

poker and game theory algorithm

yu5uke

2022 年 7 月 16 日

目次

1	アルゴリズムの疑似コード	2
1.1	Chance sampling Counterfactual Regret Minimization	2
1.2	Vanilla Counterfactual Regret Minimization	3
1.3	External Sampling Monte Carlo Counterfactual Regret Minimization	4
1.4	Outcome Sampling Monte Carlo Counterfactual Regret Minimization	5
1.5	General Fictitious Self-Play	6
1.6	Batch Fictitious Self-Play	7
1.7	Table-lookup Batch FSP with Q-learning and counting model	8
1.8	Neural Fictitious Self-Play	9
1.9	Neural Fictitious Self-Play with PPO	10
2	引用文献	11

1 アルゴリズムの擬似コード

1.1 Chance sampling Counterfactual Regret Minimization

Algorithm 1 Chance sampling Counterfactual Regret Minimization ¹

```
1: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$ 
2: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$ 
3: ChanceSamplingCFR( $h, i, t, \pi_1, \pi_2$ ):
4:   if  $h$  is terminal then return  $u_i(h)$ 
5:   else if  $h$  is a chance node then
6:     Sample a single outcome  $a \sim \sigma_c(h, a)$ 
7:     return ChanceSamplingCFR( $ha, i, t, \pi_1, \pi_2$ )
8:   end if
9: Let  $I$  be the information set containing  $h$ 
10:  $\sigma^t(I) \leftarrow \text{RegretMatching}(r_I)$ 
11:  $v_\sigma \leftarrow 0$ 
12:  $v_{I \rightarrow a}[a] \leftarrow 0$  for all  $a \in A(I)$ 
13: for  $a \in A(I)$  do
14:   if  $P(h) = 1$  then
15:      $v_{I \rightarrow a}[a] \leftarrow \text{ChanceSamplingCFR}(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$ 
16:   else if  $P(h) = 2$  then
17:      $v_{I \rightarrow a}[a] \leftarrow \text{ChanceSamplingCFR}(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$ 
18:   end if
19:    $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{I \rightarrow a}[a]$ 
20: end for
21: if  $P(h) = i$  then
22:   for  $a \in A(I)$  do
23:      $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{I \rightarrow a}[a] - v_\sigma)$ 
24:      $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$ 
25:   end for
26: end if return  $v_\sigma$ 
```

1.2 Vanilla Counterfactual Regret Minimization

Algorithm 2 Vanilla Counterfactual Regret Minimization ¹

```

1: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$ 
2: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$ 
3: VanillaCFR( $h, i, t, \pi_1, \pi_2$ ):
4:   if  $h$  is terminal then return  $u_i(h)$ 
5:   else if  $h$  is a chance node then
6:     Sample a single outcome  $a \sim \sigma_c(h, a)$ 
7:     return  $\sum_{a \in A(h)} \sigma_c(h, a) \text{ VanillaCFR}(ha, i, t, \pi_1, \pi_2)$ 
8:   end if
9: Let  $I$  be the information set containing  $h$ 
10:  $\sigma^t(I) \leftarrow \text{RegretMatching}(r_I)$ 
11:  $v_\sigma \leftarrow 0$ 
12:  $v_{I \rightarrow a}[a] \leftarrow 0$  for all  $a \in A(I)$ 
13: for  $a \in A(I)$  do
14:   if  $P(h) = 1$  then
15:      $v_{I \rightarrow a}[a] \leftarrow \text{VanillaCFR}(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$ 
16:   else if  $P(h) = 2$  then
17:      $v_{I \rightarrow a}[a] \leftarrow \text{VanillaCFR}(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$ 
18:   end if
19:  $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{I \rightarrow a}[a]$ 
20: end for
21: if  $P(h) = i$  then
22:   for  $a \in A(I)$  do
23:      $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{I \rightarrow a}[a] - v_\sigma)$ 
24:      $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$ 
25:   end for
26: end if return  $v_\sigma$ 

```

1.3 External Sampling Monte Carlo Counterfactual Regret Minimization

Algorithm 3 External Sampling Monte Carlo Counterfactual Regret Minimization ²

```

1: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$ 
2: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$ 
3: ExternalSamplingMCCFR( $h, i$ ):
4: if  $h$  is terminal then return  $u_i(h)$ 
5: else if  $h$  is a chance node then
6:   Sample a single outcome  $a \sim \sigma_c(h, a)$ 
   return ExternalSamplingMCCFR( $ha, i$ )
7: end if
8: Let  $I$  be the information set containing  $h$ 
9:  $\sigma^t(I) \leftarrow \text{RegretMatching}$ 
10: if  $P(h) = i$  then
11:   Let  $u$  be an array indexed by actions and  $u_\sigma \leftarrow 0$ 
12:   for  $a \in A(I)$  do
13:      $u[a] \leftarrow \text{ExternalSamplingMCCFR}(ha, i)$ 
14:      $u_\sigma \leftarrow u_\sigma + \sigma(I, a) \cdot u[a]$ 
15:   end for
16:   for  $a \in A(I)$  do
17:      $\tilde{r}(I, a) \leftarrow u[a] - u_\sigma$ 
18:      $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$ 
19:   end for return  $u_\sigma$ 
20: else
21:   Sample a single outcome  $a \sim \sigma(I)$ 
22:    $u \leftarrow \text{ExternalSamplingMCCFR}(ha, i)$ 
23:   for  $a \in A(I)$  do
24:      $s_I[a] \leftarrow s_I[a] + \sigma^t(I, a)$ 
25:   end for return  $u$ 
26: end if

```

1.4 Outcome Sampling Monte Carlo Counterfactual Regret Minimization

Algorithm 4 Outcome Sampling Monte Carlo Counterfactual Regret Minimization ²

```

1: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$ 
2: Initialize:  $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$ 
3: Initialize:  $\forall I \in \mathcal{I} : c_I \leftarrow 0$ 
4: OutcomeSamplingMCCFR( $h, i, t, \pi_i, \pi_{-i}, s$ ):
5: if  $h$  is terminal then return  $u_i(h)/s, 1$ 
6: else if  $h$  is a chance node then
7:   Sample a single outcome  $a \sim \sigma_c(h, a)$ 
8:   return OutcomeSamplingMCCFR( $ha, i, t, \pi_i, \pi_{-i}, s$ )
9: end if
10: Let  $I$  be the information set containing  $h$ 
11:  $\sigma^t(I) \leftarrow \text{RegretMatching}$ 
12: Let  $\sigma'(I)$  be a sampling distribution at  $I$ 
13: if  $P(h) = i$  then
14:    $\sigma'(I) \leftarrow \epsilon \cdot \text{UnIf}(I) + (1 - \epsilon)\sigma(I)$ 
15: else
16:    $\sigma'(I) \leftarrow \sigma(I)$ 
17: end if
18: Sample an action  $a'$  with probability  $\sigma'(I, a)$ 
19: if  $P(h) = i$  then
20:    $(u, \pi_{tail}) \leftarrow \text{OutcomeSamplingMCCFR}(ha', i, t, \pi_i \cdot \sigma(I, a), \pi_{-i}, s \cdot \sigma'(I, a))$ 
21:   for  $a \in A(I)$  do
22:      $W \leftarrow u \cdot \pi_{-i}$ 
23:     compute  $\tilde{r}(I, a)$ 
24:      $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$ 
25:   end for
26: else
27:    $(u, \pi_{tail}) \leftarrow \text{OutcomeSamplingMCCFR}(ha', i, t, \pi_i, \pi_{-i} \cdot \sigma(I, a), s \cdot \sigma'(I, a))$ 
28:   for  $a \in A(I)$  do
29:      $s_I[a] \leftarrow s_I[a] + (t - c_I) \cdot \pi_{-i} \cdot \sigma(I, a)$ 
30:   end for
31:    $c_I \leftarrow t$ 
32: end if return  $(u, \pi_{tail} \cdot \sigma(I, a))$ 

```

1.5 General Fictitious Self-Play

Algorithm 5 General Fictitious Self-Play ³

```

1: function FICTITIOUSSELFPLAY( $\Gamma, n, m$ )
2:   Initialize: completely mixed  $\pi_1$ 
3:    $\beta_2 \leftarrow \pi_1$ 
4:    $j \leftarrow 2$ 
5:   while within computational budget do
6:      $\eta_j \leftarrow \text{MIXINGPARAMETER}(j)$ 
7:      $\mathcal{D} \leftarrow \text{GENERATEDATA}(\pi_{j-1}, \beta, n, m, \eta_j)$ 
8:     for each player  $i \in \mathcal{N}$  do
9:        $\mathcal{M}_{RL}^i \leftarrow \text{UPDATERLMEMORY}(\mathcal{M}_{RL}^i, \mathcal{D}^i)$ 
10:       $\mathcal{M}_{SL}^i \leftarrow \text{UPDATESLMEMORY}(\mathcal{M}_{SL}^i, \mathcal{D}^i)$ 
11:       $\beta_{j+1}^i \leftarrow \text{REINFORCEMENTLEARNING}(\mathcal{M}_{RL}^i)$ 
12:       $\pi_j^i \leftarrow \text{SUPERVISEDLEARNING}(\mathcal{M}_{SL}^i)$ 
13:    end for
14:     $j \leftarrow j + 1$ 
15:  end while
16:  return  $\pi_{j-1}$ 
17: end function
18: function GENERATEDATA( $\pi, \beta, n, m, \eta$ )
19:    $\sigma \leftarrow (1 - \eta)\pi + \eta\beta$ 
20:    $\mathcal{D} \leftarrow n$  episode  $\{t_k\}_{1 \leq k \leq n}$  sampled from strategy profile  $\sigma$ 
21:   for each player  $i \in \mathcal{N}$  do
22:      $\mathcal{D}^i \leftarrow m$  episode  $\{t_k^i\}_{1 \leq k \leq n}$  sampled from strategy profile  $(\beta^i, \sigma^{-i})$ 
23:      $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \mathcal{D}$ 
24:   end for
25:   return  $\{D_k\}_{1 \leq k \leq N}$ 
26: end function

```

1.6 Batch Fictitious Self-Play

Algorithm 6 Batch Fictitious Self-Play ⁴

```

1: Initialize: completely mixed average strategy profile  $\pi_1$ 
2: Initialize: replay memories  $\mathcal{M}_{RL}, \mathcal{M}_{SL}$ 
3: Initialize: the constant numbers of episode that are sampled simulataneously,  $n$ , and
   alternatingly,  $m$ 
4: for  $k = 1, \dots, K - 1$  do
   SAMPLESIMULTANEOUSEXPERIENCE ( $n, \pi_k, \mathcal{M}_{RL}$ )
5:   Each player  $i$  updates their approximate best response strategy
6:    $\beta_{k+1}^i \leftarrow \text{REINFORCEMENTLEARNING}(\mathcal{M}_{RL}^i)$ 
7:   SAMPLEALTERNATINGEXPERIENCE ( $m, \pi_k, \beta_{k+1}, \mathcal{M}_{RL}, \mathcal{M}_{SL}$ )
8:   Each player  $i$  updates their average strategy
9:    $\pi_{k+1}^i \leftarrow \text{SUPERVISEDLEARNING}(\mathcal{M}_{SL}^i)$ 
10: end for
11: return  $\pi_k$ 
12:
13: function SAMPLESIMULTANEOUSEXPERIENCE( $n, \pi_k, \mathcal{M}_{RL}$ )
14:   Sample  $n$  episodes from strategy profile  $\pi$ 
15:   For each player  $i$  store their experienced transitions,  $(u_t^i, a_t, r_{t+1}, u_{t+1}^i)$  in their rein-
       forcement learning memory  $\mathcal{M}_{RL}^i$ 
16: end function
17:
18: function SAMPLEALTERNATINGEXPERIENCE( $m, \pi_k, \beta_{k+1}, \mathcal{M}_{RL}, \mathcal{M}_{SL}$ )
19:   for each player  $i \in \mathcal{N}$  do
20:     Sample  $m$  episodes from strategy profile  $(\beta^i, \pi^{-i})$ 
21:     Store player  $i$ 's experienced transitions,  $(u_t^i, a_t, r_{t+1}, u_{t+1}^i)$  in their reinforcement
       learning memory  $\mathcal{M}_{RL}^i$ 
22:     Store player  $i$ 's experienced own behaviour,  $(u_t^i, a_t)$  or  $(u_t^i, \beta^i(u_t^i))$  in their super-
       vised learning memory  $\mathcal{M}_{SL}^i$ 
23:   end for
24: end function

```

1.7 Table-lookup Batch FSP with Q-learning and counting model

Algorithm 7 Table-lookup Batch FSP with Q-learning and counting model ⁴

```

1: Initialize Q-learning parameters, e.g. learning stepsize
2: Initialize all agents' table-lookup action-values,  $\{Q^i\}_{1 \in N}$ 
3: Initialize all agents' table-lookup counting models,  $\{N^i\}_{1 \in N}$ 
4: Initialize and run Algorithm 6 (Batch FSP) with REINFORCEMENTLEARNING and SUPER-
   VISEDLEARNING methods below
5:
6: function REINFORCEMENTLEARNING( $\mathcal{M}_{RL}^i$ )
7:   Restore previous iteration's  $Q^i$ -values
8:   Update (decay) learning stepsize and Boltzmann temperature
9:   updated  $Q^i$ -values with Q-learning from  $\mathcal{M}_{RL}^i$ 
   return Boltzmann[ $Q^i$ ]
10: end function
11:
12: function SUPERVISEDLEARNING( $\mathcal{M}_{SL}^i$ )
13:   Restore previous iteration's counting model  $N^i$ , and average strategy,  $\pi^i$ 
14:   for each  $(u_t, \rho_t)$  in  $\mathcal{M}_{SL}^i$  do
15:      $\forall a \in \mathcal{A}(u_t) : N^i(u_t, a) \leftarrow N^i(u_t, a) + \rho_t(a)$ 
16:      $\forall a \in \mathcal{A}(u_t) : \pi^i(u_t, a) \leftarrow N^i(u_t, a) / N^i(u_t)$ 
17:   end for
18:   Empty  $\mathcal{M}_{SL}^i$ 
   return  $\pi^i$ 
19: end function

```

1.8 Neural Fictitious Self-Play

Algorithm 8 Neural Fictitious Self-Play ⁵

```

1: Initialize game  $\Gamma$  and excute an agent via RUNAGENT for each player in the game
2:
3: function RUNAGENT( $\Gamma$ )
4:   Initialize replay memories  $\mathcal{M}_{RL}$  (circular buffer) and  $\mathcal{M}_{SL}$  (reservoir)
5:   Initialize average-policy network  $\Pi(s, a|\theta^\Pi)$  with random parameters  $\theta^\Pi$ 
6:   Initialize action-value network  $Q(s, a|\theta^Q)$  with random parameters  $\theta^Q$ 
7:   Initialize target network parameters  $\theta^{Q'} \leftarrow \theta^Q$ 
8:   Initialize anticipatory parameter  $\eta$ 
9:   for each episode do
10:     Set policy  $\sigma \leftarrow \begin{cases} \epsilon - greedy(Q), \text{ with probability } \eta \\ \Pi, \text{ with probability } 1 - \eta \end{cases}$ 
11:     for  $t = 1, \dots, T$  do
12:       Observe information state  $s_t$  and sample action  $a_t$  from  $\sigma$ 
13:       Execute action  $a_t$  in game and observe reward  $r_{t+1}$  and next information state
          $s_{t+1}$ 
14:       Store transitions  $(s_t, a_t, r_{t+1}, s_{t+1})$  in reinforcement learning memory  $M_{RL}$ 
15:       if agent follows best response policy  $\sigma = \epsilon - greedy(Q)$  then
16:         Store behavior tuple  $(s_t, a_t)$  in supervised learning memory  $M_{SL}$ 
17:       end if
18:       Update  $\theta_\pi$  with stochastic gradient descent on loss
19:          $\mathcal{L}(Q^\pi) = \mathbb{E}_{(s,a) \sim M_{RL}} [-\log \Pi(s, a|\theta^\Pi)]$ 
20:       Update  $\theta_Q$  with stochastic gradient descent on loss
21:          $\mathcal{L}(Q^\pi) = \mathbb{E}_{(s,a,r,s_{prime}) \sim M_{RL}} [(r + \max_{a'} Q(s', a'|\theta^{Q'}) - Q(s, a|\theta^Q))^2]$ 
22:       Periodically update target network parameters  $\theta^{Q'} \leftarrow \theta^Q$ 
23:     end for
24:   end for
25: end function

```

1.9 Neural Fictitious Self-Play with PPO

Algorithm 9 Neural Fictitious Self-Play with PPO ⁵

```

1: Initialize game  $\Gamma$  and excute function PPO
2: Initialize number of parallel processing  $N$ 
3: Initialize number of players  $NP$ 
4: Initialize number of transitions  $T$ 
5:
6: function PPO( $\Gamma$ )
7:   Initialize replay memories  $\mathcal{M}_{RL}$  (circular buffer) and  $\mathcal{M}_{SL}$  (reservoir)
8:   Initialize average-policy network  $\Pi(s, a|\theta^\Pi)$  with random parameters  $\theta^\Pi$ 
9:   Initialize action-value network  $\beta(s, a|\theta^\beta)$  with random parameters  $\theta^\beta$ 
10:  for each iteration do
11:    for actor = 1, 2, ...,  $N$  do
12:      target-player = (actor - 1) % NP
13:      Set policy  $\sigma \leftarrow \begin{cases} \beta_{\theta_{old}} \text{ strategy} & \text{if } P(s) = \text{target-player for } s \text{ in } \sigma \\ \Pi \text{ strategy} & \text{if } P(s) \neq \text{target-player for } s \text{ in } \sigma \end{cases}$ 
14:      save-step = 0
15:      while save-step <  $T$  do
16:        Observe information state  $s_t$  and sample action  $a_t$  from  $\sigma$ 
17:        Execute action  $a_t$  in game and observe reward  $r_{t+1}$ 
18:        if  $P(s_t) = \text{target actor}$  then
19:          Store transitions  $(s_t, a_t, pd_t, r_{t+1})$  in reinforcement learning memory  $M_{RL}$ 
20:          Store behavior tuple  $(s_t, a_t)$  in supervised learning memory  $M_{SL}$ 
21:          save-step + = 1
22:        end if
23:      end while
24:      Compute adavantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
25:    end for
26:    Optimize surrogate  $L_{RL}$  wrt  $\theta^\beta$  in  $M_{RL}$ 
27:     $\beta_{\theta_{old}} \leftarrow \beta_\theta$ 
28:    Optimize  $L_{SL}$  wrt  $\theta^\pi$  with cross entropy in  $M_{SL}$ 
29:     $\Pi_{\theta_{old}} \leftarrow \Pi_\theta$ 
30:    empty  $M_{SL}$  and  $M_{RL}$ 
31:  end for
32: end function

```

2 引用文献

- [1] <http://modelai.gettysburg.edu/2013/cfr/cfr.pdf>
- [2] <https://proceedings.neurips.cc/paper/2009/file/00411460f7c92d2124a67ea0f4cb5f85-Paper.pdf>
- [3] <http://proceedings.mlr.press/v37/heinrich15.pdf>
- [4] https://discovery.ucl.ac.uk/id/eprint/1549658/1/Heinrich_phd_FINAL.pdf
- [5] <https://arxiv.org/pdf/1603.01121.pdf>