# Predicting Baseball Pitches with Recurrent Neural Networks

**Zac Sims**
zsims@uoregon.edu

**Delaney Carleton**
dcarlet2@uoregon.edu

## Abstract

In this paper, we propose that with the context of the last three pitches a pitcher has thrown in an at-bat and the context of the game situation before the pitch, we are able to adequately predict whether the pitch is a fastball or not. Specifically, we show that our Recurrent Neural Network model matches the performance of similar models that rely on more data.

## 1 Introduction

Predicting the next type of pitch a pitcher will throw in a given at-bat is valuable for understanding the strategies pitchers use to vary their pitch sequences. Most pitchers use four or five different pitch types to throw, with fastball being the most fundamental pitch and the most common pitch, with most pitchers throwing a fastball just over fifty percent of the time. The distribution of pitch types for the pitcher our models were trained on is shown in the chart below:

Justin Verlander Pitch Types

| | |
|---|---|
| Fastball: | 58.75% |
| Non-Fastball: | 41.25% |

The three other pitch types Justin Verlander throws (changeup, curveball, slider) are slower pitches with horizontal and vertical movement, justifiably called off-speed pitches. Pitchers will try to use different combinations of fastballs and off-speed pitches to fool batters. We will attempt to learn a representation of these combinations and combine that representation with information the pitcher knows about the situation of the game before the pitch to predict if the pitch-to-be-thrown will be fastball.

## 2 Background

Our approach is similar to Yifan PI, Stanford University, who in 2018 utilized a Recurrent Neural Network to predict fastball/non-fastball pitch type on the next pitch. In addition to slight differences in data features, the key difference in our approach is that we seek to learn a model that is both independent of the batter and more dependent on the pitcher himself. We do this by not attempting to learn an embedding of the pitcher-batter matchup, but instead limiting the dataset to a specific pitcher. We will also be using an RNN to encode the prior three pitches in an at-bat and run it through a dense layer, whereas Yifan uses each game as a sequence and predicts the pitch with a dense layer at each timestep in the game.

## 3 Methods

We collect the raw pitch data from MLB Advanced Media for Verlander, The top pitcher in terms of number of pitches thrown that have been tracked by a radar. The size of the datasets are below below:

Size of Training, Validation, and Test Datasets

| | |
|---|---|
| Training: | 23,685 |
| Validation: | 7,895 |
| Test: | 7,896 |

The pitches will each be represented by a length 25 vector that consists of the pitch-velocity, the coordinates of the pitch as it crossed the plate, the amount of movement in the horizontal and vertical directions, and one-hot-encodings of the pitch type and the outcome of the pitch. The game situation prior to the pitch-to-be-predicted will be represented by a length 15 vector consisting of one-hot-encodings of the presence of a runner on first, second, or third base,whether the pitcher is playing at-home or away, whether the batter is left or right-handed, the number of outs, the number of strikes, and the number of balls.

## 4 Experiments

To train an RNN to learn prior pitch representations, we define the problem as so:

$$\Pr[p_t = Fastball|\left\{p_{t-3}, p_{t-2}, p_{p_t-1}\right\}, g_t]$$

where $p_t$ is the pitch-to-be-predicted, $\{p_{t-3}, p_{t-2}, p_{p_t-1}\}$ are the prior pitches to be encoded, and $g_t$ is the current game situation. Each sequence fed into the RNN is an at-bat, at every fourth timestep in the at-bat, the output of the hidden layer at $p_{t-1}$ is concatenated with the vector representing the game situation at timestep $t$, and fed through a dense layer with Softmax output to obtain the prediction for $p_t$. To compare with this model, a Decision Tree, Support Vector Machine, Linear Regression models where also trained to compare the results obtained by the RNN.

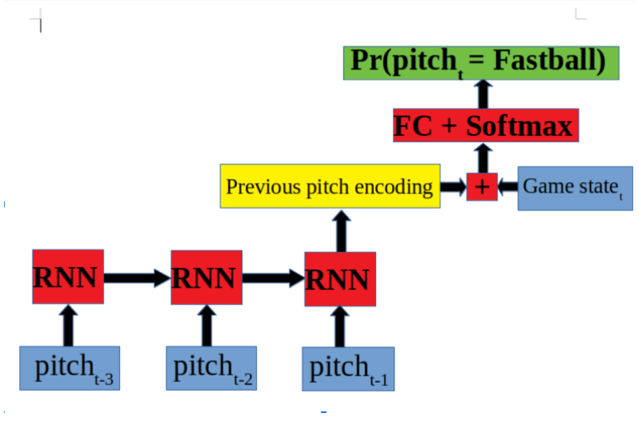

**Figure 1:** Model Computational Graph

## 5 Results

The RNN was trained using LSTM cells with one layer and 120 hidden units, The loss function used was Negative Log Likelihood, and the optimizer was ADAM with (0.9,0.99) and a learning rate of 0.0001. after 100 epochs, the training, validation, and test results are as shown below:

RNN results

| Dataset | Accuracy | % Above Naive Guess |
|---|---|---|
| Training: | 62.9% | 5.00% |
| Validation: | 62.1% | 3.77% |
| Test: | 62.8% | 4.55% |

The test results of the other models, for comparison, are as shown below:

Comparison Model Test Results

| Model | Accuracy | % Above Naive Guess |
|---|---|---|
| Lin. Regression: | 63.9% | 5.40% |
| SVM: | 61.5% | 3.04% |
| Decision Tree: | 60.5% | 2.05% |

## 6 Conclusion

The RNN model is able to accurately predict whether the next pitch will be a fastball or not, but is unable to outperform a simple linear regression model. This implies that the non-linearities in the RNN model do not improve performance and hence the design can be much further improved. One large issue with our approach that can be changed is to increase the length of the sequence considered for the previous pitch encoding. While defining the sequence of pitches as an at-bat may provide more accurate context for the pitching strategy for a specific batter, it may be better to have the sequence be defined as a game, or to include the entire at-bat as context to provide better results for shorter at-bats.

It is of interest to note that our model slightly outperforms the accuracy reported by Yifan in his/her RNN model, with a test accuracy of 66.2%, 3.5% better than the naive guess. The size of the test set Yifan used consisted of 128,000 pitches, which is a much larger dataset than the one we use since we do not combine all pitcher data. To this point, we have showed that it may be more beneficial to train models on specific player datasets to learn more accurate representations of unique pitching strategies.

Further improvements to this model would include modifying the RNN to predict the next pitch at every time step in the at-bat, rather than relying on a fixed amount of prior pitches for context. Another improvement would be to add context for the tendencies of the batter. Some players are more prone to swing-and-miss on certain pitch types due to specific traits.

## Code

https://github.com/zacharyssims/CIS472_project

## References

Yifan PI, 2018, *Predict Next Baseball Pitch Type using RNN* https://cs230.stanford.edu/projects_spring_2018/reports/8290890.pdf

MLB Advanced Media https://baseballsavant.mlb.com/statcast_search