Baiting and Ambushing in Baseball: A Game Theoretic Approach

Anton Dahbura and Jaspar Carmichael

Johns Hopkins University | Whiting School of Engineering | Baltimore, MD

Design Day 2022

Introduction

The idea of ambushing is one that has been present in the sport for as long as the game has been around but hasn't been formally acknowledged. A player can theoretically deviate from their expected strategy to surprise, or "ambush", their opponent, effectively gaining an advantage over them. For example, a pitcher may elect to throw a fastball down the middle in an 0-2 count when the hitter is expecting him to throw a "waste" pitch, catching the hitter off guard. In this project, we establish a game-theoretic model along with a simulation environment written in Python to capture this relationship and investigate questions regarding when, how, and for how long a player should deviate to gain the largest possible advantage from "ambushing" their opponent in certain situations.

Objectives

The goal is to use game theory principles and a simulation environment to investigate the details behind the idea of ambushing in the game between the hitter and the pitcher. The simulations will allow us to visualize the results of either hitter deviating and from these visuals, meaningful conclusions can be drawn.

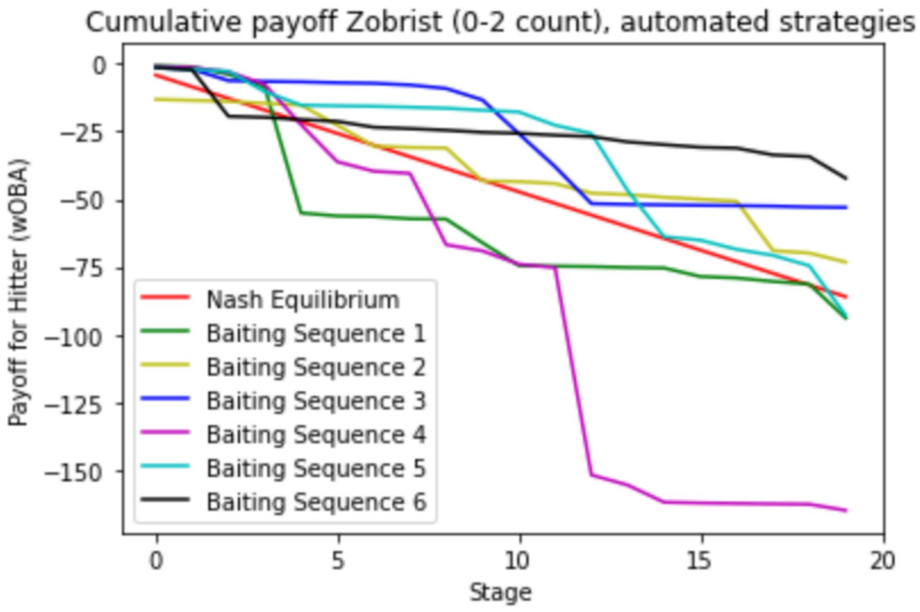
Methods

There are two aspects to the project: the game theoretical model and the simulation environment. First, I will describe the game theoretical model. The game between the hitter and the pitcher is a zero-sum mixed strategy game. The payoff matrix accounts for 4 situations: the pitcher throws either a ball or a strike and the hitter either swings at the pitch or takes it. The payoff for each outcome is given in wOBA (weighted On Base Average). The probabilities of an umpire mistake, the batter hitting a strike if he swings, the batter hitting a ball if he swings, the probability of the batter hitting a foul ball are, and the corresponding wOBA weights used to determine the payoff for each scenario. Then, the payoff matrix is used to determine the Nash Equilibrium. The Nash Equilibrium strategies for each player are viewed as a compromise between the two players, and once this compromise is broken, each player is attempting to gain an advantage over their opponent. The simulation environment was created in Python to visualize the results of two players deviating between strategies. In the simulation, the pitcher and the hitter are initially playing their Nash Equilibrium. One player deviates from their expected strategy to "bait" their opponent off of the NE strategy. After this, each player is jumping around to different strategies based on what they think their opponent is playing. The simulation goes on for 20 stages, and it keeps track of the cumulative payoff earned throughout the simulation. The output is a graph depicting this cumulative payoff in comparison to the situation where the players continue playing their Nash Equilibrium strategies. This visual is shown and discussed to the right.

There are two aspects to the simulation environment: the detection mechanism, and a mechanism to determine the optimal strategy to switch to. The detection mechanism uses the binomial distribution and a predetermined confidence threshold. Using the inferred opponent strategy, the binomial distribution for a given number of successes (strikes thrown or pitches swung at) determines the probability that the sequence could have occurred. If this probability is below the confidence threshold, then that is deemed a deviation. Once a deviation is detected, the program determines the optimal strategy to switch to in response to the inferred strategy.

To determine what strategy to switch to next, the expected cumulative payoff until detection (ECPUD) is computed for each possible strategy 0 through 1 and the optimal value is picked as the new strategy. The ECPUD is calculated by multiplying the expected events until detection by the payoff per event for the combination of strategies. In this way, the program effectively chooses an optima response for each player.

Results



Shown above is the output of multiple instances of the simulation program overlayed on a single plot. Each instance is run using the game theoretical model produced by Major Leaguer Ben Zobrist's offensive statistics from the year 2014. Additionally, each instance represents the 0-2 count for the sake of example. It is worth noting that the results would look different for each different count. As mentioned before, the simulation happens over 20 stages, where each stage is the period in which two unique strategies are being played. The current stage ends, and the next stage begins when one player detects a deviation and switches their strategy accordingly. Each line represents that instances cumulative payoff over the 20 stages. The red line represents the situation in which both the hitter and the pitcher maintain their NE strategies.

Each of these instances is unique, as there is a stochastic nature to this simulation due to the probabilistic nature of the mechanism.

When analyzing this visual, it is important to identify the meaningful comparison. In this case, comparing each instance to the Nash Equilibrium instance is useful. In each instance, it is obvious that the pitcher wins each one as the payoffs are severely negative. However, this is just inherent in the 0-2 count. The hitter is at a severe disadvantage in an 0-2 count as the pitcher can be the aggressor, so it is just natural that the hitter finds themselves at a disadvantage. Thus, it is more useful to compare each instance to the Nash Equilibrium instance to see if the deviation by the hitter gives them an advantage compared to that alternative.

Three of the instances end up on the positive side of the Nash Equilibrium line and the other three below it. In two of the cases that fall below the line, they are much closer to being positive. The single outlier happens because the state of the game stays in a position that gives the pitcher a large advantage for a large time. This is prone to happen due to the stochastic nature of the process. This visual depicts this baiting and ambushing sequence as a tug-of-war where each player can find a strategy that puts them at an advantage over their opponent. While the results don't provide clear proof that the hitter gains an advantage, it seems that when looking at the whole picture this visual provides, the hitter is more likely to put themself in a more advantageous position by first deviation to an unexpected strategy.

Conclusion and Future Work

From these results, it is difficult to make any clear assumptions. What can be concluded, however, is that the baiting and ambushing sequence unlocks players' abilities to game their opponents and possibly put themselves in a more advantageous situation. The trends depicted in the visual support this sentiment. It is clear that to determine concrete conclusions, more work is needed.

In terms of future work, there is on area of large significance in the picking of an optimal strategy mechanism. We would like to formalize the process of determining the expected cumulative payoff until detection. This value would be used to pick an optimal strategy to switch to. As the program stands, this process is completed by experimentally using the detection mechanism to determine the average expected events until detection for 1000 instances. By finding a formula to determine the ECPUD, this will drastically reduce the runtime of the program and make our results more precise. Once this aspect is resolved, exploring different hitters, counts, and other simulation configurations will allow us to gain more insight into this relationship.