Beat the Streak with Machine Learning

George Gatling,
Pratheek Jayanth, and Ajinkya Shivdikar
George Mason University
Fairfax, Virginia
Email: ggatling@gmu.edu

***Abstract*—Two one-class support vector machines with nonlinear kernels are used to choose batters for the purpose of winning MLB.com's game 'Beat the Streak'. The learning machines are trained and tested on historical data from 2010-2016, and the probability of success of the learning machine is shown to exceed that of choosing batters with uniform probability. From the performance on historical data, an expected value of the longest streak is estimated.**

I. Introduction

Since Joe DiMaggio's record streak in 1941 of achieving a hit in 56 consecutive games, baseball fans and statisticians alike have been fascinated with the probability of another player breaking his streak. In 2001 MLB.com began "Beat the Streak" (BTS), a fantasy baseball game in which website users (BTS players) attempt to predict, 57 consecutive times, when a batter will get a hit in his game. Each day that a BTS player chooses a batter who gets a hit, the BTS player's streak increases by one, and if the chosen batter fails to get a hit, the BTS player's streak is reset to zero. There are many BTS player strategies for choosing batters, including randomly, using career data of the players, and using a players' performance over the past few games to predict the outcomes of the next games.

For the purpose of this paper, we model the BTS game as a Markov chain of Bernoulli trials. On the i^{th} day the BTS player has some probability of success, p_i, and the outcome of his choice results in his Markov 'state' increasing by one with probability p_i or resetting to zero, with probability $1-p_i$. Each baseball season is a finite, but random, number of days, N_s. To simplify the model we assume the BTS player's probability of success is constant each day so that $p_i = \hat{p} \,\forall\, i$. The value of $\hat{p}$ depends on the BTS strategy used. We also assume that baseball seasons are a fixed number of days so that $N_s = \hat{N} = 184 \,\forall\, s$.

II. Historical Data

Historical data was gathered from play-by-play files on retrosheet.org. Each file corresponds to a single team, and contains the records for each play in every home game for that team. The records were aggregated into an SQL database table, shown in Table I with the following columns:

1) play result: hit or miss
2) batter's position in the batting order
3) batter's position in the field
4) if the batter was at home
5) the air temperature at the field
6) if the batter was a starter
7) the batter's name
8) the pitcher's name
9) the opposing team
10) game number that day for the batter
11) year
12) day of year

TABLE I
SOME RETROSHEET.ORG PLAY-BY-PLAY RECORDS.

year	day	temp	...	batter	pitcher	hit
2010	94	67	...	camem001	sabac001	1
2010	94	67	...	camem001	sabac001	0
2010	94	67	...	camem001	chamj002	1
2010	94	67	...	drewj001	robed002	1
...	...	...	...	...	...	...

Since the retrosheet.org data is play-by-play, but the BTS game is game-by-game, the data required pre-processing to aggregate into game-by-game records. The game-batter pair records were put into another SQL table, as in Table II, with the following columns:

1) game result: hit or all misses
2) number of plays for the batter in the game
3) batter's position in the batting order
4) batter's position in the field
5) if the batter was at home
6) the air temperature at the field
7) if the batter was a starter
8) the batter's name
9) the starting pitcher's name
10) the opposing team
11) year
12) day of year

TABLE II
SOME PREPROCESSED GAME-BY-GAME RECORDS.

year	day	temp	...	batter	pitcher	outcome
2010	94	67	...	camem001	sabac001	1
2010	94	67	...	drewj001	robed002	1
...	...	...	...	...	...	...

Table II with data from 2010-2016 represents the pool of

training and testing data used in this paper.

III. Preparing the Data

A. Partitioning

Before exploring any of the data, we partitioned it into three sets. Data from years 2010-2014, inclusive, (the training set) was used for data snooping and training learning algorithms. Data from 2015 (the validation set) was used to test the different algorithms and parameters during the exploration and training phases. Data from 2016 (the test set) was set aside for final validation and only used once, after we had settled on both an algorithm and its parameters. The authors argue that the results of the 2016 test are most representative of what one could hope to achieve in 2018 using this algorithm.

B. Average Rates

Using data from the training set, two ratios were calculated: the batters games with hits to games played and the pitchers game-batter pairs with hits to the total game-batter pairs pitched to. A larger ratio is a indicative of a better batter while a large ratio is indicative of a worse pitcher. The ratios are then taken as estimates of the batters' and pitchers' average rates of success. Since these rates were estimated from historical data, we discarded batters and pitchers for whom little historical data was available. These average rates were merged into the game-by-game database records of Table II.

C. Null Rates

The simplest way to play BTS is to assume all of the batters will get a hit each day, and choose any batter with uniform probability. We adopt this method as a benchmark and calculate its scores on the 2015 data set in this section. For each day in 2015 we fetch the record for all the batters that play, and calculate the ratio of batters that got hits to all batters that played. We use this as an estimate for of the probability of choosing a batter successfully each day, and take as the best estimate the average value over all the days. For 2015 this give 0.571 as the estimated probability of choosing a successful batter at random.

Using the Markov chain model for BTS players' performance, we may define X_i as an indicator random variable for whether the BTS player choose a batter that got a hit on the i^{th} day, with some probability of success $\hat{p}$. We also define a random variable, R, as the longest run of successes over the $\hat{N}$ days of the baseball season. The expected value of R may be approximated as [1]

$$E[R] \approx -\frac{\log(N - N\hat{p})}{\log \hat{p}}$$

.

For the benchmark case of choosing a batter randomly with $\hat{p} = 0.571$ this give an expected streak of 8.0.

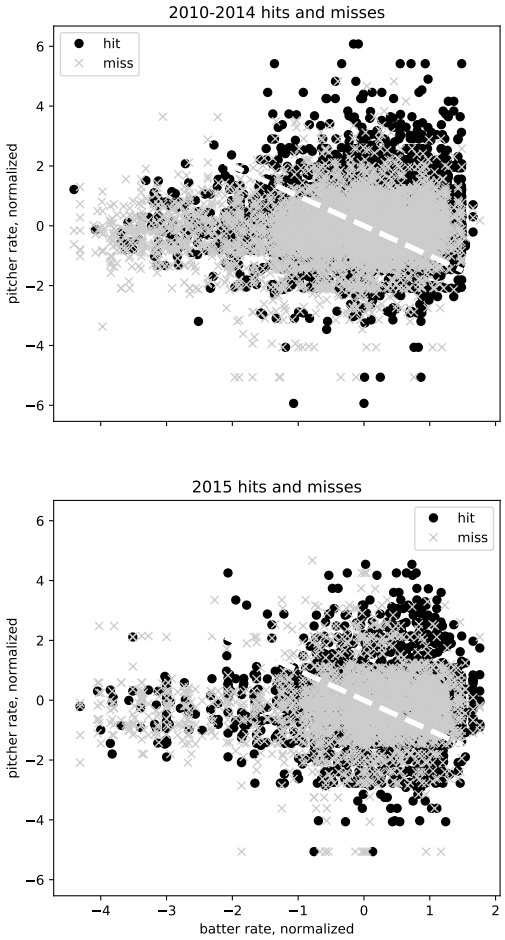

Fig. 1. Hit and miss outcomes as a function of batter and pitcher success rates.

IV. Data Snooping

A common BTS player strategy is to restrict the pool of possible batters to those who have a batting average better than some threshold set by the BTS player. To evaluate this strategy in 2015, the probability of BTS player success is estimated as in the null rate calculations, except that batters are only chosen from the pool of batters with better than average rates in the training data. In 2015 this gives $\hat{p} = 0.629$ and $E[R] = 9.3$. This improvement suggests that the training data in 2010-2014 can inform choice in 2015, and that there are correlations in the data for a learning machine to find.

From this 1-D feature space, there is a natural and intuitive extension to also consider the past performance of the pitcher. Every batter, game pair can be labeled either 'hit' or 'miss', and a plot of this data is shown in Figure 1. The significant

overlap between the two classes make them inseparable, and a supervised classifier was unable to find a suitable separating boundary between the classes, linear or otherwise, on one side of which lie exclusively 'hits'.

The upper-right corner of Figure 1 suggests the two classes do not *completely* overlap, and there may in fact be a region of feature space in which there are only 'hits', or at least in which the density of 'hits' is significantly greater than that of 'misses'. We seek a machine learning strategy to locate this region in the feature space.

V. One-Class Support Vector Machines

A. One-Class SVM

In the case where two classes have significant overlap, binary classifiers that are attempting to divide all of feature space into one of the two classes have conflicting information in the overlap region, and produce marginal to useless results when they train on such a data set. They are also highly s susceptible to noise in the training data.

Since the goal of the BTS game is to choose a single batter each day who achieves at least one hit in his game, it is not required to correctly classify most situations as 'hit' or 'miss'. Instead we only need a classifier that has a very low false-positive rate: when it predicts 'hit' the outcome is usually 'hit'. For this objective, we will use an unsupervised learning strategy to learn the regions of feature space corresponding to 'hit' and 'miss' independently. These regions are shown conceptually in Figure 2. Once these two regions are identified, we will label as 'favorable' batters that fall in the 'hit' space but not the 'miss' space.

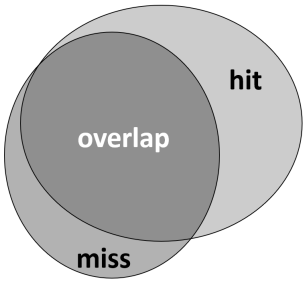

Fig. 2. Conceptual depiction of the hit and miss regions in a 2D feature space, and the overlap between them.

To learn these regions in feature space, we will use a special case of the well-studied support vector machine (SVM), the unsupervised one-class SVM. [2] This algorithm searches for the smallest surface in the feature space that contains all the the training samples, with a parameter to control how many of the points in the training set may lie outside of the learned boundary. This surface, S, is shown in Figure 3.

B. Two One-Class SVMs

Two machines were trained, one using exclusively data labeled 'miss' and the other using data labeled 'hit'. Subsequent

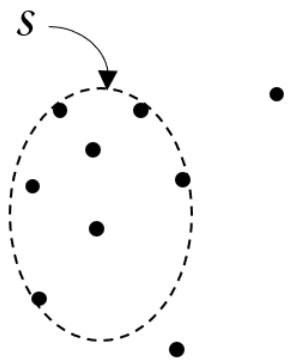

Fig. 3. The surface learned by a one-class SVM, with a parameter to allow some fraction of the training samples to lie outside the surface.

events, are then be classified twice, once to see if they belong to the miss-region, and again to see if they belong to the hit-region. Figure 4 shows the results of these two one-class hit and miss SVMs when trained on data from 2010-2014 and validated on 2015 data.

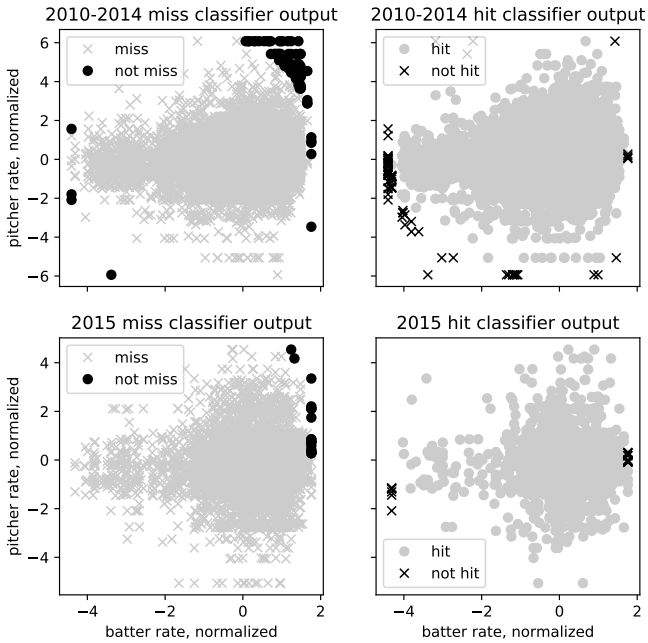

Fig. 4. One Class Output

'Favorable' events are events in the hit-region and not in the miss-region, and we will choose batters only from these events. The results of this combined classifiers on the training and validation data sets are shown in Figure 5. This classifier performance on the 2015 validation set is shown in Table III.

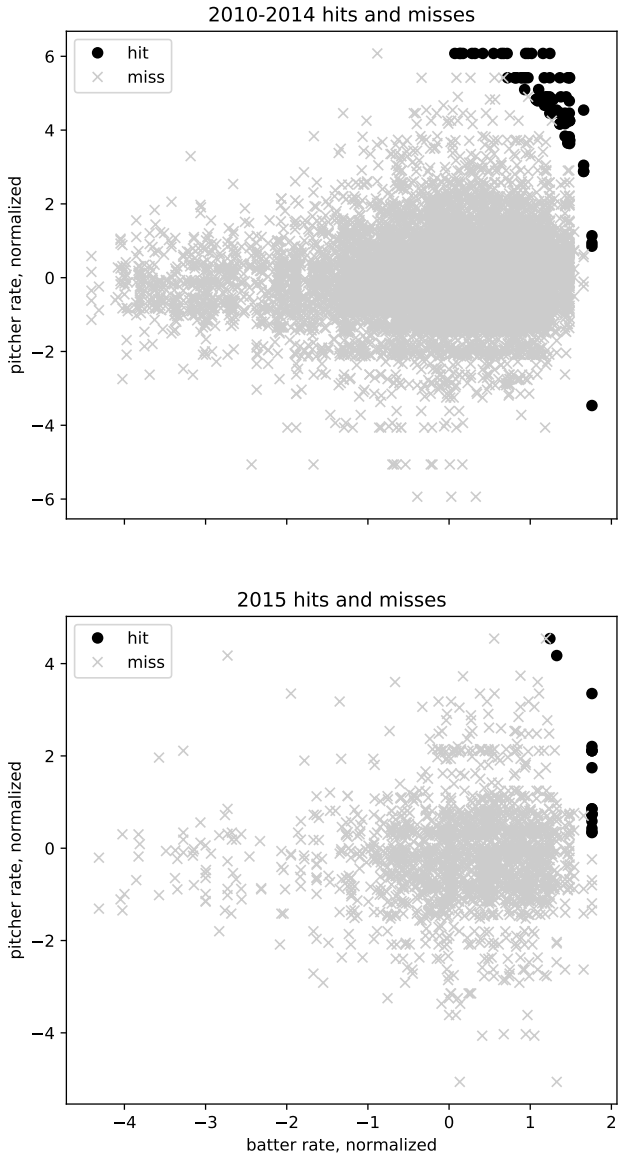

Fig. 5. Combined Classifiers

VI. Using a Higher Dimension Feature Space

The results of the preceding section were then extended to a higher dimensional space, where visualization is not possible, but the class overlap remains a problem. The same algorithm, using two one-class SVM classifiers, is used on the new feature space. The extended feature set is

1) batter rate
2) pitcher rate
3) the temperature at the ball field
4) the average plays per game for the batter
5) if the batter is a starter

The batter rate and pitcher rate are retained. The more plays-per-game a batter gets on average, the more chances he has to get a hit in the game, so this rate was added. Similarly starting batters get more plays in the game on average so this feature was included. Air temperature was added because it was available in the original data set.

Two one-class SVMs were trained on the same events as in the 2D feature section above, but with each event described by these five features. Validation results on the 2015 data set are shown in Table III.

VII. 2016 Test Results

Using the combined classifier as trained in the preceding section, the data from 2016 was processed, and the results shown in Table III. One can reasonably expect similar performance in 2018 using the methods presented in this paper. Although the region identified by the combined classifier does have a high probability of 'hit', the number of events falling in this region is inadequate to win the BTS game.

TABLE III
Two feature 2015 validation results

Null Rate

	$\hat{p}$	$E[R]$
Random	0.571	8.0

2015 Verification, 2 features

	$\hat{p}$	$E[R]$
Not a Miss	0.810	17.2
Yes a Hit	0.583	8.2
Combined	0.824	18.4

2015 Verification, 5 features

	$\hat{p}$	$E[R]$
Not a Miss	0.837	19.6
Yes a Hit	0.583	8.2
Combined	0.944	42.2

2016 Test, 5 features

	$\hat{p}$	$E[R]$
Not a Miss	0.812	17.2
Yes a Hit	0.593	8.3
Combined	0.958	48.9

VIII. Conclusion

No baseball player or BTS player has beaten the Dimaggio streak in the past 76 years of baseball or the past 17 years of BTS play. The algorithm described in this paper trains two one-class support vector machines trained individually on hits and misses from past seasons, and then classifies new events using both classifiers. Only events that are simultaneously hits and not misses are considered favorable, and, compared to

choosing batters randomly, a BTS player has an improved chance of choosing batters that will get hits during their games if the BTS player chooses batters the algorithm classified as favorable. However, using the five features in this paper, batter rate, pitcher rate, whether the batter is a starter, whether the batter is at home, and the air temperature at the ball park, the algorithm is not able to find enough favorable situations to win the BTS game. Future work could explore other unsupervised classifiers and other feature sets to expand the favorable region of feature space to enough situations the make winning BTS feasible.

ACKNOWLEDGMENT

The authors would like to thank Drs Bill Amatucci and Erik Tejero for their many lengthy conversaions on baseball games, rules, and statistics.

REFERENCES

[1] M. F. Schilling, "The longest run of heads," *The College Mathematics Journal*, vol. 21, no. 3, pp. 196–207, 1990.

[2] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.