poker and game theory algorithm

yu5uke

2022 年 7 月 26 日

目次

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]

Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$
Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$
ChanceSamplingCFR(h, i, t, π_1, π_2):
if h is terminal **then return** $u_i(h)$
else if h is a chance node **then**
    Sample a single outocme $a \sim \sigma_c(h, a)$
     return ChanceSamplingCFR(ha, i, t, π_1, π_2)
end if
Let I be the information set containing h
$\sigma^t(I) \leftarrow$ RegretMatching (r_I)
$v_\sigma \leftarrow 0$
$v_{I\rightarrow a}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(I)$ **do**
    if $P(h) = 1$ **then**
        $v_{I\rightarrow a}[a] \leftarrow \text{ChanceSamplingCFR}(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$
    else if $P(h) = 2$ **then**
        $v_{I\rightarrow a}[a] \leftarrow \text{ChanceSamplingCFR}(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$
    end if
    $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{I\rightarrow a}[a]$
end for
if $P(h) = i$ **then**
    for $a \in A(I)$ **do**
        $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{I\rightarrow a}[a] - v_\sigma)$
        $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$
    end for
end ifreturn v_σ

1.2 Vanilla Counterfactual Regret Minimization

Algorithm 2 Vanilla Counterfactual Regret Minimization [1]

Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$
Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$
VanillaCFR(h, i, t, π_1, π_2):
if h is terminal **then return** $u_i(h)$
else if h is a chance node **then**
    Sample a single outocme $a \sim \sigma_c(h, a)$
    return $\sum_{a \in A(h)} \sigma_c(h, a)$ VanillaCFR(ha, i, t, π_1, π_2)
end if
Let I be the information set containing h
$\sigma^t(I) \leftarrow$ RegretMatching (r_I)
$v_\sigma \leftarrow 0$
$v_{I \to a}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(I)$ **do**
    if $P(h) = 1$ **then**
        $v_{I \to a}[a] \leftarrow$ VanillaCFR$(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$
    else if $P(h) = 2$ **then**
        $v_{I \to a}[a] \leftarrow$ VanillaCFR$(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$
    end if
    $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{I \to a}[a]$
end for
if $P(h) = i$ **then**
    for $a \in A(I)$ **do**
        $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{I \to a}[a] - v_\sigma)$
        $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$
    end for
end ifreturn v_σ

1.3 External Sampling Monte Carlo Counterfactual Regret Minimization

Algorithm 3 External Sampling Monte Carlo Counterfactual Regret Minimization [2]

Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$
Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$
ExternalSamplingMCCFR(h, i):
if h is terminal **then return** $u_i(h)$
else if h is a chance node **then**
    Sample a single outocme $a \sim \sigma_c(h, a)$
     return ExternalSamplingMCCFR(ha, i)
end if
Let I be the information set containing h
$\sigma^t(I) \leftarrow$ RegretMatching
if $P(h) = i$ **then**
    Let u be an array indexed by actions and $u_\sigma \leftarrow 0$
    for $a \in A(I)$ **do**
        $u[a] \leftarrow$ ExternalSamplingMCCFR(ha, i)
        $u_\sigma \leftarrow u_\sigma + \sigma(I, a) \cdot u[a]$
    end for
    for $a \in A(I)$ **do**
        $\tilde{r}(I, a) \leftarrow u[a] - u_\sigma$
        $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$
    end forreturn u_σ
else
    Sample a single outocme $a \sim \sigma(I)$
    $u \leftarrow$ ExternalSamplingMCCFR(ha, i)
    for $a \in A(I)$ **do**
        $s_I[a] \leftarrow s_I[a] + \sigma^t(I, a)$
    end forreturn u
end if

1.4 Outcome Sampling Monte Carlo Counterfactual Regret Minimization

Algorithm 4 Outcome Sampling Monte Carlo Counterfactual Regret Minimization [2]

Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$

Initialize: $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$

Initialize: $\forall I \in \mathcal{I} : c_I \leftarrow 0$

OutcomeSamplingMCCFR$(h, i, t, \pi_i, \pi_{-i}, s)$:

if h is terminal **then return** $u_i(h)/s, 1$

else if h is a chance node **then**

  Sample a single outcome $a \sim \sigma_c(h, a)$

  return OutcomeSamplingMCCFR$(ha, i, t, \pi_i, \pi_{-i}, s)$

end if

Let I be the information set containing h

$\sigma^t(I) \leftarrow$ RegretMatching

Let $\sigma'(I)$ be a sampling distibution at I

if $P(h) = i$ **then**

  $\sigma'(I) \leftarrow \epsilon\cdot$ UnIf(I) $+(1-\epsilon)\sigma(I)$

else

  $\sigma'(I) \leftarrow \sigma(I)$

end if

Sample an action a' with probability $\sigma'(I, a)$

if $P(h) = i$ **then**

  $(u, \pi_{tail}) \leftarrow$ OutcomeSamplingMCCFR$(ha', i, t, \pi_i \cdot \sigma(I, a), \pi_{-i}, s \cdot \sigma'(I, a))$

  for $a \in A(I)$ **do**

    $W \leftarrow u \cdot \pi_{-i}$

    compute $\tilde{r}(I, a)$

    $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$

  end for

else

  $(u, \pi_{tail}) \leftarrow$ OutcomeSamplingMCCFR$(ha', i, t, \pi_i, \pi_{-i} \cdot \sigma(I, a), s \cdot \sigma'(I, a))$

  for $a \in A(I)$ **do**

    $s_I[a] \leftarrow s_I[a] + (t - c_I) \cdot \pi_{-i} \cdot \sigma(I, a)$

  end for

  $c_I \leftarrow t$

end ifreturn $(u, \pi_{tail} \cdot \sigma(I, a))$

1.5 General Fictitious Self-Play

Algorithm 5 General Fictitious Self-Play [3]

function FictitiousSelfPlay(Γ, n , m)
    Initialize: completely mixed π_1
    $\beta_2 \leftarrow \pi_1$
    $j \leftarrow 2$
    while within cumpuational budget **do**
        $\eta_j \leftarrow$ MixingParameter(j)
        $\mathcal{D} \leftarrow$ GenerateData($\pi_{j-1}, \beta, n, m, \eta_j$)
        for each player $i \in \mathcal{N}$ **do**
            $\mathcal{M}^i_{RL} \leftarrow$ updaterlmemory($\mathcal{M}^i_{RL}, \mathcal{D}^i$)
            $\mathcal{M}^i_{SL} \leftarrow$ updateslmemory($\mathcal{M}^i_{SL}, \mathcal{D}^i$)
            $\beta^i_{j+1} \leftarrow$ reinforcementlearning($\mathcal{M}^i_{RL}$)
            $\pi^i_j \leftarrow$ supervisedlearning($\mathcal{M}^i_{SL}$)
        end for
        $j \leftarrow j + 1$
    end while
     return π_{j-1}
end function

function GenerateData(π, β, n, m, η)
    $\sigma \leftarrow (1 - \eta)\pi + \eta\beta$
    $\mathcal{D} \leftarrow n$ episode $\{t_k\}_{1 \le k \le n}$ sampled from strategy profile σ
    for each player $i \in \mathcal{N}$ **do**
        $\mathcal{D}^i \leftarrow m$ episode $\{t^i_k\}_{1 \le k \le n}$ sampled from strategy profile (β^i, σ^{-i})
        $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \mathcal{D}$
    end for
     return $\{D_k\}_{1 \le k \le N}$
end function

1.6 Batch Fictitious Self-Play

Algorithm 6 Batch Fictitious Self-Play [4]

Initialize: completely mixed average strategy profile π_1

Initialize: replay memories $\mathcal{M}_{RL}, \mathcal{M}_{SL}$

Initialize: the constant numbers of episode that are sampled simulataneously, n, and alternatingly, m

for $k = 1, ..., K-1$ **do**

  SAMPLESIMULTANEOUSEXPERIENCE $(n, \pi_k, \mathcal{M}_{RL})$

  Each player i updates their approximate best response strategy

  $\beta^i_{k+1} \leftarrow$ REINFORCEMENTLEARNING$(\mathcal{M}^i_{RL})$

  SAMPLEALTERNATINGEXPERIENCE $(m, \pi_k, \beta_{k+1}, \mathcal{M}_{RL}, \mathcal{M}_{SL})$

  Each player i updates their average strategy

  $\pi^i_{k+1} \leftarrow$ SUPERVISEDLEARNING$(\mathcal{M}^i_{SL})$

end for

return π_k

function SAMPLESIMULTANEOUSEXPERIENCE$(n, \pi_k, \mathcal{M}_{RL})$

  Sample n episodes from strategy profile π

  For each player i store their experienced transitions, $(u^i_t, a_t, r_{t+1}, u^i_{t+1})$ in their reinforcement learning memory $\mathcal{M}^i_{RL}$

end function

function SAMPLEALTERNATINGEXPERIENCE$(m, \pi_k, \beta_{k+1}, \mathcal{M}_{RL}, \mathcal{M}_{SL})$

  for each player $i \in \mathcal{N}$ **do**

    Sample m episodes from strategy profile (β^i, π^{-i})

    Store player i's experienced transitions, $(u^i_t, a_t, r_{t+1}, u^i_{t+1})$ in their reinforcement learning memory $\mathcal{M}^i_{RL}$

    Store player i's experienced own behaviour, (u^i_t, a_t) or $(u^i_t, \beta^i(u^i_t))$ in their supervised learning memory $\mathcal{M}^i_{SL}$

  end for

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]

Initialize Q-learning parameters, e.g. learning stepsize
Initialize all agents' table-lookup action-values, $\{Q^i\}_{1 \in N}$
Initialize all agents' table-lookup counting models, $\{N^i\}_{1 \in N}$
Initialize and run Algorithm 6 (Batch FSP) with REINFORCEMENTLEARNING and SUPERVISEDLEARNING methods below

function REINFORCEMENTLEARNING($\mathcal{M}^i_{RL}$)
    Restore previous iteration's Q^i-values
    Update (decay) learning stepsize and Boltzmann temperature
    updated Q^i-values with Q-learning from $\mathcal{M}^i_{RL}$
    return Boltzmann[Q^i]
end function

function SUPERVISEDLEARNING($\mathcal{M}^i_{SL}$)
    Restore previous iteration's counting model N^i, and average strategy, π^i
    for each $(u_t, \rho_t))$ in $\mathcal{M}^i_{SL}$ **do**
        $\forall a \in \mathcal{A}(u_t) : N^i(u_t, a) \leftarrow N^i(u_t, a) + \rho_t(a)$
        $\forall a \in \mathcal{A}(u_t) : \pi^i(u_t, a) \leftarrow N^i(u_t, a)/N^i(u_t)$
    end for
    Empty $\mathcal{M}^i_{SL}$
    return π^i
end function

1.8 Neural Fictitious Self-Play

Algorithm 8 Neural Fictitious Self-Play [5]

Initialize game Γ and excute an agent via RUNAGENT for each player in the game

function RUNAGENT(Γ)

  Initialize replay memories $\mathcal{M}_{RL}$ (circular buffer) and $\mathcal{M}_{SL}$ (reservoir)

  Initialize average-policy network $\Pi(s, a|\theta^{\Pi})$ with random parameters θ^{Π}

  Initialize action-value network $Q(s, a|\theta^{Q})$ with random parameters θ^{Q}

  Initialize target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$

  Initialize anticipatory parameter η

  for each episode **do**

    Set policy $\sigma \leftarrow \begin{cases} \epsilon - greedy(Q), \text{with probability } \eta \\ \Pi, \text{with probability } 1 - \eta \end{cases}$

    for $t = 1, ..., T$ **do**

      Observe information state s_t and sample action a_t from σ

      Execute action a_t in game and observe reward r_{t+1} and next information state s_{t+1}

      Store transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in reinforcement learning memory M_{RL}

      if agent follows best response policy $\sigma = \epsilon - greedy(Q)$ **then**

        Store behavior tuple (s_t, a_t) in supervised learning memory M_{SL}

      end if

      Update θ_{π} with stochastic gradient descent on loss

        $\mathcal{L}(Q^{\pi}) = \mathbb{E}_{(s,a)\sim M_{RL}}[-log\Pi(s, a|\theta^{\Pi})]$

      Update θ_{Q} with stochastic gradient descent on loss

        $\mathcal{L}(Q^{\pi}) = \mathbb{E}_{(s,a,r,s_{prime})\sim M_{RL}}[(r + max_{a'} Q(s', a'|\theta^{Q'}) - Q(s, a|\theta^{Q}))^2]$

      Periodically update target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$

    end for

  end for

end function

1.9 Neural Fictitious Self-Play with PPO

Algorithm 9 Neural Fictitious Self-Play with PPO [5]

Initialize game Γ and excute function PPO

function PPO(Γ)
  Initialize number of parallel processing N
  Initialize number of players NP
  Initialize replay memories $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$ (reservoir)
  Initialize average-policy network $\Pi(s, a|\theta^{\Pi})$ with random parameters θ^{Π}
  Initialize action-value network $\beta(s, a|\theta^{\beta})$ with random parameters θ^{β}
  Initialize $\theta^{\beta}_{old} \leftarrow \theta^{\beta}$
  for each iteration **do**
    for actor $= 1, 2, ..., N$ **do**
      target-player = (actor -1) % NP
      Set policy $\sigma \leftarrow \begin{cases} \beta_{\theta_{old}}\ strategy & \text{if } P(s) = \text{target-player for s in } \sigma \\ \Pi\ strategy & \text{if } P(s) \mathrel{!=} \text{target-player for s in } \sigma \end{cases}$
      make environment including chance, terminal and un-target-player node
      $t = 0$
      done = False (whether one episode finish)
      observe s_0 from environment
      while not done **do**
        Observe information state s_t and sample action a_t from σ
        Execute action a_t in game and observe state s_{t+1}, reward r_{t+1} and done
        Store transitions $(s_t, a_t, pd_t, r_{t+1})$ in reinforcement learning memory M_{RL}
        Store behavior tuple (s_t, a_t) in supervised learning memory M_{SL}
        $t+ = 1$
      end while
      Compute adavantage estimates $\hat{A}_1, ..., \hat{A}_t$
    end for
    Optimize surrogate L_{RL} wrt θ^{β} in M_{RL}
    $\beta_{\theta_{old}} \leftarrow \beta_{\theta}$
    Optimize L_{SL} wrt θ^{π} with cross entropy in M_{SL}
    $\Pi_{\theta_{old}} \leftarrow \Pi_{\theta}$
    empty M_{RL}
  end for
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