs better than other, simpler models would given the same data.

VII. REFERENCES

[1] A. Blaikie, J. David, G. Abud, and R. D. Pasteur, "NFL & NCAA Football Prediction using Artificial Neural Networks," https://www.semanticscholar.org/, 2011. [Online]. Available: http://personal.denison.edu/~lalla/MCURCSM2011/4.pdf. [Accessed: 21-Mar-2020].

[2] reHOOPerate, "Training a Neural Network to fill out my March Madness Bracket," Medium, 15-Mar-2018. [Online]. Available: https://medium.com/re-hoop-per-rate/training-a-neural-network-to-fill-out-my-march-madness-bracket-2e5ee562eab1. [Accessed: 21-Mar-2020].

[3] J. Uudmae, "CS229 Final Project: Predicting NBA GameOutcomes," Stanford, 2017. [Online]. Available: http://cs229.stanford.edu/proj2017/final-reports/5231214.pdf. [Accessed: 20-Mar-2020].

[4] P. Rossotti, "NBA Enhanced Box Score and Standings (2012 - 2018)," Kaggle, 08-Nov-2018. [Online]. Available: https://www.kaggle.com/pablote/nba-enhanced-stats. [Accessed: 21-Mar-2020].

[5] "Basic regression: Predict fuel efficiency : TensorFlow Core," TensorFlow, 2017. [Online]. Available: https://www.tensorflow.org/tutorials/keras/regression. [Accessed: 21-Mar-2020].