Computing an Optimal Pitching Strategy in a Baseball At-Bat

Connor Douglas
New York University
cpd8405@stern.nyu.edu

Everett Witt
Periwinkle Trading
everett.witt@wustl.edu

Mia Bendy
Capital One
mia.bendy@wustl.edu

Yevgeniy Vorobeychik
Washington University in St. Louis
yvorobeychik@wustl.edu

Abstract

The field of quantitative analytics has transformed the world of sports over the last decade. To date, these analytic approaches are statistical at their core, characterizing what is and what was, while using this information to drive decisions about what to do in the future. However, as we often view team sports, such as soccer, hockey, and baseball, as pairwise win-lose encounters, it seems natural to model these as zero-sum games. We propose such a model for a baseball at-bat, which is a matchup between a pitcher and a batter. Specifically, we propose a novel model of this encounter as a zero-sum stochastic game, in which the goal of the batter is to get on base, an outcome the pitcher aims to prevent. The value of this game is the on-base percentage (i.e., the probability that the batter gets on base). In principle, this stochastic game can be solved using classical approaches. The main technical challenges lie in predicting the distribution of pitch locations as a function of pitcher intention, predicting the distribution of outcomes if the batter decides to swing at a pitch, and characterizing the level of patience of a particular batter. We address these challenges by proposing novel pitcher and batter representations as well as a novel deep neural network architecture for outcome prediction. Our experiments using Kaggle data from the 2015 to 2018 Major League Baseball seasons demonstrate the efficacy of the proposed approach.

Introduction

Baseball is one of the most popular team sports in the U.S., with Major League Baseball (MLB) bringing in over $10 billion in revenue in 2019 (Statista, 2020), and has considerable global popularity as well. With such a large market, teams look to gain a competitive edge, and to this end they leverage complex statistical models generated from increasingly abundant baseball data. Yet, the scope of baseball analytics has been limited primarily to statistical and machine learning approaches, rather than game-theoretic reasoning (Alcorn, 2017; Koseler and Stephan, 2017).

We present the first (to our knowledge) game-theoretic model of strategic interactions in a baseball game. Our focus is a baseball at-bat, an encounter between a pitcher and a batter. In an at-bat, a pitcher throws a series of pitches to a batter. Every at-bat ends in one of two ways: 1) the batter is out (and, in our model, the pitcher wins), for example, after receiving the third strike, or 2) the batter gets on-base, for example, by hitting a home run. In modeling an at-bat, we assume that the goal of the batter is solely to get on base, while the pitcher aims to get the batter out. This focus on the on-base-percentage (OBP) is clearly restrictive (in not accounting, say, for the difference between walks and home runs), but is nevertheless a crucial element of baseball analytics (Albert, 2002; Lewis, 2004).

We model an at-bat as a stochastic game in which the count (of balls and strikes) serves as the state, the pitcher's actions amount to which pitch to throw and where, while the batter decides whether to swing or take (not swing at) the pitch. This model introduces two principal conceptual and technical challenges. The first is that the pitcher and batter decisions are not, in fact, concurrent: the batter does get to observe the pitch as it leaves the pitcher's hand and as it travels towards home plate. On the other hand, the relevant observation window is so short (usually less than half a second) that the batter has little ability to deliberate upon their decision. The particularly salient issue here is that no batter will swing if the pitch is far outside of the strike zone, while when pitches are close, batters vary significantly in their swing propensity (what is often called a batter's "patience" or "eye"). The second challenge is that our model governs the pitcher's intent about where the pitch is thrown. However, pitching data documents only where the pitches *ended up*; we observe nothing explicit about intent.

One way to deal with the challenge of observability is to model the problem as a partially-observable stochastic game (POSG). However, POSGs are notoriously difficult to solve. We propose instead to add an explicit element to our model that allows us to both capture the most salient nature of this partial observability while keeping the main stochastic game structure. Specifically, we use data to learn a batter-specific probability that a batter swings at (relatively) borderline pitches outside the strike zone, and when this is sufficiently high, "override" a batter's decision to swing prescribed by the model by modifying the associated transition to always result in a ball.

We address the second challenge by first decomposing the distribution of outcomes given the pitcher's actions, conditional on the batter swinging, into two parts: 1) distribution of *actual* locations given *intended* locations, and 2) distribution of outcomes based on actual locations. We learn the

former by assuming that the error distribution of the pitcher is Gaussian and by taking advantage of counts in which most pitchers aim to throw a strike most of the time. For the latter, we propose a novel tensor representation of the pitcher and batter, along with a novel deep neural network architecture, and use historical baseball data to learn the outcome distribution for given pitcher-batter pairs. With all the pieces in place, we can solve the resulting stochastic game using standard methods (Filar and Vrieze, 2012; Littman, 1994).

We evaluate the parts of our approach, as well as the game theoretic equilibrium pitch distribution, using 2015-2018 Kaggle data for Major League Baseball. We show that our deep neural network models can successfully capture distributions of outcomes as well as batters' patience, and demonstrate that the proposed approach yields significantly lower predicted OBP for the batters (i.e., higher utility for the pitcher) than the empirical OBP in most counts.

Related Work Several approaches to game-theoretic modeling of sports activities have been previously proposed. In the context of baseball, approaches have tended consider very abstract player strategies Flanagan (1998); Turocy (2008); Weinstein-Gould (2009). For tennis, Walker and Wooders (2001) studied empirically whether top players at Wimbledon play each point according to a mixed-strategy equilibrium. In a later more theoretical effort, Walker, Wooders, and Amir (2011) propose a stochastic game model of win-lose encounters in which the pivotal quantity is a score, which determines the nature of each stage game, and stage games have only two possible outcomes determined by which of the two players win. This stochastic game model is related to ours, but our model has four possible outcomes in each state, violating one central assumptions of Walker, Wooders, and Amir (2011); more fundamentally, our model of a baseball at-bat does not fit their general assumed structure of the stochastic game as transitioning through a series of "point games". Moreover, our goals are different: while Walker, Wooders, and Amir (2011) focus on characterizing the structure of equilibria in this game, our goal is to solve it, and infer the structure of the game from data.

Azar and Bar-Eli (2011) study whether penalty kick interactions between the kicker and the goalie in soccer are representative of mixed-strategy Nash equilibrium play. In this setting, however, there are no environment dynamics, and the game is straightforward to empirically represent. Beal et al. (2020) and Beal et al. (2021) study strategic and tactical decision making in a soccer game. Their models are a blend of Bayesian and stochastic games, but the ultimate approaches do not investigate solutions to these games as such, but focus on a best response to a given opponent strategy.

The increasing importance of sports analytics, particularly in baseball, has given rise to a number of machine learning approaches surveyed by Koseler and Stephan (2017). However, the classes of problems investigated in such approaches have been typically limited to predicting which pitch will be thrown (Ganeshapillai and Guttag, 2012; Hoang et al., 2015), a player's batting average or other offensive statistics (Jiang and Zhang, 2010; Lyle, 2007), likelihood of catching a baseball (Das and Das, 1994), likelihood of winning (Yang and Swartz, 2004), and the like. There are no prior approaches to predict the outcomes for a pair of pitcher and batter at the level of resolution of a single pitch.

Background: the Basics of Baseball

A (U.S. major league) baseball game is an encounter between two teams and proceeds through a series of nine *innings*. Each inning is comprised of two *half-innings*: the *top* half and the *bottom* half. In the top half-inning, the away team bats, while the home team pitches and defends, and the roles reverse in the bottom half-inning. Each half-inning consists of three outs, that is, an inning proceeds until three players on the batting team have registered an out.

The most basic interaction in baseball is an at-bat, which is a faceoff between a pitcher, who throws a baseball towards the home plate area, and a batter standing near this area with the aim to "get on base". A central concept in an at-bat is a *strike zone* (see Figure 1). The vertical range of the strike zone is roughly from the batter's knees to their shoulders, while the horizontal range is the width of the home plate—a white pentagon drawn on the ground. While the strike zone depends on the batter, we'll treat it as a fixed entity for simplicity. Any pitch that is in the strike zone when it crosses the home plate is considered a *strike*, and when the batter observes, or swings through, strike 3, he is automatically out. Conversely, a pitch that is outside the strike zone is called a *ball*, and the batter is automatically on base whenever they observe a fourth ball. A *count* keeps track of the number of balls and strikes in an at-bat, starting at 0-0 (balls-strikes).

At any point, if a batter swings at a pitch, one of four things can happen: 1) a hit, which happens whenever the ball lands in the field of play, or is a home run (leaves the stadium within the field of play), and cannot be reached by a defensive fielder before the batter reaches first base; 2) an out, which happens whenever the a fielder either catches a hit ball on the fly, or can throw it to first base (i.e., to the defender standing with one foot on first base) prior to the batter stepping on it; 3) a strike, if the batter does not make contact with the baseball; and 4) a foul, if the batter makes contact, but the ball lands behind the field of play.

A Baseball At-Bat as a Stochastic Game

Consider an *at-bat*, an encounter between a pitcher and a batter. We start with a 0-0 count (0 balls, 0 strikes). The pitcher throws a pitch, and we anticipate a number of possible outcomes. A home run, a base hit, or a walk, all end an at-bat, as does a strikeout. However, a ball, a strike, and a foul ball may (in the latter case, will always) continue the at-bat, potentially changing the count. For example, a strike or a foul ball in a 0-0 count will always progress the count to 0-1, and if a ball follows, the count progresses again to 1-1. On a 3-2 count, a foul ball returns us to the same 3-2 count, while a ball necessarily results in a walk.

Since there is no evident private information in an at-bat, it is natural to model it as a zero-sum stochastic game. However, a good model is not obvious. First, what are the action sets for the players? For example, the batter does observe the pitch as it traverses the airspace between the pitcher's mound and the catcher's glove, and could use this information (e.g.,

spin, location, speed of the pitch) to decide whether, and how, to swing. Aside from the considerable complexity this introduces, none of this information is available in public baseball data at the level of necessary detail.

We propose a novel model of a baseball at-bat as the following stochastic game involving two players, a pitcher and a batter. First, we consider only two types of outcomes of at-bats: on-base (a hit or a walk) and out. Second, we define the actions of the two players as follows. For the pitcher, let P be the set of all pitch types they can throw, and L the set of possible locations (both in the strike zone, and outside), which we assume to be finite (discretizing the strike zone, and the "non-strike" zone). Let $Z_s \subset L$ be the strike zone, and $Z_b \subset L$ be locations outside the strike zone (see Figure 1 for a particular discretization of both Z_s and Z_b). The pitcher's action space is then $A_p = P \times L$. Note that in

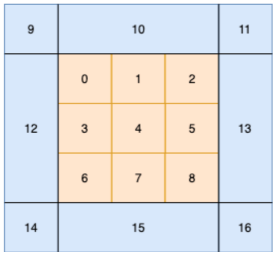

Figure 1: Pitch zone model from the pitcher's point of view: the orange center is the strike zone, while the blue periphery is outside the strike zone. The numbers are numerical labels for associated discrete zone segments.

this context, locations $l \in L$ are *intended* locations: that is, locations that the pitcher is *aiming* for; of course, the pitch may often end up elsewhere, and we will come back to this below. For the batter, we define actions as binary: whether to swing at the pitch (denoted by σ), or to take (not swing; τ), that is, $A_b = \{\sigma, \tau\}$. There is an important complication here: clearly, sometimes the pitch will be so far out of the zone that no batter would plausibly swing; we come back to this below as well. We use $a = (a_p, a_b) \in A_p \times A_b$ to denote the action profile (joint actions) of both players.

We define the state space S as follows. Let $u \in U \equiv \{0, 1, 2, 3\}$ be the number of balls and $v \in V \equiv \{0, 1, 2\}$ the number of strikes, and let b represent the terminal state in which the batter is on-base, while o is the state with the batter being out. The state space is then $S = U \times V \cup b \cup o$. We denote states by $s \in S$.

Since the game is zero-sum, it suffices to define the utility function $u(s, a)$, where $s \in S$ and $a \in A_p \times A_b$ just for the batter. Moreover, since we only care about on-base or outs, $u(s, a) = 0$ for any $s \in U \times V$ (active at-bat), and $u(b, a) = 1$ while $u(o, a) = 0$. Therefore, the expected utility (i.e., probability the batter gets on base) has the natural interpretation as *on-base percentage (OBP)*.

Our final task is to define the transition distribution $T^a_{ss'}$ in this stochastic game. Central to this are two sources of uncertainty conditional on the action choices a of both players: 1) uncertainty about where the chosen pitch $p \in P$ ends up given its intended location l, and 2) uncertainty about the outcome if the batter chooses to swing, i.e., $a_b = \sigma$. Let's start with pitch location uncertainty. Let $D_l(p, l)$ be the distribution over locations $l' \in L$ *where the pitch ends up*, given that a pitch $p \in P$ was aimed at location $l \in L$. To deal with outcome uncertainty, first note that if the batter takes, the outcome is fully determined by pitch location: either the pitch is in the strike zone, in which case it's a strike, or not, and it's a ball (we ignore here the additional complication of umpire mistakes). If the batter swings, there are four possibilities: 1) the batter swings-and-misses (a strike), 2) foul ball, 3) hit, and 4) out after putting the ball in play. Let $\Omega = \{\omega_s, \omega_f, \omega_h, \omega_o\}$ be the set of these four outcomes, respectively. Let $D_\omega(p, l, s)$ be the probability distribution over Ω conditional on the batter swinging, if the pitch is p and the *actual* (not necessarily intended) location of the pitch is l (it also in general depends on count; hence the explicit dependence on state s).

With D_l and D_ω in hand, we can now completely define the transition distribution $T^a_{ss'}$. If $a_b = \tau$, then transition is deterministic and only depends on the count and D_l: 1) if the actual location $l \in Z_s$, the pitch is a strike; then if $v = 2$, $s' = o$, and otherwise $v' = v + 1$; 2) if $l \in Z_b$ (the pitch is a ball), then if $u = 3$, $s' = b$, and otherwise $u' = u + 1$. If the batter swings, transitions are now determined by D_ω. Thus, if the (stochastic) outcome $\omega = \omega_s$, the transition is exactly the same as if the batter took the strike. If $\omega = \omega_f$, then if $v = 2$, $v' = 2$, and otherwise, $v' = v + 1$. If $\omega = \omega_h$, then $s' = b$ (if it's a hit, the batter ends up on base), and, finally, if $\omega = \omega_o$, then $s' = o$ (the batter is out).

If we are given all of the information above, including D_l and D_ω, we can solve this game using the combined value iteration and linear programming approach proposed by Littman (1994), which we review below. The challenge is that not only are these not given explicitly, but D_l is pitcher-specific, and D_ω depends on the particular matchup between the pitcher and batter.

An additional challenge is that in our model above, we effectively assumed that the batter decides whether to swing or take *at the same time* as the pitch is thrown. In practice, batters can partially detect where the pitch will end up, a characteristic commonly referred to as a batter's *patience* (their propensity to swing at pitches outside the strike zone). This ability to discern varies with batter ability. We model this feature by introducing a binary patience function $G(l)$ for $l \in Z_b$, where $G(l) = 0$ means that the batter will in fact take that pitch *even if they intended to swing initially*, while $G(l) = 1$ means that the batter will proceed with swinging. Of course, this, too, is not give a priori.

Next, we describe in detail how we can use baseball data to arrive at estimates of D_l, D_ω, and G. We then put everything together in solving the resulting stochastic game.

Solution Approach

We now describe our approach to computing an equilibrium of the stochastic game representing a baseball at-bat, given a specific pair of pitcher and batter. We start by describing in detail how we learn the aspects of the game model that determine the state transition distribution from past data of

baseball at-bats. Subsequently, we describe how we solve the stochastic game using the well-known solution approach that combines linear programming with value iteration.

Predicting Outcomes

The first missing piece of the game model that must be inferred from data is the distribution $D_\omega(p, l, s)$ that predicts outcomes for a pitch p that is thrown (whether intended to or not) at location l in state (count) s. One of the central aspects of the game model is that this distribution also clearly depends on who is pitching, and who is batting. Let x_p and x_b represent the pitcher and batter, respectively; our goal, more precisely, is to learn $D_\omega(p, l, s; x_p, x_b)$.

A naive idea is to represent players by some 1-dimensional attributes, say batting average for the batter and earned-run average for the pitcher. However, such simplistic representations lose a great deal of information. For example, it is often conventional to talk about some batters as, say, "good low-ball hitters", others as "good fastball hitters", and so on—clearly capturing information that is not simply reflected in simple aggregate statistics. We therefore propose a novel representation of both batters and pitchers, aiming to capture as much readily available information about their past experience as possible, thereby avoiding making specific assumptions about them a priori.

Let's start with batters. We represent batters using a 3D tensor in which 2 dimensions correspond to the orientation of the (strike and ball) zone Z (height and width; see Figure 1, but with zones 10, 12, 13, 15 split up into 3 each to correspond to a matrix, but with associated matrix entries taking on identical values) from the pitcher's perspective. We then associate each pitch with a pair of slices in the third dimension: one for the relatively frequency of swinging at that pitch, and another for batting average if that pitch was thrown in the associated location. Thus, across the x zones, y zones, and 2 values for each of 6 pitch types, we arrive at tensor shape of (5x5x12).

Now, the pitchers. We represent pitchers, just as batters, using a 3D tensor with 2 dimensions corresponding to the zone. We then associate each pitch with a pair of slices in the third dimension: one for the relative frequency of throwing that pitch in the particular location in the zone; consequently, the sum of the entries over the zone and all pitches is 1; and the second for the average velocity of the associated pitch.

Input pitch type and location are represented by tensors with a similar shape, (5,5,6), to pitchers and batters, but use a one-hot encoding, with a 1 in the position corresponding to the pitch and its location, and 0s elsewhere. Finally, we represent count s by two integers, one for the number of balls, and the other for the number of strikes.

Next, we propose a novel deep convolutional neural network architecture for predicting outcomes shown in Figure 2. While the primary innovation is in our representation of batter and pitcher as inputs which we described above, it is not a priori self-evident how to combine them into a single neural network architecture. This problem is reminiscent of sensor fusion (Feng et al., 2021), where the architecture is often intricate, and the ideas do not necessarily transfer here. What we opt for is essentially a *late fusion* architecture. Specifically, as illustrated in Figure 2, we first create separate convolutional architectures separately for the pitcher, batter, and pitch, which (after convolutional layers) ultimately embed each into real vectors. These are then concatenated with the pitch vector z and the count representation s, with the resulting vector passed through several fully connected layers and, eventually, through the softmax layer to obtain the probability distribution over the four outcomes.

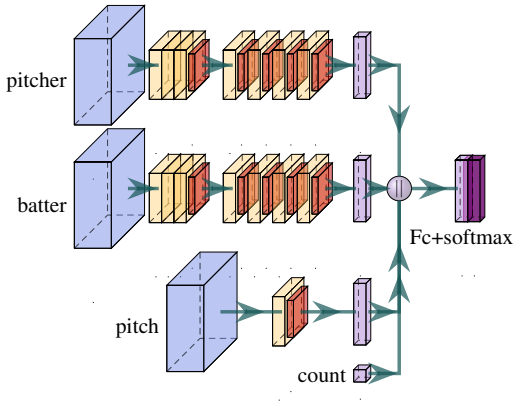

Figure 2: Deep neural network architecture for predicting at-bat outcomes. Input tensors are in blue, conv layers in yellow, max-pooling in red, and fully-connected layers in purple. The circle concatenates the fully connected "embedding" layers. Final softmax layer is in dark purple. All convolutional layer activations are ReLU, while activations in Fc (fully-connected) layers are sigmoid.

Learning Pitcher Control

A typical dataset of baseball at-bats consists of pitches thrown and documentation about where they ended up upon crossing the home plate, whether the batter swung, and the outcome. What is conspicuously missing from such data is *the intended location of the pitch*. Given this state of affairs, it appears at first hopeless to get any handle on the distribution D_l of actual pitch locations, given expected locations. However, we now leverage some baseball conventional wisdom—imperfect, to be sure, but generally quite reasonable—that in certain counts the pitcher just wants to throw a strike. In particular, suppose that we have a 3-0 count. If the pitcher fails to throw a strike, and the batter does not swing, the batter is automatically on base. It is, therefore, generally expected that a pitcher intends to throw a strike. Now, *where* in the strike zone the strike would be thrown remains uncertain, but here we add another reasonable assumption: if the pitcher aims to throw a strike, they will typically aim at the same part of the zone. The final crucial assumption we make is that the spatial distribution of the *error* (in 2 vertical dimensions of the strike zone) is zero-mean Gaussian, and only depends on the pitch type.

With the assumptions above, if we aggregate pitches thrown on 3-0 count for a particular pitcher, we can estimate a mean and the co-variance matrix Gaussian distribution, which in this case consists of 5 parameters for each pitch type p: $(\mu_x^p, \mu_y^p, \sigma_x^p, \sigma_y^p, \sigma_{xy}^p)_{p \in P}$, where x is the hori-

zontal and y the vertical dimension of the strike zone viewed from the pitcher's perspective, with the origin in its center.

The challenge with the approach above is that we do not necessarily have sufficient data for all pitchers on 3-0 counts *for every pitch type* to effectively estimate the Gaussian distribution. To address this issue, we use a deep neural network regression to estimate the Gaussian error model parameters for each pitcher and pitch type. For this, we use the same pitcher representation as described in Section as input.

Representing and Learning Batter Patience

We represent the problem of batter "patience" as the task of learning the probability that the batter swings on a pitch in a particular area outside of the zone. For this purpose, we first split our zones outside the strike zone further, with the presumption that when the pitch is far outside the zone, most batters will not swing, and the variation among batters stems primarily when pitches are relatively close to the strike zone. For each such *borderline* location $l \in Z_b$ and pitch $p \in P$, we learn a binary classifier to predict whether a given batter x_b, represented exactly as described above, will swing. We use the same neural network architecture as the "batter" portion of the network in Figure 2 for this purpose, and learn a separate neural network for each pitch and location.

Solving the At-Bat Game

Littman (1994) presented a general approach for solving finite zero-sum stochastic games which blends value iteration with linear programming. We provide it here for completeness. Note that in general, an equilibrium in a stochastic game entails randomization (*mixed strategies*), and a mixed-strategy equilibrium in which policies depend only on current state, always exists Filar and Vrieze (2012). For our purposes, it suffices to introduce notation for the pitcher's mixed strategies. Let $\Delta(A_p)$ denote the set of mixed strategies for the pitcher. We denote a particular mixed strategy (probability distribution) by $r_p \in \Delta(A_p)$.

The value function $V_i(s)$, which is initialized arbitrarily at iteration $i = 0$, is updated as follows in iteration $i > 0$:

$$V_i(s) = \min_{r_p \in \Delta(A_p)} \max_{a_b \in A_b} \sum_{a_p \in A_p} r_p(a_p) U_i(s, a_p, a_b), \quad (1)$$

where $r_p(a_p)$ denotes the probability of choosing a_p under the distribution r_p and

$$U_i(s, a_p, a_b) = \left[u(s, a_p, a_b) + \sum_{s' \in S} T_{ss'}^{a_p, a_b} V_{i-1}(s') \right].$$

Now, we can observe that in any iteration i, $U_i(s, a_p, a_b)$ is fixed for every state, and therefore Equation (1) defines a zero-sum game, which can be solved for the equilibrium mixed strategy for the pitcher, r_p, using linear programming Shoham and Leyton-Brown (2008).

Experiments

Experiment Setup To train and test our models, we use a Kaggle dataset comprised of at-bats from 2015-2018 MLB seasons (Kaggle, 2018). For each at-bat, the dataset includes the pitcher and batter, the pitch thrown, along with its final coordinates as it crosses the home plate, count, and outcome (hit, foul, ball, etc). Our total dataset includes 2696132 pitches thrown over 730585 at-bats. Our training/test splits for each model ensured no overlap in pitchers or batters between the two. Experiments were run on a Macbook Air 2020 (1.1 GHz dual-core i3, 8 GB RAM) running macOS 10.15.5. Neural networks were built in TensorFlow 2.5.0, and all source code was written in Python.

Throughout, we evaluate our approaches using batches of at-bats involving pitchers and batters that we categorize as *strong* (well above average), *average* (around average), and *weak* (well below average). The categorization is based on ranking either pitchers or batters in terms of empirical OBP (for pitchers, this would be OBP against). We present results for two such pairing groups: 1) strong pitchers vs. weak batters and 2) weak pitchers vs. strong batters.

Outcome Predictions We begin by evaluating the efficacy of our neural network model for predicting outcomes of thrown pitches conditional on the batter swinging. Our test set of these outcomes comprised 147799 pitches. Since predictions are distributions, we compare the average predicted probabilities of the four outcomes (strike, foul, out, and hit) for two large batches of at-bats: one pairing a strong pitcher with a weak batter, and another pairing a weak pitcher and a strong batter. Both the predicted probabilities and empirical observations, are averaged over all corresponding counts and at-bats from test data.

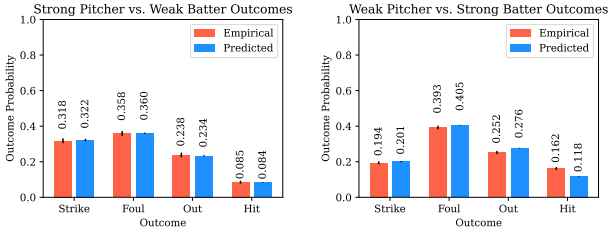

Figure 3: Average predicted probabilities and empirical observations of the four outcomes of swinging at a pitch (strike, foul, out, and hit).

The results are shown in Figure 3. First, we can observe that the predicted outcome distribution tracks empirical values very closely. Of particular importance to us is that the relative probabilities are closely preserved: for example, foul balls are the most common whatever the matchup, and a strong pitcher facing a weak batter is far more likely to get a swing-and-miss strike than if the relative pitcher-batter strength is reversed. Figure 4 offers additional demonstrations of the prediction efficacy in terms of tracking relative probabilities for different settings. Comparing 3-0 and 0-2 counts, for example, one would expect a higher likelihood of a hit in the former than the latter, which we indeed observe (for both predicted and empirical distributions). Similarly, we observe a significantly higher likelihood of a strike in an 0-2 than a 3-0 count, and we again observe this. Analogously, contrasting outcomes when the pitch thrown ends up outside the zone (a ball) vs. in the zone (a strike), we see a

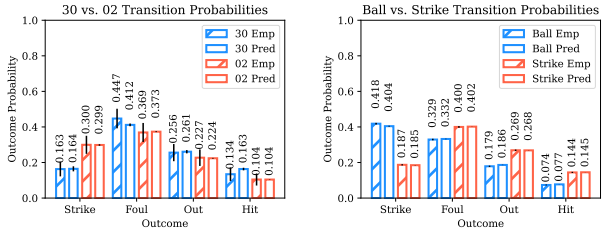

Figure 4: Average predicted probabilities and empirical observations of the four outcomes of swinging at a pitch (strike, foul, out, and hit).

far greater likelihood of a swing-and-miss in the former case than the latter, but lower likelihood of a hit.

Control Predictions We evaluate our model of control in two ways: first quantitatively, and then qualitatively, using 118 pitchers observed in 3-0 counts. First, we consider the accuracy of predicted Gaussian model covariance matrix parameters in terms of mean-squared error (MSE). The MSE in the x-dimension is 0.036, in the y-dimension it is 0.053, and the MSE for the covariance of x and y is 0.025. These are quite small compared to empirically observed variances in either the x (0.45-0.55) and y (0.55-0.67) dimensions for the 4-seam fastball (these are higher for other pitch types).

We also see predicted variances and empirical variances drop in both the x and y dimensions with improving pitcher quality. That our model captures this trend suggests that our tensor representations of pitchers do indeed reflect the innate skill of players.

Patience Predictions Next, we evaluate the efficacy of our approach of batter patience prediction over 126145 test pitches thrown outside of the strike zone, but sufficiently close for batters to swing at them. The results are shown

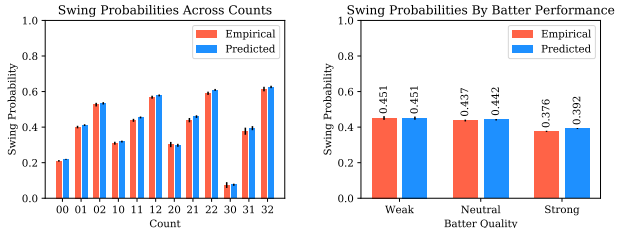

Figure 5: Average predicted probabilities and empirical observations of the outcomes of swinging at a pitch.

in Figure 5, where we consider variation by counts (left), and by batters with differing degrees of success (right). First, note that predictions are remarkably close to empirical swing probabilities. Second, we can observe qualitative trends for both empirical and predicted swing propensities that are consistent with common baseball intuition. For example, batters very rarely swing at pitches outside the zone on a 3-0 count, and very frequently on 2-strike counts. Second, the top batters (those in our *strong* category) are indeed noteworthy in that they tend to be more patient.

Overall Effectiveness Finally, we compare the effectiveness of the proposed *stochastic game (SG)* approach in terms of on-based percentage (OBP) in equilibrium with empirical OBP. Our first observation is that, overall, *SG* significantly reduces OBP: over 3600 matchups on test data, the empirical OBP is 0.329, whereas the OBP in *SG* equilibrium is **0.242**, more than a 25% reduction!

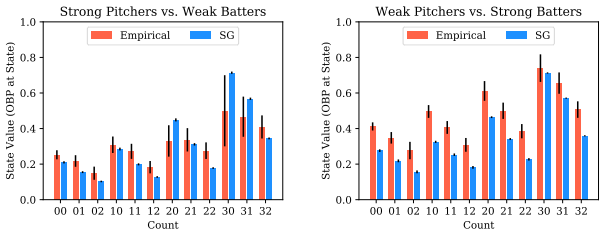

Figure 6: Stochastic game (SG) OBP vs. empirical OBP, for different matchups and counts.

In Figure 6, we delve deeper by comparing *SG* and empirical OBP for different counts, and contrasting, in particular, two classes of matchups: 1) strong pitchers paired with weak batters, and 2) weak pitchers paired with strong batters. First, note that we see expected variation for different counts: for both *SG* and empirical OBP, counts that favor the batter (2-0, 3-0, and 3-1) have dramatically higher OBP than those that favor the pitcher (0-2, 1-2). What is especially interesting, however, is that *SG* does not exhibit much advantage in batter-favored counts. However, *SG* offers a considerably greater advantage when the count is roughly even, or the pitcher is ahead. Furthermore, *SG* holds a far greater advantage for weaker pitchers: for example, on 0-0 count, empirical OBP in these matchups is, on average, 0.412, while *SG* yields an OBP of 0.276. Indeed, we see this in a number of counts, with an advantage of *SG* in a 1-0 and 1-1 counts particularly significant.

Conclusion

We proposed a novel game-theoretic model of a baseball at-bat as a stochastic game, and showed that it yields significantly lower OBP in nearly every count. The key challenge of this model is deriving the transition probability from empirical at-bat data for an arbitrary pitcher-batter pair. We present a novel approach for doing so which is based on first decomposing the transition distribution into several parts, and propose deep neural network approaches for learning the key parts from data. Our experiments demonstrate the efficacy of the proposed approach, showing, in particular, that the distribution of pitches and locations we compute in the resulting stochastic game yields significantly higher utility for the pitcher (lower on-base percentage for the batter) in nearly every count. Moreover, we show that the proposed approach is particularly beneficial for average and below-average pitchers.

Acknowledgments This work was partially supported by the NSF (IIS-1905558, IIS-2214141) and ARO (W911NF1910241, W911NF1810208).

References

Albert, J. 2002. A baseball statistics course. *Journal of Statistics Education* 10(2).

Alcorn, M. A. 2017. (batter/pitcher)2vec: Statistic-free talent modeling with neural player embeddings. In *MIT Baseball Analytics Conference*.

Azar, O. H., and Bar-Eli, M. 2011. Do soccer players play the mixed-strategy nash equilibrium? *Applied Economics* 43(25):3591–3601.

Beal, R.; Chalkiadakis, G.; Norman, T. J.; and Ramchurn, S. D. 2020. Optimising game tactics for football. In Seghrouchni, A. E. F.; Sukthankar, G.; An, B.; and Yorke-Smith, N., eds., *International Conference on Autonomous Agents and Multiagent Systems*, 141–149.

Beal, R.; Chalkiadakis, G.; Norman, T. J.; and Ramchurn, S. D. 2021. Optimising long-term outcomes using real-world fluent objectives: An application to football. In Dignum, F.; Lomuscio, A.; Endriss, U.; and Nowé, A., eds., *International Conference on Autonomous Agents and Multiagent Systems*, 196–204.

Das, R., and Das, S. 1994. Catching a baseball: a reinforcement learning perspective using a neural network. In *AAAI Conference on Artificial Intelligence*, 688–693.

Feng, D.; Haase-Schütz, C.; Rosenbaum, L.; Hertlein, H.; Gläser, C.; Timm, F.; Wiesbeck, W.; and Dietmayer, K. 2021. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *Transactions on Intelligent Transportation Systems* 22(3):1341–1360.

Filar, J., and Vrieze, K. 2012. *Competitive Markov decision processes*. Springer Science & Business Media.

Flanagan, T. 1998. Game theory and professional baseball: mixed-strategy models. *Journal of sport behavior* 21(2):121.

Ganeshapillai, G., and Guttag, J. 2012. Predicting the next pitch. In *Sloan Sports Analytics Conference*.

Hoang, P.; Hamilton, M.; Murray, J.; Stafford, C.; and Tran, H. 2015. A dynamic feature selection based lda approach to baseball pitch prediction. In *Trends and Applications in Knowledge Discovery and Data Mining*. 125–137.

Jiang, W., and Zhang, C.-H. 2010. Empirical bayes in-season prediction of baseball batting averages. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown*. 263–273.

Kaggle. 2018. Mlb pitch data 2015-2018. https://www.kaggle.com/pschale/mlb-pitch-data-20152018.

Koseler, K., and Stephan, M. 2017. Machine learning applications in baseball: A systematic literature review. *Applied Artificial Intelligence* 31(9-10):745–763.

Lewis, M. 2004. *Moneyball: The art of winning an unfair game*. WW Norton & Company.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 157–163.

Lyle, A. 2007. *Baseball prediction using ensemble learning*. Ph.D. Dissertation, University of Georgia.

Shoham, Y., and Leyton-Brown, K. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Statista. 2020. Major league baseball total league revenue from 2001 to 2020. https://www.statista.com/statistics/193466/total-league-revenue-of-the-mlb-since-2005/.

Turocy, T. L. 2008. In search of the" last-ups" advantage in baseball: A game-theoretic approach. *Journal of Quantitative Analysis in Sports* 4(2).

Walker, M., and Wooders, J. 2001. Minimax play at wimbledon. *American Economic Review* 91(5):1521–1538.

Walker, M.; Wooders, J.; and Amir, R. 2011. Equilibrium play in matches: Binary markov games. *Games and Economic Behavior* 71(2):487–502.

Weinstein-Gould, J. 2009. Keeping the hitter off balance: Mixed strategies in baseball. *Journal of Quantitative Analysis in Sports* 5(2).

Yang, T. Y., and Swartz, T. 2004. A two-stage bayesian model for predicting winners in major league baseball. *Journal of Data Science* 2(1):61–73.