Predicting Pitch Outcome in Major League Baseball Games

Edison Huang [*1] **Michael Chen** [*1] **Yin-Hsun Huang** [*1]

Abstract

In this project, we compare the importance of various features on predicting the umpire call of a pitch in baseball games using machine learning methods. In particular, we evaluate models such as logistic regression, gradient boosting machine, random forest and neural network in terms of categorical cross-entropy and accuracy. We also implement recurrent neural network utilizing sequential features, which achieves the best accuracy.

1. Introduction

Watching baseball game is a popular leisure in the United State. Americans spend more than $30 million on baseball gambling along. Therefore, getting small improvements in the prediction of any aspect of the game is valuable. In this paper, we try to predict the umpire call outcome of a pitch by using game statistics and features known beforehand. There are three possible outcomes: 'ball', 'strike' and 'hit'.

As any baseball fan can tell, 'balls' and 'strikes' are the majority results of a pitch. Even when the batter hits the ball, there are a lot of factors need to be considered besides the physical capability of players. Thus, we expect this problem to be difficult due to the nature of the sports.

2. Dataset and Features

The dataset PitchF/X [2] is a pitch tracking system, created by Sportvision, to track the velocity, movement, release point, spin, and pitch location for every pitch thrown (Fangraphs, 2018).

In particular, the data is divided into 3 groups per year: pre-season, regular-season, post-season. Although the features are identical in all of them, we won't be using any of preseason data in this project.

[1]Computer Science Department, New York University, New York, NYU. Correspondence to: Edison Huang <tyh273@nyu.edu>, Michael Chen <yc1483@nyu.edu>, Yin-Hsun Huang <yhh303@nyu.edu>.

[2]Provided by our advisor, Mr.David Frohardt-Lane.

2.1. Target Label

"umpcall" is chosen as the target to predict. It stands for umpire calls, which means the decision of the outcome of a given pitch made by umpires.

It is a categorical variable consisting of 3 classes: "B" for balls, "S" for strikes, and "X" for hits. In our dataset, there is about 45.8% of "S", 36% of "B", and only 18.2% of "X".

2.2. Feature Selection

With many features given, we first identify what features are potentially useful and suited for our purpose.

Since the final goal is to predict the outcome of a pitch, we first remove any feature that could only be gathered after the pitch is thrown such as "start_speed", "pitch_type" ... etc. This principal keeps us from using all precise statistical features provided by PitchF/X and greatly limits our selections.

After the preliminary feature selection, our candidates are as follows:

FEATURES	TYPE	DESCRIPTION
DATE	DATE	WHEN IT TAKES PLACE
STADIUM	STRING	VENUE
INNING	INTEGER $\in [1, 19]$	
SIDE	CATEGORICAL	
PITCHER*	STRING	PITCHER ID
BATTER*	STRING	BATTER ID
PITCH_COUNT	INTEGER	PITCH COUNT
BALLS	INTEGER $\in [0, 4]$	CURRENT BALLS COUNT
STRIKES	INTEGER $\in [0, 2]$	CURRENT STRIKES COUNT
ON_1B	STRING	PLAYER ON 1ST BASE
ON_2B	STRING	PLAYER ON 2ND BASE
ON_3B	STRING	PLAYER ON 3RD BASE

Table 1. Details of input features.
*For missing data, we categorize them as "unknown"

2.3. Exploratory Data Analysis

After deciding the input features, we started examining whether there's any discrepancy in "umpcall" among different years in Figure 1. Interestingly, the percentage of hits, balls and strikes are consistent throughout. However, the total number of data we have for 2013 is significantly less than that of the other years. Thus, to keep the data uniform, we exclude data from the year 2013.

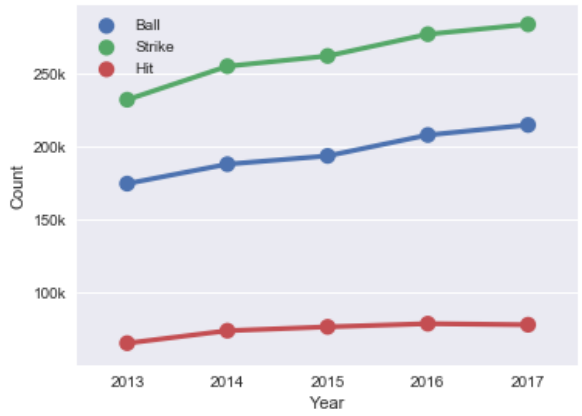

Figure 1. Number of balls, strikes, hits over the years

Next, suppose we combine "balls" and "strikes" as single categorical variable, how would it affect the outcome of umpire call? In Figure 2, we plotted the combined variable vs. the ratio of each outcome within their respective category. If we pay close attention to group 0-0 and 3-0, we notice that "hit" is very low compare to that of the other groups and that's probably due to the nature of baseball, which a batter tends to be more passive when the ball count is 0 or 3.

Therefore, with this finding, we are certain that the combined variable has a great potential to improve our model.

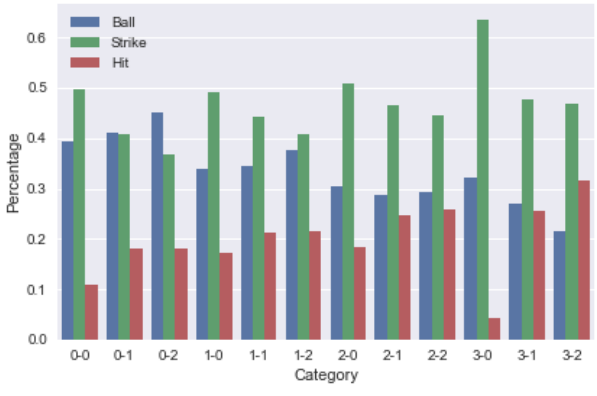

Figure 2. Percentage of balls, strikes and hits within each ball-strike combination

2.4. Preprocessing

2.4.1. Feature Generation

Based on data in 2015 and 2016 to analyze and generate features.

Hit Ratio and Strike Ratio

A given batter B's hit ratio is calculated as follows

$$\begin{aligned} E_B &= \text{Entries with B as batter} \\ E_{hit} &= \text{Entries with "umpcall" = "X"} \\ H_{ratio}(B) &= Logit\left(\frac{Count(E_B \cap E_{hit})}{Count(E_B)}\right) \end{aligned}$$

And a given pitcher P's strike ratio is calculated as follows

$$\begin{aligned} E_P &= \text{Entries with P as pitcher} \\ E_{strike} &= \text{Entries with "umpcall" = "S"} \\ S_{ratio}(P) &= Logit\left(\frac{Count(E_P \cap E_{strike})}{Count(E_P)}\right) \end{aligned}$$

We join these results to the original data as 2 new features. Using "batter" to bind the corresponding batter's hit ratio and "pitcher" for corresponding strike ratio.

2.4.2. Feature Transformation

Many models can not handle categorical features directly such as logistic regression, and neural network. Hence, transforming the data into another form to proceed is the first step. In addition, discrete numeric features are also transformed as categorical such as "balls" and "strikes" for better result.

One-Hot Encoding Features "pitcher", "batter", "balls", "strikes", "on_1b", "on_2b", "on_3b" are all applied with one-hot encoding before feeding into any model. "strikes" and "balls" are merged into single category before encoding, so do "on_1b", "on_2b", "on_3b".

Problem with one-hot encoding is that the large feature dimension after encoding. Specifically, "pitcher" and "batter" both have roughly 2000 unique categories which can increase the feature dimension from $20 \sim 30$ to almost 3000. This enormous number of features has greatly increased the difficulty of training models.

NN Embedding Features "pitcher" and "batter" are treated as word in (Mikolov et al., 2013). We use a simple look-up table to store the vector representation of each entity of "pitcher" and "batter". This representation is further refined through the process of back-propagation.

Feature Simplification "on_1b", "on_2b", "on_3b" features are originally presented in batter's id. This level of detail is not necessary for our application and could only increase feature dimension. Thus, we decided to replace id with a Boolean variable representing if someone is on that base or not. We then combined the simplified "on_1b",

"on_2b", and "on_3b" before the encoding process mentioned earlier.

Forming Sequence In order to setup the training data for the RNN model, we have to group the training data to form sequences. A natural choice is to group them game by game. To enable batch training, we also pad the short sequence so that each batch training input has a shape $(\text{batch\_size}, \text{max\_seq\_len}, \text{feature\_dim})$.

3. Methods

Our problem is a multi-class classification problem. We compare a variety of classification models applicable to multi-class scenario and rank them using cross-entropy and accuracy.

3.1. Baseline

We first group all $x \in \mathcal{X}$ in the input space into a disjoint union set, $G_1, \ldots, G_m$. Then each training data (x_i, y_i) belongs to some group G_j such that $x_i \in G_j$. To predict an input x, the predicting function first decides the group it belongs to, say G_k, and output $(\frac{n_B}{\|G_k\|}, \frac{n_S}{\|G_k\|}, \frac{n_X}{\|G_k\|})$ as the predicting distribution. n_B is the number the training data in group G_k that their y is "balls". We use two grouping schemes as our baseline algorithms.

Baseline 1. All x belongs to one group.

Baseline 2. All x with the same "balls" and "strikes" are in the same group.

3.2. Models

Logistic Regression(Peng et al., 2002) is a statistical model for binary classification problem. We use one-vs-all technique to apply the binary classifier on multi-class setting.

Gradient Boosting Machine(Friedman, 2001) is a sequential ensemble method to create a prediction model on top of regression tree in a stage-wise fashion.

Random Forest(Breiman, 2001) is an ensemble model by taking bootstrap samples and randomizing sub-sampling to create a multitude of decision trees at training time and predicting with the mode of classes from each tree.

Neural Network consists of five layers of fully connected layer with SELU(Klambauer et al., 2017) activation function. Categorical variable are first processed with a simple lookup table embedding layer.

Recurrent Neural Network consists of five layers of fully connected layer with SELU activation function + one LSTM(Hochreiter & Schmidhuber, 1997) layer.

3.3. Performance Metrics

Categorical Cross-Entropy is the first performance metrics we use to evaluate the performance of our models. Given a prediction function $f : \mathcal{X} \to \mathcal{D}_{\mathcal{A}}$ that takes a vector of features $x \in \mathcal{X}$ to a probability distribution over 'umpcall', $\mathcal{D}_{\mathcal{U}}$. Let training data be $(x_1, y_1), \ldots, (x_n, y_n)$. The cross-entropy of this prediction function is defined as follows:

$$\ell(f) = \frac{1}{n} \sum_{i=1}^{n} -\log \mathbb{P}[f(x_i) = y_i]$$

Accuracy, also known as the $0/1$-loss, is the metric we value the most at the end. We want our model to predict the "umpcall" correctly as often as possible. The accuracy of a prediction function is:

$$acc(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[f(x_i) = y_i]$$

4. Experiments and Results

4.1. Experiment Setup

We train our models on over 2 million pitches in both regular season and post season from 2014 to 2016 then evaluate them on the regular season and post season on 2017. 10% of the training data are randomly sampled as the validation set for tuning hyper-parameter and performing early stop in neural network-based models.

4.2. Input Features

We have experiment with different combinations of input features. The common feature is "Balls_Strikes", which is the same as in baseline models. The goal in this stage is to see if adding any additional feature help making the model improve performance.

MODEL	BALLS_STRIKES	+ON_BASE	+INNING
BASELINE1	45.907/1.0314	—	—
BASELINE2	46.639/1.0160	—	—
LR	46.633/1.0159	46.482/1.0157	46.640/1.0157
GBM	46.633/1.0159	46.638/1.0153	**46.676/1.0156**
RF	46.633/1.0159	46.638/1.0153	46.667/1.0154
NN	46.448/1.0161	46.623/1.0155	46.668/1.0156
RNN	46.725/1.0150	46.683/1.0147	46.675/1.0151

Table 2. MODEL PERFORMANCE WITH SELECTED FEATURES IN THE FORM OF (ACCURACY/ENTROPY)

4.3. Experiment Verdict

According to Table 4.2, "pitcher" and "batter" is the best feature to be included and boost the accuracy of almost all

MODEL	+PITCHER, BATTER	+PITCH_COUNT
BASELINE1	—	—
BASELINE2	—	—
LR	**47.104/1.0116**	46.625/1.0157
GBM	46.633/1.0150	46.602/1.0155
RF	**46.732/1.0155**	46.546/1.0159
NN	**47.056/1.0108**	46.591/1.0160
RNN	**47.128/1.0115**	46.659/1.0145

Table 3. (CONT.) MODEL PERFORMANCE WITH SELECTED FEATURES IN THE FORM OF (ACCURACY/ENTROPY)

Figure 3. This figure show the performance of models trained using the best combination of variable of their own. Models with randomness in their training process are repeated several times.

models. Both one-hot encoding and embedding work well and the choice comes down to the model. Here are some observation of the results:

- RNN generally does the best across the board.
- The difference among different models is minuscule when only using "Balls_Strikes" as input.
- "Pitcher, Batter" improves the performance the most.
- Logistic Regression performs better than expected considering it is a lot less complex than the neural network model and it only takes a fraction of the time to train.

5. Discussion

5.1. Theoretical Limit

To find the best prediction a model can generate, we train a neural network model with all data in the PitchF/X dataset. How much performance boost can we obtain? A quick experiment shows that this model can achieve 69% accuracy and 0.687 categorical entropy, which is a big improvement compared to the result in Section 4.2.

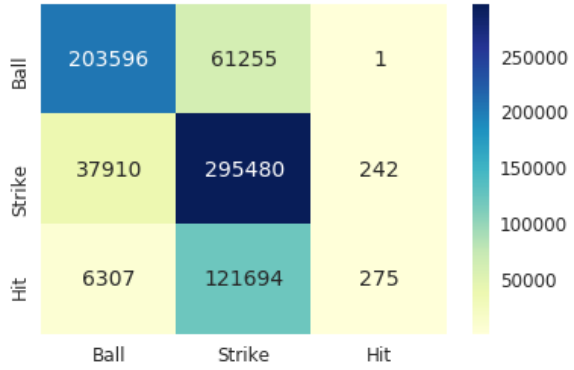

Figure 4. The confusion matrix of the neural network model trained with all PitchF/X data. The y-axis indicates the original label and the x-axis indicates the predicted label.

5.2. Why Does It Perform Better?

Let's compare this model with the best result from Logistic Regression in Section 4.2.

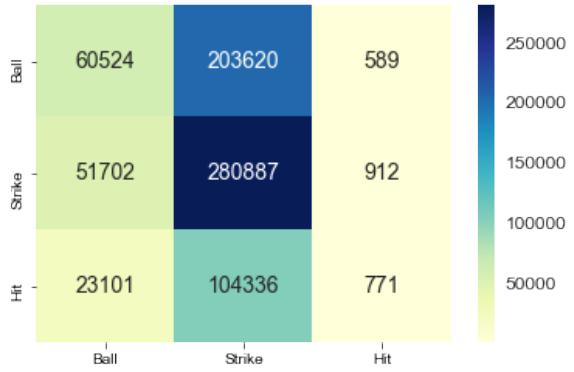

Figure 5. The confusion matrix of the logistic regression model trained without post-pitch data. The y-axis indicates the original label and the x-axis indicates the predicted label.

One can quickly observe that both models have decent accuracy in predicting "strikes". However, the major difference is in predicting "balls". The model with post-pitch feature not only is good at predicting "stirkes" but also "'balls". That said, we can conclude that the accuracy boost comes down to model's better capability at marking the correct "balls".

6. Conclusion and Future Work

This problem is hard due to the nature of baseball. We do see some improvements in using more sophisticated model and the RNN model performs the best. The RNN can utilize

information across one sequence. Following this method, one can try using other sequence-based model, e.g. Markov chain.

We only considers conditional probability models without latent variable. One extension would be trying to model the problem with a more sophisticated probability model that encodes human understanding of the sport into it. Another direction is to gather more data that is equally accessible as the current dataset.

One can analyze the learned embedding of pitchers and batters to see if it entails some of the properties of the players. Furthermore, how to make high quality embedding is itself an interesting topic.

References

Breiman, Leo. Random forests. *Machine Learning*, 45 (1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Fangraphs. What is pitchf/x?, 2018. URL https://www.fangraphs.com/library/misc/pitch-fx/.

Friedman, Jerome H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29 (5):1189–1232, 2001. ISSN 00905364. URL http://www.jstor.org/stable/2699986.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-Normalizing Neural Networks. *ArXiv e-prints*, June 2017.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

Peng, Chao-Ying Joanne, Lee, Kuk Lida, and Ingersoll, Gary M. An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96(1):3–14, 2002. doi: 10.1080/00220670209598786. URL https://doi.org/10.1080/00220670209598786.