Andrew Doyle

August 6, 2017

Capstone Proposal

**Domain Background**

For my Capstone project I will be applying several different unsupervised learning algorithms to Major League Baseball data in order to classify pitches thrown into balls and strikes. There are several reasons I chose the domain of baseball for my final project. First and foremost, I am a big fan of baseball, particularly the analytical and quantitative side of the game. It was this interest that lead me to the field of machine learning in the first place, and even helped me learn some machine learning and statistical software (Marchi & Albert, 2013). Statistical analysis is in the DNA of the game of baseball, as game data has been collected for as long as the game has been around. Because of the vast amount of data, the field lends itself very well toward machine learning.

Secondly, with the advent of the Statcast system, recently collected data can offer insights into the game that were previous unavailable. Statcast “is a state-of-the-art tracking technology, capable of measuring previously unquantifiable aspects of the game. Set up in all 30 Major League ballparks, Statcast collects data using a series of high-resolution optical cameras along with radar equipment. The technology precisely tracks the location and movements of the ball and every player on the field, resulting in an unparalleled amount of information covering everything from the pitcher to the batter to baserunners and defensive players” (What is Statcast?, 2016).

Finally, the datasets for MLB data are both large and free to use by anyone, making it very conducive toward this type of analysis. Machine learning has been applied to baseball data both publicly among the baseball community for pleasure and privately among professional teams with the goal of gaining a competitive advantage. An example of using this field of research on a Statcast dataset would be using a Random Forest classifier to predict whether a batted ball will be a hit or an out (Petti, 2016).

**Problem Statement**

The problem I will be attempting to solve is “can unsupervised Machine Learning algorithms correctly classify pitches thrown by Major League Pitchers into balls and strikes?” Using Statcast data, consisting of perceived velocity, spin rate, release point and pitch type, I will apply unsupervised machine learning algorithms in my effort to accomplish this. Statcast tracks the above features for every pitch that is thrown in major league baseball, as well as the result of each pitch and numerous other metrics. The metric I will be attempting to identify is the resulting classification of the pitch, which is ball or strike.

**Datasets and Inputs**

Statcast tracks many different features of each pitch thrown in a major league baseball game, with all data being free to use and publicly available via [www.BaseballSavant.com](http://www.baseballsavant.com). The impetus for the project came from watching the game - it is the job a hitter to identify whether each pitch thrown is going to be a ball or a strike before the pitch cross the plate, and then in turn decide whether to swing at the pitch. This identification is based on some combination of the velocity of the pitch, the spin direction and rate, and the release point, as well as some randomness (it is likely guesswork is involved to some degree though how much is unknown). To replicate the decision-making process, pitch classification is conducive to using unsupervised algorithms because it is a form of binary classification with many features with which the algorithm can use.

**Solution Statement**

The feature I will be trying to predict is whether a pitch is a ball or a strike, which is identified along with each other feature that Statcast tracks. I will be using accuracy score and f1-score to determine how well the algorithms perform at classifying each pitch.

There are two possible methods to classify each pitch as a ball and a strike. The first would be to use the call made by the umpire, and the second would be to determine where the ball crosses the plate and whether that position is in the defined strike zone. There are pros and cons to each method, which I will detail below.

The first method to classify pitches is to use the call made by the umpire. The advantage of this method is that is requires no processing on the part of the user – the call is marked in the data set already – and that it potentially plays into the decision made by the hitter. The strike zone of an umpire may not match that of the rulebook exactly, yet the umpire is the final authority. This means that a hitter may take into consideration the strike specific to that umpire in determining whether a pitch will be a ball or a strike. The downside to using this method is that it is an inconsistent method of classification which would cause issue for a machine learning algorithm.

The second option is to use the rulebook definition of the strike zone in conjunction with the coordinates of where the ball cross the plate. The benefits of this are a consistent strike zone for every pitch in the dataset, which would be more conducive toward accurate classification. The downside is additional manual processing of the data set, as the classification in the data set is based on the call made by the umpire, and not the location of the pitch as it crosses the plate.

For this project I have chosen to use the second method of classification, which classifies each pitch by where it crosses the plate and not the call made by the umpire.

**Benchmark Model**

In this case, the benchmark model is that actual classification of the pitch. The results are recorded for every pitch thrown, so I can easily separate the data set into the features and the variable I am attempting to predict. I can then use that variable as a method for evaluating the classification made by the algorithm.

**Evaluation Metrics**

Accuracy score computes the accuracy of the prediction, presented as a fraction of the correct predictions out of the total number of predictions (3.3. Model evaluation: quantifying the quality of predictions, 2010-2016). F1-score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal (3.3. Model evaluation: quantifying the quality of predictions, 2010-2016).

**Project Design**

The initial step in the process is to download the data set. The data set will consist of a defined list of major league pitchers and every pitch they have thrown in the 2016 and 2017 seasons, along with all of the features measured by Statcast. This data set will be too large to download together, so each pitcher’s data will have to be downloaded separately and then manually merged.

Once the dataset is complete the data needs to be cleaned and preprocessed. This consists of extracting only those features which are relevant to the classification. Those features are pitch type (fastball, curveball, etc.), release speed in miles-per-hour), the x, y and z dimensional coordinates of the release point of the pitch, the x and y coordinates of where the pitch cross the plate, the spin rate of the pitch, and the amount of extension the pitcher has when throwing the ball. Once the relevant features have been extracted the values need to be converted to a format that can be used by the algorithm. This is composed of 3 steps.

The first step is to classify each pitch into ball and strike based on where the pitch crossed the plate. The dimensions of the strike zone, according to the rule book, are 1.66 feet horizontally and 1.96 feet vertically (<http://www.hardballtimes.com/strike-zone-fact-vs-fiction/>). I can then use the X and Z coordinates identified statcast to see if it crosses the plate in that 3 dimensional zone. Once this is finished, the data can be separated into features and the variable I am attempting to predict. The second step in this process is to normalize the numerical features to account for skewed values. The third step is to convert all categorical using one hot encoding.

Once the data has been preprocessed, it can then be separated into a training set and a testing set.

Once these steps have been completed the data is ready to be processed. I will be using unsupervised learning algorithms on this data set. In this case I will be trying the following on the data set:

* Gaussian Naive Bayes (GaussianNB)
* Decision Trees
* Ensemble Methods (Random Forest)
* Support Vector Machines (SVM)
* Logistic Regression

After the algorithms are run on the curated and preprocessed dataset, I will test the accuracy and F-score of each algorithm to view how well each algorithm performed.