Success of Libratus

Jiren Z

January 26, 2018

Introduction

In (fill in month) 2017, four world class poker players engaged in a 20(check duration)-day long battle of texas poker (put in the correct name). However, they were not competing against each other. Instead, they were fighting against a common foe, an AI for the game called "Libratus", developed by (name) and (name) from Carnegie Mellon University. As the game progresses, the computer held a solid advantage, beating the humans with (fill in margin). This is the first time that an computer agent beats professional players in heads-up no limit Texas Poker (check name). Different from other AI agents like Alpha Zero, Libratus did not use Reinforcement Learning or Deep Learning that were very popular these days. Instead, their method is rather classical and involves a lot of Game Theory. This article aims to provide necessary backgrounds for Game Theory so that one can understand what made Libratus so powerful and then overview the innovations of Libratus.

Background

Games and Game Theory Every one loves playing games. While one can play games for fun, the notion of a game can be applied to many important aspects of life. Whether it is you negotiating wages with an employer, companies setting prices or governments deploying armies, a game can be any scenario where two or more parties can take various actions to maximize their own outcomes. Game theory is a branch of Mathematics/Economics that formalizes and studies games. It is a rich subject that has influenced numerous fields.

Normal form game To understand what made Libratus surpass human performance. It is time that we go into a bit of formalisms of game theory. For our purpose we will limit our discussion to two player games only. A very simple yet very important type of game is called **Normal Form Games**. In these games, each player has a collection of available actions: A_1, A_2. They choose actions $a_1 \in A_1$, $a_2 \in A_2$ simultaneously and receives rewards $u_1(a_1, a_2)$, $u_2(a_1, a_2)$ as functions over actions of both players.

As an example, let's look at how Rock-Paper-Scissors would look like under this formulation. Let's use R for rock, P for paper and S for Scissor. We will also assume that for both players, winning a game would yield one dollar in return and losing will yield -1. A draw would mean 0 return for both players. The game would then look like Figure 1. If we want to see the outcome of a

	P	S	R
P	0/0	-1/1	1/-1
S	1/-1	0/0	-1/1
R	-1/1	1/-1	0/0

Figure 1: Normal form game formulation of the game Paper Scissor Stone

game, we would go to the row corresponding to action of player 1 and column corresponding to player 2. Say that player 1 plays paper and player 2 plays rock. We would find the $P-R$ cell and get $1/-1$. These numbers correspond to the rewards received by player 1 and player 2, respectively.

A strategy describes how the player chooses his actions. If a player chooses strategies deterministically, the strategy is called a pure strategy. A strategy that involves randomization is called a mixed strategy. We usually describe the strategy as a probability distribution over all actions, using $\sigma_1(a)$ to describe the probability that player 1 plays action a. Thus the expected payoff for this player is

$$u_1(\sigma_1, \sigma_2) := \sum_{a_1 \in A_1, a_2 \in A_2} u_1(a_1, a_2)\sigma_1(a_2)\sigma_2(a_2)$$

Nash and Nash equilibrium Since we are talking about game theory, we must introduce Nash, a genius Mathematician who laid the foundation of this thriving subject. Perhaps the most fundamental concept for Game Theory is that of a Nash Equilibrium, which Nash described. A **Nash Equilibrium** describes the scenario where none of the participants of a game can improve by changing his/her own strategy. Formally speaking, a Nash equilibrium consists of strategies σ_1 and σ_2 such that for player 1, if he chooses another strategy σ_1', we have

$$u_1(\sigma_1, \sigma_2) \geq u_1(\sigma_1', \sigma_2)$$

and vice versa for player 2. This captures the fact that a rational player would do whatever is in his/her control to maximize the outcome from a game.

It is not necessary that Nash equilibria always exist for a game when we only consider pure actions. For example, in the game of Rock-Paper-Scissors, if player 2 chooses Paper all the time. Player 1 can choose Scissors to win. But then player 2 should play Rock instead and the reasoning does not end. The only Nash equilibrium is one with mixed strategies where both player randomizes between all three actions with equal probability. Nash's important contribution is the "Nash Existence Theorem" where he showed that there always exists at least a mixed strategy Nash Equilibrium for games with finite actions. This makes Nash Equilibrium a general concept that applies to various different games. For our discussion we will always assume that we are always considering mixed strategies along with pure strategies.

Zero-sum games An shared feature among poker, rock-paper-scissors and many other competitive game is that whatever one player get is negative of what the other player gets. Such a game is called a **Zero-Sum Game**. Formally speaking, we have

$$u_1(a_1, a_2) + u_2(a_1, a_2) = 0$$

An important property of zero-sum games is that any Nash equilibrium can be computed efficiently. A **maxmin** value of a player 1 is

$$\max\min_1 = \max_{a_1}(\min_{a_2} u_1(a_1, a_2))$$

the maximum payoff that player 1 can guarantee if he chooses action before player 2. The corresponding **minmax** value of player 1 is

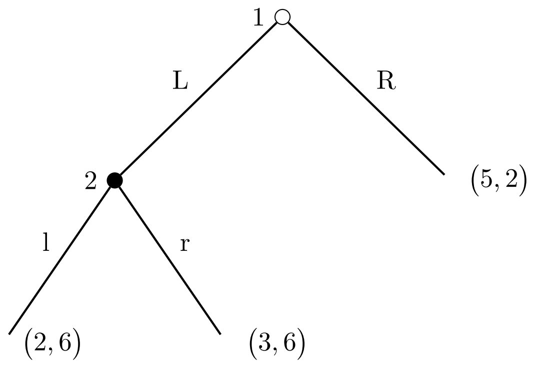
$$\min\max_1 = \min_{a_2}(\max_{a_1} u_1(a_1, a_2))$$

The minimum payoff that player 2 can cause player 1 to suffer if he actions before player 1.

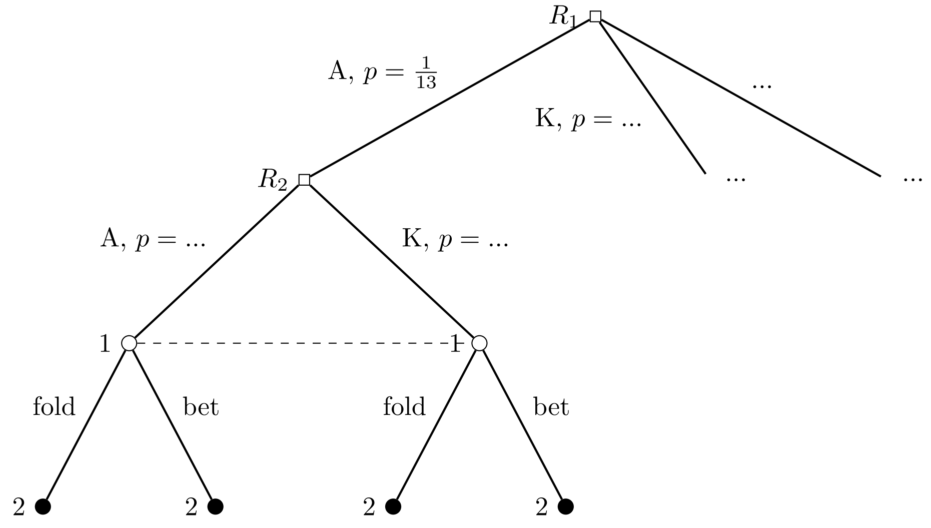
The **"Minmax Theorem"** states that minmax and maxmin are equal for a zero-sum game (with mixed strategies) and that Nash equilibria consists of both players playing maxmin strategies. An important corollary is that zero-sum games can be computed **efficiently** since the maxmin strategy can be solved by a linear program maximizing v over $v \in \mathbb{R}$, σ_1 a probability distribution over A_1, satisfying

$$\sum_{a_1} \sigma_1(a_1)u_1(a_1, a_2) \geq v \forall a_2 \in A_2$$

Extensive form game and imperfect information While normal form game can describe a variety of games. A more general formalism is necessary for studying poker. Games like poker are usually studied as an extensive form game where multiple actions may take place. See Figure for an example where player 1 first takes action between L and R. R ends the game with payoff $(5, 2)$. L continues the game, offering player 2 choice between l, r and corresponding rewards. All the possible games states are specified by the game tree and it is easy to read off what is going on.



We take a simplified game tree of a Poker game for an example. See Figure . In a game of poker, cards are dealt randomly. Randomness is depicted by the chance node (R) where a dice is rolled and possible outcomes happen with some probability. First a card is dealt to player 1 with some probability specified at chance node R_1, next a card is dealt to player 2 at chance node R_2. Then it is player 1's turn to bet. Note that in a poker game player 1 does have perfect information. He does not know what card is dealt to player 2. In the game tree, this is denoted by the **information set**. An information set is a collection of game states that a player cannot distinguish when making decisions. In terms of strategies, it is required that each player only have one strategy in each information set. When player 1 finds himself in the information set, the true state of the world is a distribution over all possibilities specified by the information set. So he needs to evaluate over all possible underlying states, what is the payoff of a potential action.



Poker and Libratus

Now that we have the necessary game theoretic terminology, let's take a look at Poker and why Poker is such a challenge. In Poker, each players are dealt cards that are secret to the other player, so the game is an imperfect information game. The variant of Poker that Libratus solved is called heads up no-limit Texas Hold'em. Heads up means that there are only two players playing against each other, making the game a two-player zero sum game.

As described in [cite short report], Libratus has three main modules, game abstraction, subgame solving and self improvement. We shall discuss them in the next three sections.

Game abstraction

No-limit means that there is no fixed bet sizes and no limit on the number of raises. In contrast, in a limited version of the game, bet sizes are fixed and number of raises are fixed. This reduces the number of possibilities of the game state, which makes the game easier to solve. This variant is solved in [put year here], see article [cite].

As a comparison, the limit version has 1.38×10^{13} information sets. According to the report, to solve the limited version, an algorithm, named "CFR+", "was executed on a cluster of 200 computation nodes each with 24 2.1-GHz

AMD cores, 32 GB of RAM, and a 1-TB local disk. The algorithm takes 61 min on average to complete one iteration. The computation was then run for 1579 iterations, taking 68.5 days, and using a total of 900 coreyears of computation (43) and 10.9 TB of disk space, ...". This is already a lot of computation.

In the no limit version, a bet size can be $501 or $502 and they are different game states. This results in a whopping 10^161 different decision points from a naive view of the game. Nevertheless, it is not necessary to construct a new betting strategy for a single dollar difference in the bet. One of the first task that Libratus must accomplish is to abstract the game to bring it down to a more realistic abstraction.

As described in [cite short report], the Libratus abstracts the game by grouping similar actions and similar cards, bring the game size down to 10^{12} [refer to the short report]. This results in a manageable game that can thus be solved by a variation of the CFR method. The abstraction is referred to as the **blueprint**.

Solving the blueprint: Counterfactual Regret Minimization Recall that earlier we mentioned that Nash equilibriums in normal form two player zero sum games can be solved by linear program. The ideal can be directly applied to an extensive form game where a possible strategy of a player describes what he does in all information sets. This approach, however, does not apply to any game of reasonable size as such possible strategies grow exponentially with respect to the number of information sets. Later methods exploits the tree-like structure of extensive form games and represent strategies as the actions at each node.

Counterfactual Regret Minimization is an iterative algorithm that solves for Nash equilibriums in extensive form games which has linear run time with respect to the number of information sets. Libratus uses a Monte Carlo (sampling) based variant where additional pruning is applied so that the algorithm does not sample really bad actions. For more information, see [Add citations]

Nested safe subgame solving While many Poker bots use an abstraction of the game to reduce the size of the game, they typically stick to the abstraction. Suppose that the blueprint evaluates the scenarios where the opponent bets $100 and $200 beforehand. If the opponent bets $134 in the actual game, the bot would treat it as if the opponent bet $100 through rounding.

Libratus takes an alternative approach. It expands the game tree at the off-blueprint action in real time and solves for that subgame exactly. [Possibly add image here]. This method, "subgame solving", was proposed earlier but were not shown to improve the performance of an agent. The difficulty originates from the fact that different subtrees in the game state are not independent in an inperfect information game, and you cannot look at a subgame in isolation. In a perfect information normal form game, if you have reached a subgame, your best strategy in this subgame is independent of what you do in other subgames.

When you are holding a pair of K and your opponent calls your blind, there are multiple possible situations, maybe he has 2 and 7, or he has a pair of A. All of his possible actions form a propability distribution conditioned on this information set being reached. But your opponent's action depends on how this subgame compares to other subgames. And if the opponent changes his strategy according to your new strategy in the subgame, the probability distribution of his possible hands at this information set would be different. This effectively invalidates the previously solved best strategy.

"Unsafe" subgame solving refers to the naive approach where one assumes that the opponent's strategy is fixed and computes a best strategy for the subgame. A "safe" subgame solving method augments the subgame by augmenting the subgame by giving player 1 some alternatives where he can choose not to enter the subgame being solved and receive expected payoff under the blueprint. This ensures that for any possible situation, player 1 is no better-off reaching the subgame after the new strategy is computed. This requirement turns out to be too conservative. Libratus uses "reach" subgame solving which only requires that player 1 becomes no better off for those cases where he is likely to reach this subgame.

Self improvement Finally, during the day when Libratus is playing against human players, it looks for most frequent off-blueprint actions. At night, Libratus refines its blueprint to solve for equilibrium of these actions. Thus Libratus continuously narrows the gap between actions taken by the human players and its blueprint abstraction of the game. This way, as the game goes on, it is harder and harder to exploit the fact that Libratus is only solving an approximate version of the game.

Final words

Work in progress. I will base it on reader editor backs about contents above.