Pu Ke, pk2532, Electrical Engineering Department, Columbia University

Explore Batting Strategies using Markov Decision Process

***Abstract*—This paper concentrates on using statistical and machines learning methods in baseball area. It models a pitcher's behavior by a Markov Process on the pitch count, estimate the transition probabilities by historical MLB seasonal play-by-play data, models the batting strategy as a Markov Decision Process to swing or stay on each pitch, gives the optimal strategies using Reinforcement Learning's Value Iteration algorithm, and evaluates the obtained strategies on data from a different season using Policy Evaluation algorithm. This paper also aims to compare the performance of batting strategies calculated using data from a single pitcher and data from a group of pitchers, explore the changes of pitching style along different MLB periods, and also propose a way to tune the parameters of reward function used to get the batting strategies.**

***Index Terms*—baseball statistics, Markov Decision Process, reinforcement learning**

I. Introduction

STATISTICS are very important to baseball, perhaps more than any other sport. Since the game of baseball has a very structured flow to it, the game lends itself to easy record keeping and statistics. This makes comparisons between players on field performance relatively easy, and therefore gives statistics about baseball more importance than in most other sports.

Statistical analysis has been around as long as baseball has been played competitively. Long before Moneyball became a worldwide phenomenon in the 21st century and before Bill James' baseball writings gained mainstream popularity in the 1980s, Hall of Fame manager Earl Weaver was using index cards to fine-tune his platooning system and pitching changes with the Baltimore Orioles in the 1960s, while Branch Rickey hired statistician Allan Roth in the 1940s to evaluate player performance with the Brooklyn Dodgers. A generation before that, Baseball Magazine editor F.C. Lane was creating new statistical methods to measure offensive production, culminating in his classic book of essays, Batting. In the mid-19th century, Henry Chadwick is credited with developing the box score and his tabulation of hits, home runs and total bases led to the formulation of metrics such as batting average and slugging percentage.

Among all the problems the statistic analysis of baseball aims to solve, how to get a hit off of a MLB pitcher is one of the hardest [1]. A game of prediction based on the obtained information is hidden behind the battle between the batter and pitcher. The batter is guessing what ball the pitcher will be throwing next, and select his actions based on this information, while the pitcher is doing the same thing. Instead of guessing blindly, we can use historical game data to probe into certain patterns of pitchers' behavior [2] and contribute the gathered information to producing better batting strategies.

This paper uses statistical and machine learning technique to exploit pitchers' decision making by modeling it as a Markov Process on the ball count and strike count pair at each pitch. The transition probabilities are estimated by the conditional proportions in MLB pitch-by-pitch data, which can be downloaded from http://www.retrosheet.org/. Two types of transition probabilities can be calculated. One uses only data from a single pitcher, which is called 'pitcher-specific' later in this paper, and the other uses data from a group of pitchers, which is called 'general'. A batter informed of the transition probabilities is presented with a Markov Decision Process (MDP) to swing or stay at each given ball count and strike count pair. Optimal batting strategies for MDP can be solved by Reinforcement Learning algorithms. Specifically, in this paper, the Value Iteration algorithm is used to find the optimal strategy, and the Policy Evaluation algorithm is used to evaluate the optimal strategies. This method has also been used in sports gamesmanship, via the study of offensive play calling in American football [3].

Baseball play-by-play data from 1991 to 2017 for 87 pitchers is explored using the above method in this paper. In addition to evaluating the batting strategies on new data, I also compare the performance of the pitcher-specific, general and intuitive batting strategies, explore changes of pitching style along the years through different policy learned from data, and also propose an approach to tune the parameter's of reward functions.

This paper is structured as follows. Section 2 introduces the methodology used in this paper, including the project architecture and the details of MDP model. Section 3 describes the details of the techniques and steps to implement the actual project. Section 4 explores and interprets the results produced. At last, Section 5 gives a brief conclusion.

II. Methodology

The methodology used in this paper is similar to [3].

A. Project Architecture

The project uses 27 years (1991-2017) of play-by-play seasonal MLB data. For each year's data, the following steps are applied:

(1) Represent specific pitcher and general pitching behavior by Markov Process whose probabilities are estimated by counting the transitions of the pitching data in one season.
(2) Generate optimal batting strategies against these process, both in general and pitcher-specific sense.
(3) Evaluate both strategies on data in another year.

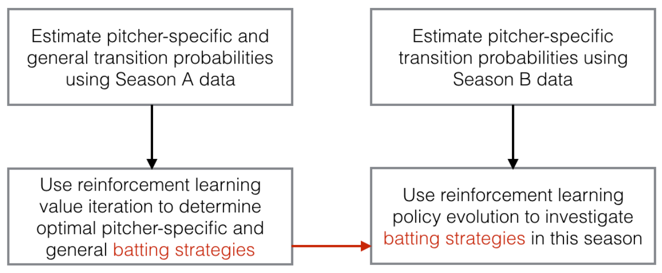
(4) Explore the results in terms of comparison between general batting strategies & pitcher-specific batting strategies, changes of batting strategies during these 27 years, the performance of this data-driven strategies on elite and non-elite pitchers, etc.

For each pitcher, we assume that their pitch behaviors are stochastic and governed by one-step Markovian assumptions on the pitch count. Different pairs of strike and ball count used as different states of a Markov Process. The transition probabilities can be estimated for a particular pitcher based on their own data in a year, or data for a collection of pitchers which are called general transition probabilities and pitcher-specific transition probabilities in this paper.

Then the batter informed of the relevant transition probabilities is presented with a Markov Decision Process (MDP) to swing or stay at a given pitch count. Optimal batting strategies for MDP can be found by a Reinforcement Learning algorithm [5].

For each two seasons of MLB data, The schematic analysis flowchart is presented as below:

Fig. 1. The flowchart of Value Iteration and Policy Evaluation analysis of two years of MLB data.


Estimate pitcher-specific and general transition probabilities using Season A data
Estimate pitcher-specific transition probabilities using Season B data
Use reinforcement learning value iteration to determine optimal pitcher-specific and general batting strategies
Use reinforcement learning policy evolution to investigate batting strategies in this season

B. Markov Decision Process and related algorithms

In my paper, I present different status of an at-bat with 18 states, 6 of which are absorbing states. And a pitcher's decision is represented by conditioning the pitch outcome at the previous pitch count. They are showed below:

S = {
(strike count: 0, ball count: 0),
(strike count: 1, ball count: 0),
(strike count: 2, ball count: 0),
(strike count: 3, ball count: 0),
(strike count: 0, ball count: 1),
(strike count: 0, ball count: 2),
(strike count: 1, ball count: 1),
(strike count: 1, ball count: 2),
(strike count: 2, ball count: 1),
(strike count: 2, ball count: 2),
(strike count: 3, ball count: 1),
(strike count: 3, ball count: 2),
Out,
Single,
Double,
Triple,
Homerun,
Walk
}

The Value Iteration algorithm used to calculate optimal policy using season A's data can be expressed as below:

Fig. 2. Value Iteration (P^{TRAINING} , g, S, U)

```
repeat
  $\Delta = 0$
  for each $i \in \mathcal{S} \cap \mathcal{E}^c$
    do
    $v \leftarrow J(i)$
    $J(i) \leftarrow \max_u \sum_j P_u^{\text{TRAINING}}(i,j)[g(i,u,j) + J(j)]$
    $\Delta \leftarrow \max(\Delta, |v - J(i)|)$
until $(\Delta < \varepsilon)$
output (deterministic policy $\pi = \{u^0, \ldots, u^{n-1}\}$)
```

$P_u^{\text{TRAINING}}(i,j)$ is an estimated probability of transitioning from state i to j when selecting batting action u (stay or swing) on the pitch-by-pitch data. As mentioned above, the probabilities can be computed from either a single pitcher's data or a population of at-bats for one season. J(i) is the expected reward of state i when following the best batting strategy. g(i, u, j) is the reward function where u is the current action, i is the previous state, and j is the current state. The initial function in this paper is defined as below, and a reward function parameter tuning method is also presented in this paper later.

g(i, u, j) = {
0, if j = {Out} or other non-absorbing states,
1, if j = {Walk} and u = Stand,
2, if j = {Single} and u = Swing,
3, if j = {Double} and u = Swing,
4, if j = {Triple} and u = Swing,
5, if j = {Homerun} and u = Swing
}

Intuitively, the algorithm keeps iterating until the states' expected reward is close to their optimal reward. The algorithm terminates upon satisfying the convergence criterion [6].

The Value Iteration algorithm outputs the batting strategy pi that maximizes the expectation at each state over the input data. And then this batting strategy π is used in policy evaluation algorithm showed below, which uses the best batting strategy π on a different season of the respective pitcher's pitch-by-pitch data to calculate the expected reward of each state in the at-bat which is represented as J^{π}:

Policy Evaluation therefore allows a quantitative comparison of the pitcher-specific and general batting strategies, π^p and π^g .

Fig. 3. Policy Evaluation (P^{TEST}, g, S, π)

```
repeat
  $\Delta = 0$
  for each $i \in \mathcal{S} \cap \mathcal{E}^c$
    do
    $v \leftarrow J(i)$
    $J(i) \leftarrow \sum_j P^{\text{TEST}}_{\pi(i)}(i,j)[g(i,\pi(i),j) + J(j)]$
    $\Delta \leftarrow \max(\Delta, |v - J(i)|)$
until $(\Delta < \varepsilon)$
output $(J^{\pi})$
```

III. Implementation

The basic steps for implementation is showed as below:

A.Data Preparation

The MLB seasonal play-by-play files from 1991 to 2017 are downloaded from http://www.retrosheet.org. The information used to calculate transition probabilities is contained in play records and sub records in files with extension of EVN or EVA for each year, and it is expressed in the format:

"play, 1, 0, margm001, 12, CCBFFFX, HR/78/F".

The interpretation of each field is as follows:

(1) The first field is the inning, an integer starting at 1.

(2) The second field is either 0 (for visiting team) or 1 (for home team).

(3) The third field is the Retrosheet player id of the player at the plate.

(4) The fourth field is the count on the batter when this particular event (play) occurred. (since records before 1991 only contains '??' in this field, so I only use files starting from 1991)

(5) The fifth field is of variable length and contains all pitches to this batter in this plate appearance and is described below. If pitches are unknown, this field is left empty, nothing is between the commas.

(6) The sixth field describes the play or event that occurred.

(In process.py file)

B. Calculate transition probabilities

Sub records (which indicates when players are substituted) and start records (which indicates the initial players) are used to filter out the play records when specific player is on the field as the pitcher. Based on information of the fifth field and sixth field of play record, I count the transition frequency of each states. For example, from the play record example above the fifth and sixth fields are 'CCBFFFX' and 'HR/78/F', which means "called strike, called strike, ball, foul, foul, foul, homerun". So each transition below is added by one:

(the first field of the tuple is ball count, the second field of the tuple is strike count)

(0, 0) -> (0, 1)
(0, 1) -> (0, 2)
(0, 2) -> (1, 2)
(1, 2) -> (1, 2)
(1, 2) -> (1, 2)
(1, 2) -> (1, 2)
(1, 2) -> Homerun

In implementation, I use a dictionary to match actual different circumstances with the states. (In process.py file)

The transition probabilities are calculated for 1991-2017 and 87 pitchers in history. Both general and pitcher-specific transition probabilities are calculated and stored separately in dictionary generalProb and pitcherProb, where the key in generalProb is 'year', and the key in pitcherProb is 'year + pitcher'. Those probabilities are used in the Value Iteration and Policy Evaluation. It should be noted that the actual pitchers for each year vary since each pitcher has different active years. (In stats.py file)

C.Value Iteration

Using the Value Iteration algorithm provided above (implemented in bellman.py file), I calculate batting policies using general or pitcher-specific transition probabilities for each year. The result is expressed as a 1 * 12 dimension vector.

For example, for pitcher Max Scherzer, the calculated general batting strategy for year 2017 is:

[0 0 0 1 0 0 0 0 0 0 1 0]

(for non-absorbing state [0,0], [1,0], [2,0], [3,0], [0,1], [0,2], [1,1], [1,2], [2,1], [2,2], [3,1], [3,2])

which suggests the batter to swing when there are three balls and no strike, and when there are three balls and one strike.

The calculated pitcher-specific batting strategy is:

[0 0 0 1 0 0 0 0 0 0 0 0]

which suggests the batter to swing only when there are three balls and no strike.

The general batting policies and pitcher-specific batting policies are stored separately in dictionary generalPolicy and pitcherPolicy, where the key in generalPolicy is 'year', and the key in pitcherPolicy is 'year + pitcher'. Those policies are used in the Policy Evaluation. (The code is in calc.py. The whole policies logging results for each pitcher and each year can be found in file diff27pitcher.txt)

D.Policy Evaluation

Using policies calculated for a specific year or specific year + specific pitcher, I evaluate on another year using that year's pitcher-specific transition probabilities.

So for each year, there are 26 (other years) * 2 (general or pitcher-specific) * N (actual pitchers number in that year) train/test pairs. It should be noted that since the actual pitchers appearing in each year vary, only the pitchers appearing in both training and testing years are calculated for pitcher-specific train/test pair. And all the pitchers appearing in test year can be calculated for general train/test pair. Evaluation for all train/test pairs is stored in resultDict. The key is 'pitchername-trainYear-testYear-label(general/pitcher-specific)', and the value is the evaluation result of that train/test pair.

(The code is in calc.py. The whole evaluations logging results for each pitcher for each year can be found in file diff27evaluation.txt)

IV. Experiments and results

Below are several explored results and the possible interpretations for them.

A.Compare pitcher-specific batting strategies and general batting strategies

The results show that policies calculated based on general transition probabilities and pitcher-specific transition probabilities are different. This is reasonable considering different pitcher has different style and capability.

Taking the example used above, for pitcher Max Scherzer, the calculated general batting strategy for year 2017 suggests the batter to swing when there are three balls and no strike, and when there are three balls and one strike. But the pitcher-specific batting strategy suggests the batter to swing only when there are tree balls and no strike. Since we know Max Scherzer performed very well in 2017, the pitcher-specific result takes a very conservative choice.

To evaluate the general batting strategies and pitcher-specific batting strategies more quantitively, we can take a look at the Policy Evaluation results. I also compare those two strategies with intuitive strategies, which means only swinging at batter's count (when the number of balls are greater than the number of strikes, excluding the full count state). When certain strategy's evaluation result of the train/test pair is larger than the other two strategies, I define that strategy wins for this train/test pair. After iterating through these 27 years, the winning count of intuitive strategies is 3069, the winning count of pitcher-specific strategies is 3502, and the winning count of general strategies is 6324. From the result, we can see that in average, the general strategies have the best result, while the pitcher-specific strategies only exceed the intuitive strategies by a small amount. It should be noted that winnings are only counted for the train/test pairs where pitchers appear in both training and testing years.

Another interesting finding is when I separate the pitchers into elite and non-elite groups and calculate the general strategies for two groups, the pitcher-specific strategies winning count increases by a large amount for the non-elite pitchers. It can be interpreted as the non-elite pitchers are more exploitable than the elite ones.

B.Explore changes of general batting strategies along the years

Since the general batting strategies have the best evaluation results, I use them to explore the policy changes from year 1991 to 2017. The result is showed as below:

```
for year:  1991 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  1992 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  1993 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  1994 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  1995 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  1996 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  1997 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  1998 , the policy is:  [0 0 1 0 0 0 0 0 0 0 0 0]
for year:  1999 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  2000 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2001 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2002 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2003 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2004 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  2005 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  2006 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2007 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  2008 , the policy is:  [0 0 0 0 0 0 0 0 0 0 0 0]
for year:  2009 , the policy is:  [0 0 1 0 0 0 0 0 0 0 0 0]
for year:  2010 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2011 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2012 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2013 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2014 , the policy is:  [0 0 0 1 0 0 0 0 0 0 0 0]
for year:  2015 , the policy is:  [0 0 1 0 0 0 0 0 0 0 1 0]
for year:  2016 , the policy is:  [0 0 0 1 0 0 0 0 0 0 1 0]
for year:  2017 , the policy is:  [0 0 0 1 0 0 0 0 0 0 1 0]
```

From the result above, there are some similarities although the year changes: At most time, the policy suggests to swing when there are three balls and no strike, and sometimes it suggests to swing when there are two balls and no strike, or when there are three balls and one strike. These suggestions are reasonable since in these pitch counts the batter is largely favored.

There are also changes along these 27 years. We can see that from 1991 to 1993, the batting strategies are very conservative, and then the result gives a very stable swinging suggestion at three balls and no strike until 2004. From 2004 to 2008, it takes a very conservative period again. We can also see that in recent years, the batting strategies have a tendency to be more aggressive. This may indicate some possible style and power changing of pitchers, since pitchers and batters adapt to each other.

(It should be noted that, the result is also influenced by the correctness of data source, the parameters in reward function I am using, etc. So the result can be only interpreted comparatively. eg. an all-zero result may not really suggest standing all the time at all pitch counts)

C.Experiment with reward function tuning method

Reward function tuning is also experimented in this paper. Used as baseline, the reward for Out state and other non-absorbing states is 0, and the reward for Walk state is 1. The rewards for other absorbing states are tuned in this section. The iteration step is 0.5 for each parameter, and the experimented reward range for each state is

Single: [0, 3]
Double: [reward for Single state + 0.5, 4.5]
Triple: [reward for Double state + 0.5, 6]
Homerun: [reward for Triple state + 0.5, 7.5]

These parameter ranges are justified in the sense that a more privileged state should get higher reward.

The indicator to define if certain group of parameters is better than the others is calculated using the following steps:

(1) Use the experimented parameter with Value Iteration to calculate general batting strategies for each year from 1991 to 2017.

(2) Use Policy Valuation to evaluate train/test pair for all the pitchers and for all the 27 years separately based on general batting strategies learned above and intuitive batting strategies.

(3) For all train/test pairs, count the number where the batting strategies win over the intuitive batting strategies, and

divide it by the total number of train/test pairs. I use this percentage as the indicator of how good this group of parameter is.

(4) Iterate through the range of all the parameters, record the group of parameters that has the highest percentage.

After conducting the steps above, the result obtained is (1.5, 2.0, 2.5, 3.0, 1), which means the reward for Single state is 1.5, the reward for Double state is 2.0, the reward for Triple state is 2.5, the reward for Homerun state is 3.0, and the reward for Walk state is 3.0. The percentage of general batting strategies winning over the intuitive strategies is 0.9938, when using this group of parameters.

However, this parameter tuning method is just an experiment, and doesn't guarantee to find the optimal parameters. First of all, the indicator I use seems reasonable but may not be the most appropriate one in reality. The parameters are smaller than expected, especially when compared to the initial parameters I use in other parts of this paper as (2, 3, 4, 5, 1). In my opinion, it's more reasonable to give a much higher reward to a hit than walk. So the correctness of this parameter tuning method should be explored and justified in the future work.

V. Conclusion

In this paper, I have described the system architecture and elaborated on the Markov Decision Process (MDP) model used to explore the batting strategies when modeling the pitcher behavior as a Markov Process. The states utilized in the MDP, Value Iteration and Policy Evaluation algorithm are also presented with details in the paper. Then Several explorations are described, such as comparisons of general batting strategies, pitcher-specific batting strategies & intuitive batting strategies, changes of pitching styles along the years, reward function parameters tuning methods, etc.

This paper can be seen as a exploration of baseball batting strategies using MDP. And there are several possible ways to improve its correctness and richness in the future:

First, the dataset I use is from http://www.retrosheet.org's play-by-play files. It's an impressive dataset collected and sorted by baseball fans and contains pitch-by-pitch records. However, it doesn't contain information about the types of pitches in terms of fastball, curveball, etc. So I can only simply present the states in MDP as the ball count and strike count. If a richer dataset can be found (it can be possibly joint with some tracking systems), then I can have a more covered state expression of MDP, which may lead to more correct results.

Second, this paper only looks at the pitchers when deciding on the batting strategies. However, as we know, when interpreting the pitching results, the capability of the batter should also be considered. For example, when counting a Homerun state for certain pitcher, we should know it's the result by both the batter and pitcher. And also, when we calculate the batting strategies, each strategy should vary for different batters. So it should be noted that the results from this paper are limited in the sense it just gives a very general batting strategy.

In the last, the reward function tuning method proposed in the paper is just an experiment, a more accurate and verified method should be implemented using some reinforcement learning approach in the future.

References

1. Stallings, J., Bennett, B. and American Baseball Coaches Association, "Baseball Strategies: American Baseball Coaches Association," Human Kinetics, Champaign, IL, 2013.
2. Bickel, J. E., "On the decision to take a pitch,"Decis. Anal. 6 186–193, 2009.
3. Patek, S. D. and Bertsekas, D. P., "Play selection in American football: A case study in neuro-dynamic programming," Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Oct. 1996.
4. Gagan Sidhu and Brian Caffo, "MONEYBaRL: Exploiting pitcher decision-making using Reinforcement Learning," Ann. Appl. Stat. vol. 8, Num. 2, 926-955, 2014.
5. Sutton, R. S. and Barto, A. G., "Reinforcement Learning: An Introduction. ," Massachusetts Institute of Technology, 1998
6. Sidhu, G. and Caffo, B., "Supplement to MONEYBaRL: Exploiting pitcher decision-making using Reinforcement Learning," DOI:10.1214/13-AOAS712SUPP, 2014.