EECS598 Project: Pitch Tunneling

Eva Kraegeloh, Matthew Winchester

1 Overview

The goal of this project is to predict the efficiency of a pitch based on how well tunneled it is. The efficiency or success of a pitch is measured by its Run Value, which we will try to predict with our models.

2 Pitch tunneling metric and feature selection

Drawing inspiration from [HPL17], we first transform six predictors from the pitch-tunneling database to create two new quantities for each pitch: the "break:tunnel (b:t) ratio" and "release:tunnel (r:t) ratio". The b:t ratio of a pitch is the ratio of the pitch's break differential to the tunnel differential, while the r:t ratio compares the release differential to the tunnel differential. Differentials for each pitch are calculated using the $\boldsymbol{\ell}^2$-norm with respect to the average pitch trajectory. Since pitch tunneling suggests that the most effective pitches are those that look similar at the tunneling point but diverge at the base, we expect that pitches with a large b:t ratio and small r:t ratio will produce the smallest Run Value. In the following sections we will use a variety of algorithms to test if this claim is true. We also divide the data into the four combinations of pitcher/batter handedness and either fastballs/sinkers or all other pitch types as suggested in the project description. All results quoted in this summary are using the left handed pitcher, left handed batter, fastballs/sinkers subset, but the full set of results can be found in the following summary tables.

3 Linear Regression

The first model we use is a simple linear regression which minimizes the least squares loss function $\sum_{i=1}^{n} r_i^2$, where r_i is the ith residual between the predicted Run Value and the measured Run Value of the ith pitch from the test dataset. The solution to the minimization problem $\|Ax - b\|^2$ is given by $x = A^+ b$, where the matrix A contains columns of the b:t and r:t ratios and the vector b contains the corresponding Run Values.

After performing the linear regression on a subset of the training data, the estimated coefficient of the b:t ratio is 0.007 and the estimated coefficient of the r:t ratio is -0.006. This simple model indicates that increasing the b:t ratio of a pitch will result in a higher Run Value, while increasing the r:t ratio will result in a lower Run Value as expected. The magnitude of these coefficients also shows us that the b:t ratio has a slightly greater impact on the Run Value of the pitch compared to the r:t ratio. If we use this model to predict the Run Values of the test data, we get a mean squared error (MSE) of 0.0423.

4 Neural Network

After experimenting with various neural network topologies, we found that we could achieve a reasonable regression performance and run time with a single hidden layer consisting of 8 neurons using the sigmoid activation function $\sigma(x)$. We trained the network with both the b:t and r:t ratios and used mse as the loss function. Using this network, if we are given both ratios of a pitch we can predict the Run Value (RV) of the pitch using the equation:

$$RV = \sum_{i=1}^{8} w_i \sigma\Big(\sum_{j=1}^{8} a_i \sigma(\boldsymbol{x}^T \boldsymbol{b_j} + c_j) + d_i \Big), \tag{1}$$

where $\boldsymbol{x}$ is the column vector containing the two ratios of the pitch, and w_i, a_i, $\boldsymbol{b_i}$, c_i, and d_i are all weights that have been learned from the training data. Once we have used the network to estimate RV for each test pitch we can compare to the actual RVs and compute the mse as before. Using this network and 500 learning epochs, we are able to get an MSE of 0.0421.

5 k-Nearest Neighbors

We also implement a k-nearest neighbors regression algorithm, which predicts the Run Value of a pitch in the test dataset by finding the k nearest neighbors in the training dataset and taking the weighted average of the neighbors' Run Values. We used both the $\boldsymbol{\ell}^2$-norm and $\boldsymbol{\ell}^1$-norm to determine the nearest neighbors, but the $\boldsymbol{\ell}^2$-norm produced a model with a lower MSE. We found the optimum value of k by running the algorithm over a range of k and selecting the model that produced the smallest MSE. The minimum MSE was 0.0590 and occurred at $k = 16$.

6 Evaluation metrics

We compared our algorithms using MSE and a function that evaluates the "accuracy" of our algorithms, i.e. what fraction of the test data was predicted correctly. Since the RunValues are continuous, we need to decide what we define as "correctly predicted". It is unclear to us how precise we need to predict the RunValues. Intuitively one might think that predicting the correct integer is sufficient, in which case we get accuracies of about 97% for both the linear regression and the neural net, and about 96% for the nearest neighbor classification. However, the vast majority of the data has a RunValue that will round to 0 if we round to integers, so it seems that it is potentially necessary to predict more precisely. If we want to predict at least the first decimal point correcly, our accuracies drop to around 30%.
MSE losses for linear regression and the neural net are consistently around 0.04 and a little higher, around 0.05, for the nearest neighbor classification.
A DataFrame with all metric results can be found at the end of the Julia notebook.

7 Timing

Timing information for the linear regression, neural network, and k-nearest neighbor algorithms are shown in the following table. We were able to improve the speed of all algorithms by reducing the number of predictors from 36 to 2 (b:t and r:t ratios). k-Nearest Neighbors appears to be the slowest algorithm since we need to calculate the distance from every test pitch to every train pitch (so runtime scales as $\mathcal{O}(n^2)$). Linear Regression is by far the quickest algorithm, especially for large datasets.
We use both the fastball/sinker - both left-handed dataset for timing (because it is the dataset we use for all other numbers) as well as the fastball/sinker - both right-handed dataset since it is the largest dataset.

algorithm	Linear Regression	Neural Network	k-Nearest Neighbors
LL total time (s)	0.016	42.97	53.28
RR total time (s)	0.02	369.43	2273.29

8 Training Issues

The most prominent issue with data from the pitch-tunneling database was that a fraction of the entries had missing values of the Run Value (around 1%). We chose to ignore these pitches since they only made up a small portion of the full dataset, but we could potentially improve the accuracy of our algorithms by implementing an imputation method to the data processing stage.
As an alternative feature selection, we explored the difference of the current trajectory to the trajectory of

the previously thrown pitch, but all algorithms performed similarly to the "averaged" features. Since the sequencing of the data requires a lot more code and we "lose" a lot of data due to the fact that it only makes sense to compare pitches that are close in time, we decided against this method.

References

[HPL17] Jonathan Judge Harry Pavlidis and Jeff Long. Prospectus feature: Introducing pitch tunnels, 2017.