Neural Net for Classifying MLB Pitches

INSY 7130 Data Mining Project

Darryn Pasqua

12/1/2024

Problem Description and Motivation

In Major League Baseball (MLB), the ability to accurately classify pitch types is of paramount importance for players, coaches, and analysts. Understanding the type of pitch thrown can significantly influence a batter's strategy, a pitcher's game plan, and a team's defensive alignment. Historically, pitch classification was a manual process, relying on the expertise of scouts and analysts who observe games and categorize pitches based on visual cues and basic statistical measures. Beginning in 2006, MLB began developing their own tools for tracking and classifying pitches in real time. I wanted to tackle this project because of its practical significance in sports analytics and the opportunity it presents to apply advanced data mining techniques. Prior to developing the model for this project, I did no prior research on MLBs methods. The literature review contained within this report was done after developing my own model for comparison.

Data mining approaches, particularly deep learning models, are appropriate for this problem due to their ability to handle large datasets and capture complex, non-linear relationships among features. Neural networks, in particular, are well-suited for classification tasks where the input data is high-dimensional and intricate patterns exist. With the advent of sophisticated tracking technologies like PITCHf/x, Statcast, and Hawk-Eye, a wealth of detailed pitch data is now available. This data includes measurements of pitch speed, spin rate, movement, and trajectory, among others.

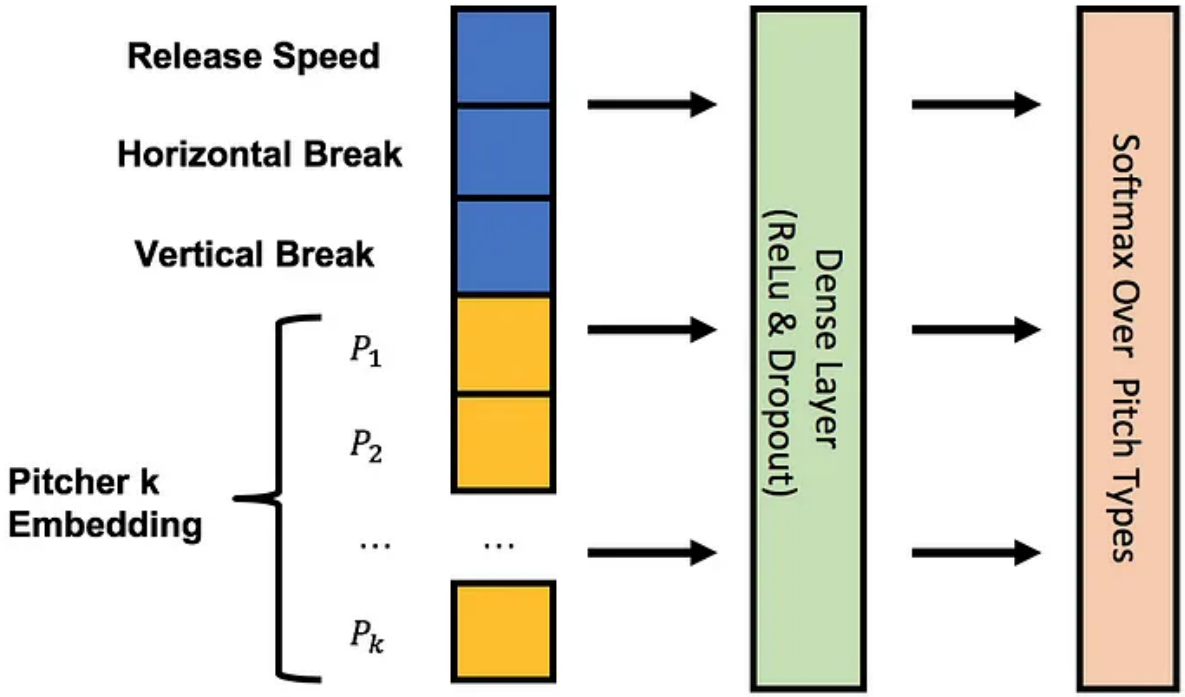
Research Review

Major League Baseball (MLB) has been utilizing automated pitch classification systems since the 2006 postseason, employing neural networks to categorize pitches in real-time (1). Initially, the system used only two neural networks: one for

left-handed pitchers and another for right-handed pitchers. However, this approach proved inadequate due to significant variations in pitching repertoires among different pitchers.

MLB now employs a patented pitch classification neural network software customized for every pitcher in Major League Baseball called Pitch Net (1, 4). This approach addresses the limitations of the initial system by accounting for individual pitcher characteristics. The current system classifies nearly 750,000 pitches each season in real-time. These classifications are displayed on various platforms, including Gameday, At-Bat, MLB.TV, local and national broadcasts, and in-stadium displays. The neural networks generate pitch classification outputs, each representing the likelihood that the input corresponds to a particular type of pitch (e.g., knuckleball, two-seam fastball). The specific system platform may then apply post-processing to the outputs based on additional factors such as pitcher identification, handedness, pitch speeds, favorite pitches, environmental conditions, and game situation.

The current neural networks use various pitch properties as inputs, including pitch speed, spin rate, and vertical/horizontal movement.



(1) - PitchNet Architecture

Researchers have explored various machine learning techniques for pitch classification, each with unique challenges and considerations. Clustering is a common approach for defining pitcher arsenals, offering simple visualizations but limited scalability across league-wide datasets. RNN models have been developed to predict the next pitch but are more complex than traditional classification models, while Gaussian Mixture Models aim to account for movement not generated by spin, addressing misclassifications from spin-based metrics like spin-induced vertical break. However, challenges such as overfitting, class imbalance, contextual factors (e.g., game situations and weather), and the computational complexity of creating accurate real-time models for MLB broadcasts and Statcast systems remain significant hurdles.

The accurate classification and prediction of pitches have significant implications for baseball strategy and player development. Teams can use these insights to improve their batting performance and pitching strategies. Scouting and player development can now utilize robust models to train players to fit certain “models” or reach for outlier criteria that are valuable to an organization.

Project Definition

The dataset for this project consists of all detailed pitch-level data collected during the 2024 MLB season. The data was obtained from Statcast via the pybaseball Python library (https://github.com/jldbc/pybaseball) and was downloaded via a python script I wrote called scrapev3.py. The data was filtered to include the following key pitch characteristics:

- `release_speed`: The velocity of the pitch at the point of release (in mph).
- `release_spin_rate`: The spin rate of the pitch upon release (in rpm).
- `spin_axis`: The axis of rotation of the pitch (in degrees).
- `pfx_x` and `pfx_z`: Horizontal and vertical movement of the pitch due to spin (in inches).
- `plate_x` and `plate_z`: Location of the pitch as it crosses home plate (in feet).
- `vx0`, `vy0`, `vz0`: Initial velocity components of the pitch along the x, y, and z axes (in feet per second).
- `ax`, `ay`, `az`: Acceleration components of the pitch along the x, y, and z axes (in feet per second squared).
- `p_throws`: Pitcher's throwing hand ('L' for left, 'R' for right).
- `sz_top` and `sz_bot`: Vertical boundaries of the batter's strike zone (in feet).
- `api_break_z_with_gravity`: Vertical break of the pitch accounting for gravity (in inches).
- `api_break_x_arm`: Horizontal break of the pitch from the pitcher's arm side (in inches).
- `api_break_x_batter_in`: Horizontal break of the pitch into the batter (in inches).
- `pitch_type`: The type of pitch thrown (e.g., fastball, curveball, slider).

Goals and Objectives

The primary goal of this project is to develop a model for classifying MLB pitch types based on pitch-level data. The specific objectives include:

1. **Data Preprocessing**: Clean and preprocess the data to ensure it is suitable for modeling. This includes handling missing values, encoding categorical variables, and scaling features.

2. **Model Development**: Design and train a neural network optimized for classification of pitch types for both lefties and righties.
3. **Model Evaluation**: Assess the model's performance using appropriate metrics and validate its generalizability with the target of 90% or greater accuracy.
4. **Model Deployment**: Save the trained model and associated preprocessing objects for future use in real-time pitch classification applications.

Constraints and Assumptions

We assume the data is accurate and representative of the 2024 MLB season, with relevant features selected based on domain knowledge. Separate models will be trained for left- and right-handed pitchers to account for differences in mechanics. Computational resources will be optimized to balance efficiency and predictive performance while handling the resource-intensive nature of training deep neural networks.

Data Preparation for Modeling

The dataset was prepared for modeling through filtering, encoding, scaling, and splitting. It was segmented by the pitcher's handedness ('L' for left-handed, 'R' for right-handed) to account for variations in pitch mechanics. Rows with missing values in features or the target variable were removed. The `pitch_type` variable was label-encoded into numerical values, and features were standardized using `StandardScaler` for normalization. Finally, the data was split into training (80%) and validation (20%) sets to evaluate the model on unseen data.

Hyperparameter Tuning with Hyperband

We optimized the neural network using the Hyperband algorithm from the `keras_tuner` library, which efficiently explores hyperparameter configurations by allocating resources to promising candidates while stopping less viable ones early. Key hyperparameters tuned included the LeakyReLU activation's `alpha` values `[0.01, 0.05, 0.1, 0.2, 0.3]`, the number of neurons in dense layers `[512, 1024, 2048]`, and dropout rates `[0.2, 0.3, 0.4]`. The best configuration identified was `alpha=0.01`, `dense_units=2048`, and `dropout_rate=0.2`.

The model's architecture includes an input layer for the standardized feature vector and five hidden layers with decreasing neurons: 2048, 1024, 512, 256, and 128. Each hidden layer applies LeakyReLU activation (α=0.01), batch normalization for stability,

and dropout (rate=0.2) for regularization. The output layer uses softmax activation for multi-class classification, with the number of neurons matching the unique pitch types.

We employed the `AdamW` optimizer with a learning rate of 1e-3 for efficient weight decay integration and `categorical_crossentropy` as the loss function for one-hot encoded multi-class labels. Mixed precision training (`mixed_float16`) was enabled to accelerate computation on compatible hardware. This architecture was consistently optimal for models tailored to both left-handed and right-handed pitchers, leveraging 18 preprocessed input features.

Results

Left-Handed Pitchers

The left-handed pitcher model achieved the following performance metrics:

- **Mean Training Accuracy**: 89.54%
- **Mean Validation Accuracy**: 91.08%
- **Mean Training Loss**: 0.281
- **Mean Validation Loss**: 0.239
- **Final Validation Accuracy**: 94.84%
- **Epochs for Convergence**: 25

Key observations from the learning curves:

- Training accuracy steadily increased across epochs, while validation accuracy closely tracked it, indicating minimal overfitting.
- Validation loss consistently decreased, supporting robust generalization.

Left Handed Model Training Validation and Accuracy

Right-Handed Pitchers

The right-handed pitcher model exhibited similar performance:

- **Mean Training Accuracy**: 89.76%
- **Mean Validation Accuracy**: 91.12%
- **Mean Training Loss**: 0.278
- **Mean Validation Loss**: 0.237
- **Final Validation Accuracy**: 94.84%
- **Epochs for Convergence**: 25

Insights from the learning curves:

- The trends mirrored those of the left-handed pitcher model, with a tight alignment between training and validation metrics, signifying good generalization.
- Both training and validation losses converged smoothly.

Right Handed Model Training Validation and Accuracy

Interpretation of Results

The hyperparameter tuning process led to a model with enhanced capacity and regularization:

- **Alpha Value (`0.01`)**: A small `alpha` in the LeakyReLU activation function allowed the model to benefit from a minor negative slope, preventing neuron inactivity and facilitating better learning.
- **Dense Units (`2048`)**: A higher number of neurons increased the model's ability to capture complex, non-linear relationships within the data.
- **Dropout Rate (`0.2`)**: A lower dropout rate provided sufficient regularization without overly diminishing the model's learning capacity.
- The tuned models achieved a validation accuracy of between 92 and 94%, a significant improvement over models without hyperparameter tuning (baseline was 85%).
- Training accuracy closely matched validation accuracy, suggesting that the models generalized well to unseen data or new pitchers.

This performance is better than MLBs deprecated pitch classification models but still not as accurate as the new PitchNet system.

Measure	Known Pitchers	
	Current Network	PitchNet
Accuracy	89.78%	96.25%
In-Arsenal %	94.35%	99.95%

(1) - Pitch Net Performance

The final models are robust and efficient, suitable for deployment in real-time pitch classification systems. Improvements could be made to tune the models for each pitcher and pre-cluster the data according to each arsenal prior to classification.

Strengths of the Approach

1. **High Accuracy**: Both models achieved high validation accuracy (92+%), demonstrating their effectiveness in pitch classification tasks.
2. **Efficient Tuning**: The Hyperband tuning approach successfully identified the optimal hyperparameters (`alpha=0.01`, `dense_units=2048`, `dropout_rate=0.2`) within a reasonable timeframe.
3. **Generalization**: Close alignment between training and validation metrics suggests robust generalization, making the models suitable for deployment in real-world applications.

Unexpected Observations

- **Validation Performance**: Validation accuracy slightly surpassed training accuracy in the early epochs, which could be attributed to the regularization techniques (dropout, batch normalization) effectively reducing overfitting.
- **Consistency Across Handedness**: The left handed model was more accurate (~94% vs ~92%), but this is likely due to there being more right handed pitchers with a wider variety of mechanics in MLB.

Drawbacks and Limitations

1. **Computational Costs**: The training and hyperparameter tuning process was computationally intensive, taking over an hour for each handedness subset.
2. **Model Interpretability**: Neural networks are inherently less interpretable compared to simpler models, which could limit insights into feature importance

and decision-making.

3. **Season-Specific Data**: The models were trained exclusively on 2024 data, potentially limiting their applicability to other seasons or varying pitching conditions or data collection methods.

Future Work

- **Cross-Validation**: Implement k-fold cross-validation to further evaluate model robustness.
- **Real-Time Deployment**: Integrate the models into live systems to assess performance in real-time scenarios.
- **Expanded Feature Set**: Explore additional features, such as pitcher-specific tendencies or game context variables.
- **Temporal Analysis**: Analyze model performance over different seasons to assess adaptability.
- **Cluster Integration**: Implement pre-processing via clustering to tune the model for each pitcher.

Code and Research Links

https://github.com/dpasqua17/pitch_classifier

Research Review Sources:

[1] https://technology.mlblogs.com/mlb-pitch-classification-64a1e32ee079

[2] https://content.iospress.com/articles/journal-of-sports-analytics/jsa200559

[3] https://github.com/J-Douglas/MLB-Pitch-Analytics

[4] https://patents.google.com/patent/US8876638B2/en

[5] https://cs230.stanford.edu/projects_spring_2018/reports/8290890.pdf

[6] https://assets-global.website-files.com/5f1af76ed86d6771ad48324b/606e515d0e2589d01b6946ba_ArnavPrasad-DecodingMLB-RPpaper.pdf

[7] https://arxiv.org/pdf/1304.1756