"Stealing signs" of MLB pitchers through sparse learning.

Daniel Cusworth
MIT 9.520 Fall 2016

There are three general types of pitches an major league (MLB) pitcher makes: ***fastball***, ***offspeed***, and ***breaking***.

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 pitch types given a particular pitcher's in-game and historical pitching performance.

I train sparse algorithms to understand essential features that a future could put in their memory bank as they approach a future at bat.

Learning Algorithms

Case 1: Two pitches

$$\mathcal{Y} \in \{\text{fastball, non-fastball}\} = \{0, 1\}$$

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

Logistic loss function with elastic-net penalty.

For each pitcher and a given α, solve using coordinate descent

Case 2: Three pitches

$$\mathcal{Y} \in \{\text{fastball, breaking ball, offspeed pitch}\} = \{0, 1, K = 2\}$$

$$Z = \mathbb{B}^{n \times (K+1)} \qquad z_{i,k} = \mathbb{I}(y_i = k)$$

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

For each pitcher for a given α, minimize $\mathcal{E}$ by only allowing (w_0, w_k) to vary for a single class at a time. Then follow the partial Newton algorithm to minimize.

For both Case 1 and Case 2, optimize value of λ doing 10-fold cross validation.

Also, find a unique solution for each $\alpha \in [0, 1]$

Features $x_i \in \mathbb{R}^d$

Pitch Count: $x_i^{(1)} \in [-1, 1]$
-1 = favorable pitcher's count (0-2)
+1 = favorable hitter's count (3-0)

Runners on base: $x_i^{(2)} \in [-1, 1]$
-1 = no runners on base
+1 = bases loaded

Inning: $x_i^{(3)} \in [1, 9]$
Can theoretically exceed nine if game tied after nine innings.

Percent of pitches thrown in game:

$$x_i^{(r)} = \frac{\sum_{j=1}^{i-1} \mathbb{I}(\phi_j = L_\ell)}{i - 1} \qquad r \in [4, 4 + K]$$

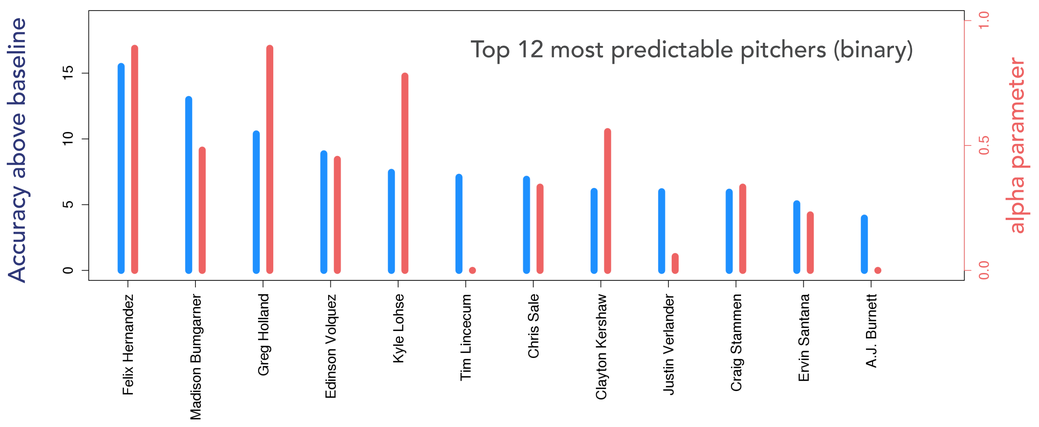
$$\phi_1, \cdots, \phi_{i-1} \in L \qquad L = \{0, 1, \cdots, K\}$$

Physical deviation from history:

$$x_i^{(p)} = \left| \frac{\sum_{j=1}^{i-1} \beta_p}{i - 1} - \overline{\beta_p} \right| \qquad p \in [4 + k + 1, d]$$

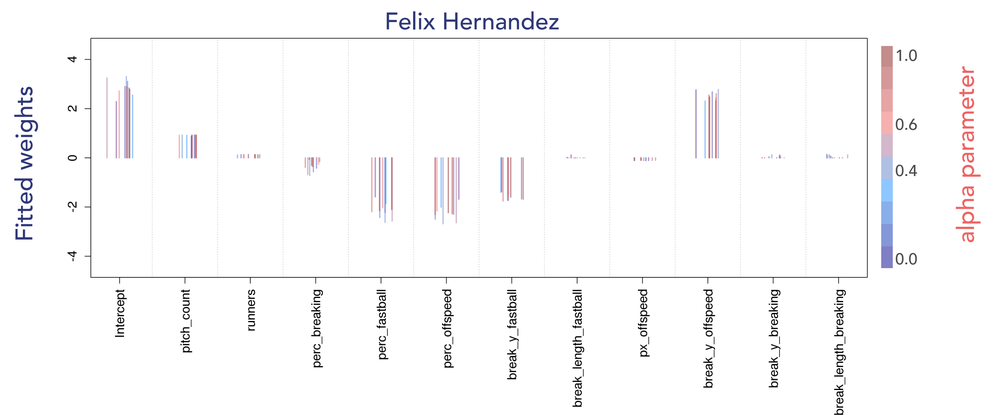
Where each β_p represents a physical attribute of a certain pitch (e.g., speed, break), and $\overline{\beta_p}$ is the attribute averaged over a pitcher's entire career.

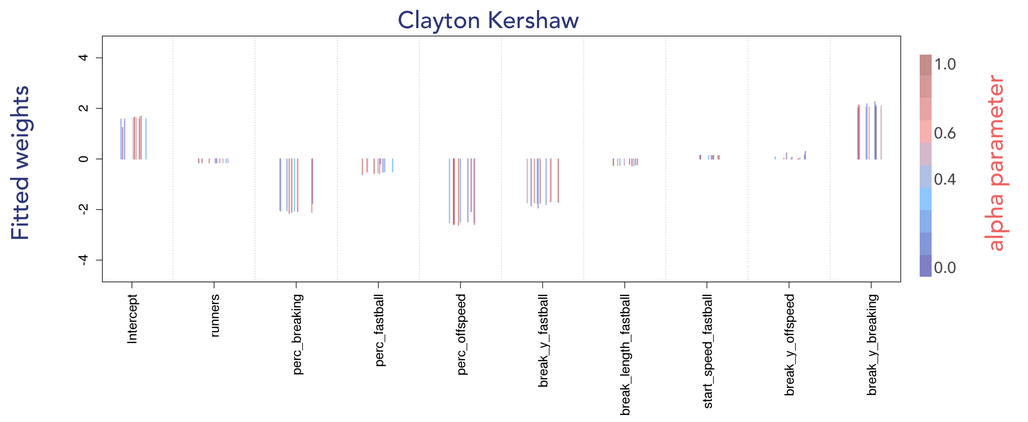
Binary Results



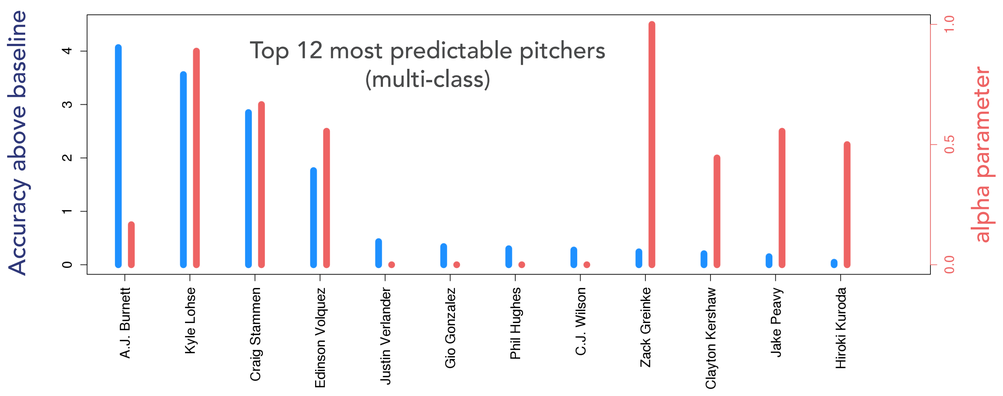
Fit unique model onto 70 pitchers:

42 models (60%) outperformed baseline

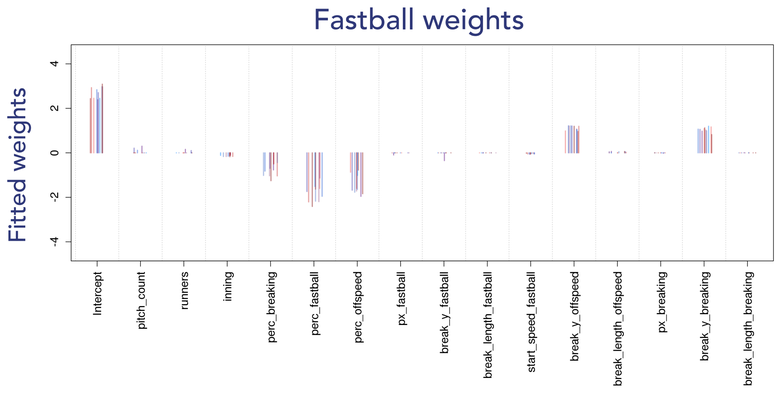


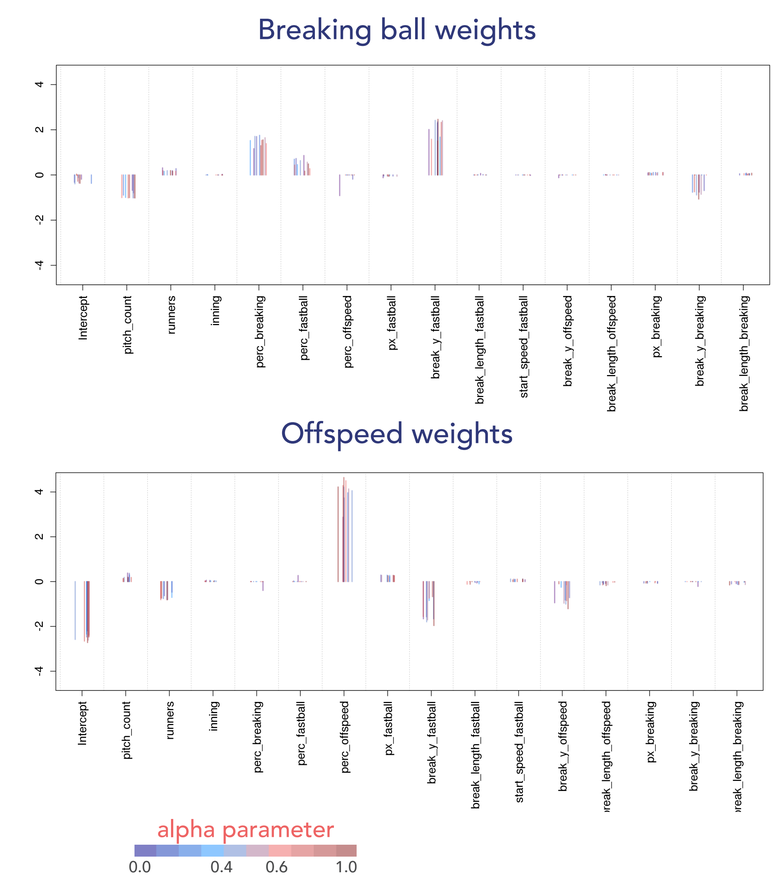


Multi-class Results



12 models (17%) outperformed baseline





Conclusions

Using the features described without a kernel, I find that a binary classification problem can outperform a baseline model (always predict the majority class) for the majority of pitchers.

For multi-classification, the number of pitcher models that outperform the baseline diminishes, and the accuracy above the baseline is much lower than in the binary case.

Predictability does not necessarily correlate with how "good" a pitcher is. For example, Clayton Kershaw here would be described as predictable, but has also been one of the most successful pitchers from 2012-present.

The results seem independent of the value of the elastic net parameter alpha

References

[1] Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

[2] Sievert, Carson. "Taming PITCHf/x Data with XML2R and pitchRx." A peer-reviewed, open-access publication of the R Foundation for Statistical Computing (2014): 5.