Faculty of Mathematics & Science
Department of Computer Science

COSC 4P80 Project: At-Bat Predictor

Prepared By:
Alex Duclos (6738884)
Ibrahim Hashmi (6352926)
Monty Oshinov (6759286)

Instructor:
D. Bockus

Brock
April, 2023

Abstract

This project aims to develop a neural network for predicting the Weighted On-Base Average (wOBA) value for at-bats in Major League Baseball (MLB). The model makes use of a comprehensive dataset consisting of data from all at-bats from every MLB season 2010 and 2015, as well as the latest available data from 2022. Additionally, weather information from the previous games is also included to help make more accurate predictions. This data is obtained from Pybaseball, a Python library for accessing baseball data, as well as Weatherapicom, an API for gathering weather data. We explored and assessed two distinct network topologies, one being larger in size while the other smaller. Both networks were trained and tested using the same sets of data, and then evaluated based on their efficiency and accuracy. After careful analysis, the smaller topology was chosen for it's faster training time while still maintaining an acceptable prediction accuracy. Mean squared error (MSE) and loss were used to evaluate the model's performance. The results show the feasibility of using the network for predicting wOBA values in upcoming games, providing potential but limited insights for baseball analytics and decision-making.

Table of Contents

Chapter 1

Introduction and Motivation

Contents

1.1 Motivation

Sports betting is a major global industry grossing billions of dollars, which creates a constant demand for improved predictive software and technologies to assist organizations and bettors. This software can be focused on predicting various aspects of a game. In Major League Baseball, wOBA (Weighted On-Base Average) measures a player or team's overall performance by measuring how often and by what means they reach base. In betting this can be used as a valuable statistic for making informed decisions
Player wOBA can be used to find opportunities for betting arbitrage. For example, if predictive software suggests that a player will have a particularly high wOBA value against a pitcher in an upcoming game, this could be used to bet that the batter will hit a home run in an upcoming game, as well as betting against that player hitting a home run if the wOBA is low.

1.2 Problem Statement

The goal of this project is to create and train an Artificial Neural Network model capable of predicting a batter's wOBA against a pitcher based on a set of player statistics and weather data. This should provide usable data to make better decisions in sports betting.

1.3 Methodology

1.3.1 Training Data

The **Pybaseball** library was used to collect data pertaining to the players and teams themselves. The following parameters for this data were used for training the model:

- `batter` (unique player ID)
- `pitcher` (unique player ID)
- `BABIP_value` (Batting Average on Balls in Play)
  - measurement of how often batted balls result in hits, excluding home runs
- `ISO_value` (Isolated Power)
  - measure of batter's raw power, indicates how often a player hits for extra bases
- `wOBA_value` (Weighted On-Base Average)
  - on-base percentage that accounts for how a player reached base

The outcomes of at-bats can be very dependent on weather conditions during the games. It's important to understand that baseball is a game of inches when it is played at the level of Major League Baseball, and thus it is important to take these factors into account as even the slightest change in conditions can have a significant effect on the play. Thus, we chose to collect the following weather data using **weatherapi.com**:

- `game_temp` (air temperature during the game)
  - higher temperatures can affect ball speed and pitcher grip
- `game_wind` (wind speed)
  - higher wind speeds can significantly change the trajectory of pitches
- `game_humidity` (humidity during the game)
  - pitches and throws are faster on humid days than on dryer days

Initially, our dataset consisted of a massive 10 million pitches across 20 seasons in the MLB. To improve efficiency, we decided to reduce the dataset to 1.5 million at-bats from the years 2010 to 2022. This dataset was still problematic however, as it included the years where "juiced balls" were in effect. This refers to when the MLB used altered baseballs which were designed to bounce off the bat at higher speeds, resulting in a 3-4% increase in total home runs across the league[1]. Seeing as this could greatly skew the data, we chose to reduce our dataset even further to only include at-bats from 2010-2015, as well as 2022, while omitting the 2016-2021 seasons where both "juiced ball" and shortened seasons were in effect. This effectively reduced our dataset down to just over 1 million at-bats across seven different seasons.

Chapter 2

Stats

Contents

This chapter will provide a brief overview of baseball statistics used, and what they represent. It will further discuss their value for the model.

2.1 WOBA

WOBA is a modern stat that aims to provide contextual value for a plate appearance. It weights different outcomes at different weight.

$$\text{wOBA} = \frac{0.691 \times \text{uBB} + 0.722 \times \text{HBP} + 0.884 \times \text{1B} + 1.257 \times \text{2B} + 1.593 \times \text{3B} + 2.058 \times \text{HR}}{\text{AB} + \text{BB} - \text{IBB} + \text{SF} + \text{HBP}}$$

Here we can see that a home run is valued at 2.058 and a double is valued at 1.257, despite a home run accounting for twice as many total bases.

2.2 ISO

ISO is similar to wOBA where it values total bases, but the formula is different. This statistic shows the total bases acquired per hit. So a higher ISO would be indicative of a slugging power hitter, but said power hitter might walk a lot.

$$\text{ISO} = \frac{\text{TB} - \text{H}}{AB}$$

Where TB is the total bases, 4*HR, 3*Triple, 2*Double, and 1*Single, and hits is how many times the player reached on a hit.

2.3 BABIP

Baseball is an inherently random game, and BABIP is an attempt to work towards normalizing that. BABIP is the batting average on only balls in play, so it normalizes to remove home runs and strikeouts.

A high BABIP is indicative of hitting the ball hard or getting lucky, while a low BABIP is the opposite. Over a large enough sample size however, usually 600+ PAs, we can generally figure this out.

2.4 Statcast

Modern baseball allows for some more advanced statistics, including exit velocity and launch angle of the baseball.

These statistics are only available for use post-2015, which results in a training problem as MLB changed the balls up for a few years in the late 2010s resulting in an higher than expected home run rate for balls with a lower EV/LA.

For this reason we did not use statcast statistics for our sample set, however, a future trial for this task could make good use of these statistics.

Chapter 3

Methodology and Performance

Contents

This chapter discusses the methodologies used to implement the Neural Network, including it's tuning parameters and dependencies, along with an overview of performance based on its settings.

3.1 Architecture

The network we developed for our project was a Feed-Forward Neural Network composed of multiple layers. In total, it consisted of seven layers, with five of them functioning as hidden layers. We conducted experiments with two different network topologies: the first one had a structure of `7x64x128x128x64x32x1`, while the second one had a thinner configuration of `7x16x32x32x16x8x1`. It's worth mentioning that both network configurations were trained for the same number of epochs to ensure a fair comparison.

During the training process, we closely monitored the performance of both networks, focusing on the mean squared error (MSE) metric, which provides insights into the quality of predictions. We observed that at the end of training, both networks achieved a remarkably low MSE value, approximately ranging from `0.2` to `0.25`. This indicates a high level of accuracy in predicting network outcomes for both topologies.

Despite the similar MSE values, our analysis revealed that the smaller network outperformed the larger one in terms of training time. The smaller topology not only required less time to train but also demonstrated comparable accuracy to the larger topology. Therefore, taking into account the MSE values and the training efficiency, we concluded that the smaller topology emerged as the superior network configuration for our project.

3.1.1 Tuning parameters

Each neural network was trained using the same hyper-parameters, other than their layer sizes:

- Number of Layers: Five hidden layers were used with varying input sizes.
- Activation function: The activation function used was a Sigmoid function.
- Initialization weights Weights were initialized randomly using a Xavier distribution between .
- Learning rate: The learning rate was set to 0.001
- Epochs The Neural Networks were trained over 500 epochs
- Training and Test sets The training and test sets were split randomly so that the model was trained on 80 percent of the data and tested on the remaining 20 percent. The batch size for testing was only 128 data points.

3.1.2 Dependencies

Several dependencies were used for this project. The project required a Machine Learning framework, and in this class' presentations several different frameworks were discussed, including TensorFlow, Pytorch, and Microsoft Cognitive Toolkit. It was decided by our group that TensorFlow was the best option due to familiarity with the system and its extensive community.

The project also required the use of Pybaseball, a set of baseball data scraping tools, and weatherapi.com for obtaining weather and player data. Pandas and BeautifulSoup4, were used for data collection.

3.2 Network Efficiency and Results

As previously mentioned, during the initial stages of training, our network was designed with a larger topology of 7x64x128x128x64x32x1. However, we encountered a notable challenge due to the size of our training data, which resulted in considerably longer training times. On average, each epoch took between 40 to 60 seconds to complete when trained for 500 epochs. Consequently, we had to patiently wait before being able to thoroughly analyze the network's predictions and evaluate its accuracy. Often, this compelled us to let the network train overnight and evaluate its predictions the following morning.

To address this issue, we decided to make adjustments to the network architecture, opting for a smaller topology of `7x16x32x32x16x8x1`. This modification significantly reduced the training time, shaving off approximately 10-15 seconds per epoch. Both network configurations were trained for an equal number of epochs, specifically 500, to ensure a fair comparison.

Throughout the training process, we observed that as the number of epochs increased, the training speed of both network configurations gradually slowed down. However, it was evident that the smaller network consistently outperformed its larger counterpart in

terms of efficiency. Despite its faster training time, the smaller network maintained a similar level of accuracy when predicting outcomes on the testing set.

Ultimately, both network configurations exhibited a final Mean Squared Error (MSE) value of approximately 0.2 to 0.25, indicating a commendably low level of error at the end of training. Taking into account the training efficiency and the comparable accuracy achieved, or close to it, the smaller topology was deemed the more efficient and practical choice. Therefore, it was determined to be the preferred network.

After thoroughly training and testing the data with an 80/20 training-testing split, it was found that the network demonstrated an overall accuracy of approximately 80%. However, it is important to note that there are certain observations which could affect one's interpretation of these results. For instance, one observation made is that the network frequently predicted a wOBA value of 0.00, which matched the vast majority of expected outcomes in the testing set. This trend could potentially be inflating the network's accuracy to be more accurate than it really is.

Another important thing to note is that when it comes to expected output values over 0.00, there are some noticeable inconsistencies with the network's predictions. For instance, the network tends to over-predict wOBA values of 1.00, particularly for expected outputs of 3.00. These findings inidicate that additional refinement of the dataset could further enhance the network's accuracy.

Chapter 4

Conclusions

Contents

4.1 Performance

Two topologies featuring different layer sizes were tested for the model. The two network configurations of sizes `7x64x128x128x64x32x1` and 7x16x32x32x16x8x1 were trained, and found to yield similar results after testing. It was determined that the increased computing power and time required for training the larger model did not justify it's use, as it took ten to fifteen seconds longer per epoch to back-propagate. Hence, it was determined that employing a larger topology was unnecessary, if not counter-productive.

4.2 Feed Forward Network

The 7-layer Feed-Forward Neural Network was found to be fairly accurate at predicting wOBA values for potential at-bats involving MLB batters and pitchers. Despite our dataset consisting of data from over one million historical at-bats, it is important to acknowledge that the model's reported accuracy of approximately 80% might be inflated or not entirely indicative of its true performance. Several factors contribute to this uncertainty, including its tendency to over-predict values of 0.00 and 1.00. However, even with this limitation, the model was able to achieve a reasonably high accuracy, making the provided data valuable for assisting with sports-betting. The margin of error, ranging from 0.2 to 0.25 mean squared error, was found to be relatively low, and thus it was determined that the network could be used to assist with sports-betting

4.3 Bibliography

[1] The history of juiced balls and how today's home run binge fits