Xinyu Zhang Follow

Dec 26, 2020 · 12 min read · Listen

Learn AI Game Playing Algorithm Part III — Counterfactual Regret Minimization

In Post I and II, I have introduced some basics of AI game-playing algorithms, and Monte Carlo Tree Search (MCTS) [1], a popular algorithm suitable for solving perfect information games.

In **Part III**, which is this post, I will introduce Counterfactual Regret Minimization (CFR) [2], a popular algorithm for solving imperfect information games in current literature, and some of its variants [3]. Specifically, I will talk about

1. **Information Set**, the key definition that models partially observable game state
2. **Nash Equilibrium Approximation**, the framing idea for CFR, which we briefly introduced in Part I. Here we will revisit this topic and provide some extra details.
3. **Counterfactual Regret Bound**, an optimizable upper bound that measures the distance to Nash equilibrium
4. **CFR Procedure** with Pseudo-Code
5. **Monte Carlo CFR** [3], a widely-used variant of CFR, more efficient and suitable for larger games [5]

Because the CFR algorithm is not a widely known algorithm, we need to introduce some new definitions along the way, making this subject seem a little more complicated than previous posts. Nevertheless, I will try my best to write this post as straightforward and intuitive as possible but with enough depth.

I want to note again that the mathematical language is not rigorous enough in this series, and please excuse me for this! If any problems found, please point them out 😃

> References of many of the following points can be found at [7], which I find very well written and helpful for the current subject.

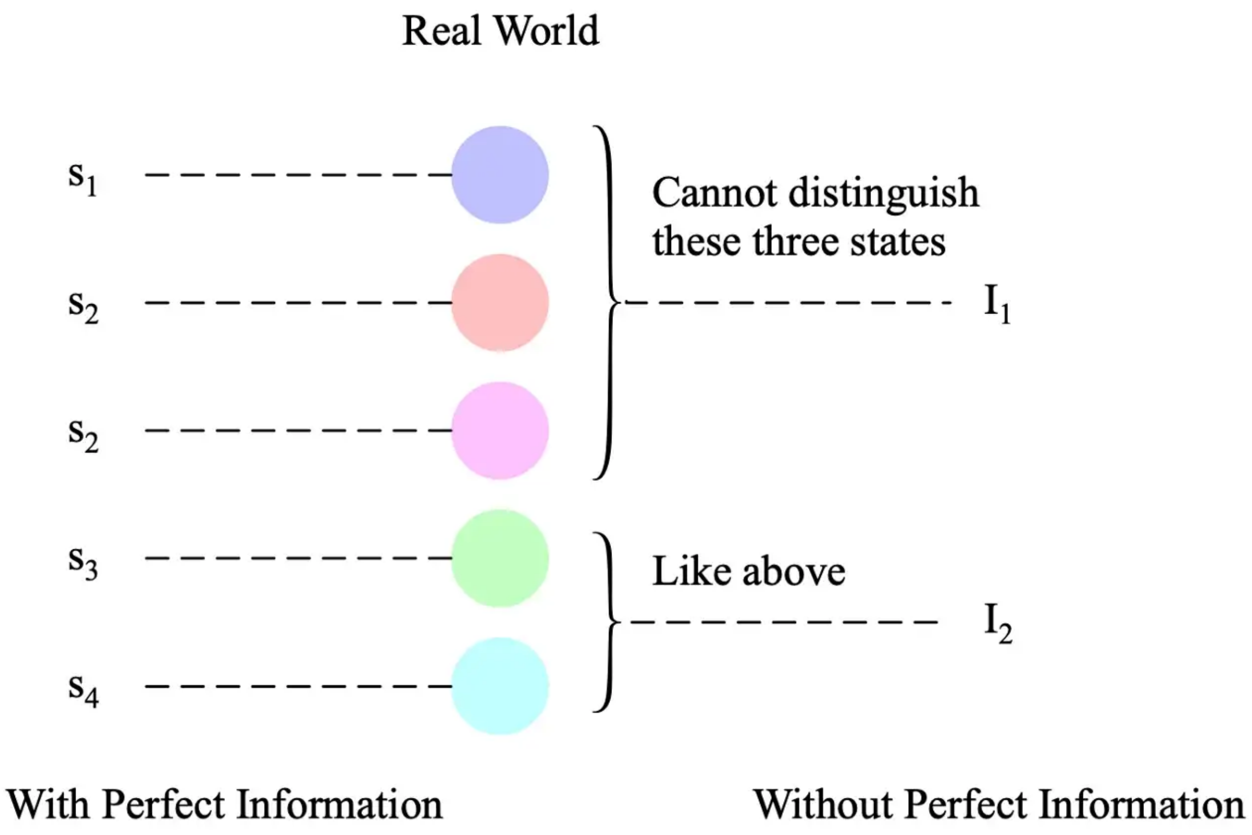
Table of Content

Information Set

Recall Part I, we talk about the critical difference between a perfect information game and an imperfect one:

> In a game with imperfect information, players cannot observe the actual game state s, which means we cannot use $\pi(a|s)$ to represent strategy.

The insight is that **enclosure of information makes players cannot distinguish certain individual states.**



The idea of Information Set (on the right)

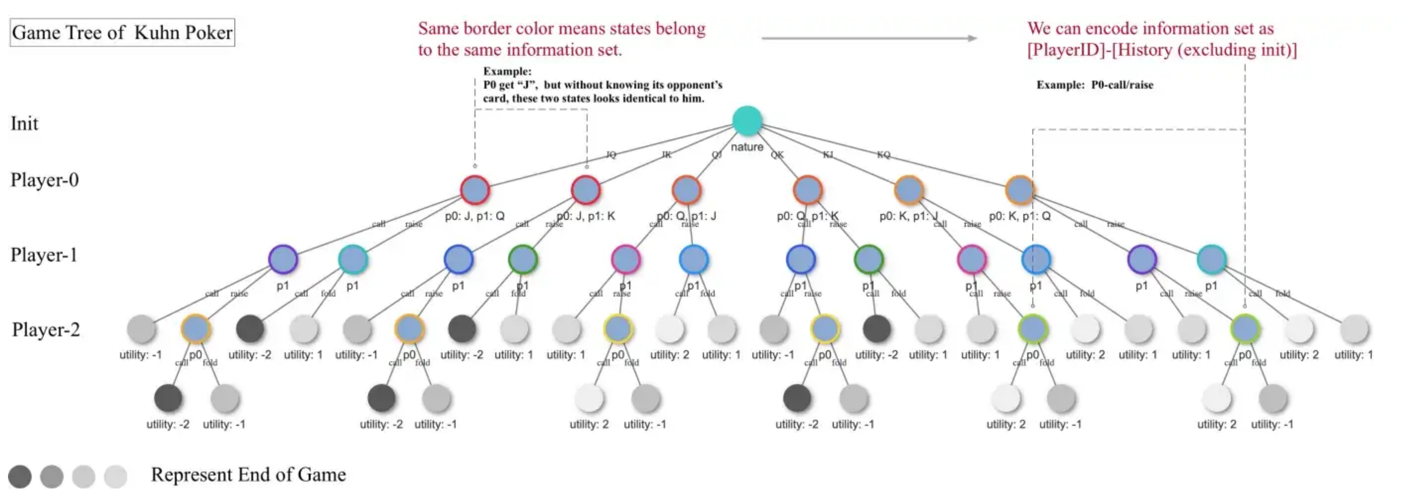
As illustrated from the above figure, we can define our strategy as $\pi(a|I)$ instead of $\pi(a|s)$, where I is the information set. According to [4], an information set is a set of decision nodes such that:

> When the game reaches the information set, the player with the move cannot differentiate between nodes within the information set. All nodes in an information set belong to one player.

To explain it more clearly, I draw a game tree of Kuhn Poker [6] below. Kuhn Poker is an extremely simplified form of poker.

> **How Kuhn Poker Play**

> Two players each receive a card but do not show it to the other. Each player may initiate a card match (`action=call, bet=1`), and the other player may either accept the match (`action=call`), concede (`fold`) or raise the bet (`action=raise, bet=2`).
>
> - If they both call, the larger one wins the bet. The bet can be either 2 or 1, depending on whether the raise happens.
> - If either one of two folds, another player wins 1.
> - The game is zero-sum.



Game Tree and Information Set Demonstration in Kuhn Poker

It should be very clear from the above figure that what an information set is. In the following, we will define **some symbols that used throughout this article**:

- *I* denotes an information set
- *h* denotes an action sequence (or game history), $h \in I$
- *A(h), A(I)* denote the set of available actions at *h* or *I*
- P*(h)* and P*(I)* denote functions that return the player with the move, given action sequence or information set.
- *Z* denotes the set of terminal histories (like the gray nodes in above figure). We write $h \in Z$ to denote the game history that ends in Z.
- We will use σ to denote strategy, and σ_i denotes the strategy for player *i*, $\sigma(a|I)$ is the the probability of choose action *a* at *I*.

- We will use π to denote probability, $\pi(h|\sigma)$ is the probability of reaching h with strategy σ.

$$\text{we will also use } \pi^{\sigma}(h) \text{ to denote } \pi(h|\sigma) \text{ for simplicity}$$

- We use $-i$ to denote players other than i, thereby define π_i and π_{-i} as contributions to probability π by player i and other players, i.e. $\pi(h|\sigma) = \pi_i(h|\sigma_i)\ \pi_{-i}(h|\sigma_{-i})$; σ_i and σ_{-i} as strategies of player i and other players, u_i and u_{-i} as utilities of player i and other players.
- $u(z)$, $z \in Z$ denotes the utility of terminal history
- $u(h|\sigma)$ denotes the utility of a game history, it can be recursively defined as below, and $u(root|\sigma)$ can be written as $u(\sigma)$.

$$u^{\sigma}(h) = \sum_{h' \in \text{children}(h), (a,h) \to h'} u^{\sigma}(h')\sigma(a|h)$$

Nash Equilibrium Approximation

Recall Part I, we talk about the concept of Nash equilibrium and its corresponding strategy

> Nash equilibrium strategy is a conservative but good strategy, though not optimal, like in prisoner dilemma, it works well in practice. Many successful works on solving imperfect information games are based on the idea of optimizing strategy to approximate Nash equilibrium state.

An approximation of a Nash equilibrium is called ϵ-Nash equilibrium, which is formalized in an example of a two-player game below:

$$u_1(\sigma) + \epsilon \geq \max_{\sigma'_1 \in \Sigma_1} u_1(\sigma'_1, \sigma_2) \quad u_2(\sigma) + \epsilon \geq \max_{\sigma'_2 \in \Sigma_2} u_2(\sigma_1, \sigma'_2)$$

However, **how do we find a strategy σ like the above**? The entry point lies in the concepts of average strategy and average strategy regret.

Average Strategy and Regret

Given a player i plays from time *1* to T, let σ_i^t denotes his strategy in t, we can derive the **average strategy** over time as

$$\begin{aligned}\bar{\sigma}_i^t(a|I) &= E_{\pi_i^{\sigma^t}(I)/C}(\sigma^t(a|I)) && \text{where } C \text{ is a normalizing constant } \sum_{t=1}^{T}\pi_i^{\sigma^t}(I)\\ &= E_{\pi_i^{\sigma^t}(I)/\sum_{t=1}^{T}\pi_i^{\sigma^t}(I)}(\sigma^t(a|I)) && \text{that turns } \pi_i^{\sigma^t}(I) \text{ a distribution over } t\\ &= \frac{\sum_{t=1}^{T}\pi_i^{\sigma^t}(I)\sigma^t(a|I)}{\sum_{t=1}^{T}\pi_i^{\sigma^t}(I)}\end{aligned}$$

The equation is expanded to explain how "average over time" is derived.

At the same time, we can define the **average strategy regret** at time T for player i as

$$R_i^T = \frac{1}{T}\max_{\sigma_i^*}\sum_{t=1}^{T}\left(u_i\left(\sigma_i^*, \sigma_{-i}^t\right) - u_i\left(\sigma^t\right)\right)$$
$$\text{where } \left(\sigma_i^*, \sigma_{-i}^t\right) \text{ denotes the combined strategy of } \sigma_i^* \text{ for player } i \text{ and } \sigma_{-i}^t \text{ for others}$$

The closer σ_i^t to σ^*, the smaller the R_i^t is

We can see that the R_i^t measures the utility distance between current strategy σ_i^t and the Nash equilibrium strategy σ^*_i for player i. Then it comes the key theorem that opens a possibility of Nash equilibrium approximation:

> *Theorem-1: In a zero-sum game at time t, if both player's average overall regret is less than ϵ , then the average strategy is a 2ϵ equilibrium strategy.*

This theorem tells us that if we can **minimize R_i^t to near zero**, then we just need to **average all the strategies used over time**, which will be the **Nash equilibrium strategy**. It is fascinating 🤪

Proof of above Theorem

After some surveys, I cannot find official proof for the theorem, which is more like a folk theorem. Therefore, I present my proof steps. Let us assume u_i is convex:

$$
\begin{aligned}
\epsilon \geq R_i^T &= \max_{\sigma_i^*} \frac{1}{T} \sum_{t=1}^{T} \left(u_i\left(\sigma_i^*, \sigma_{-i}^t\right) - u_i\left(\sigma^t\right) \right) \\
&= \max_{\sigma_i^*} \left(\frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma_i^*, \sigma_{-i}^t\right) - \frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma^t\right) \right) \\
&\because u_i \text{ is convex} \therefore \frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma_i^*, \sigma_{-i}^t\right) \geq u_i\left(\sigma_i^*, \bar{\sigma}_{-i}^t\right) \\
&\geq \max_{\sigma_i^*} u_i\left(\sigma_i^*, \bar{\sigma}_{-i}^t\right) - \frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma^t\right) \\
&\star \text{ let } h \text{ be the root of game tree, } h' \text{ be children of } h \\
&\because u_i(\bar{\sigma}^t) = \sum_{a \in A(h), h'} u_i(h') \bar{\sigma}^t(a|h) = \sum_{a \in A(h), h'} u_i(h') E_{\pi^{\sigma^t}(I)/C}(\sigma^t(a|I)) \\
&\because \frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma^t\right) = \sum_{a \in A(h), h'} u_i(h') \frac{1}{T} \sum_{t=1}^{T} \sigma^t(a|h) \\
&\because \frac{1}{T} \sum_{t=1}^{T} \sigma^t(a|h) \text{ is an unbiased estimator of } E_{\pi^{\sigma^t}(I)/C}(\sigma^t(a|I)) \\
&\therefore \text{ we can always find a } T \text{, which makes } \left| \frac{1}{T} \sum_{t=1}^{T} u_i\left(\sigma^t\right) - u_i\left(\bar{\sigma}^t\right) \right| \leq \epsilon \\
\therefore \epsilon &\geq \max_{\sigma_i^*} u_i\left(\sigma_i^*, \bar{\sigma}_{-i}^t\right) - u_i\left(\bar{\sigma}^t\right) - \epsilon \\
2\epsilon &\geq \max_{\sigma_i^*} u_i\left(\sigma_i^*, \bar{\sigma}_{-i}^t\right) - u_i\left(\bar{\sigma}^t\right) \\
\therefore \bar{\sigma}^t & \text{ is a } 2\epsilon\text{-Nash Equilibirum strategy}
\end{aligned}
$$

Amateur Proof of Theorem 1

Counterfactual Regret Bound

With the theorem above, now our task is to find a method that can **minimize** R_i^t **by updating** σ^t **iteratively**, which is precisely the counterfactual regret minimization (CFR) method. There are two main points of CFR:

1. Discover an upper bound for R_i^t with counterfactual regret $R_i^t(I, a)$
2. Propose to use an online optimization method called **regret matching** to minimize each $R_i^t(I, a)$ individually by updating $\sigma^t(a|I)$.

The **counterfactual regret** $R_i^t(I, a)$ is defined on the level of the information set as below

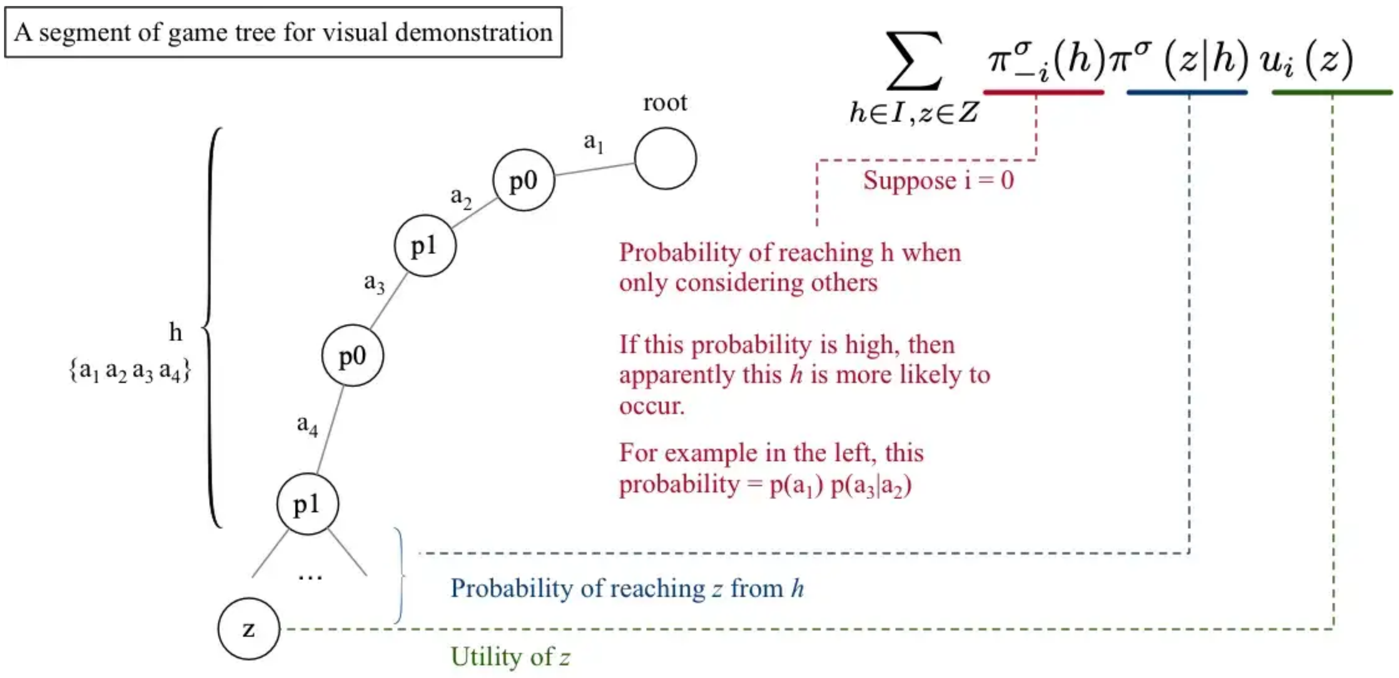
$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T}\left(v_i\left(\sigma^t_{I\to a}, I\right) - v_i\left(\sigma^t, I\right)\right)$$

$$\star\text{where } v_i(\sigma, I) \text{ is the counterfactual value function}$$
$$v_i(\sigma, I) = \sum_{h\in I, z\in Z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(z|h)\, u_i(z)$$

where $\sigma^t\{I \to a\}$ is the strategy that identical to σ^t except that it will **always choose action *a* when encountering** *I*, $\pi(z|h, \sigma)$ is the probability of reaching *z* given current history *h*.

Understanding Counterfactual Regret

We first look at the counterfactual value function $v_i(\sigma^t, I)$, which consists of three parts, as illustrated in the following figure:


A segment of game tree for visual demonstration
$\sum_{h\in I,z\in Z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(z|h)\,u_i(z)$
root
a_1
p0
a_2
p1
a_3
p0
a_4
p1
h
$\{a_1\, a_2\, a_3\, a_4\}$
Suppose i = 0
Probability of reaching h when only considering others
If this probability is high, then apparently this h is more likely to occur.
For example in the left, this probability = $p(a_1)\, p(a_3|a_2)$
...
z
Probability of reaching z from h
Utility of z

We can see that $v_i(\sigma^t, I)$ is a reasonable definition to **measure the utility of information set *I*** based on the possibility of reaching *I* and potential terminal utility from *I*.

> *"Counterfactual" means contrary to fact. Intuitively, the counterfactual regret closely measures the* ***potential gain*** *if we do something contrary to the fact, such as deliberately making action a at I instead of following σ^t.*

Upper Bound for Average Strategy Regret

Upon the definition of counterfactual regret, we discover the following theorem, which makes us one more step closer to approximate the Nash Equilibrium.

$$\text{Theorem-2: } R_i^T \leq \sum_I R_{i,\text{imm}}^{T,+}(I)$$

where we define

$$R_{i,imm}^T(I) = \max_{a \in A(I)} R_i^T(I,a), \text{ as the immediate counterfactual regret}$$

$$R_{i,imm}^T(I) = \max\left\{R_{i,imm}^T(I), 0\right\}$$

Counterfactual Regret Bound for Average Strategy Regret

To understand this bound, we can treat $R^t_{i\ imm}$*(I, a) as a distance between the current strategy* σ^t *and another strategy* σ^t*{I → a*}*, which can act **perfectly** at information set *I* while identical to σ^t in other places. Therefore, the **smaller the** $R^t_{i\ imm}$*(I, a)* **is, the better the strategy** σ^t **is at** *I*. By summing over all information sets, we can optimize the overall performance of σ^t.

The above theorem tells us that we can **minimize** R^t_i **by just minimizing each** $R_i{}^t$*(I, a)*, which significantly simplifies the problem.

Regret Matching

After the simplification, we still need a method to minimize each $R_i{}^t$*(I, a)* by updating σ^t*(a|I)*. It turns out the current problem is in line with the definition of an online convex optimization algorithm called **regret matching** [8], which provides us the following iterative formula to update the strategy σ^t*(a|I)* for each timestep.

$$\sigma^{T+1}(a|I) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a' \in A(I)} R_i^{T,+}(I,a')} & \text{if the denominator is positive (non-zero);} \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases}$$

$$\text{where } R_i^{T,+}(I,a) = \max\left\{R_i^T(I,a), 0\right\}$$

At this point, we have a complete picture of CFR. In CFR, we use **regret matching to update our strategy** σ^t iteratively and **return the average strategy** as the approximated

Nash equilibrium strategy **after enough iterations.** It is that simple 😃

Notes about Theorem-2 Proof

The full proof of theorem 2 can be found in the appendix of [2]. This section does not intend to be a full introduction of the proof and only takes a little note on one of the proving steps.

Meanwhile, sorry for skipping the discussion of the proof of regret matching; that topic might need a separate post.

> *The crucial step in the proof is Eq (10) in A.1 [2], which seems not to be evident at first glance, while other steps are relatively easier to follow. I expand the key segment of Eq (10) in the following.*

$$
\begin{aligned}
\max_{\sigma'} v_i\left(\sigma^t|_{D(I)\to\sigma'}, I\right) &= \max_{\sigma'} \sum_{h\in I, z\in Z} \pi_{-i}^{\sigma^t|_{D(I)\to\sigma'}}(h)\,\pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h)\,u_i(z) \\
&\because D(I) \text{ represents information sets reachable from } I \\
&= \max_{\sigma'} \sum_{h\in I, z\in Z} \pi_{-i}^{\sigma^t}(h)\,u_i(z)\,\pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h) \\
&\star \text{a common but smart trick in the next line} \\
\max_{\sigma'} \pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h) &= \max_{\sigma', a\in A(h)} \left(\pi^{\sigma^t|I\to a}(z|h) - \pi^{\sigma^t|I\to a}(z|h) + \pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h)\right) \\
&\star \text{let } h' \text{ be children of } h \\
&= \max_{\sigma', a\in A(h)} \left(\pi^{\sigma^t|I\to a}(z|h) + \sum_{a\in A(h), h'} \pi(h'|h,a)\left(\pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h') - \pi^{\sigma^t|I\to a}(z|h')\right)\right) \\
&\because h' \text{ passes over } I \text{ and we only keep the best } a \\
&= \max_{\sigma', a\in A(h)} \left(\pi^{\sigma^t|I\to a}(z|h) + \sum_{h'} \pi(h'|h,a)\left(\pi^{\sigma^t|_{D(I)\to\sigma'}}(z|h') - \pi^{\sigma^t}(z|h')\right)\right) \\
&\star \text{from the last line, Eq(10) can easily follow.}
\end{aligned}
\tag{1}
$$

Algorithm Procedure

After we have a basic understanding of the mathematical principles of CFR, we now introduce some implementation details. Recall the definition of counterfactual value function:

$$\sum_{h\in I,\underbrace{z\in Z}_{\text{All possible endings } z \text{ from } h}} \pi^\sigma_{-i}(h)\underbrace{\pi^\sigma(z|h)}_{\text{Probability from } h \text{ to } z} u_i(z)$$

Decomposition of $v_i(\sigma, I)$

Therefore, to cover links to all possible endings *z* from *h,* we need a **full depth-first traversal** of the game tree to compute counterfactual value and regret. We usually implement the traversal using recursion.

The pseudo-code of CFR is below.

```
def cfr(
    h=[],    # history
    π_i={},  # a mapping from player_id to π_i(h)
    i=0,     # player_id
    t=0      # timestep
):
    # here I skip the chance node case for simplicity
    if h is terminal:
        z = h
        return u(i, z)  # utility u(z) for player i

    # information set of the history, h ∈ I
    I = h.I

    # R[t] is counterfactual regret (without maximum), R^t
    # σ[t] is the strategy from regret matching at time t, σ^t
    σ[t][I] = regret_matching(R[t][I])

    # counterfactual value divided by π_{-i}(h), v(σ|I -> a) / π_{-i}(h)
    # I use `**` dict unpack syntax like here https://www.codespeedy.com/merge-two-dictionaries-in-python-update-double-star/
    v[I] = {a: cfr(h + [a], {**π_i, P(h): π_i[P(h)] * σ[t][I][a]}, i, t)
                                                  for a in A[I]}

    # counterfactual value v(σ) divided by π_{-i}(h)
    # expectation of v(σ|I -> a) / π_{-i}(h) over actions
    vI = sum(σ[t][I][a] * via for a,via in v[I].items())

    for a in A[I]:
      # π_{-i}(h), probability of reaching h when only
      # considering other players
      π_neg_i_h = prod(prob for pid, prob in π_i.items() if pid != P(h))

      # counterfactual regret at time t + 1
      R[t+1][I][a] += π_neg_i_h * (v[I][a] - vI)

      # strategy average over time
      σ_avg[I, a] += π_i[P(h)] * σ[t][I][a]
    return vI

# using
for iteration in range(10000):  # 10000 iterations
  for player_id in range(2):    # 2 player
    cfr(i=player_id, t=iteration)
```

cfr.py hosted with ❤ by GitHub view raw

procedures we explained in the previous section. Therefore, I will not go into details. But there are two things to mention:

1. Every call of `cfr` returns the counterfactual value divided by $\pi_{-i}(h|\sigma)$ to simplify its recursive calculation.
2. We omit the denominator of the average strategy, which is a constant that will not affect the resulting probability.

Monte Carlo CFR

I will introduce a popular variant of CFR, Monte Carlo CFR (MCCFR) [3]. But before we dive into that topic, let us first calculate the time and space complexity of CFR.

Clearly, the space complexity is $O(|I| * |A|)$, where $|I|$ stands for the total number of information sets because we need to store $R^t_i(I, a)$ and $\sigma^t(a|I)$. The time complexity is $O(|h|* N)$, where $|h|$ stands for the number of histories (or nodes) in the game tree and N denotes the number of iterations. For a large game like Texas Hold'em, $|h|$ can go up to 10^{18} [9], which CFR is not efficient enough to solve. This is where MCCFR comes in to solve.

General Idea

The idea of MCCFR is very straightforward. **Instead of** traversing the **entire game tree** and reaching every possible game endings z, we **sample a small number of** z according to a prior distribution $q(z)$ and only traverse links that reach them.

Specifically, we split Z into subsets of $\{Q_1, \ldots, Q_r\}$; we call one of these subsets a **block**, where each block Q_i is a smaller set of terminal histories.

Let q_j be the probability of choosing Q_j, then at each iteration, we first sample a block Q_j and only traverse parts of the game tree that reach game ending $z \in Q_j$.

$$q(z) = \sum_{j:z\in Q_j} q_j$$

Sampling Distribution of q(z)

Meanwhile, we calculate a **sampled counterfactual value** as below:

$$v_i^s(\sigma, I \mid j) = \sum_{h \in I, z \in Q_j} \frac{1}{q(z)} u_i(z) \pi_{-i}^{\sigma}(h) \pi^{\sigma}(z|h)$$

We can easily prove that the expectation of sampled counterfactual value is equal to the original one, which means MCCFR can work just as well as CFR, but simultaneously has great flexibility in designing the sampling strategy, making it much more efficient.

$$E_{j \sim q_j}\left[v_i^s(\sigma, I \mid j)\right] = v_i(\sigma, I)$$

> When there is only one Q_1, $Q_1 = Z$, then $v^s_i(\sigma, I |j) = v_i(\sigma, I)$, MCCFR is equal to CFR.

Theoretically, we only need to **change the calculation of** $v_i(\sigma, I)$ **and sampling of** z **in CFR to become MCCFR.** In practice, it is difficult to sample z without actually reaching all of them by recursion. Therefore, we must cleverly design $q(z)$ in order to sample z but without an extra step of reaching them. I will briefly introduce two sampling methods that meet the requirement.

Outcome Sampling

In outcome sampling, we make the following designs:

1. Each block Q_j only contains a single terminal history z
2. Set $q(z) = \pi(z|\sigma^t)$

It seems simple, but they indicate we only need to **sample one action at each step,** instead of all actions like in CFR pseudo-code

```
# in cfr, we simulate all actions
v[I] = {a: cfr(h + [a], {**π_i, P(h): π_i[P(h)] * σ[t][I][a]}, i, t)
                                                 for a in A[I]}

# in outcome sampling mccfr, we only need to sample one a from A[I]
a = sample(A[I], σ[t][I]) # or use `ϵ * uniform + (1-ϵ) * σ[t][I]`
v[I][a] = mccfr(h + [a], {**π_i, P(h): π_i[P(h)] * σ[t][I][a]})
```

outcome_sampling.py hosted with ❤ by GitHub view raw

Intuitively, it is like we **merge the sampling of z with the process of recursive computation**. At the deepest level of recursion, we return $\pi(z|\sigma^t)$ as $q(z)$, which will be used by the upper level to calculate $v^s_i(\sigma, I\,|j)$.

> *The time complexity of Outcome Sampling MCCFR is* O(d * N), *where d is the depth for the game tree, much smaller than* $|h|$.

By outcome sampling, the more likely a terminal history z happens under strategy σ, the more we will sample it, which is reasonable. In practice, to balance between exploitation and exploration, we can also use ϵ-on-policy, which sample actions from a linear combination of $\sigma^t(a|I)$ and uniform distribution.

External Sampling

In external sampling, we set $q(z)=\pi_{-i}(z|\sigma)$, which looks similiar to outcome sampling, but we exclude the influence of current player strategy σ_i over $q(z)$.

To archive that, we will need to traverse all available actions at $\{h \mid P(h) = i\}$, for game histories where other players with the move, we sample an action according to σ_{-i}, just like the outcome sampling.

```python
if P(h) == i:
  # traverse all available actions, to illiminate influence of σ_i
  v[I] = {a: mccfr(h + [a], {**π_i, P(h): π_i[P(h)] * σ[t][I][a]}, i, t)
                                                  for a in A[I]}
else:
  # sample one a from A[I]
  a = sample(A[I], σ[t][I])
  v[I][a] = mccfr(h + [a], {**π_i, P(h): π_i[P(h)] * σ[t][I][a]})
```

external_sampling.py hosted with ❤ by GitHub view raw

Further Notes for MCCFR

There are many more details we could discuss further about MCCFR, such as

- The method of calculating the average strategy. Because we have biased the σ^t with sampling strategy on z, and $R^t_i(I, a)$ is unable to update for all information sets at every timestep. Therefore, we need a new way to compute the average strategy.
- Convergence speed for different sampling methods.

However, I will not discuss them here for reasons of space; this post has already gotten way longer than I expected 😂. If you are interested, I highly recommend you to read the phd thesis from Marc Lanctot [7], which is very well written and covers this topic comprehensively.

Summary

This post introduces Counterfactual Regret Minimization (CFR), a popular method for solving imperfect information games. I hope it is helpful to you.

CFR is a truly fascinating work, which starts from profound principles and arrives at a simple but powerful algorithm framework. The background knowledge and mindset are quite different from other classical machine learning methods. It is an exciting challenge and big fun to write this article.

At the same time, I have finished the plan to write three blogs about the AI game-playing algorithm. I will probably write more blogs on this topic in the future. See you then :D

Reference

[1] A Survey of Monte Carlo Tree Search Methods
http://ccg.doc.gold.ac.uk/ccg_old/papers/browne_tciaig12_1.pdf

[2]: Regret Minimization in Games with Incomplete Information
http://citeseerx.ist.psu.edu/viewdoc/download?
doi=10.1.1.131.8230&rep=rep1&type=pdf

[3] Monte Carlo Sampling for Regret Minimization (MCCFR) in Extensive Games
http://mlanctot.info/files/papers/nips09mccfr.pdf

[4] Information set (game theory)
https://en.wikipedia.org/wiki/Information_set_(game_theory)

[5]: Superhuman AI for multi-player poker
https://science.sciencemag.org/content/365/6456/885

[6]: Kuhn poker https://en.wikipedia.org/wiki/Kuhn_poker

[7]: Monte Carlo Computation Sampling And Regret Minimization For Equilibrium And Decision-making In Large Extensive Form Games, Marc Lanctot, Phd Thesis
http://mlanctot.info/files/papers/PhD_Thesis_MarcLanctot.pdf

[8]: A SIMPLE ADAPTIVE PROCEDURE LEADING TO CORRELATED EQUILIBRIUM
http://www.dklevine.com/archive/refs4572.pdf

[9] Tartanian5: A Heads-Up No-Limit Texas Hold'em Poker-Playing Program
https://www.cs.cmu.edu/~sandholm/Tartanian_ACPC12_CR.pdf

Artificial Intelligence Machine Learning Probability Game Theory

Reinforcement Learning