STUDY NOTES ON COUNTERFACTUAL REGRET MINIMIZATION: TOWARDS KOI-KOI

Hiroki Takizawa

October 5, 2020

ABSTRACT

ナッシュ均衡は、ゲーム理論における基本的な概念の一つであり、経済学だけでなくゲーム AI の設計や数理生物学などの多くの分野で活用されている。Counterfactual Regret Minimization (CFR) は展開型ゲームのナッシュ均衡解を近似的に求める手法のひとつであり、ポーカー AI の設計で成功を収めてきた。この記事では最初の CFR アルゴリズムを詳しく説明し、後続の発展的な手法たちの解説も行う。最後に、こいこいの AI 設計へ各手法を適用できる可能性について論じる。

1 イントロダクション

Counterfactual Regret Minimization (CFR) は有限不確定不完全情報ゲームのナッシュ均衡解を反復的に近似計算する手法のひとつである。初出は 2008 年 [1] である。近似度 ϵ を達成するのに必要な反復回数 δ の理論的な既知の上界は他の手法より悪い [2] のだが、実際は速く収束することが経験的に知られている。それに加えて理解も実装も比較的易しいことから、ポーカーソフトなどに広く用いられており [3, 4]、研究も進んでいる。

この記事では CFR とその発展手法のアルゴリズムを解説する。CFR 系手法に対する既存の解説記事は日本語でも英語でも存在するので、この記事は既存の記事と合わせて読んでも価値があるように書いた。すなわち、原論文を読む際の助けになるように用語の説明を充実させ、アルゴリズムの導出における式変形なども詳しく説明するよう心がけた。また、シュードコードも分かりやすく自己完結的なものとした。

2 原論文で出てくる用語

以降の章では、論じる対象を二人零和ゲームに限定する。CFR は本来は二人零和ゲームに限定されない手法だが、こいこいが二人零和ゲームであることからそのようにする。

2.1 プレイヤー / players

対局している 2 人のこと。プレイヤーの有限集合は N で表される。論文によって $N = \{1, 2\}$ のときと $N = \{0, 1\}$ のときがあるので注意。

2.2 チャンスプレイヤー / chance

ある種の擬人化である。直観的に言うと、サイコロを振ったり山札からカードを引いたりめくったりする行為（をする人）のこと。考察の対象は 2 人ゲームだが、以降の議論ではチャンスプレイヤーを入れた 3 人ゲームとして考えると見通しが良くなる。c で表されることが多い。$N \cup \{c\}$ と書かれることがあり、その場合 $c = -1$ のときと $c = 0$ のときがあるのでシュードコードを読むときに注意。

2.3 アクション / action

合法手のこと。a で表される。プレイヤーが行為するとは、有限のアクションたちの中からひとつを選ぶことである。

2.4 ヒストリー / history

直観的に言うと、神の視点から見た「棋譜」のこと。終了時だけでなく、開始時や途中も含む。ヒストリーの有限集合は H で表される。1 つのヒストリーは $h \in H$ で表される。

- h は a の 0 個以上のリストである。
- 開始時のヒストリーは $h = \emptyset$ で表される。
- 終了時のヒストリーの集合は Z で表される。$Z \subseteq H$ である。
- h の直後にアクション a を行ったヒストリーは (h, a) とか ha と書いて表される。リストの末尾にくっつけているだけである。（C++の push_back や Python の append のように）
- $h \in H$ かつ $h \neq \emptyset$ とする。ここで、h の末尾のアクションを取り除いたヒストリー h' を考える。すなわちあるアクション a が存在して $h = (h', a)$ だとする。このとき必ず $h' \in H$ となる。一般に、ヒストリー $h \in H$ をリストとして見たとき、その任意のプレフィックス h' について $h' \in H$ が成り立つ。
- ヒストリー h' が h のプレフィックスであることを $h' \sqsubseteq h$ で表す。
- $h \in Z$ はいかなるヒストリーのプレフィックスでもない。
- 任意の $h \in H \setminus Z$ に対しあるアクション a が存在して、$(h, a) \in H$ が成り立つ。
- $(h, a) \in H$ であることは a が h における合法手であることと同値である。

2.5 アクション関数 / action function

直観的に言うと、合法手を全列挙する関数である。$A(h)$ で表される。ヒストリー $h \in H \setminus Z$ を引数に取り、$(h, a) \in H$ なる a を全て出力する。すなわち

$$A(h) = \{a : (h, a) \in H\}$$

である。

2.6 プレイヤー関数 / player function

ヒストリー $h \in H \setminus Z$ を引数に取り、次にアクションするのが誰かを返す関数である。$P(h)$ で表される。

$$P(h) \in \{N \cup \{c\}\}$$

である。論文中でシュードコードを書くときに便利。

2.7 チャンスプレイヤーの確率分布

直観的に言うと、ランダム性のあるゲームにおけるランダムな行為の確率分布は、ゲームのルールとして全てのプレイヤーに予め知らされているということ。堅く言うと以下のようになる。

$P(h) = c$ なる任意の h に対して、離散確率分布関数 $f_c(\cdot|h)$ が定義される。確率値 $f_c(a|h)$ は、ヒストリー h においてチャンスプレイヤーがアクション a を取る確率を意味する。

2.8 情報分割 / information partition

各プレイヤー $i \in \{N \cup \{c\}\}$ について、そのプレイヤーの手番であるようなヒストリー全てからなる集合を $\mathcal{I}_i$ で表す。すなわち

$$\mathcal{I}_i = \{h \in H : P(h) = i\}$$

である。任意のヒストリー $h \in H$ は Z か $\mathcal{I}_i$ のどれか一つだけに必ず含まれる。換言すると、H は $\mathcal{I}_i$ と Z とによって相互排他的かつ網羅的に分割される。

2.9 情報集合 / information set

直観的に言うと、プレイヤー $i \in N$ の視点から見た「局面」であり「棋譜」でもある概念のこと。「現在のヒストリーとしてありうるもの全ての集合」とも言える。情報集合は I_i で表される。$I_i \subseteq \mathcal{I}_i$ である。任意の 2 ヒストリー $h, h' \in \mathcal{I}_i$ に関して、プレイヤー $i \in N$ の視点（i は自分が現在 I_i にいると知っている）から見て h と h' を区別できないことは $h, h' \in I_i$ であるための必要十分条件であるとする。すると、

- $A(h) = A(h')$ が成り立つ。
- （後述の perfect recall を仮定するならば）任意のヒストリー $h \in \mathcal{I}_i$ に対して、それを含むような情報集合 I_i が一意に存在する。換言すると、$\mathcal{I}_i$ は I_i によって相互排他的かつ網羅的に分割される。
- そのため、I_i は $\mathcal{I}_i$ の要素であるとして $I_i \in \mathcal{I}_i$ と書くこともできる。（$I_i \subseteq \mathcal{I}_i$ ではなく）
- $A(h)$ や $P(h)$ と同様に $A(I_i)$ や $P(I_i)$ を定義できる。

不完全情報ゲームにおいては一般に、対局中のプレイヤー i の視点からは現在のヒストリーを一意に特定できない。なぜなら（相手の手札とかの）見えない部分に複数通りの可能性がありうるからである。そのため、i から見たときの「現在のヒストリーとしてありうるもの全ての集合」が要請される。それが情報集合である。

神の視点からは全プレイヤーの全情報が見えるので、ヒストリーは常に一意に定まることに注意。また、完全情報ゲームではプレイヤー視点でもヒストリーは常に一意に定まる。（すなわち常に $|I_i| = 1$）

以上がヒストリーと情報集合の本質的な違いであり、完全情報ゲームと不完全情報ゲームの差でもある。

不完全情報ゲームの議論で「局面」という単語を使うとヒストリーのことなのか情報集合のことなのか曖昧になってしまうので、できるだけ使わないほうがいいと思われる。この記事でも極力避けた。

2.10 perfect recall

全プレイヤー $i \in N$ が対局中の全時点において「過去の全てのアクションとそのときの情報集合」を完全に覚えているという問題設定を parfect recall と呼ぶ。これは一般的に用いられる設定であり、この記事でも parfect recall を仮定して議論する。

情報集合は「i 視点での、現在のヒストリーとしてありうるもの全ての集合」なわけだが、これをフォーマルかつ端的に言い表すことは難しい。論文 [1, 5] などでは $A(h) = A(h')$ をもって情報集合の定義としていたが、これは「アクション」に「合法手」以上の意味をもたせていることを暗黙的な前提としている。

perfect recall の仮定のもとでは、「アクション」の定義を「その（ヒストリー | 情報集合）における合法手」というだけではなく「『過去の全てのアクションとそのときの（ヒストリー | 情報集合）』と『現在の（ヒストリー | 情報集合）における合法手』のタプル」とすれば、情報集合の形式的な定義は前述の「任意の二局面 $h, h' \in I_i$ に関して $A(h) = A(h')$」で問題ないと言える。

2.11 利得関数 / utility function

直観的に言うと、勝ち点を決めるルールのこと。u_i で表される。$u_i(h)$ は $h \in Z$ を引数に取り、プレイヤー i が得る得点を返す。

2.12 ゼロサムゲーム / zero-sum game

$N = \{1, 2\}$ とする。任意の $h \in Z$ に対して $u_1(h) = -u_2(h)$ のとき、そのゲームをゼロサムゲームと呼ぶ。

2.13 プレイヤー i の戦略 / strategy of player i

直観的に言うと、プレイヤー i が各情報集合でどのアクションをどの確率で選ぶかの分布のことである。プレイヤー i の戦略は σ_i で表される。これは $I_i \in \mathcal{I}_i$ においての $A(I_i)$ 上の確率分布である。つまり、情報集合とアクションを引数に取って確率を返す関数 $\sigma_i(a|I_i)$ などと書く。

任意の $i \in N, I_i \in \mathcal{I}_i, a \in A(I_i)$ について $\sigma_i(a|I_i)$ が 0 か 1 であるような戦略を純粋戦略と呼ぶ。さもなくば混合戦略と呼ぶ。換言すると、どの情報集合においても決定論的に行動する戦略が純粋戦略である。

任意の $h \in H \setminus Z$ に対して $h \in I_{P(h)}$ なる $I_{P(h)}$ が一意に存在することから、$\sigma_{P(h)}(a|h)$ と書くこともできる。

チャンスプレイヤーについては常に、$\sigma_c(a|h) = f_c(a|h)$ である。

ちなみに、σ_i をコンピュータ上で表現する場合、全ての可能な I_i をキーとして、各アクションの確率（具体的には double 型配列とか）を値とする巨大なハッシュテーブルとかになる。後でまた触れるが、Vanilla CFR では実際にこれを保持する必要があり、そのメモリへの負荷が Vanilla CFR の問題点となっている。

可能な σ_i 全てからなる非可算無限集合は Σ_i で表される。

2.14 strategy profile

全プレイヤーの σ_i を合わせて strategy profile と呼び、σ と書く。また、プレイヤー i 以外全員の strategy を σ_{-i} と書く。

2.15 到達確率

全プレイヤーが strategy profile σ に従ったときにヒストリー h に到達する確率を $\pi^\sigma(h)$ と書く。すなわち、$\pi^\sigma(h) = \prod_{h'a \sqsubseteq h} \sigma_{P(h')}(a|h')$ である。

- $h \sqsubseteq z$ なる h と z について、h に到達したという条件のもとで h から z に到達する確率を $\pi^\sigma(h,z)$ と書く。すなわち、$\pi^\sigma(h,z) = \dfrac{\pi^\sigma(z)}{\pi^\sigma(h)}$ である。
- $\pi^\sigma(h) = \prod_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ というように、各プレイヤーの寄与率で分解することができる。
- $\pi_{-i}^\sigma(h)$ は、プレイヤー i 以外が σ に従い、プレイヤー i は h に向かうアクションを可能な限り取り続けたときに h に到達できる確率を意味する。すなわち、$\pi_{-i}^\sigma(h) = \dfrac{\pi^\sigma(h)}{\pi_i^\sigma(h)}$ である。
- 情報集合 $I \subset H$ について $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$ と定義する。
- $\pi_i^\sigma(I)$ や $\pi_{-i}^\sigma(I)$ を同様に定義できる。

2.16 期待利得 / expected payoff

全プレイヤーが strategy profile σ に従ったときにプレイヤー i が得る利得は $u_i(\sigma)$ で表される。すなわち

$$u_i(\sigma) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$$

である。

プレイヤー 1 が σ_1 に従い、プレイヤー 2 が σ_2 に従うときにプレイヤー i が得る利得を $u_i(\sigma_1, \sigma_2)$ と書いてもよい。σ_c はルールによって決まっているので省略している。

2.17 (イプシロン) ナッシュ均衡 / (epsilon-) Nash equilibrium

2 人ゲームでの ϵ ナッシュ均衡解とは、以下の条件を満たす strategy profile σ のことである。

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

$\epsilon = 0$ のときが古典的なナッシュ均衡解である。

混合戦略のナッシュ均衡解は必ず一つ以上存在する。[6]

2.18 average overall regret

あるゲームを T ラウンド繰り返しプレイしたとして、t ラウンド目における strategy profile が σ_t だったとする。このとき、プレイヤー i の T ラウンド後における **average overall regret** を以下のように定義する。

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t))$$

また、プレイヤー i の T ラウンド後における **average strategy** を以下のように定義する。

$$\overline{\sigma}_i^T(a|I) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^T(a|I)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}$$

ここで、論文 [1] で「よく知られた定理」として紹介されている以下の定理が成り立つ。

- ゼロサムゲームのラウンド T において、両プレイヤーの **average overall regret** が ϵ 未満ならば、$\bar{\sigma}_i^T$ は 2ϵ ナッシュ均衡である。

この σ_i^* も R_i^T も現実には計算できないので、代わりに R_i^T の上界となるような指標を上手く定義してそれを最小化することをこれから目指していく。「真に最小化したい値 X をまず定義して、別の扱いやすい値 Y を導入し、Y が X の上界であることを証明し、Y を最小化するアルゴリズム Z を作る」というよくある戦術が用いられる。

プレイヤー i は相手の strategy を全て知っているとして、「自分がもし別の strategy を採用していたらより大きな利得が得られたのになあ」と、反実仮想によって後悔することができる。その仮想において、最大の利得が得られるような strategy が σ_i^* である。それを採用したときの仮想と現実の差分の最大値が R_i^T である。

2.19 効用の期待値 / expected utility

ゲームがヒストリー h に到達したとして、以降プレイヤーたちが strategy σ に従うとしたときの、プレイヤー i から見た expected utility $u_i(\sigma, h)$ を定義する。それは以下のように書ける。

$$u_i(\sigma, h) = \sum_{z \in Z, h \sqsubseteq z} \pi^\sigma(h, z) u_i(z)$$

2.20 counterfactual utility

プレイヤー視点では現在のヒストリーは不明であり、情報集合しかわからない。プレイヤーの行動選択は情報集合に基づいて行われる。よって、ゲームが情報集合 $I \in \mathcal{I}_i$ に到達したとして、以降プレイヤーたちが strategy σ に従うとしたときの、プレイヤー i から見た counterfactual utility $u_i(\sigma, I)$ を定義する。それは以下のように書ける。

$$u_i(\sigma, I) = \sum_{h \in I} u_i(\sigma, h)$$

これをなぜ **counterfactual** と呼ぶかというと、プレイヤー i は情報集合 I に向かうアクションを選び続けるという想定だが、相手もそうしてくれるというのは仮想だからである。

counterfactual utility 自体は意味のある値ではないと思う。相手が I を目指してくれる確率が h によって違うことを無視しているので、これ自体を何か意味のある概念に紐づく値として見るのはナンセンスだろう。逆に言えば、$v_i(\sigma, I) = \sum_{h \in I} \pi_{-i}^\sigma(h) u_i(\sigma, h)$ を counterfactual utility の代わりとして導入するという手もある。（実際そういう流儀が後で登場する。counterfactual value と呼ばれている）でも今回そうすると、後に出てくる **immediate counterfactual regret** の式において $\pi_{-i}^\sigma(h)$ を括り出せていないような形になってしまい収まりが悪い。

この論文 [1] の主張としては **immediate counterfactual regret** が最重要で、counterfactual utility はその説明のための準備でしかないのだ。

2.21 一箇所だけ違うストラテジー

任意の $I \in \mathcal{I}_i$ 中の任意の $a \in A(I)$ について、$\sigma|_{I \to a}$ という記法を導入する。$\sigma|_{I \to a}$ は、I 以外の情報集合では σ に従うが、I では必ず a を選ぶような strategy を意味する。

2.22 immediate counterfactual regret

あるゲームを T ラウンド繰り返しプレイしたとして、プレイヤー i から見た、情報集合 I における **immediate counterfactual regret** $R_{i,imm}^T(I)$ を以下のように定義する。

$$R_{i,imm}^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I))$$

2.23 原論文の key result, Theorem 3

論文 [1] で以下の定理が示されている。(Theorem 3)

$$R_i^T \leq \sum_{I \in \mathcal{I}_i} \max(R_{i,imm}^T(I), 0)$$

原論文では上付きの+で $\max(\cdot, 0)$ を表現していたりするが、その表記法は上付きの $T+1$ とかと紛らわしいと思い採用しなかった。

2.24 Blackwell の接近性定理 / Blackwell's approachability theorem

average overall regret R_i^T の最小化は現実的に計算できなかったが、**immediate counterfactual regret** $R_{i,imm}^T(I)$ の最小化は、個別の $\sigma_i(I)$ に注目するだけで計算できる。論文 [1] によると、このことの証明には Blackwell の接近性定理が使える。

2.25 変数 R(I,a) (cumulative counterfactual regret)

immediate counterfactual regret 最小化アルゴリズムにおいて、我々は全ての情報集合 $I \in \mathcal{I}_i$ の全てのアクション $a \in A(I)$ について、以下の変数 $R_i^T(I,a)$ を保持する。

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I))$$

論文 [1] ではこの変数に名前はついていなかったが、後の論文では **cumulative counterfactual regret** と呼ばれている。以下は tips である：

- 論文 [1] には以上のように書いてあるが、実装上は $R_i^T(I,a) = \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I))$ でよい。つまり、T ラウンド目までの総和の値だけ保持すればよい。
- 実装上、すべての可能な情報集合 I について、ハッシュテーブルかなにかで $R_i^T(I,a)$ を陽に持っておく必要があり、ゲームの規模によっては大量のメモリを必要とする。これが Vanilla CFR の大きな問題点であり、後の論文たちの課題になっている。
- この変数 $R_i^T(I,a)$ は、前に出てきた **average overall regret** R_i^T とは異なる。紛らわしいし、前述した通り $\frac{1}{T}$ が実装上無意味だというのと合わせて、この変数の表記法はあまり良くないと思う。

2.26 各ラウンドでの strategy

$T+1$ ラウンド目での strategy $\sigma_i^{T+1}(I,a)$ を以下のように定める。

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

最初のラウンドでは常に後者の一様な strategy とする。このような方法を一般に **regret matching** と呼ぶらしい。

論文 [1] の Theorem 4 によると、上記の strategy で反復的に自己対戦すれば、T を無限大に飛ばしたときに average strategy がナッシュ均衡解に収束する。

3 Vanilla CFR Algorithm

ヒストリー h の全体がなすゲーム木を深さ優先探索するという形で記述した。

Algorithm 1 Vanilla CFR

Declare cumulative regret tables: $r_I[a]$ for all I and a.
Declare cumulative strategy tables: $s_I[a]$ for all I and a.
Declare strategy profiles: $\sigma^n(I,a)$ for all I and a and $n \in \{1,2,3,...\}$.

function CFR_DFS$(h, t, \pi_1, \pi_2, \pi_c)$
if $h \in Z$ **then**
  return $u_1(h)$
else if $P(h) = c$ **then**
  $u \leftarrow 0$
  for $a \in A(h)$ **do**
    $u \leftarrow u + \sigma^t(a|h) \cdot \text{CFR\_DFS}(ha, t, \pi_1, \pi_2, \sigma^t(a|h) \cdot \pi_c)$
  end for
  return u
end if
Let I be the information set s.t. $h \in I$ and $I \in \mathcal{I}_{P(h)}$.
$u_\sigma \leftarrow 0$
$u_{\sigma_{I \to a}}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(h)$ **do**
  if $P(h) = 1$ **then**
    $u_{\sigma_{I \to a}}[a] \leftarrow \text{CFR\_DFS}(ha, t, \sigma^t(a|I) \cdot \pi_1, \pi_2, \pi_c)$
  else if $P(h) = 2$ **then**
    $u_{\sigma_{I \to a}}[a] \leftarrow -\text{CFR\_DFS}(ha, t, \pi_1, \sigma^t(a|I) \cdot \pi_2, \pi_c)$
  end if
  $u_\sigma \leftarrow u_\sigma + \sigma^t(a|I) \cdot u_{\sigma_{I \to a}}[a]$
end for
if $P(h) = 1$ **then**
  $\pi_{-i} \leftarrow \pi_2 \cdot \pi_c$
else if $P(h) = 2$ **then**
  $\pi_{-i} \leftarrow \pi_1 \cdot \pi_c$
end if
for $a \in A(h)$ **do**
  $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (u_{\sigma_{I \to a}}[a] - u_\sigma)$
  $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(a|I)$
end for
return u_σ

Algorithm 2 Vanilla CFR (続き)

```
Extern: r_I[a] for all I and a.
Extern: s_I[a] for all I and a.
Extern: σ^n(I, a) for all I and a and n ∈ {1, 2, 3, ...}.

function CFR_REGRET_MATCHING(t)
for all I do
    x ← 0
    for a ∈ A(I) do
        x ← x + max(r_I[a], 0)
    end for
    for a ∈ A(I) do
        if x > 0 then
            σ^{t+1}(I, a) ← max(r_I[a], 0)/x
        else
            σ^{t+1}(I, a) ← 1/|A(I)|
        end if
    end for
end for

function CFR_GET_PROBABILITY(I, a)
x ← 0
for a' ∈ A(I) do
    x ← x + max(s_I[a'], 0)
end for
if x > 0 then
    return max(s_I[a], 0)/x
end if
return 1/|A(I)|

function CFR_SOLVE(T)
for all I do
    for a ∈ A(I) do
        r_I[a] ← 0
        s_I[a] ← 0
        σ^1(I, a) ← 1/|A(I)|
    end for
end for
for t = {1, 2, 3, ..., T} do
    CFR_DFS(∅, t, 1, 1, 1)
    CFR_REGRET_MATCHING(t)
end for
```

3.1 使い方

CFR_SOLVE 関数に十分大きな整数 T を引数として与えると、Vanilla CFR の計算が行われて、グローバル変数 $s_I[a]$ に strategy が入力される。その後、所望の (I, a) について CFR_GET_PROBABILITY 関数を呼ぶことで、（ナッシュ均衡解に近いであろう）行動確率が得られる。

3.2 tips

シュードコードの記述では全ての n について $\sigma^n(I, a)$ を保持するように書いてあるが、実装上は $\sigma^1(I, a)$ だけメモリ確保しておいて、CFR_REGRET_MATCHING のときに上書きしてしまってよい。そのように実装するならば、イテレーション番号 t を引数として与える必要はなくなる。

perfect recall を仮定しているので、ヒストリー h の全体がなすゲーム木は決して DAG にはならず、純粋な木構造になる。言い換えると、『指し手の種類や順番』は異なるが『盤面のスナップショット』が同じになるような複数通りの『局面』がもしあっても、手順が異なるならばヒストリーは必ず異なるので、それらは決して同一視されない。（同一視されうるならばゲーム木は木ではなく DAG になりうる）

4 CFR+で出てくる用語

原論文 [7] は非常に短い論文 (technical report) で、peer-review のプロセスを経てもいないが、後続の研究論文たちに大きな影響を与えている。この論文の少し後に、同じ著者らが同じ手法を用いてある種のポーカーを解いたという学会論文 [8] が publish された。

notation や数式の組み立て方は Vanilla CFR の原論文 [1] ではなく別の論文 [5] に従っている。

理論的な解析は原論文 [7] には記されていない。同じ筆頭著者による学会論文 [8] に記されている。それらを引用して詳しく解説している学位論文 [2] もある。

この記事の以下の subsection では、notation の一部分で 2009 年の論文 [5] ではなく別の 2018 年の論文 [9] を参考にした。

4.1 counterfactual value

Vanilla CFR の原論文 [1] での数式の組み立てかたをおさらいする。expected utility(2.19)、counterfactual utility(2.20)、一つだけ違うストラテジー (2.21)、immediate counterfactual regret(2.22)、cumulative counterfactual regret(2.25) という順番だった。

$$u_i(\sigma, h) = \sum_{z \in Z, h \sqsubseteq z} \pi^\sigma(h, z) u_i(z)$$

$$u_i(\sigma, I) = \sum_{h \in I} u_i(\sigma, h)$$

$$R^T_{i,imm}(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I))$$

$$R^T_i(I, a) = \frac{1}{T} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I))$$

一方で、CFR+論文が依拠しているところの論文 [5] では、同じ immediate counterfactual regret を定義するための準備として、新しい counterfactual value $v_i(\sigma, I)$ という値を以下の通りに定義する。

$$v_i(\sigma, I) = \sum_{h \in I} \sum_{z \in Z, h \sqsubseteq z} \pi^\sigma_{-i}(h) \pi^\sigma(h, z) u_i(z)$$

これを用いると、immediate counterfactual regret は以下のように書ける。（実際の CFR+論文には書かれていないが、書こうと思えば書けるという意味で）

$$R^T_{i,imm}(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} (v_i(\sigma^t|_{I \to a}, I) - v_i(\sigma^t, I))$$

counterfactual value $v_i(\sigma, I)$ は新しい値といっても、新規なコンセプトではない。Vanilla CFR 論文 [7] の流儀の定式化を変形して、前半部分を合体させて一つの値にしただけである。（式をよく見比べるとわかる）

4.2 cumulative counterfactual regret+

CFR+では、counterfactual value を用いて、cumulative counterfactual regret $R^T_i(I, a)$ の代わりになる新規なコンセプトである **cumulative counterfactual regret+** $R^{+,T}_i(I, a)$ を以下の通りに定義する。

$$R^{+,T}_i(I, a) = \begin{cases} \max(v_i(\sigma^t|_{I \to a}, I) - v_i(\sigma^t, I), 0) & \text{if } T = 1 \\ \max(R^{+,T-1}_i(I, a) + v_i(\sigma^t|_{I \to a}, I) - v_i(\sigma^t, I), 0) & \text{otherwise} \end{cases}$$

ちなみに、Vanilla CFR の原論文の流儀により、**cumulative counterfactual regret+**と等価な式を以下のようにも書き下せる。

$$R^{+,T}_i(I, a) = \begin{cases} \max(\pi^{\sigma^t}_{-i}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I)), 0) & \text{if } T = 1 \\ \max(R^{+,T-1}_i(I, a) + \pi^{\sigma^t}_{-i}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I)), 0) & \text{otherwise} \end{cases}$$

この式による regret matching を **regret matching+**と呼ぶ。

4.3 各ラウンドでの strategy

$T+1$ ラウンド目での strategy $\sigma_i^{T+1}(I,a)$ を以下のように定める。

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

最初のラウンドでは常に後者の一様な strategy とする。

4.4 weighted averaging

Vanilla CFR では、ナッシュ均衡解に収束するのは average strategy であって、各ラウンドでの strategy ではなかった。しかし CFR+論文 [7] の主張によると、経験的にではあるが、CFR+の strategy は averaging するまでもなくそれ自体がナッシュ均衡解にほぼまたは完全に収束する。

それに加えて CFR+論文 [7] では、各ラウンド t における $\sigma_i^{T+1}(I,a)$ たちを重み付け平均 (**weighted averaging**) すればさらに速く収束させられると主張していた。重みベクトル w^T はシンプルに $\max(T-d,0)$（ただし d はハイパーパラメータ）を提案していた。後の学会論文 [8] のほうでは $d=0$ に固定する形になっていた。

CFR+論文 [7] が課題としていたのは収束の速さ（イテレーションの数）だけでなく、メモリ消費量の削減もであった。彼らはポーカーソフトの実装上のテクニックに言及していた。すなわち、一般に CFR 系の手法では全ての情報集合 I について諸々の値を保持する必要があるわけだが、実際には大部分の I についてその値はゼロであるから、『可逆圧縮機能付きのハッシュテーブル』みたいなものを用意してゼロの保持を省メモリ化するテクニックがあったらしい。そのうえで、**cumulative counterfactual regret+**は Vanilla CFR の諸々の値よりもスパースになりやすいゆえにメモリ消費量が削減されると主張されていた。

4.5 simultaneous update と alternating update

simultaneous update とは、1 度のイテレーションで両プレイヤーの strategy を更新することである。一方、片方のプレイヤーの strategy だけを更新するのを交互に行う方法も考えられる。これは alternating update と呼ばれる。

Vanilla CFR 論文では暗黙的に simultaneous update を想定したような感じで数式が組み立てられれており、一部の実装者が勝手に alternating update で実装しているという実情だったようだ。（ちなみにこの記事で前述した Algorithm 1は simultaneous update である）

それに対して CFR+論文では alternating update を採用すると明言していた。しかしその筆者らの意図は不明であった。つまり、筆者らは alternating update のほうが収束が速いと知っていたのかもしれないし、ポーカーソフトの実装上のテクニックと alternating update の相性が良いからというだけだったのかもしれない。この件について、ある学位論文 [2] で詳細な実験結果が報告されている。結果としては Vanilla CFR と CFR+の両方において、alternating update を採用するほうが大幅に収束が速くなることが示されていた。

4.6 理論的な解析

この記事の最初のほうで、「真に最小化したい値 X をまず定義して、別の扱いやすい値 Y を導入し、Y が X の上界であることを証明し、Y を最小化するアルゴリズム Z を作る」という戦術があると書いた。

Vanilla CFR 論文 [1] の構成では、X が **average overall regret** R_i^T で、Y が **immediate counterfactual regret** $R_{i,imm}^T(I)$ で、そして Z が Vanilla CFR であった。「Y が X の上界であることの証明」と「Z が Y を最小化することの証明」も Vanilla CFR 論文 [1] に記されていた。

CFR+論文 [7] では、「扱いやすい値」として **cumulative counterfactual regret+** $R_i^{+,T}(I,a)$ が導入されたが、これは変数 $R_i^T(I,a)$ の代わりであって、**immediate counterfactual regret** の代わりではなかった。Z は CFR+だが、何かを最小化するわけでもなく、Y が何かの上界であるとも主張されていなかった。

後の学会論文 [8] では、**weighted averaging** の $d=0$ の場合において、**regret matching+**が Vanilla CFR の **regret matching** と同様にナッシュ均衡解へ収束することの証明がなされていた。

ただしその証明は、収束するという点において Vanilla CFR と同等だと示しているだけだった。CFR+が Vanilla CFR より遥かに「速く」収束するという経験的観測に対する理論的な解析は無かった。

5 CFR+ Algorithm

CFR+論文 [7] の Algorithm 1 は、純粋な CFR+アルゴリズムの説明になっておらず、ポーカーソフトの実装上のテクニックとごちゃまぜになっている。

そこで以下のシュードコードでは、実装上のテクニックにはあまり立ち入らずにシンプルなコードを目指した。Vanilla CFR の流儀に沿いつつ、**cumulative counterfactual regret+**、**weighted averaging**、alternating update の 3 つを導入した。

メモ：CFR+論文 [7] の Algorithm 1 の line 37 の π_{-i} は typo じゃないか？　正しくは π_i だと思う。たぶん。

Algorithm 3 CFR+

Declare cumulative regret tables: $r_I[a]$ for all I and a.
Declare cumulative strategy tables: $s_I[a]$ for all I and a.
Declare strategy profiles: $\sigma^n(I,a)$ for all I and a and $n \in \{1,2,3,...\}$.

function CFR_PLUS_DFS($h, i, t, w, \pi_i, \pi_{-i}$)
if $h \in Z$ **then**
  return $u_i(h)$
else if $P(h) = c$ **then**
  $v \leftarrow 0$
  for $a \in A(h)$ **do**
    $v \leftarrow v + \sigma^t(a|h) \cdot \text{CFR\_PLUS\_DFS}(ha, i, t, w, \pi_i, \sigma^t(a|h) \cdot \pi_{-i})$
  end for
  return v
end if
Let I be the information set s.t. $h \in I$ and $I \in \mathcal{I}_{P(h)}$.
$u_\sigma \leftarrow 0$
$u_{\sigma_{I \to a}}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(h)$ **do**
  if $P(h) = i$ **then**
    $u_{\sigma_{I \to a}}[a] \leftarrow \text{CFR\_PLUS\_DFS}(ha, i, t, w, \sigma^t(a|I) \cdot \pi_i, \pi_{-i})$
  else
    $u_{\sigma_{I \to a}}[a] \leftarrow \text{CFR\_PLUS\_DFS}(ha, i, t, w, \pi_i, \sigma^t(a|I) \cdot \pi_{-i})$
  end if
  $u_\sigma \leftarrow u_\sigma + \sigma^t(a|I) \cdot u_{\sigma_{I \to a}}[a]$
end for
for $a \in A(h)$ **do**
  if $P(h) = i$ **then**
    $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (u_{\sigma_{I \to a}}[a] - u_\sigma)$
  else
    $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(a|I) \cdot w$
  end if
end for
return v_σ

Algorithm 4 CFR+(続き)

Extern: $r_I[a]$ for all I and a.
Extern: $s_I[a]$ for all I and a.
Extern: $\sigma^n(I,a)$ for all I and a and $n \in \{1,2,3,...\}$.

function CFR_PLUS_REGRET_MATCHING(i,t)
for all I $s.t.$ $P(I)=i$ **do**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow \max(r_I[a],0)$
  end for
  $x \leftarrow 0$
  for $a \in A(I)$ **do**
    $x \leftarrow x + r_I[a]$
  end for
  for $a \in A(I)$ **do**
    if $x > 0$ **then**
      $\sigma^{t+1}(I,a) \leftarrow r_I[a]/x$
    else
      $\sigma^{t+1}(I,a) \leftarrow 1/|A(I)|$
    end if
  end for
end for

function CFR_PLUS_GET_PROBABILITY(I,a)
$x \leftarrow 0$
for $a' \in A(I)$ **do**
  $x \leftarrow x + \max(s_I[a'],0)$
end for
if $x > 0$ **then**
  return $\max(s_I[a],0)/x$
end if
return $1/|A(I)|$

function CFR_PLUS_SOLVE(T,d)
for all I **do**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow 0$
    $s_I[a] \leftarrow 0$
    $\sigma^1(I,a) \leftarrow 1/|A(I)|$
  end for
end for
for $t = \{1,2,3,...,T\}$ **do**
  for $i = \{1,2\}$ **do**
    CFR_PLUS_DFS$(\emptyset, i, t, \max(t-d,0), 1, 1)$
    CFR_PLUS_REGRET_MATCHING(i,t)
  end for
end for

5.1 使い方、tips

Vanilla CFR のときと同様である。

CFR_PLUS_SOLVE 関数に十分大きな整数 T と適当なハイパーパラメータ d（$d=0$ で良い）を引数として与えると、CFR+の計算が行われて、グローバル変数 $s_I[a]$ に strategy が入力される。その後、所望の (I,a) について CFR_PLUS_GET_PROBABILITY 関数を呼ぶことで、（ナッシュ均衡解に近いであろう）行動確率が得られる。

6 Monte Carlo CFR (MCCFR) で出てくる用語

Vanilla CFR の原論文 [1] ですでに、ポーカーソフトのテクニックとしてのサンプリングへの言及があった。2009 年の論文 [5] で詳しく研究された。

チャンスプレイヤーのアクションを 1 つだけ sampling する方法 (**chance sampling**) と、それに加えて alternative update においてターゲットではない側（相手 / opponent）のアクションも 1 つだけ sampling する方法 (**external sampling**) と、ターゲット側のアクションも含めた全てのアクションを 1 つだけ sampling する方法 (**outcome sampling**) が提案された。

outcome sampling では、1 回の深さ優先探索で 1 つのヒストリーしか探索されない。深さ優先探索というより、シミュレーションでトラジェクトリを得ると呼ぶべきかもしれない。

2009 年の論文 [5] の著者らもまたポーカーに大いに注目していたが、彼らによるポーカーを含むいくつかのゲームでの実験結果では概ね **external sampling** が最も優れていた。しかしゲームごとに細かな優劣が異なっており、その理論的な解明は課題として残されていた。

Li らの論文 [9] で、いくつかのコンセプトが提案された。

- **robust sampling**: **outcome sampling** において、ターゲット側で 1 アクションだけ sampling するのではなく複数（$\min(k, |A(I)|)$ 個、ただし k はハイパーパラメータ）を sampling する。$k = 1$ において **outcome sampling** に一致し、$k = \infty$ において **external sampling** に一致する。
- **Mini-batch MCCFR**:1 イテレーションごとに 1 回だけ sampling して update するのではなく、複数（b 回、ただし b はハイパーパラメータ）回 sampling した結果を集めて update する。$b = 1$ で MCCFR に一致する。
- **Mini-batch MCCFR+**:彼らは robust sampling に基づく Mini-batch MCCFR の収束性を証明していたが、それに CFR+の regret matching+を加えた Mini-batch MCCFR+についても収束性を証明していた。

6.1 sampling block Q

terminal history $h \in Z$ の集合の集合 $\mathcal{Q}$ を定義する。すなわち、$\mathcal{Q} = \{Q_1, Q_2, ..., Qr\}$ s.t. $Q_i \subseteq Z$ for all $i \in \{1, 2, ..., r\}$ とする。ただし、$\mathcal{Q}$ の要素全ての和集合は Z に等しいとする。$\mathcal{Q}$ の要素 Q_i のことを **sampling block** と呼ぶ。

outcome sampling では $r = |Z|$ であり、常に $|Q_i| = 1$ である。また、一切 sampling しない手法は $\mathcal{Q} = \{Z\}, r = 1, |Q_1| = |Z|$ という特殊ケースとみなすことができる。

6.2 sampling probability

サンプリングによって Q_j が得られる確率を $f_{\mathcal{Q}}(j) > 0$ と表す。確率なので当然に $\sum_{j=1}^{r} f_{\mathcal{Q}}(j) = 1$ が成り立つ。

あるターミナルヒストリー $z \in Z$ に注目する。サンプリング結果に z が含まれている確率 $f_{\mathcal{Q}}(z)$ を考える。$\mathcal{Q}$ 中の複数の Q が z を要素として含みうることから、$f_{\mathcal{Q}}(z) = \sum_{j:z \in Q_j} f_{\mathcal{Q}}(j)$ となる。

一切 sampling しない手法では常に $f_{\mathcal{Q}}(z) = 1$ となる。

6.3 sampled counterfactual value

counterfactual value の代わりに、**sampling block** Q_j に基づく **sampled counterfactual value** $\tilde{v}_i^{\sigma}(I_i|Q_j)$ を以下のように定義する。

$$\tilde{v}_i^{\sigma}(I_i|Q_j) = \sum_{h \in I_i} \sum_{z \in Q_j, h \sqsubseteq z} \frac{1}{f_{\mathcal{Q}}(z)} \pi_{-i}^{\sigma}(h) \pi^{\sigma}(h, z) u_i(z)$$

ちなみに（sampled ではない）**counterfactual value** $v_i(\sigma, I)$ は以下の通りであった。

$$v_i(\sigma, I) = \sum_{h \in I} \sum_{z \in Z, h \sqsubseteq z} \pi_{-i}^{\sigma}(h) \pi^{\sigma}(h, z) u_i(z)$$

sampled counterfactual value の期待値が **counterfactual value** と一致することは論文 [5] の Lemma 1 で証明されていた。

$\tilde{v}_i^\sigma(I_i|Q_j)$ は σ を $\tilde{v}$ の肩に乗せる表記となっており、sampled でないほうの表記 $v_i(\sigma, I)$ とは異なる。この変更には理由がある。一箇所だけ違うストラテジーを使った $\tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j)$ という項が後で登場するのだが、$\sigma^t|_{I\to a}$ を肩に乗せないと、カッコの中に二個の縦棒が入ることになり見栄えが悪くなってしまう。

6.4 External-Sampling MCCFR の特徴

External-Sampling MCCFR では、ターゲットプレイヤー i のアクションは全て探索し、i 以外全員（つまり相手とチャンス）のアクションは strategy σ_{-i} に比例する確率で 1 つだけサンプルする。注目すべきことに、このとき $f_\mathcal{Q}(z) = \pi_{-i}^\sigma(z)$ が成立する。ゆえに、sampled counterfactual value は以下のように書ける。

$$\begin{aligned}\tilde{v}_i^\sigma(I_i|Q_j) &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\frac{1}{f_\mathcal{Q}(z)}\pi_{-i}^\sigma(h)\pi^\sigma(h,z)u_i(z)\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\pi_i^\sigma(h,z)u_i(z)\end{aligned}$$

cumulative counterfactual regret $r^t(I,a) = v_i(\sigma^t|_{I\to a}, I) - v_i(\sigma^t, I)$ に相当するサンプル期待値 $\tilde{r}_i^t(I_i, a|Q_j)$ は

$$\begin{aligned}\tilde{r}_i^t(I_i,a|Q_j) &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \tilde{v}_i^{\sigma^t}(I_i|Q_j)\\ &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\pi_i^{\sigma^t}(h,z)u_i(z)\\ &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\sigma_i^t(a|I_i)\pi_i^{\sigma^t}(ha,z)u_i(z)\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}1\cdot\pi_i^{\sigma^t}(ha,z)u_i(z) - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\sigma_i^t(a|I_i)\pi_i^{\sigma^t}(ha,z)u_i(z)\\ &= (1-\sigma_i^t(a|I_i))\sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\pi_i^{\sigma^t}(ha,z)u_i(z)\end{aligned}$$

とシンプルに書ける。以上の議論は 2009 年の論文 [5] による。

ちなみに、以前の数式で I に下付きの i が付いていないのは、simultaneous update か alternating update か曖昧だったからである。下付きの i が付いていることは、alternating update であることを暗に示している。

6.5 Outcome-Sampling MCCFR の特徴

Outcome-Sampling MCCFR では、ターゲットプレイヤー i のアクションは sampling policy σ_i' に従って 1 つだけサンプルし、i 以外全員（つまり相手とチャンス）のアクションは strategy σ_{-i} に従って 1 つだけサンプルする。このことは $\sigma'_{-i} = \sigma_{-i}$ とも表現できる。

sampling policy σ_i' は σ_i とほぼ同じでかまわないが、全てのヒストリーを $\delta > 0$ の確率でサンプルしうることを保証する必要がある。そのため例えば確率 ϵ（ハイパーパラメータ）で uniformly at random とし、確率 $1-\epsilon$ で σ_i に従うといった policy を用いることが多い。

この方法ではただひとつの z がサンプルされる。よって、まず $f_\mathcal{Q}(z) = \pi_i^{\sigma'}(z)\pi_{-i}^\sigma(z)$ が成立する。

そして、Q_j と z を同一視しつつ、**sampled counterfactual value** $\tilde{v}_i^\sigma(I_i|Q_j)$ は以下の通りに書ける。

$$\begin{aligned}\tilde{v}_i^\sigma(I_i|Q_j) &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\frac{1}{f_\mathcal{Q}(z)}\pi_{-i}^\sigma(h)\pi^\sigma(h,z)u_i(z)\\ &= \sum_{h\in I_i}\frac{1}{\pi_i^{\sigma'}(z)}\pi_i^\sigma(h,z)u_i(z)\\ &= \frac{u_i(z)}{\pi_i^{\sigma'}(z)}\sum_{h\in I_i}\pi_i^\sigma(h,z)\end{aligned}$$

ただし、$\sum_{h \in I_i} \pi_i^{\sigma}(h,z)$ は形式上では総和の形になっているが、実際には要素が 1 つしかない点に注意。

$r^t(I,a) = v_i(\sigma^t|_{I \to a}, I) - v_i(\sigma^t, I)$ に相当するサンプル期待値 $\tilde{r}_i^t(I_i, a|Q_j)$ は、アクション a がサンプルされたか、すなわち $ha \sqsubseteq z$ となる $h \in I_i$ が存在するかどうかで場合分けする必要がある。仮に

$$F(I_i, a|z) = \begin{cases} 1 & \text{if } \exists h \in I_i \, s.t. \, ha \sqsubseteq z \\ 0 & \text{otherwise} \end{cases}$$

と定義する（つまり、F は a がサンプルされていれば 1 を、さもなくば 0 を返す関数である）と、

$$\begin{aligned}
\tilde{r}_i^t(I_i, a|Q_j) &= \tilde{v}_i^{\sigma^t|_{I \to a}}(I_i|Q_j) - \tilde{v}_i^{\sigma^t}(I_i|Q_j) \\
&= \tilde{v}_i^{\sigma^t|_{I \to a}}(I_i|Q_j) - \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \pi_i^{\sigma^t}(h,z) \\
&= \tilde{v}_i^{\sigma^t|_{I \to a}}(I_i|Q_j) - \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \sigma_i^t(a|I_i) \pi_i^{\sigma^t}(ha,z) \\
&= \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} F(I_i, a|z) \pi_i^{\sigma^t}(ha,z) - \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \sigma_i^t(a|I_i) \pi_i^{\sigma^t}(ha,z) \\
&= (F(I_i, a|z) - \sigma_i^t(a|I_i)) \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \pi_i^{\sigma^t}(ha,z)
\end{aligned}$$

と書ける。以上の議論も 2009 年の論文 [5] による。

6.6 Robust Sampling のための sampled strategy profile

Robust Sampling を説明するための準備として、**sampled strategy profile** $\sigma^{rs(k)}$ を定義する。ただしこの k は Robust Sampling のハイパーパラメータである。External Sampling や Outcome Sampling では特に意識されないが、Robust Sampling ではアクション a が選ばれる確率が非自明になるので注目する必要がなる。

$\sigma_i^{rs(k)}$ はプレイヤー i がサンプリングするときの strategy であり、$\sigma_{-i}^{rs(k)}$ はプレイヤー i 以外がサンプリングするときの strategy である。2009 年の論文 [5] に従い、$\sigma_{-i}^{rs(k)} = \sigma_{-i}$ とする。

Robust Sampling では、プレイヤー i は情報集合 I_i において、$\sigma_i^{rs(k)}$ に従って $\min(k, A(I))$ 個のアクションをサンプルする。i 以外のプレイヤーは σ_{-i} に従って 1 個のアクションをサンプルする。$\sigma_i^{rs(k)}(a|I_i)$ は、Robust Sampling において I_i で a が選ばれる確率である。ヒストリー $h \, s.t. \, P(h) = i$ についても同様に $\sigma_i^{rs(k)}(a|h)$ が定義される。

例えば、もしプレイヤー i が uniformly at random に $\min(k, A(I))$ 個サンプルするならば、

$$\sigma_i^{rs(k)}(a|I_i) = \frac{\min(k, |A(I_i)|)}{|A(I_i)|}$$

となる。

sampling によってヒストリー h に到達する確率のうちプレイヤー i の寄与率である $\pi_i^{\sigma^{rs(k)}}(h)$ は以下の通りに定義される。

$$\pi_i^{\sigma^{rs(k)}}(h) = \prod_{h' \sqsubseteq h, h'a \sqsubseteq h, P(h')=i} \sigma_i^{rs(k)}(a|h')$$

論文 [9] では、$k = 1$ や $k = \infty$ と特定の σ を与えることで、Outcome Sampling と External Sampling を Robust Sampling の特殊ケースとして位置づけられることが Lemma 3 と Lemma 4 で示されていた。

6.7 Robust Sampling の特徴

Robust Sampling では、$f_{\mathcal{Q}}(z) = \pi_i^{\sigma^{rs(k)}}(z)\pi_{-i}^{\sigma}(z)$ が成立する。ゆえに、sampled counterfactual value は以下のように書ける。

$$\begin{aligned}\tilde{v}_i^{\sigma}(I_i|Q_j) &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\frac{1}{f_{\mathcal{Q}}(z)}\pi_{-i}^{\sigma}(h)\pi^{\sigma}(h,z)u_i(z)\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\pi_i^{\sigma}(h,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\end{aligned}$$

$r^t(I,a) = v_i(\sigma^t|_{I\to a}, I) - v_i(\sigma^t, I)$ に相当するサンプル期待値 $\tilde{r}_i^t(I_i, a|Q_j)$ は、アクション a がサンプルされたか、すなわち $ha \sqsubseteq z$ となる $h \in I_i$ が存在するかどうかで場合分けする必要がある。上述の場合分け関数 F を以下のように修正して用いることにすると、以下の通りに書ける。

$$F(I_i, a|Q_j) = \begin{cases}1 & \text{if } \exists h \in I_i, \exists z \in Q_j \; s.t. \; ha \sqsubseteq z\\ 0 & \text{otherwise}\end{cases}$$

$$\begin{aligned}\tilde{r}_i^t(I_i, a|Q_j) &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \tilde{v}_i^{\sigma^t}(I_i|Q_j)\\ &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\pi_i^{\sigma}(h,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\\ &= \tilde{v}_i^{\sigma^t|_{I\to a}}(I_i|Q_j) - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\sigma_i^t(a|I_i)\pi_i^{\sigma}(ha,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}F(I_i, a|Q_j)\pi_i^{\sigma}(ha,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)} - \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}\sigma_i^t(a|I_i)\pi_i^{\sigma}(ha,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}(F(I_i, a|Q_j) - \sigma_i^t(a|I_i))\pi_i^{\sigma}(ha,z)\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\\ &= \sum_{h\in I_i}\sum_{z\in Q_j, h\sqsubseteq z}(F(I_i, a|Q_j) - \sigma_i^t(a|I_i))\frac{\pi_i^{\sigma}(z)}{\pi_i^{\sigma}(h)\sigma_i^t(a|I_i)}\frac{u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\\ &= \sum_{h\in I_i}\frac{(F(I_i, a|Q_j) - \sigma_i^t(a|I_i))}{\pi_i^{\sigma}(h)\sigma_i^t(a|I_i)}\sum_{z\in Q_j, h\sqsubseteq z}\frac{\pi_i^{\sigma}(z)u_i(z)}{\pi_i^{\sigma^{rs(k)}}(z)}\end{aligned}$$

メモ : 論文 [9] の Algorithm 3 の MCCFR-NN 関数のシュードコードでは $F = 0$ の場合の更新をしていないが、あれはバグだと思う。Robust Sampling の特殊ケースとして Outcome Sampling に一致することの証明が Appendix に載っているが、そこでは $F = 0$ の場合も更新するように書いてあり、シュードコードと矛盾している。

6.8 Mini-batch Technique

b 個の Q をサンプルしてミニバッチで更新することを考える。**mini-batch counterfactual value** $\tilde{v}_i^{\sigma}(I_i|b)$ を以下の通りに定義する。

$$\tilde{v}_i^{\sigma}(I_i|b) = \frac{1}{b}\sum_{j=1}^{b}\tilde{v}_i^{\sigma}(I_i|Q_j)$$

論文 [9] ではこれが unbiased であることの証明がなされている。

また、$\tilde{r}_i^t(I_i, a|Q_j)$ のミニバッチ版も以下のように書ける。

$$\tilde{r}_i^t(I_i, a|b) = \sum_{j=1}^{b} \tilde{r}_i^t(I_i, a|Q_j)$$

そのうえで、**cumulative mini-batch regret** $\tilde{R}_i^T((a|I_i)|b)$ は以下のように書ける。

$$\tilde{R}_i^T((a|I_i)|b) = \tilde{R}_i^{T-1}((a|I_i)|b) + \tilde{r}_i^t(I_i, a|b)$$

ただし $\tilde{R}_i^0((a|I_i)|b) = 0$ とする。実装上、このミニバッチたちは並列にサンプルできるので実装の高速化に寄与する。

6.9 Mini-batch MCCFR+

CFR+論文 [7] では regret-matching+という手法が提案された。これは Vanilla CFR などの refret-matching よりも速く収束すると主張されていた。ある学位論文 [2] は、regret-matching+は mini-batch と同時に用いないとむしろ収束が遅くなることを示した。Li ら [9] によると、適切な mini-batch size のもとで regret-matching+は実際に regret-matching よりも早く収束する。

cumulative mini-batch regret+ $\tilde{R}^{+,T}$ は以下の通りに定義される。

$$\tilde{R}^{+,T} = \begin{cases} \max(\tilde{r}_i^T(I_i, a|b), 0) & \text{if } T = 0 \\ \max(\tilde{R}^{+,T-1} + \tilde{r}_i^T(I_i, a|b), 0) & \text{if } T > 0 \end{cases}$$

7 Monte Carlo CFR Algorithm

Algorithm 5 MCCFR+ (Robust Sampling, Mini-Batch)

Declare cumulative regret tables: $r_I[a]$ for all I and a.
Declare cumulative strategy tables: $s_I[a]$ for all I and a.
Declare strategy profiles: $\sigma^n(I, a)$ for all I and a and $n \in \{1, 2, 3, ...\}$.

function MCCFR_PLUS_SAMPLE(I_i, i, t, k, ϵ)
for $a \in A(h)$ **do**
    $\sigma_i'(a|I_i) \leftarrow \epsilon \frac{1}{|A(h)|} + (1-\epsilon)\sigma_i^t(a|I_i)$
end for
$A^{rs(k)}(I_i) \leftarrow \emptyset$
while $|A^{rs(k)}(I_i)| < k$ **do**
    Sample single action $a \sim \sigma_i'(a|I_i)$.
    $A^{rs(k)}(I_i) \leftarrow A^{rs(k)}(I_i) \cup \{a\}$
end while
Calculate $\sigma_i^{rs(k)}(\cdot|I_i)$ using Dynamic Programming or something.
return $(\sigma_i^{rs(k)}(\cdot|I_i), A^{rs(k)}(I_i))$

function MCCFR_PLUS_DFS($h, i, t, k, \epsilon, w, \pi_i, \pi_i^{rs(k)}$)
if $h \in Z$ **then**
    return $\frac{\pi_i u_i(h)}{\pi_i^{rs(k)}}$
else if $P(h) \neq i$ **then**
    sample single action $a \sim \sigma^t(a|h)$
    return MCCFR_PLUS_DFS($ha, i, t, k, \epsilon, w, \pi_i, \pi_i^{rs(k)}$)
end if
Let I_i be the information set s.t. $h \in I_i$ and $I_i \in \mathcal{I}_i$.
$u_\sigma \leftarrow 0$
$(\sigma_i^{rs(k)}(\cdot|I_i), A^{rs(k)}(I_i)) \leftarrow$ MCCFR_PLUS_SAMPLE(I_i, i, t, k, ϵ)
for $a \in A^{rs(k)}(I_i)$ **do**
    $u_\sigma \leftarrow u_\sigma +$ MCCFR_PLUS_DFS($ha, i, t, k, \epsilon, w, \sigma^t(a|I_i) \cdot \pi_i, \sigma_i^{rs(k)}(a|I_i) \cdot \pi_i^{rs(k)}$)
end for
for $a \in A(h)$ **do**
    if $a \in A^{rs(k)}(I_i)$ **then**
        $r_{I_i}[a] \leftarrow r_{I_i}[a] + \frac{(1-\sigma^t(a|I_i))u_\sigma}{\pi_i \sigma^t(a|I_i)}$
    else
        $r_{I_i}[a] \leftarrow r_{I_i}[a] - \frac{u_\sigma}{\pi_i}$
    end if
    $s_{I_i}[a] \leftarrow s_{I_i}[a] + \pi_i \sigma^t(a|I_i) w$
end for
return v_σ

Algorithm 6 MCCFR+ (Robust Sampling, Mini-Batch)(続き)

Extern: $r_I[a]$ for all I and a.
Extern: $s_I[a]$ for all I and a.
Extern: $\sigma^n(I,a)$ for all I and a and $n \in \{1,2,3,...\}$.

function MCCFR_PLUS_REGRET_MATCHING(i,t)
for all I $s.t.$ $P(I)=i$ **do**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow \max(r_I[a],0)$
  end for
  $x \leftarrow 0$
  for $a \in A(I)$ **do**
    $x \leftarrow x + r_I[a]$
  end for
  for $a \in A(I)$ **do**
    if $x > 0$ **then**
      $\sigma^{t+1}(I,a) \leftarrow r_I[a]/x$
    else
      $\sigma^{t+1}(I,a) \leftarrow 1/|A(I)|$
    end if
  end for
end for

function MCCFR_PLUS_GET_PROBABILITY(I,a)
$x \leftarrow 0$
for $a' \in A(I)$ **do**
  $x \leftarrow x + \max(s_I[a'],0)$
end for
if $x > 0$ **then**
  return $\max(s_I[a],0)/x$
end if
return $1/|A(I)|$

function MCCFR_PLUS_SOLVE(T,d,b,k,ϵ)
for all I **do**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow 0$
    $s_I[a] \leftarrow 0$
    $\sigma^1(I,a) \leftarrow 1/|A(I)|$
  end for
end for
for $t = \{1,2,3,...,T\}$ **do**
  for $j = \{1,2,3,...,b\}$ in parallel **do**
    for $i = \{1,2\}$ **do**
      MCCFR_PLUS_DFS$(\emptyset,i,t,k,\epsilon,\max(t-d,0),1,1)$
    end for
  end for
  MCCFR_PLUS_REGRET_MATCHING(i,t)
end for

7.1 使い方、tips

以前のと同じ。j のループは並列実行できるが、このシュードコードでは $r_I[a]$ と $s_I[a]$ がグローバル変数になっているので、排他制御とかアトミック操作とかでデータ競合を防ぐ必要がある。（単にスレッドごとにメモリ確保しておいて後で総和するのでもいい。MCCFR はゲーム木全体をパースするわけではないので不可能な案ではない）

8 Online Outcome Sampling CFR (OOS CFR)

論文は 2015 年に publish されている。[10]

「Online」と呼ばれる問題設定では、プレイヤーは主に対局中に計算して現在の情報集合の strategy を求めなければならない。何らかの事前計算が許されるかは個々の論文の問題設定による。限られた計算時間とメモリでナッシュ均衡解に近い strategy を求めたい。自プレイヤー i から見た現在の情報集合 I_i は既知だが、真のヒストリー $h \in I_i$ は不明である。

最大の問題は、論文 [10] の中で **The Problem of Non-Locality** と呼ばれている。これは、求解のためのシミュレーションを I_i からスタートさせても unbiased な解は得られないというものである。なぜなら、真のヒストリーがどれであるかの確率は過去の履歴によって決まるが、相手の過去の行動はその時点での相手方の情報集合に由来しており、その strategy はその時点での未来予測に基づいているからである。

ゆえに、Online で unbiased な解を得たいとしても、シミュレーションは開始状態から始めなければいけない。しかも、Online であるからにはただの MCCFR はよろしくなくて、現在の情報集合 I_i を通るサンプルが多めに得られるように MCCFR を改変するべきである。そこで OOS CFR が提案された。

3 種類のゲームで実験して、先行研究との性能比較などしていた。しかしそのどれも、$|A(\emptyset)|$ が組合せ論的に莫大な値になることが決してないゲームであった。これがあると OOC CFR は上手く働かないと考えられる（私の予想にすぎないが）。そう考えたため、詳細の説明は省略する。

こいこいやブリッジではゲームの最初に多数のカードが山札からプレイヤーたちに配られる。すなわち $P(\emptyset) = c$ であり、$|A(\emptyset)|$ は組合せ論的に莫大な値になる。

9 Double Neural Counterfactual Regret Minimization

Double Neural Counterfactual Regret Minimization という論文 [9] は 2018 年に arXiv に掲載され、ICLR2020 で採択されている。テーブル引きだった regret と strategy をディープラーニングの関数近似で置き換えるものである。（この論文でも当然いくつかに触れられているが）同様のコンセプトの「Neural CFR 系手法」みたいなやつは多数ある。そのなかでこの論文は（前の章で説明した通り）MCCFR 自体への改善手法をいくつか提案していて実験も多いことから評価されている。ただし、実験のソースコードも計算機環境も公開されていない。

アルゴリズムを端的に説明すると、前述の MCCFR_PLUS_SOLVE 関数の t ループの最後で、2 つのニューラルネット：**RegretSumNetwork(RSN)** と **AvgStrategyNetwork(ASN)** を学習する。これら次のイテレーションにおける $r_I[a]$ と $s_I[a]$ に相当する機能を果たす。

ネットワークアーキテクチャは Attention 機構付き (LSTM | GRU | RNN) と Fully Connected を実験していて、Attention 付きのほうが slightly better だったらしい。

10 Discussion：こいこいを解くにはどうすればいいのか

10.1 最初の手を特別扱いしないといけないのでは？

Outcome Sampling における sampled counterfactual value は以下の通りであった。

$$
\begin{aligned}
\tilde{v}_i^{\sigma}(I_i|Q_j) &= \sum_{h \in I_i} \sum_{z \in Q_j, h \sqsubseteq z} \frac{1}{f_{\mathcal{Q}}(z)} \pi_{-i}^{\sigma}(h) \pi^{\sigma}(h,z) u_i(z) \\
&= \sum_{h \in I_i} \frac{1}{\pi_i^{\sigma'}(z)} \pi_i^{\sigma}(h,z) u_i(z) \\
&= \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \pi_i^{\sigma}(h,z)
\end{aligned}
$$

ここで、$P(\emptyset) = c$ であり、$|A(\emptyset)|$ は組合せ論的に莫大な値になるケースについて考える。これは、こいこいやブリッジなど、最初に多数のカードを配るゲームに当てはまる想定である。このアクションを特別扱いするために、以下のように式変形する。

$$\tilde{v}_i^{\sigma}(I_i|Q_j) = \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i} \pi_i^{\sigma}(h,z)$$
$$= \frac{u_i(z)}{\pi_i^{\sigma'}(z)} \sum_{h \in I_i, h \neq \emptyset} \pi_i^{\sigma}(h,z) \sum_{a \in A(\emptyset), a \sqsubseteq h} \frac{f_c(a|\emptyset)}{\sigma_c'(a|\emptyset)}$$

具体的にどのように特別扱いすれば Online でうまく扱えるようになるのかは今後の課題である。

10.2 abstraction, またはその自動化としてのディープラーニング

ポーカーソフトでは、ゲーム木のノード数を減らすために abstraction と呼ばれるテクニックが広く使われる [3, 4]。例えば掛け金を数パターンに限定するとか（アクションを減らす）、「近い」情報集合を同一視するとか、perfect recall 仮定をやめるとか、いろいろある。

しかしながら、abstraction のやりかたは各ゲームの性質に応じて個別に設計する必要があるうえ、こいこいのように役が複雑なゲームでは人力での設計は困難だと思われる。

regret や strategy をテーブル引きから関数近似に置き換えることは、ある意味で abstraction を自動化していると見ることができる [3]。Neural CFR なら複雑なゲームの abstraction を上手く見つけられるかもしれない。可変長の情報集合を入力するには Attention 付き LSTM とか Transformer とかを使うのが良いと考えられる。ここでまず問題になるのは、それらの学習は現状かなり重い計算になることである。また、既存の Neural CFR の枠組みでは、CFR の反復ステップごとにニューラルネットの学習をやり直す形になっている。具体的な反復ステップ数は 1000 回とかである。さらには、CFR のサンプリングの際に、regret や strategy をテーブル引きする代わりにニューラルネットの推論を行う形にもなっている。

11 Conclusion

この記事では CFR、CFR+、MCCFR を詳しく説明した。加えて、Online sampling と Neural CFR 系の手法にも触れた。

強いこいこいの AI を既存手法ベースで作るとすると、最も有望なのは Neural CFR 系の手法だと考えられる。しかし、既存研究が実験に用いてきたポーカー等のゲームに比べて、こいこいの性質はいくつかの点で非常に異なるため、上手く行かない可能性は十分にある。とはいえディープラーニングには普遍近似定理がある。ゆえにおそらく、任意の ϵ-ナッシュ均衡に対してあるミニバッチ数 b とパラメータ数 n が存在して、n 以上の大きなネットワークを b 以上のデータ量で学習すれば ϵ を達成できるだろう。

しかし、そのような方法論が安価な計算機環境で実現可能かどうかは別の問題である。こいこいはポーカーと違いプロシーンが無いので、人間のチャンピオンに勝つことはできない。日本国外での知名度が低いため、強いこいこい AI ができても偉業とはみなされないかもしれない。そのため大量の計算機を調達するのも難しいだろう。Attention 付き LSTM や Transformer モデルなどの大きなやつの学習と推論を民主化可能な程度まで高速化することこそが、強いこいこい AI を作るための目の前にある課題なのかもしれない。

References

[1] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

[2] Neil Burch. Time and space: Why imperfect information games are hard. 2018.

[3] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[4] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[5] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

[6] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[7] Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

[8] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[9] Hui Li, Kailiang Hu, Zhibang Ge, Tao Jiang, Yuan Qi, and Le Song. Double neural counterfactual regret minimization. *arXiv preprint arXiv:1812.10607*, 2018.

[10] Viliam Lisỳ, Marc Lanctot, and Michael Bowling. Online monte carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 27–36, 2015.