Pokerbots Course Notes

Massachusetts Institute of Technology
6.176: Pokerbots Competition*

IAP 2022

Contents

*Contact: pokerbots@mit.edu

1 Lecture 3: Game Theory

This lecture is taught by Matthew Mcmanus.[1] and Alexander Zhang [2] All code from lecture can be found at the public GitHub repository https://github.com/mitpokerbots/reference-lecture-3-2023. The slides from this lecture are available for download on GitHub.

At this lecture (and all others), we raffle off a pair of Sony MX4s. Attend lectures in person if you want a chance to win similar prizes like these!

We'd also like to thank our 12 Pokerbots 2023 sponsors for making Pokerbots possible. You can find information about our sponsors in the syllabus and on our website, and drop your resume at pkr.bot/drop to network with them. Our sponsors will also be able to see your progress on the scrimmage server, giving you a chance to stand out over the course of our competition.

Finally, a reminder that Mini-Tournament 1 is tomorrow! Be sure to **submit your bot by 11:59pm EST on 01/13/2023** to receive credit for this course, the winner of this week will win $1000.[3] If you're pressed for time, feel free to incorporate some of the code we wrote in lectures 1 or 2, available at our public GitHub repository.

1.1 Pre–Lecture Game Rules

Before the lecture formally begins, we have two games for you to play. The first, which will be today's giveaway, is everyone submits a number between 1 and 1000 inclusive, and the person who guesses closest to $\frac{2}{3}$ of the average wins.[4] The second game winner is the person with the highest last 4 digits of their MIT ID.

1.2 What is a game?

There are many kinds of games, but we will only consider two–player zero–sum games. Two–player, very naturally, means there are only 2 players. The technical definition of a zero–sum game is one where any value won by you is lost by your opponent, and vice versa; there is no value created by winning the game. Examples of these games are: rock, paper, scissors (referred to "RPS" going forward); "my number is bigger than yours" (a game where two players say a positive integer and the one with the largest integer wins, referred to as "MNIBTY" going forward), and even chess or heads–up poker.

A *game* between players 1 and 2 consists of a pair of strategy sets S_1 and S_2, and a utility function u. Examples of possible strategy sets are your bot going all in, or only check–calling. The utility function u outputs R, the utility for player 1, so the negative of this is the utility for player 2. In poker, u is viewed as simply the number of chips. Note that strategies are submitted simultaneously. Naturally, player 1 wishes to maximize u, and player 2 seeks to minimize u. This notation works very well for two–player zero–sum games, as each player is simply trying to maximize their own utility function.

1.3 Pure and mixed strategies

Let's use RPS as our main example. RPS has 3 pure strategies: always rock, always scissors, or always paper. A *mixed strategy* is any that uses randomness (a probability distribution) to interpolate between *pure strategies* (e.g. flip a coin, and heads implies scissors and tails implies rock). If you're using a mixed strategy, for example, $p(rock) = .4$, $p(paper) = .3$, $p(rock) = 3$, you must choose one of the three pure strategies for a specific round. The reason we analyze mixed strategies is that a lot of fundamental game theory is only truly applicable when we let people use mixed strategies, choosing between various pure strategies. A pure strategy in RPS is not good, as it is easily exploitable if your opponent knows the strategy (e.g. always rock is beaten by always paper). A mixed strategy is, on the other hand, undeterministic decreasing your opponent's ability to predict your actions.

[1] Email: andyzhu@mit.edu.
[2] Email: haijiaw@mit.edu.
[3] Bots must be submitted to our scrimmage server at pkr.bot/scrimmage.
[4] This is called the "Beauty Contest" because there was initially a newspaper advertisement where you had to rank people's attractiveness, and the winner was the person with $\frac{2}{3}$ of the average rank.

One of the many ways to represent games is through a matrix. The rows correspond to the pure strategies player 1 is allowed to take, and the columns to the pure strategies player 2 is allowed to take. The payoffs we usually write represent the payoffs to player 1. RPS is also easy to create a matrix out of, because each player has a finite number of possible strategies at every stage of the game—thus, it is a matrix form game. Doing this allows us to compute the utility of a strategy S_1, as we can average payoffs of the pure strategies using the probability of picking such a pure strategy. We use $+1$ to represent the payoff of winning rock paper scissors, and -1 to represent losing; note that the magnitude does not truly matter as long as we are consistent.

	Rock	Paper	Scissors
Rock	0	-1	$+1$
Paper	$+1$	0	-1
Scissors	-1	$+1$	0

Another thing about RPS that makes it special is that it is a symmetric game, as switching the players or payouts doesn't affect the strategies.[5] This can be seen by the matrix as its transpose (switching the players) equates its negative (switching the payouts).[6]

1.3.1 The Keynesian Beauty Contest

Now, we're going to talk about the first game that we played. This is also a matrix form game, as each player has 1001 possible strategies; however, it has a multidimensional matrix, as there are multiple players each with their own strategies.

We'll analyze "rational" play for this game. The first thing that you notice about the game is we're trying to guess $\frac{2}{3}$ of the average, and the maximum possible number is 1000; therefore, whatever we play, we shouldn't pick above $\frac{2}{3}$ of the max, or 667. So, if no one is going to play over 667, the average won't be higher than $\frac{2}{3}$ of this, or 445. By the same logic, we shouldn't guess more than $\frac{2}{3}$ of this number. Continuing in this fashion, no rational player should play anything other than 0 or 1, depending on how you round. Notice how "rational" here is in quotes—this is because you're trying to win the Sony headphones, and not necessarily play rationally, as this does not always translate well to practice. This demonstrates why there's a distinction between playing game–theoretic optimally, and playing to win Pokerbots—something interesting to keep in mind while developing your bot's strategy.

1.3.2 Dominance

Next we'll talk about dominance: we say that strategy A "dominates" strategy B if playing A is `always` a better idea. Quantitatively, the utility of strategy A is always greater than B. In the Beauty Contest, submitting 667 dominates the strategy of submitting 1000, as we multiply the average by $\frac{2}{3}$. We can take this idea further by talking about "second–order dominance;" once we've crossed out the first dominant strategy, we can do the same consideration again. This is where we multiplied 667 by $\frac{2}{3}$ again to get 445. You can recursively apply this definition to even get to "infinite-order" dominance, which is when you only play strategies that are not dominated to any order. Considering the beauty contest, we cannot make the assumption that every player is hyper-rational. Therefore we wish to find the numbered order of dominance that best estimates the number of these hyper-rational players.

This takes us to a point where no strategy can dominate another any more. It's helpful to define the point of an "equilibrium:" an *equilibrium* is a set of strategies, one for each player, such that nobody has an incentive to switch.[7]

Lets' look at a quick example (slide 17). If player 1 plays A and player 2 plays C then there is a positive output of +3 for player 1 and -3 for player 2. In this case this is the worst possible output for player 2. So, if player 2 instead plays D the they will only lose 2 instead of 3. So, it is in their best interest to always go with D if player 1 plays A. Now lets look at the second row, player 1 playing B. Without going into the examples

[5] Note that being a symmetric game does not imply that its matrix is symmetric.

[6] This is referred to as the matrix being "skew–symmetric."

[7] Note that by "strategies," here, we are talking about mixed strategies. The "incentive to switch" means they cannot increase their expected utility by modifying their mixed strategy in any way.

we can see player could either get +1 or -1. Compared to going with option A where they can either get +3 or +2 it will always be their best interest to play A instead of playing B. As we discussed earlier if player 1 plays A then player 2 will always play D. Seeing this we then know there is a Nash Equilibria at (A,D).

1.4 Nash Equilibria

You've probably heard the words "Nash Equilibrium" thrown around before. It may be very easy to get confused and think that a Nash Equilibrium refers to a perfect strategy; this is incorrect. A Nash equilibrium simply means a set of strategies such that no player has a point in switching, but this does not always make it the best strategy. You will see an example of this when we go over the beauty contest statistics.

A Nash equilibrium happens when everyone plays 0 in the beauty contest game (definition on slide 13). Even if you know that everyone else is playing 0, 0 is still $\frac{2}{3}$ the average of 0 so you have no incentive to switch, nor does anyone else. In the game RPS, there's no pure strategy equilibrium, as for every pair of player strategies players will be incentivized to switch to the pure strategy that dominates their opponent.

There is an "equilibrium" for RPS, which is playing each pure strategy $\frac{1}{3}$ of the time (slide 20). Let r, p, s be our probabilities of playing rock, paper, and scissors, respectively. Slide 21 shows the calculation for why both us and our opponent playing this same strategy is an equilibrium, as neither of us has an incentive to switch with constant guaranteed EV. Our opponent wants to pick a strategy r, p, s which minimizes our utility, and we want to maximize our utility given that our opponent is attempting to do this. The solution to our min–max relation guarantees us at least 0 value, which you may think isn't very good, but this means that on average we are never losing. If a pokerbot was able to get this sort of performance in a tournament, it would almost guarantee them winning as they would never be losing money on average.

Now we'll analyze a more complicated game that is asymmetric, themed around military battle plans (slide 23).[8] I can choose to either charge or sneak attack against my opponent, and they can respond with either a full defense or defend in shifts. The matrix below displays my utility for each case:

	Full Defense	Defend in shifts
Charge	0	+3
Sneak attack	+1	−1

Looking at this matrix, there are mostly positive numbers; thus, before mathematically analyzing, we'll intuitively expect player 1 to have positive utility on average. Also, this matrix is asymmetric, as switching the players and strategies changes the payoffs more than just multiplying by negative 1 (as one can easily tell by the +3 payoff).

Let c, s be the probabilities of me launching a charge and sneak attack, respectively. My expected values for utility as a function of my opponent's strategy then becomes:

$$\mathbb{E}[\text{opp full}] = c \cdot 0 + s \cdot 1 = s,$$

$$\mathbb{E}[\text{opp shifts}] = c \cdot 3 + s \cdot (-1) = 3c - s.$$

Let's suppose we want to find the values of c and s so that we are always guaranteed a certain expected value in this game no matter what our opponent picks—that is, we want to find c, s such that $\mathbb{E}[\text{opp full}] = \mathbb{E}[\text{opp shifts}]$. Calculating, we find that $c = 0.4, s = 0.6$ satisfies this constraint; substituting into either expected value equation tells us that no matter what our opponent picks, we have an expected utility of 0.6. Hence, the "value" of this game is 0.6 for us, or we are claiming value by playing this game.[9,10]

A famous quote, that many of you may have seen in *A Beautiful Mind*, is the guaranteed existence of a Nash equilibrium in every finite game using mixed strategies. This was proven by Nash himself, in 1951, using topology.[11] Note that this theorem only applies to finite games, which MNIBTY is not—for every

[8] This game is themed around military battle plans, as game theory was actually developed significantly by military strategists at the RAND Corporation, a large think tank, during the 1950s!

[9] Note that if you use this value to guarantee a minimum expected value of utility in other games, you can get nonsensical numbers if dominated strategies are present.

[10] You can always use this computational method, even when more strategies are available, using linear programming for those of you who are familiar with it.

[11] The specific theorem is called the fixed point theorem, which some of you mathematicians in the room may have heard of.

strategy you can shift yours over to beat your opponent. We are also only referring to games as we defined them, where they are matrix form, you have a set of strategies for each player, and a deterministic utility function which given these strategies assigns a utility for each player.

1.5 Applications to poker

I've shown you that Nash equilibriums exist, which may sound very weak for practical applications, but this information is very useful for finding it in different games; now let's look at it for poker. First, we have to prove that poker is a *finite game*, but then lets talk more about what equilibrium actually means in the context of poker. RPS is very easy to represent as a matrix, but other times its more natural to represent games as a *tree*. This representation may be more natural to you if you've taken computer science classes using graph theory.

We'll look at the concept of "extensive form games;" we can represent a game that's played sequentially, such as poker, by game trees (slide 27). Consider the game of tic-tac-toe, which has its tree displayed on slide 28. You'll notice that tic-tac-toe is very symmetric, so we've made the tree smaller already—however, its tree is still very large. An extensive form game can be represented by a finite (possibly large) game tree where the nodes represent game states, levels of nodes are alternating players, and eventually you reach a leaf which is a number representing player 1's utility, rather than another game stage. Poker is an extensive form game, where you're using the information your pokerbot gets from the `GameState` to build your game tree. It's clear that chess is another extensive form game, but with chess the game tree is incomprehensibly large compared to tic-tac-toe's given the sheer number of possible moves. These game trees can get really big because they grow exponentially, especially permutation poker with its combinatorial explosion; however, they can be dealt with through a technique called *"backwards induction."*

Looking at tic-tac-toe's game tree, we can start at some point late in the game and consider a specific section of the tree (slide 30). Note that the leaf nodes have their player 1 payoffs marked in blue. Next, note the black number next to certain nodes—this represents from that game stage onward, only the black number is possible as a payoff. We can do this by going upwards from leaf nodes, calculating what a player playing optimally would rationally choose between the game stages available to them at the level below that state. This upwards traversal means that eventually we'll reach the start node of the game, giving the game itself a payoff for each player. This approach has been applied to games as big as Connect Four, which is strongly solved, but not much bigger than that.

Every extensive form game—such as tic-tac-toe—is also a "normal form game," meaning that it can be expressed in matrix form. The problem with writing this matrix, however, is that it grows exponentially in nature, because the number of strategies for each player is equivalent to every possible non-end-stage of the game. This matrix in fact is doubly exponential, as the number of both row and column strategies is exponential. This existence, however, means that tic-tac-toe has a Nash equilibrium in existence.

However, there's a problem–we're still not working with poker! Poker has an unimaginably large doubly exponential matrix, but since all bets are in integral amounts of chips, it has a finite matrix. Therefore we know the existence of a Nash equilibrium for poker. There is one additional caveat for poker, however: the reason we can't explain poker with this type of game tree is because poker has incomplete information. With poker, you can have a situation like the one displayed on slide 35. Nodes in the poker game tree can be reached not only by player actions, but also by "nature:" elements of randomness. When the engine tells you there are three new cards and that's what the flop is, this is randomness, as well as when your opponent flips their cards over. Randomness is the following tree representation of our poker game is represented by player R. The dotted lines on the game tree explain that players don't know which node they're situated at because of what hand their opponent has, which is why the game tree becomes imperfect. R always leads to a node labeled 1, which represents player 1's turn. They then have two options (to fold or to bet), which leads to a player 2 scenario. One of the key aspects of poker is that our opponent does not know our hand—that is our secret knowledge—so this is represented in the game tree by our player 2 nodes being linked by a dashed line. Our opponent cannot tell whether we were dealt the AA or the AK, so the game states are effectively equivalent to our opponent. They can see our game tree movements, but not our particular game state. This is why poker is such a beautiful game to play. The question then becomes whether we can still apply backwards induction to these types of game trees; the answer is we cannot, due to these dashed lined which we refer to as "information sets." These dashed lines throw off the backwards induction process.

Poker is also a matrix game, as for every type of strategy that our opponent takes—betting, calling, or folding, for example—it's possible for us to create a responsive strategy based on the information that we have. This shows that Poker does have a Nash equilibrium after all, since it can be expressed in matrix form! This equilibrium strategy will have average payout 0, which will guarantee that we do not lose on average—that is, we cannot lose; however, this matrix is double exponential in size. Thus, computing the Nash equilibrium is near–infeasible due to computational complexity. Methods do exist to do this using linear programming, however, but even then it's still very difficult due to complexity reasons.[12] The answer is also no for a different reason. If some of you were there for our pre–registration game Blotto, you may realize that game also has a Nash equilibrium; however, in Blotto, the optimal strategy is not to solve for the Nash equilibrium but rather to play against the other players. Similarly, in poker, your goal is to beat the other players, not solve for the Nash equilibrium. If you cannot solve for the Nash equilibrium, that is totally okay; your goal is to take your opponents chips. If you can play strategies that beat multiple opponents and take their chips, then you have a great shot at winning this tournament. If you target specific opponents, we call this an "exploitation strategy;" sometimes, these can do better than approximating or solving for the Nash equilibrium.

You can attempt is to approximate the Nash; this strategy will do pretty well on average. In practice, playing the Nash equilibrium in RPS for example is not always the best idea. For example, if you play an opponent who uses rock with probability 0.6, you want to play paper more than the Nash equilibrium suggests you do. The Nash will guarantee that on average you play the value of the game, and in each of the subgames in the game tree it will also have you play optimally in that situation. What this means is that you could definitely come up with a strategy that is not the Nash equilibrium that plays equally against the Nash, but finding it is computationally infeasible. The Nash equilibrium in poker, on the other hand, will always do pretty well, but might not be the best strategy against certain strategies which are really suboptimal that you could crush with a different strategy—it's up to you to determine when to approximate the Nash and when to do something different that makes your bot great rather than good.

Examples: Applied Game Theory

YouTube links:

1. https://www.youtube.com/watch?v=hRIXXCe0Hi0
2. https://www.youtube.com/watch?v=WqP3fj3nvOk

1.6 MIT ID Game Discussion

Just like the beauty contest game, we're going to analyze rational play for the MIT ID game. There's again a cutoff—this time, people would rationally only play if their last four digits are in the 90,00+ range. Again, however, people play to win, not necessarily rationally. People will always behave in ways that are hard to model mathematically, but it will be advantageous for you if you can figure out how to exploit this behavior in some way. This game is certainly one where it makes sense to deviate from the Nash equilibrium.

This game is a great example of "adverse selection," which happens anytime a "buyer" and "seller" have asymmetric information. You likely don't know anyone else's MIT ID, so you're only competing with people when they *want* to compete against you—selecting adversely to your chances of winning the game. There's a game called the "market for lemons"[13]: suppose everyone is trying to sell a car in a lot, where the cars vary wildly in quality. The prices range from $1000 to $10,000, and you're trying to determine the fair price for a car on this market. If the fair price is $5,500, all cars worth above this value will leave the market as their sellers don't want to lose so much money. Now, the average value of cars on this market will drop again, and as a result the fair price will drop as well. This process continues until all cars left on the market

[12] The MIT course 6.853, "Topics in Algorithmic Game Theory," covers many ways to find and approximate Nash equilibria; we highly recommend it.

[13] The word "lemon" is slang for a car that is discovered to have problems only after it's bought. This refers to the fact that when you drive a new car off the lot it loses 20% of its value instantly. If you try to sell your new car the day after you purchased it, people will wonder why you're selling it since they have less information about it than you.

are the worst of them all (the "lemons"). This happens because there's so much adverse selection—every seller has private information about the true worth of their car.[14]

There are many sources of adverse selection in the game of poker: e.g. choosing to bet on the first action since your opponent is much more likely to call when they have good cards, multiple bets in a row, staying in the pot during later stages of the round, etc. You should always ask in poker that when you bet, am I happy with my bet conditioned on my opponent calling. Players can also exploit against a deterministic betting strategy such as going all in with a top X% hand by going all in with a top X/2% hand, another example of adverse selection.

1.7 Reference Bot

In the the prior lecture we were able to understand how strong our hands were using a Monte Carlo strategy. Today we will modify of this a little bit by leveraging some of the Poker Theory we covered by creating strong hands and the Monte Carlo Strategy.

We will try to update our bot from last lecture so that it not only runs Monte Carlo to determine the strength of our pre-flop hand, but also use the Community Cards in order to test our hand strength during the later stages in the game.

In order to achieve that, we added a parameter to our `calc_strength` method called community. Community is a list of the community cards we are looking at. First, we need to run `eval7` through our community cards, and remove them from our deck.

We also need to make a small change in the number of community cards we randomize to 5 - len(community). We then incorporate the community card into our hand using an if statement. At the end we compare our hand value with our opponent's and calculate hand strength the same way.

Comparing the results of this bot to `reference-lecture-2` yields a large chip lead over our opponent, demonstrating the power of game theory in improving your bot. As always the lecture bot is posted at: https://github.com/mitpokerbots/reference-lecture-3-2023

[14] A common economics joke is that if you find a $20 bill on the street, you should pick it up; if you find a $20 bill in the middle of Grand Central, you should leave it be since if all the rational people around it are walking past, there must be some adverse selection to picking it up.