Artificial Intelligence: Through the Lens of Poker

Max Chiswick

2024-09-11

Table of contents

1 AI Poker Tutorial

New AIPT by Poker Camp coming soon.

Part I

INTRO

Hello

2 Why Poker?

Hello

3 Poker Camp

Hello

4 Problem Solving

Hello

5 Games

Hello

6 Basic Strategy

Hello

7 Ethical Considerations

Hello

Part II

CODING FOUNDATIONS

Hello

8 Python

Hello

9 AI Agents

Hello

10 Building AI Agents

Hello

Part III

MATH FOUNDATIONS

Hello

11 Algebra

Hello

12 Probability

Hello

13 Combinatorics

Hello

14 Expected Value

Hello

15 Bayes' Rule

Hello

16 Statistics

Hello

17 Monte Carlo Methods

Hello

Part IV

KNOWLEDGE

Hello

18 Logic

Hello

19 Knowledge Representation

Hello

20 Rationality

Hello

21 Psychology and Mindset

Hello

Part V

DECISION MAKING UNDER UNCERTAINTY

Hello

22 Probabilistic Thinking

Hello

23 Decision Theory

Hello

24 Risk

Hello

25 Regret

Hello

26 Multi-armed Bandits

Hello

Part VI

GAME THEORY

Hello

A Nash equilibrium is a set of strategies, one for each player in the game. If the game is two-player and zero-sum, and if the players alternate positions to even out the advantage of playing in each position (as in poker games), then a Nash equilibrium has a useful theoretical property: it can do no worse than tie, on expectation, against any other opponent strategy. In a game such as poker, that "on expectation" is important: due to the luck in the game from the cards being randomly dealt, there is no guarantee that a Nash equilibrium (or any strategy!) will win every single hand. However, if you average over a large set of hands, or compute the expectation exactly, then it cannot to any worse than tie against anyone.

If the opponent also plays a Nash equilibrium strategy then they will tie; if the opponent carefully considers the Nash equilibrium strategy and computes a perfect counter-strategy then they will also tie. If the opponent makes mistakes, however, then they can lose value, allowing the Nash equilibrium strategy to win. In other words, a Nash equilibrium just plays perfect defence: it doesn't try to learn about or exploit the opponent's flaws, and instead just wins when the opponent makes mistakes. This is on purpose, since attempting to find and exploit an opponent's mistakes usually makes it possible for an even smarter opponent to exploit your new strategy. There's a tradeoff between playing defence and offence.

Since a Nash equilibrium is an unbeatable strategy for this type of game, it is considered to be an optimal strategy, and "solving" a game is equivalent to computing a Nash equilibrium. In this sense, "solve" is a technical term, meant in exactly the same sense that one might "solve for X" in a mathematical equation.

27 Nash Equilibrium

Hello

28 Game Theory Optimal (GTO)

Hello

29 Mixed Strategies

Hello

Part VII

GAME TREES

Hello

30 Perfect Information

Hello

31 Minimax

Hello

32 Imperfect Information Games

Hello

Part VIII

SOLVING TOY GAMES

Hello

33 Analytical Solutions

Hello

34 Normal Form

Hello

35 Optimization

Hello

36 Sequence Form

Hello

Part IX

COUNTERFACTUAL REGRET MINIMIZATION (CFR)

The most popular method for iteratively solving poker games is the Counterfactual Regret Minimization (CFR) algorithm. CFR was developed in 2007 at the University of Alberta. It converges to Nash equilibrium in two player zero-sum games.

Counterfactual

Counterfactual means “relating to or expressing what has not happened or is not the case”. For example, if in reality I didn’t bring an umbrella and got wet in the rain, I could say counterfactually, “If I had brought an umbrella, I wouldn’t have gotten wet.”

Actual event: I didn’t bring an umbrella, and I got wet in the rain

Counterfactual event: If I had brought an umbrella, I wouldn’t have gotten wet

Regret

Regret we previously touched on in Section 25.

In brief, it’s a way to assign a value to the difference between a made decision and an optimal decision. For example, if you choose to play a slot machine that returns a value of 5 rather than the best machine that returns a value of 10, then your regret would be $10 - 5 = 5$.

Minimization

Minimization refers to minimizing the difference between the made decision and the optimal decision. Playing the optimal slot machine, i.e. the one that returns a value of 10, would minimize the regret to 0.

What is CFR?

In brief, CFR is a self-play algorithm that learns by playing against itself repeatedly. It starts play with a uniform random strategy (each action at each decision point is equally likely) and iterates on these strategies each round to nudge closer to the game theory optimal Nash equilibrium strategy.

Each action at each information set in the game has an associated regret value that is calculated based on how that action performs compared to how the overall strategy at that infoset performs. Positive regret means that the action did better than the overall strategy and negative means worse.

The regrets get updated each iteration and upon returning to that node for the next iteration, the strategy is calculated according to the proportion of the positive regrets, meaning that actions with higher regret that performed well will get played more often.

At the end, the average of all strategies played converges to the equilibrium strategy.

The strategy is computed offline and then can be used in play – it's a fixed, opponent-agnostic strategy that can't be beaten in the long-run, but also doesn't take advantage of opponent weaknesses.

See Mike Johanson's intuitive explanation of CFR for more details.

37 CFR Algorithm

37.1 Algorithm Overview

Each information set maintains a `strategy` and `regret` tabular counter for each action. These accumulate the sum of all strategies and the sum of all regrets.

In a game like Rock Paper Scissors, there is effectively only one infoset, so only one table for `strategy` over each action (Rock, Paper, Scissors) and one table for `regret` over each action (Rock, Paper, Scissors).

Regrets are linked to strategies through a policy called *regret matching*, which selects strategies proportionally to their positive regrets.

After using regret matching and after many iterations, we can minimize expected regret by using the *average strategy* at the end, which is the strategy that converges to equilibrium.

If two players were training against each other using regret matching, they would converge to the Nash Equilibrium of 1/3 for each action using the average strategy in Rock Paper Scissors.

37.2 Simplified CFR Example: Regret Updates in RPS

Let's see how that works in Rock Paper Scissors.

In general, we define regret as:

Regret = u(Alternative Strategy) $-$ u(Current Strategy)

We prefer alternative actions with high regret and wish to minimize our overall regret. That second part might sound counterintuitive, but as we play more of the better actions, we will consequently minimize overall regret.

We play Rock and opponent plays Paper $\implies$ u(rock,paper) $= -1$

Regret(paper) = u(paper,paper) $-$ u(rock,paper) $= 0 - (-1) = 1$

Regret(rock) = u(rock,paper) $-$ u(rock,paper) $= -1 - (-1) = 0$

Regret(scissors) = u(scissors,paper) $-$ u(rock,paper) $= 1 - (-1) = 2$

We play Scissors and opponent plays Paper $\implies$ u(scissors,paper) = 1

Regret(paper) = u(paper,paper) − u(scissors,paper) = 0 − 1 = −1

Regret(rock) = u(rock,paper) − u(scissors,paper) = −1 − 1 = −2

Regret(scissors) = u(scissors,paper) − u(scissors,paper) = 1 − 1 = 0

We play Paper and opponent plays Paper $\implies$ u(paper,paper) = 0

Regret(paper) = u(paper,paper) − u(paper,paper) = 0 − 0 = 0

Regret(rock) = u(rock,paper) − u(paper,paper) = −1 − 0 = −1

Regret(scissors) = u(scissors,paper) − u(paper,paper) = 1 − 0 = 1

To generalize:

- The action played always gets a regret of 0 since the "alternative" is really just that same action
- When we play a tying action, the alternative losing action gets a regret of −1 and the alternative winning action gets a regret of +1
- When we play a winning action, the alternative tying action gets a regret of −1 and the alternative losing action gets a regret of −2
- When we play a losing action, the alternative winning action gets a regret of +2 and the alternative tying action gets a regret of +1

After each play, we accumulate regrets for each of the 3 actions.

We decide our strategy probability distribution using regret matching, which means playing a strategy that normalizes over the *positive* accumulated regrets, i.e. playing in proportion to the positive regrets.

Here's an example from Marc Lanctot's CFR Tutorial. Assume that the regret counters start at 0 for each action.

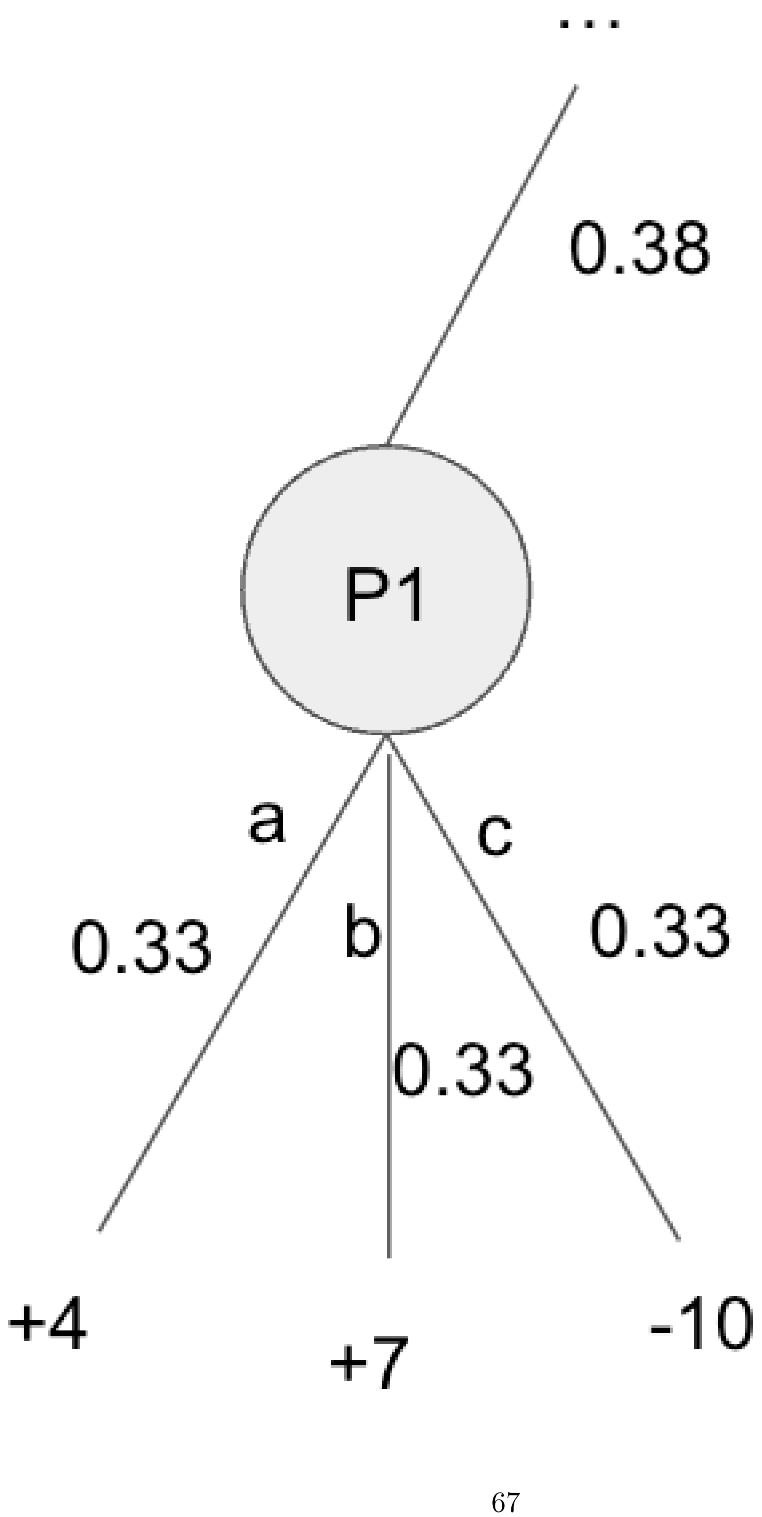
- Game 1: Choose Rock and opponent chooses Paper
  - Lose 1
  - Rock: Regret 0
  - Paper: Regret 1
  - Scissors: Regret 2
- Next Action: Proportional

$$\begin{pmatrix} \text{Rock} & 0/3 = 0 \\ \text{Paper} & 1/3 = 0.333 \\ \text{Scissors} & 2/3 = 0.667 \end{pmatrix}$$

- Game 2: Choose Scissors (probability 2/3) and opponent chooses Rock
  - Lose 1
  - Rock: Regret 1
  - Paper: Regret 2
  - Scissors: Regret 0
- Cumulative regrets:
  - Rock: 1
  - Paper: 3
  - Scissors: 2
- Next Action: Proportional

$$\begin{pmatrix} \text{Rock} & 1/6 = 0.167 \\ \text{Paper} & 3/6 = 0.500 \\ \text{Scissors} & 2/6 = 0.333 \end{pmatrix}$$

37.3 Simplified CFR Example: Counterfactual Values



Here we have a very simple game tree showing a Player 1 node at the end of the game. For simplicity, we assume that this is the first iteration of the algorithm so each action is assigned a uniform 1/3 probability.

The counterfactual reach probability of information state I, −i (I), is the probability of reaching I with strategy profile except that, we treat current player i actions to reach the state as having probability 1. In all situations we refer to as “counterfactual”, one treats the computation as if player i’s strategy was modified to have intentionally played to information set Ii . Put another way, we exclude the probabilities that factually came into player i’s play from the computation.

Regular expected value: We can compute the regular expected value for each action of P1:

$\mathbb{E}(\mathrm{a}) = 0.33 * (4) = 1.33$

$\mathbb{E}(\mathrm{b}) = 0.33 * (7) = 2.33$

$\mathbb{E}(\mathrm{c}) = 0.33 * (-10) = -3.33$

Counterfactual value: The counterfactual value includes the reach probability $\sigma_(-i) = 0.38$ of the opponent and chance playing to this node.

$$CV()v_i(I,a) = \pi^{\sigma}_{-i}(h)\pi^{\sigma:I\to a}(h,z)u_i(z)v$$

37.4 Core CFR Algorithm

High quantity of definitions (conventions primarily sourced from 2015 paper *Solving Heads-up Limit Texas Hold’em*):

P: Set of players

H: Game state, represented as history of actions from start of game

Tabular storing strategies and regrets at each infoset

Regrets based on action values compared to node EV, which is based on counterfactual values

Regret minimization, usually regret matching, to get new strategies

Average strategy converges to Nash equilibrium

As we think about solving larger games, we start to look at iterative algorithms.

37.4.1 Regret and Strategies

A strategy at an infoset is a probability distribution over each possible action.

Regret is a measure of how much each strategy at an infoset is preferred and is used as a way to update strategies.

For a given P1 strategy and P2 strategy, a player has regret when they take an action at an infoset that was not the highest-EV action at that infoset.

Regret Exercise

What is the regret for each action?

Action	Regret
A	
B	
C	

Solution

Action	Regret
A	4
B	2
C	0

Expected Value Exercise

If taking a uniform strategy at this node (i.e. $\frac{1}{3}$ for each action), then what is the expected value of the node?

Solution

$\mathbb{E} = \frac{1}{3} * 1 + \frac{1}{3} * 3 + \frac{1}{3} * 5 = 0.33 + 1 + 1.67 = 3$

Poker Regret Exercise

In poker games, the regret for each action is defined as the value for that action minus the expected value of the node. Give the regret values for each action under this definition.

Solution

Action	Value	Poker Regret
A	1	-2
B	3	0
C	5	2

At a node in a poker game, the player prefers actions with higher regrets by this definition.

Regret matching definitions:

- a is actions
- σ is strategy
- t is time
- i is player
- R is cumulative regret

$$\sigma_i^t(a) = \begin{cases} \frac{\max(R_i^t(a),0)}{\sum_{a' \in A} \max(R_i^t(a'),0)} & \text{if } \sum_{a' \in A} \max(R_i^t(a'), 0) > 0 \\ \frac{1}{|A|} & \text{otherwise} \end{cases}$$

This is showing that we take the cumulative regret for an action divided by the cumulative regrets for all actions (normalizing) and then play that strategy for this action on the next iteration.

If all cumulative regrets are ≤ 0 then we use the uniform distribution.

If cumulative regrets are positive, but are are < 0 for a specific action, then we use 0 for that action.

In code:

```
    def get_strategy(self):
  #First find the normalizing sum
        normalizing_sum = 0
        for a in range(NUM_ACTIONS):
            if self.regret_sum[a] > 0:
                self.strategy[a] = self.regret_sum[a]
            else:
                self.strategy[a] = 0
            normalizing_sum += self.strategy[a]

    #Then normalize each action
        for a in range(NUM_ACTIONS):
            if normalizing_sum > 0:
                self.strategy[a] /= normalizing_sum
            else:
                self.strategy[a] = 1.0/NUM_ACTIONS
            self.strategy_sum[a] += self.strategy[a]

        return self.strategy
```

37.4.2 Iterating through the Tree

The core feature of the iterative algorithms is self-play by traversing the game tree over all **infosets** and tracking the strategies and regrets at each.

From above, we know how to find the strategy and regret in the simple Rock Paper Scissors environment.

In poker:

- Strategies are determined the same as above, through regret matching from the previous `regret` values at the specific information set for each action
- CFR definitions:
  - a is actions
  - I is infoset
  - σ is strategy
  - t is time
  - i is player
  - R is cumulative regret
  - z is a terminal node
  - u is utility (payoffs)
  - p is the current player who plays at this node
  - $-p$ is the the opponent player and chance
  - v is counterfactual value
- Counterfactual values are effectively the value of an information set. They are weighted by the probability of opponent and chance playing to this node (in other words, the probability of playing to this node if this player tried to do so).
  - Counterfactual value: $v^\sigma(I) = \sum_{z \in Z_I} \pi^\sigma_{-p}(z[I]) \pi^\sigma(z[I] \to z) u_p(z)$
  - $\sum_{z \in Z_I}$ is summing over all terminal histories reachable from this node
  - $\pi^\sigma_{-p}(z[I])$ is the probability of opponents and chance reaching this node

  - $\pi^\sigma(z[I] \to z)$ is the probability of playing from this node to terminal history z, i.e. the weight component of the expected value
  - $u_p(z)$ is the utility at terminal history z, i.e. the value component of the expected value

- Instantaneous regrets are based on action values compared to infoset EV. Each action EV then adds to its `regret` counter:
  - $r^t(I,a) = v^{\sigma^t}(I,a) - v^{\sigma^t}(I)$

- Cumulative (counterfactual) regrets are the sum of the individual regrets:
  - $R^T(I,a) = \sum_{t=1}^{T} r^t(I,a)$

37.5 More Exercises

Maximize Against non-Nash Fixed Opponent

How would you maximize in RPS knowing the opponent plays a fixed non-Nash strategy that you don't know?

Solution

One option is to play the equilibrium strategy until you get a significant sample on your opponent and then to exploit their strategy going forward.

Strategy Against No-Rock Opponent

What is the optimal play if your opponent can't play Rock?

Solution

Player 1/2	Paper	Scissors
Rock	(-1, 1)	(1, -1)
Paper	(0, 0)	(-1, 1)
Scissors	(1, -1)	(0, 0)

We can see that Player 1 playing Paper is dominated by Scissors, so Player 1 should never play Paper.

Player 1/2	Paper	Scissors
Rock	(-1, 1)	(1, -1)
Scissors	(1, -1)	(0, 0)

In the reduced game, we see that if Player 2 plays Paper with probability p and Scissors with probability s, then:
$\mathbb{E}(\text{P1 R}) = -1 * p + 1 * s = -p + s$ $\mathbb{E}(\text{P1 S}) = 1 * p + 0 * s = p$
Setting these equal, $-p + s = p \Rightarrow s = 2p$.
We also know that $s + p = 1$.
Therefore $s = 1 - p$ and $1 - p = 2p \Rightarrow 1 = 3p \Rightarrow p = 1/3$.
Therefore, $s = 1 - 1/3 = 2/3$.
For Player 2, we have $s = 2/3$ and $p = 1/3$.
For Player 1, we can solve similarly:
$\mathbb{E}(\text{P2 P}) = 1 * r - 1 * s = r - s$ $\mathbb{E}(\text{P2 S}) = -1 * r + 0 * s = -r$
$r - s = -r \Rightarrow 2r = s$
We also know that $r + s = 1$.
Therefore $s = 1 - r$ and $1 - r = 2r \Rightarrow 1 = 3r \Rightarrow r = 1/3$.
Therefore, $s = 1 - 1/3 = 2/3$.
For Player 2, we have $s = 2/3$ and $p = 1/3$.
Inserting these probabilities, we have:

Player 1/2	Paper (1/3)	Scissors (2/3)
Rock (1/3)	(-1, 1) (1/9)	(1, -1) (2/9)
Scissors (2/3)	(1, -1) (2/9)	(0, 0) (4/9)

Therefore Player 1 has payoffs of: $1/9 * -1 + 2/9 * 1 + 2/9 * 1 + 4/9 * 0 = 3/9 = 1/3$. Therefore the player that can still play Rock has an advantage of 1/3 at equilibrium.

Maximize Against Adapting Rock Opponent

1. Suppose that your opponent is forced to play Rock exactly

Suppose that your opponent gets a card with probability X such that $X\%$ of the time they are forced to play Rock What if the opponent is adapting to you, but 10% of the time they are forced to play Rock?

Solution

Soon

Skewed Rock Payoff

What is the equilibrium strategy if the payoff for Rock over Scissors is 2 (others stay the same)?

Solution

Soon

38 CFR Interactive

Hello

39 Solving RPS and Poker with CFR

Bet capping: Even when holding the strongest hand — a pair of aces — the strategy caps the betting less than 0.01% of the time

39.0.1 RPS Regret Matching Experiment

Here we show that regret matching converges only using the average strategy over 10,000 iterations:

The bottom shows both players converging to 1/3, while the top shows Player 1's volatile current strategies that are cycling around.

Suppose that your opponent Player 2 is playing 40% Rock, 30% Paper, and 30% Scissors. Here is a regret matching 10,000 game experiment. It shows that it takes around 1,600 games before Player 1 plays only Paper (this will vary).

We see that if there is a fixed player, regret matching converges to the best strategy.

But what if your opponent is not using a fixed strategy? We'll talk about that soon.

39.1 Data: Talk Paper Scissors

eieio games made a Rock Paper Scissors over voice game in which players call a phone number and get matched up with another player for a 3 game RPS match.

They published their 40,000 round data on X:

Overall: R 37.2%, P 35.4%, S 27.4%

Round 1: R 39.7%, P 37.6%, S 22.7%

Round 2: R 34.0%, 33.4%, 32.6%

Round 3: R 37.2%, 34.7%, 28.1%

Expected Value Against TPS Player

What is the best strategy per round against the average TPS player? What is your expected value per round and overall?

Solution

The best strategy is to always play Paper.
$\mathbb{E}(\text{Round 1}) = 0.397 * 1 + 0.376 * 0 + 0.227 * -1 = 0.17$
$\mathbb{E}(\text{Round 2}) = 0.34 * 1 + 0.334 * 0 + 0.326 * -1 = 0.014$
$\mathbb{E}(\text{Round 3}) = 0.372 * 1 + 0.347 * 0 + 0.281 * -1 = 0.091$
$\mathbb{E}(\text{Round 4}) = 0.17 + 0.014 + 0.091 = 0.275$

40 CFR Proof

Hello

Intuitions

https://www.reddit.com/r/GAMETHEORY/comments/nq7gly/doubts_about_counterfactual_regret_minimi
https://www.quora.com/

Epsilon-Nash Equilibrium HULHE is solved

A game is said to be ultraweakly solved if, for the initial position(s), the game-theoretic value has been determined; weakly solved if, for the initial position(s), a strategy has been determined to obtain at least the game-theoretic value, for both players, under reasonable resources; and strongly solved if, for all legal positions, a strategy has been determined to obtain the game-theoretic value of the position, for both players, under reasonable resources. In an imperfect-information game, where the game-theoretic value of a position beyond the initial position is not unique, Allis's notion of "strongly solved" is not well defined. Overall, we require less than 11 TB of storage to store the regrets and 6 TB to store the average strategy during the computation, which is distributed across a cluster of computation nodes. This amount is infeasible to store in main memory, and so we store the values on each node's local disk. But: Finally, unlike with CFR, we have empirically observed that the exploitability of the players' current strategies during the computation regularly approaches zero. Therefore, we can skip the step of computing and storing the average strategy, instead using the players' current strategies as the CFR+ solution.

A strategy's "exploitability" is the maximum amount that a perfect counter-strategy could win on expectation against a strategy. A Nash equilibrium has an exploitability of zero, since it cannot be beaten by anyone on expectation, and having a lower exploitability is good. When you run CFR, the average strategy's exploitability converges towards zero, driving its worst-case loss lower and lower. Note that this is a pessimistic way to measure how good your strategy is: our best poker programs started beating the world's best human players in heads-up limit hold'em in 2008, even though our programs at that time were still massively exploitable by this worst-case measure.

In our January 2015 Science paper, we've announced that we've produced a strategy that has essentially weakly solved the game. That means that we have computed a strategy with such a low exploitability (0.000986 big blinds per game) that it would take more than a human lifetime of play, using the perfect counter-strategy, for anyone to have 95% statistical confidence that they were actually winning against it. So it's not an exactly perfect strategy, but it is so close to perfect that the game is essentially solved, as it's now outside of any human's ability to beat it for a statistically meaningful amount by playing games against it.

41 CFR Algorithm Improvements

Hello

CFR+ variation such that regrets can't be <0

Linear CFR such that regrets are weighted by their recency

41.1 CFR+

41.2 Discounted CFR

https://www.cs.cmu.edu/~sandholm/cs15-888F23/ lecture 6 When is CFR+ worse? High variance play keeps getting very positive

CFR+ does exhaustive iterations over the entire game tree and uses a variant of regret matching (regret matching+) where regrets are constrained to be non-negative. Actions that have appeared poor (with less than zero regret for not having been played) will be chosen again immediately after proving useful (rather than waiting many iterations for the regret to become positive). Finally, unlike with CFR, we have empirically observed that the exploitability of the players' current strategies during the computation regularly approaches zero. Therefore, we can skip the step of computing and storing the average strategy, instead using the players' current strategies as the CFR+ solution.

What are other ways to improve CFR?

42 Monte Carlo CFR

Hello

https://www.cs.cmu.edu/~sandholm/cs15-888F21/ lecture 10 Improve upon CFR with Monte Carlo sampling Monte Carlo CFR versions (focus on chance sampling and external sampling) MCCFR Mini-Project: MCCFR implemented https://www.cs.cmu.edu/~sandholm/cs15-888F23/ lecture 14 https://ai.stackexchange.com/questions/26345/how-exactly-is-monte-carlo-counterfactual-regret-minimization-with-external-samp

Sampling methods

External: Sample chance and opponent nodes

Chance: Sample chance only

Outcome: Sample outcomes

43 Vector CFR

Hello

MCCFR worse for headsup postflop, slows down at lower exploitability Vector alg efficient at headsup postflop (pass down ranges, update regrets for entire range at once), evaluate showdowns O(n) instead of O(n^2)

Part X

ABSTRACTING LARGE GAMES

Hello

44 Game Size

Hello

45 Card Abstraction

Hello

46 Bet Abstraction

Hello

47 Agent Evaluation

Hello

Part XI

POKER SOLVERS

Hello

48 How Solvers Work

Hello

49 Using Solvers

Hello

50 Generalizing Solver Outputs

Hello

51 Studying Populations

Hello

52 Solver Limitations

Hello

53 Advanced Strategy

Hello

54 Tournaments

Hello

Part XII

AI MATH

Hello

55 Calculus

Hello

56 Linear Algebra

Hello

57 Information Theory

Hello

Part XIII

AI FOUNDATIONS

Hello

58 Machine Learning

Hello

59 Deep Learning

Hello

60 Reinforcement Learning

Hello

Part XIV

recentgameaiadvances/recentgameaiadvances.

61 Types of Games

Hello

62 Multi-agent Systems

Hello

Part XV

STATE OF THE ART POKER AI

Hello

63 Deep CFR

Hello

64 Top Poker Agents

Hello

65 Variance Reduction

Hello

66 Human vs. AI

Hello

67 New Research

Hello

Part XVI

LLMS

Hello

68 Transformers

Hello

69 OthelloGPT

Hello

70 PokerGPT

Hello

71 Interpretability

Hello

Part XVII

OPPONENT MODELING

Hello

72 Best Response

Hello

73 Exploitative Strategies

Hello

Part XVIII

AI RISKS AND SAFETY

Hello

74 Ethics and Short-term Risks

Hello

75 Alignment and Long-term Risks

Hello

Part XIX

THE RIVER

Hello

76 Trading

Hello

77 Prediction Marketes

Hello

78 Other Betting

Hello

Part XX

PROJECT IDEAS

Hello

79 Projects

Hello