"Stealing signs" of MLB pitchers through sparse learning.

Daniel Cusworth
Final Project
MIT 9.520 - Fall 2016
dcusworth@fas.harvard.edu

Abstract

The game of baseball is fraught with rules, but can be reduced to essentially one main conceit: A pitcher throws a ball that a batter tries to hit. The success of the batter is related to the difficulty of the pitch the pitcher makes. There are three general types of pitches a major league (MLB) pitcher makes: fastball, offspeed pitch, and breaking ball. Being able to anticipate the pitch type is advantageous to a hitter, thus many batters "steal signs," or orchestrate schemes to learn a pitcher's pitch selection before it is thrown. I develop a learning algorithm to predict a pitcher's next pitch given a particular pitcher's in-game and historical pitching performances. I use a logistic loss function with an elastic net penalty, and train a unique model on several pitchers individually. I train each learning algorithm for each pitcher several times while varying the elastic net parameter α. I use 10-fold cross validation to optimize the value of λ, the regularization parameter. I find that for binary classification (fastball, non-fastball), 42 out of 70 trained pitchers outperformed their baseline models. For multi-class classification (fastball, offspeed, breaking), only 12 models outperformed the baseline. I find relatively small influence of the elastic net parameter in improving accuracy of the model, and see that even the non-sparse algorithm has a similar fitted weight structure as sparse solutions.

1 Building Features

Major League Baseball (MLB) data for each pitch a pitcher throws is catalogued and archived by Sportvision (`http://www.sportvision.com/baseball/pitchfx`). These data are maintained in a SQL database accessible in R through the package "PitchRx" [Sievert, 2014]. I queried all pitches thrown since 2012 during MLB games. I then find the 70 pitchers who threw the most pitches during this time frame. These pitchers will each have a unique set of learning algorithms fit to their data. For each pitcher, I build a training/testing (70/30 split) matrix game-by-game that describes both their in-game pitching performance and how their in-game performance compares to their historical pitching performance. Let N be the number of pitches in one game, and $\hat{X} \in \mathbb{R}^{N \times d}$ be the feature matrix. I construct the matrix row by row, such that $x_i \in \mathbb{R}^d$, $i \in \{1, \cdots, N\}$. Each element in x_i are as follows:

1.1 Pitch Count

An MLB at bat allows three swings (strikes) for a batter to make contact with a pitch. If the pitcher throws a pitch into the "strike zone" and a batter does not swing, it is also considered a strike. If a batter does not swing at a pitch outside the strike zone, he is awarded a "ball." If a pitcher pitches four balls during an at bat, the batter is awarded first base. To assign "pitch counts" as a feature, I create a

map where $x_i^{(1)} = -1$ if the pitch count highly favors the pitcher (i.e., zero balls and two strikes), and $x_i^{(1)} = +1$ if the count highly favors the batter (i.e., three balls and no strikes). I then assign all other pitch counts to a discrete value in the range of -1 and +1, depending on the favorability of those pitch counts, which I derive from historical batting averages across all baseball players for a given pitch count. Thus any pitch count that falls more on the negative spectrum represents pitcher favorability, and vice versa.

1.2 Runners on base

There are three bases that previous batters occupy if they were successful in getting a "hit" or "walk" off of a pitcher. In a similar fashion as pitch count, I create an index for runners on base where $x_i^{(2)} = -1$ if no runners are on base (favorable to the pitcher) and $x_i^{(2)} = +1$ if the bases are loaded (unfavorable to the pitcher). All other combinations of base runners are assigned a value in this range depending on their historical favorability.

1.3 Inning

I index the current inning in play $x_i^{(3)} \in [1, 9]$. Note that baseball can exceed nine innings in case of a tie, and the range is adjusted accordingly.

1.4 Outs

I index the current number of outs in the inning $x_i^{(4)} \in \{0, 1, 2\}$.

1.5 Percent of pitches of given pitch type thrown in game

I track how much a pitcher has been favoring a certain type of pitch in their pitch arsenal during the course of the game. Thus let $L = \{1, \cdots, K\}$ be the different pitch types in a pitcher's arsenal, with K being the total number of pitches in their arsenal. Let $\phi_1, \cdots, \phi_{i-1} \in L$ be the pitch types of all previously thrown pitches by the pitcher. Then features $x_i^{(r)}$ for $r \in [5, \cdots, 5 + \ell, \cdots, 5 + K]$ are defined as follows:

$$x_i^{(r)} = \frac{\sum_{j=1}^{i-1} \mathbb{1}(\phi_j = L_\ell)}{i - 1} \tag{1}$$

1.6 Physical deviation from history

I track how much during a game the pitch type for a given pitcher is deviating from the historical performance of that pitch. The physical attributes assessed are pitch speed, pitch ending location (in x-y plane), pitch "break" (measure of the vertical movement of a pitch along its trajectory), break angle (angle made between ending location and highest y-location along its trajectory), and ball spin rate. Thus I define the feature $x_i^{(p)}$ for $p \in [5 + K + 1, \cdots, d]$ as

$$x_i^{(p)} = \left| \frac{\sum_{j=1}^{i-1} \beta_{p,j}}{i - 1} - \overline{\beta_p} \right| \tag{2}$$

where $\beta_{p,j}$ is the value of the pth physical attribute of the jth pitch and $\overline{\beta_p}$ is the historical average of the pth attribute.

Lastly, I concatenate each matrix $\hat{X}$ made for each game the pitcher has pitched during 2012-2016 (P) to make the full training/testing matrix $\hat{X}_{full} \in \mathbb{R}^{\sum_{\ell=1}^{P} N_\ell \times d}$.

I note that the PitchFx database includes many other pieces of information that can be transformed into more features. However, for the sake of this analysis, I start with these features keep open the possibility of expanding to more features in the future.

2 Learning Algorithms

2.1 Binary Classification

For the binary classification case (fastball, not-fastball), I train a learning algorithm using the logistic loss function, linear kernel, and elastic net penalty:

$$\min_{w_0, w \in \mathbb{R}^d} \left\{ \frac{1}{N} \left[\sum_{i=1}^{N} y_i(w_0 + \langle x_i, w \rangle) - \log(1 + \exp((w_0 + \langle x_i, w \rangle))) \right] + \lambda \left[\frac{(1-\alpha)}{2} ||w||_2^2 + \alpha ||w||_1 \right] \right\} \tag{3}$$

I find the minimum using the coordinate descent method [Wright, 2015], which is achieved by minimizing along one direction at a time.

2.2 Multi-class Classification

For the multi-classification case (fastball, offspeed pitch, breaking ball), I also train a learning algorithm using logistic loss and elastic net penalty, however now over K classes. Let $Z = \mathbb{B}^{n \times K}$ with $z_{i,k} = \mathbb{1}(y_i = k)$, where $k \in \{1, \cdots, K\}$, then

$$\mathcal{E}(\{(w_{0,k}, w_k)\}_1^K) = \frac{1}{N} \left[\sum_{i=1}^{N} \left(\sum_{k=1}^{K} z_{i,k}(w_0 + \langle x_i, w_k \rangle) - \log \left(\sum_{k=1}^{K} \exp(z_{i,k}(w_0 + \langle x_i, w_k \rangle)) \right) \right) \right] + \lambda \left[\frac{(1-\alpha)}{2} ||w||_2^2 + \alpha ||w||_1 \right] \tag{4}$$

I use a partial Newton algorithm to find the minimum of $\mathcal{E}$. This is achieved by only allowing $(w_{0,k}, w_k)$ to vary one class at a time [Friedman et al., 2010]. Then Equation 4 becomes the following (after Taylor expanding around $(w_{0,k}, w_k)$):

$$\mathcal{E}_k((w_{0,k}, w_k)) \approx -\frac{1}{2N} \sum_{i=1}^{N} w_{i,k}(q_{i,k} - w_{0,k} - \langle w_k, x_i \rangle)^2 \tag{5}$$

with

$$q_{i,k} = w_0 + \langle w, x_i \rangle + \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))} \tag{6}$$

where

$$p(x_i) = Pr(\{1, \cdots, K\} = \ell | x_i) = \frac{\exp(w_{0,\ell} + \langle w_\ell, x_i \rangle)}{\sum_{k=1}^{K} \exp(w_{0,k} + \langle w_k, x_i \rangle)} \tag{7}$$

Then $\mathcal{E}_k$ can be again minimized using coordinate descent.

For both the binary and multi-class cases, I find the optimal value of λ^* using 10-fold cross-validation. Also, for each pitcher I fit the learning algorithm and additional 10 times while varying $\alpha \in [0, 1]$ in order to gauge the influence of sparsity on the optimal solution.

3 Results

3.1 Binary Classification

Of the 70 pitchers who considered, 42 (60%) were more predictable than a baseline classifier on the reserved testing set. Here I take the baseline to be a prediction of the majority class of all pitches, which generally means always predicting a fastball. Figure 1 shows the top 12 most predictable pitchers, and the absolute difference between the best fit classification and the baseline classification. One interesting result is that many pitchers who are perceived as elite are also predictable. For example, the most predictable pitcher here is Felix Hernandez, who was voted as a top-10 pitcher in baseball during 2012-2015. This result gives evidence that great pitchers get batters out even if their pitch selection is predictable, because the pitches they throw are extraordinary by themselves.

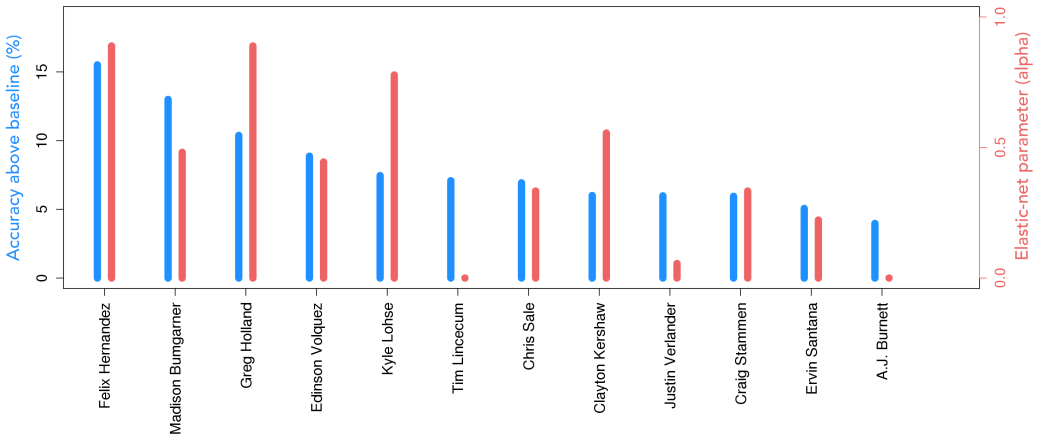

Figure 1: Top pitchers who outperformed their baseline classifiers on the testing set in the binary setting.

As noted, Felix Hernandez best outperforms his baseline classification. The values of his non-sparse weighted features are shown in Figure 2. I get roughly the same response independent of the elastic net parameter. For less sparse solutions, the small fitted weights are still much smaller than the larger weights. For Hernandez, we can interpret Figure 2 showing that the percentage of breaking balls, percentage of offspeed pitches, fastball break in the y direction, and breaking ball break in the y direction are the most prominent features. More simply, it appears that if he has not been using his breaking or offspeed pitches during a game, he will continue to not favor those pitches. Also, if his breaking pitches are deviating a lot in the vertical direction for a particular game, he will start to favor his fastball instead. This is seen in the opposite signs of "break y fastball" and "break y breaking."

3.2 Multi-class Classification

For the multi-class classification, I find that 12 pitchers out of 70 (17%) outperform their baseline on the testing set. The results are shown in Figure 3. Whereas the binary classification produced models up to 15% above the baseline, the best model in the multi-class setting performs 4% above the baseline, here for pitcher A.J. Burnett. Again exceptional pitchers are seen to outperform their baseline (e.g., Zack Greinke, Justin Verlander, Clayton Kershaw). Also, as in Figure 1, we see that there is no discernible trend in α across pitchers.

I look at the fitted weights for the A.J. Burnett. Since this is the multi-class setting, there are fitted weights for each pitch category (Figure 4). The strongest feature across classes is the percent offspeed feature in the offspeed class. This shows that if he is favoring his offspeed pitch during a given game, he is most likely to stick with that pitch. For his breaking ball (panel C), we see that if he is

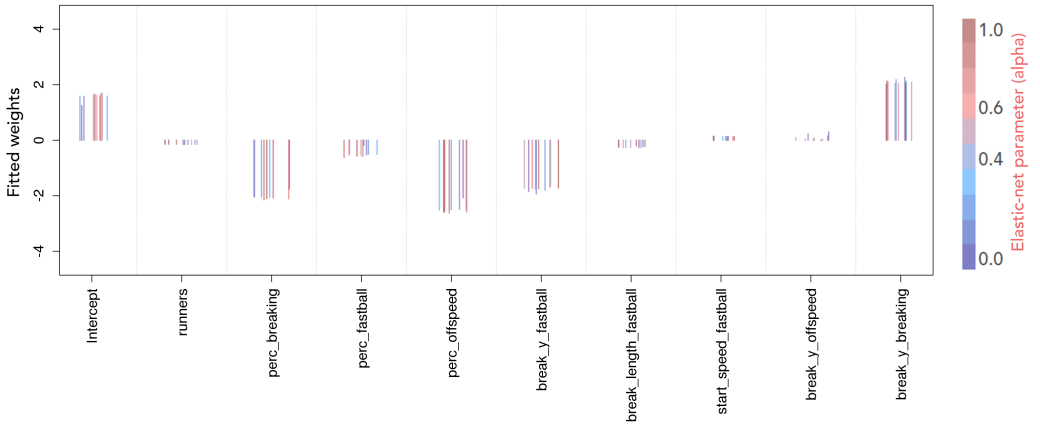

Figure 2: Felix Hernandez's non-sparse fitted weights over the elastic net grid $\alpha \in [0, 1]$.

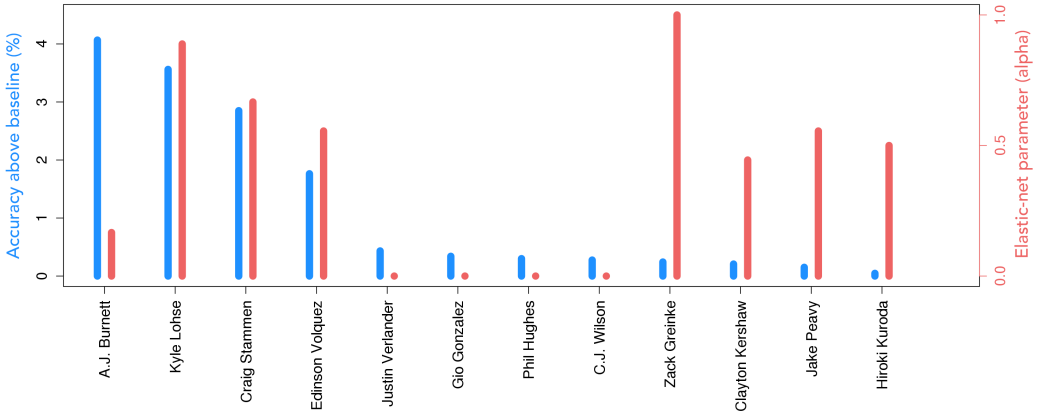

Figure 3: Top pitchers who outperform their baseline classifiers in the multi-class setting.

in a pitcher's count (i.e., index is negative), he is more likely to throw it, especially if there is little deviation in his breaking ball.

4 Conclusion

Using a selection of features retrieved and transformed from the vast PitchFx database, I create both binary and multi-class classifiers to predict a pitcher's next pitch. I find the best performance above baseline in the binary case. Many elite pitchers are also predictable, which may indicate that elite pitchers possess such elite pitches that confusing the batter with pitch variety is of lesser importance. My results also show an independence of the elastic net parameter α on the optimal solution.

The results of this work were performed using a logistic loss function and linear kernel. However the problem could easily be extended to other learning algorithms and more complicated kernels.

The PitchFx database contains more information about pitches that could be used to expand the feature space. One element not considered in this project is how historical pitching attributes change with respect to the batter, the batter's handedness, and the umpire. Since a pitcher may locate his pitches differently given which hand a batter uses, a future study should take this attribute into account. Similarly, a pitcher may change their approach depending on the elite status of a batter.

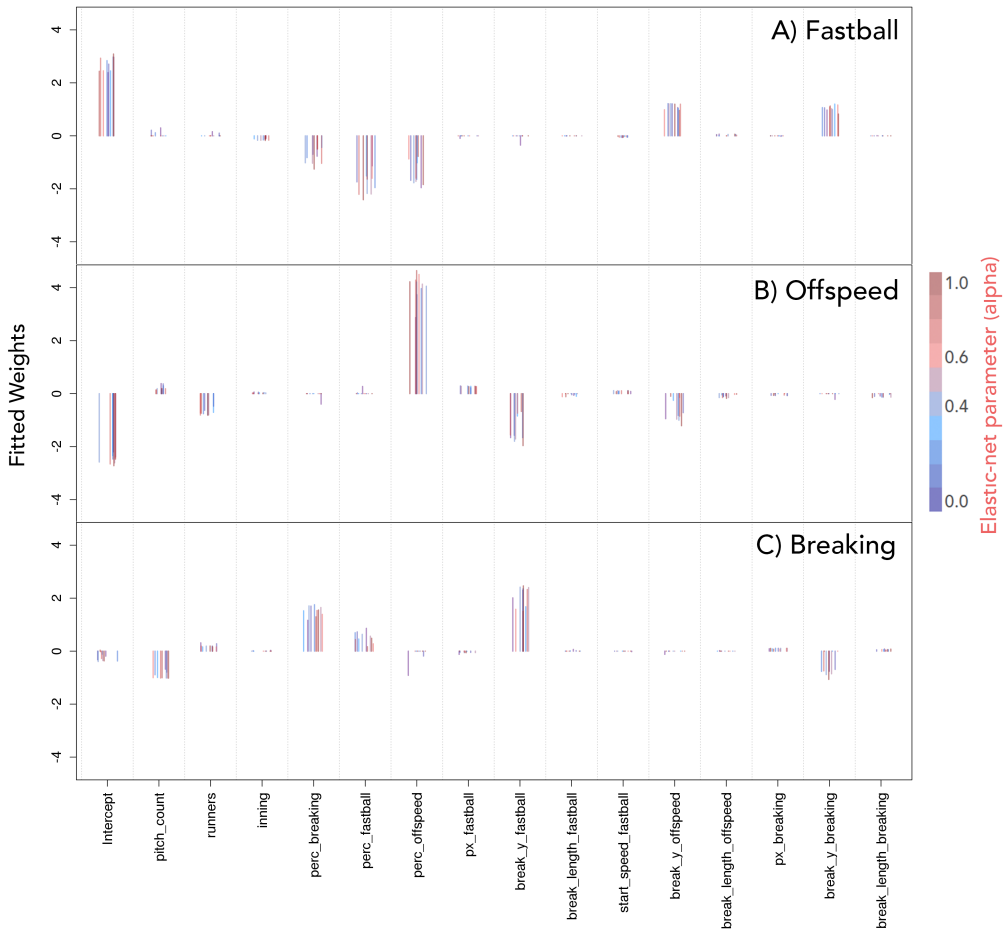

Figure 4: A.J. Burnett's non-sparse fitted weights over the elastic net grid for classes fastball, offspeed, and breaking.

Thus physical attributes may need to be adjusted based on a batter's (or group of elite batters') history with a given pitcher.

Finally, a future study could fit a model that predicts which pitch type a batter is least likely to hit given their historical proclivity to certain pitches. Then one could compare this work (predicting the next pitch) versus what a pitcher ought to have thrown to that batter. Such a result could inform a pitcher how to approach a particular batter.

References

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

C. Sievert. Taming pitchf/x data with xml2r and pitchrx. *A peer-reviewed, open-access publication of the R Foundation for Statistical Computing*, page 5, 2014.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.