Vol.2018-GI-39 No.7
2018/3/2

大貧民の状態価値(期待順位)の強化学習

桑原 和人[1,a)]　保木 邦仁[1,b)]

概要：現在コンピュータ大貧民では，機械学習した方策関数を用いたモンテカルロ法が有効とされている．一方，バックギャモンや囲碁などでは状態価値や方策関数の強化学習で強いプレイヤの作成に成功している．本研究では大貧民を題材として，畳込みニューラルネットワークを用いた順位予測による状態価値の推定と，状態価値に基づくグリーディ方策の作成を行った．「過去の UEC コンピュータ大貧民大会 (UECda) 優勝プレイヤ同士の対戦の順位予測」と「ランダムプレイヤから開始した自己対戦による強化学習」の 2 つのアプローチを実施し，どちらも UECda 優勝プレイヤに準ずる強さのグリーディ方策が得られた．

Reinforcement learning of state value (expected rank) of *Daihinmin*

Kazuto Kuwabara[1,a)]　Kunihito Hoki[1,b)]

1. はじめに

大貧民 (大富豪とも呼ばれる) は，トランプを用いて行うゲームである．場に出ているカードより強いカードを出さなければならないという制約の下で，各プレイヤは手札からカードを場に出し他のプレイヤよりも手札を早くなくすことを目指す．複数人でプレイするゲームで，相手の手札が公開されていないという点から，大貧民は多人数不完全情報ゲームに分類される．

大貧民には多くのローカルルールがあるが，本研究では UEC コンピュータ大貧民大会 (UECda)[*1] のルールを採用した．その理由は，UECda の過去の優勝プログラムが公開されており，それらを用いた学習や性能の比較を用意に行うことができるからである．

UECda は, 電気通信大学で毎年開催されているコンピュータ大貧民の強さを競う大会である．2006 年に第 1 回大会が開催され，2017 年 11 月 25 日に第 11 回大会が開催された．UECda にはライト級と無差別級の 2 つの部門が存在し，ライト級はヒューリスティックな手法のみを用いたプレイヤ，無差別級はこれに加えて機械学習・モンテカルロ法などを適用したプレイヤが参加する部門である．

現在，UECda ではモンテカルロ法が有効とされており，2009 年の第 4 回大会にモンテカルロ法を適用したプレイヤである fumiya [1] が登場し優勝して以来，UECda の無差別級ではモンテカルロ法を適用したプレイヤの優勝が続いている [2], [3], [4].

UECda における標準ルール（UEC ルール 2007）では，対戦は 5 人で行われる．1 プレイにおける獲得点は，1 位が 5 点，2 位が 4 点，3 位が 3 点，4 位が 2 点，5 位が 1 点であり，最終順位は複数回プレイしたときの総得点で決まる．

2. 強化学習

強化学習とは，相互作用から学習して目標を達成する問題の枠組みである．この枠組みを，文献 [5] を抜粋して述べる．

エージェントと環境が離散的な時間ステップ $t = 0, 1, 2, 3, \ldots$ において相互作用を行うような状況を考える．各時間ステップ t において，エージェントは何らかの環境の状態の表現 $s_t \in \mathcal{S}$ ($\mathcal{S}$ は可能な状態の集合) を受け取り，行動 $a_t \in \mathcal{A}(s_t)$ を選択する ($\mathcal{A}(s_t)$ は状態 s_t において選択することのできる行動の集合)．1 時間ステップ後にエージェントはその行動の結果として数値化された報酬

[1] 電気通信大学
The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[a)] kaz03301@gmail.com
[b)] k.hoki@uec.ac.jp

[*1] UEC コンピュータ大貧民大会 http://www.tnlab.ice.uec.ac.jp/daihinmin/2017/ (last access, 2018)

$r_{t+1} \in \mathbb{R}$ を受け取り，新しい状態 s_{t+1} を観測する．

各時間ステップにおける状態から可能な行動を選択する確率の写像を方策と呼び、π_t で表す．$\pi_t(s,a)$ はもし $s_t = s$ ならば $a_t = a$ となる確率である．

強化学習法の目的は，エージェントがこの方策を変更し，最終的に受け取る報酬の総量を最大化することである．即ち，目的は，時間ステップ t の後に受け取った報酬の系列を $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$ として，期待収益を最大化することである．収益 R_t は，最も単純には，報酬の合計 $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$ で表される．T は最終時間ステップである．この種の最終時間があるエピソードを伴うタスクはエピソード的タスクと呼ばれる．対して，エージェントと環境の相互作用が限界なく継続的に進行していくタスクは連続タスクと呼ばれるが，これは本研究では扱わない．

マルコフ決定過程では，時刻 $t+1$ における環境の応答が t における状態と行動表現のみに依存し，つまりすべての s', r, s_t, a_t に対して $\Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$ のみを指定することで環境のダイナミクスを定義することができる．特に，状態と行動の空間が有限ならば，有限マルコフ決定過程 (有限 MDP) と呼ばれる．

有限 MDP は主に状態と行動の集合と，環境の 1 ステップダイナミクスから定義される．任意の状態と行動 s, a が与えられたとき，次に可能な各状態 s' の確率 $\mathcal{P}^a_{ss'}$ を遷移確率と呼び，現在の任意の状態 s と行動 a が与えられたとき，次の報酬の期待値は $\mathcal{R}^a_{ss'}$ で表される．

2.1 価値関数と方策の改善

強化学習アルゴリズムでは，価値関数に基づく評価を行う．価値関数は，状態の関数で，エージェントがある状態にいることがどれだけ良いのかを評価する．正確には，特定の方策に関する期待収益で定義される．

方策 π のもとで状態 s の価値 $V^\pi(s)$ は，状態 s にいて方策 π に従ったときの期待収益であり，$V^\pi(s) = E_\pi\{R_t | s_t = s\}$ で表される．$E_\pi\{\}$ は，エージェントが π に従うとしたときの期待値を表す．また，エピソードが終わった後の時刻 t では $r_t = 0$ とする．終端状態の価値は 0 である．関数 V^π を方策 π に対する状態価値関数と呼ぶ．

同様に，方策 π のもとで状態 s において行動 a を取ることの価値を $Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\}$ で表す．$Q^\pi(s,a)$ を方策 π に対する行動価値関数と呼ぶ．

方策とその価値関数が与えられた時，すべての状態で $Q^\pi(s,a)$ が最大になる行動を選択することで，元の方策と同等かより良い方策を得ることができる．価値が最大の行動を取る方策はグリーディ方策と呼ばれる．元の方策の価値関数に従ってグリーディな行動を選択し，その方策を改善するような方策を作り出す過程を方策改善と呼ぶ．任意の π に対する状態価値 V^π を計算する方法を方策評価と呼ぶ．

状態価値を使って方策を改善しより優れた方策を得て，今度はその方策による状態価値を使ってまた方策を改善する，というように方策と価値関数を改善する方法を方策反復と呼ぶ．方策反復では，一方が終了した段階でもう一方が開始されるように，2 つの過程が交互に行われる．対して，評価と改善の過程の粒度を細かくして価値関数の収束を待たずに方策改善を実行する方法も存在し，一般的な概念として一般化方策反復と呼ぶ．

2.2 方策オフ型モンテカルロ法

ここでは，標本平均をもって収益の期待値を推定して強化学習問題を解くモンテカルロ法の一種である方策オフ型モンテカルロ法について説明する．方策 π に従って状態 s を通過するエピソードの集合が与えられたとして，π のもとでの s の価値 $V^\pi(s)$ を推定することを考える．方策オフ型モンテカルロ法は，挙動を生成するための方策 (挙動方策) と評価され改善される方策 (推定方策) が分離させる方法である．この利点は，グリーディ方策のようなすべての可能な行動をサンプリングすることが困難な方策に対して方策評価が可能となることである．方策オフ型モンテカルロ法のアルゴリズムは以下に示す．

すべての $s \in \mathcal{S}, a \in \mathcal{A}$ に対して初期化：
  $Q(s,a) \leftarrow$ 任意の値
  $N(s,a) \leftarrow 0 (Q(s,a)$ の分子)
  $D(s,a) \leftarrow 0 (Q(s,a)$ の分母)
$\pi \leftarrow$ 任意の決定論的方策
永久に繰り返し：
  (a) π' を選択し，それを用いて 1 個のエピソード
$s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T, s_T$ を生成する
  (b) $\tau \leftarrow$ 最も最近 $a_\tau \neq \pi(s_\tau)$ となった時刻
  (c) 時刻 τ あるいはそれ以降にエピソード中に出現した各 s,a の対について：
    $t \leftarrow t \leq \tau$ であるような s,a が最初に発生した時刻
    $w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k,a_k)}$
    $N(s,a) \leftarrow N(s,a) + wR_t$
    $D(s,a) \leftarrow D(s,a) + w$
    $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
  (d) 各 $s \in \mathcal{S}$ について：
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$

ここで，w は π' を用いたことによる R_t の重みで，π, π' に対する相対生起確率である．挙動方策 π' は $s \in \mathcal{S}, a \in \mathcal{A}(s)$ に対して $\pi'(s,a) > 0$ が満たされるような方策が用いられる．例えば，確率 ϵ でランダムに行動を選択し，確率 $1-\epsilon$ で $\pi(s)$ に従う ϵ-グリーディ方策が考えられる．

これまで考えていた方策オフ型モンテカルロ法は，行動価値をテーブル形式で保存する方法である．この形式は，メモリの問題やテーブルの値を正確に求めるための時間や

データ数の問題により，状態数と行動数が小さいタスクでの利用に限られてしまう．より大きい状態と行動をもつタスクに対しては一般化を行う必要があり，強化学習では既存の一般化手法である関数近似と組み合わされることが主である．

ここで，行動価値関数をパラメータベクトル $\boldsymbol{\theta}$ を用いた関数で近似することとし，行動価値関数を $Q^{\boldsymbol{\theta}}$ と表すとする．この場合における方策オフ型モンテカルロ法では，テーブルの値を変更する代わりに価値が収益 R_t の期待値に近づくようにパラメータの調整を行う．例えば，ニューラルネットワークを用いた関数近似であれば R_t との損失関数値を小さくするような方法が考えられる．関数近似を用いる方策オフ型モンテカルロ法のアルゴリズムの一例を以下に示す．

$\boldsymbol{\theta} \leftarrow$ 任意のパラメータベクトル

$\mathcal{D} \leftarrow$ 空集合 (学習データの集合)

$\pi \leftarrow$ 任意の決定論的方策

$E \leftarrow$ 任意の損失関数

永久に繰り返し：

(a) π' を選択し，それを用いて 1 個のエピソード $s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T, s_T$ を生成する

(b) $\tau \leftarrow$ 最も最近 $a_\tau \neq \pi(s_\tau)$ となった時刻

(c) 時刻 τ あるいはそれ以降にエピソード中に出現した各 s, a の対について：

$t \leftarrow t \leq \tau$ であるような s, a が最初に発生した時刻

$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, R_t, w)\}$

もし $\mathcal{D}$ の要素数が一定数以上なら

損失関数 $\sum_{(s_t, a_t, R_t, w) \in \mathcal{D}} wE(s_t, a_t, R_t, \boldsymbol{\theta})$ で $\boldsymbol{\theta}$ を調整

$\mathcal{D}$ をある程度削除

(d) 各 $s \in \mathcal{S}$ について：

$\pi(s) \leftarrow \arg\max_a Q^{\boldsymbol{\theta}}(s, a)$

3. 先行研究

コンピュータ大貧民では，ランダムシミュレーションを用いてプレイの期待獲得点を算出するモンテカルロ法が有効とされている．実際，UECda では 2009 年にモンテカルロ法を適用したプレイヤが登場し優勝して以来 [1]，モンテカルロプレイヤが優勝を続けている．大渡らは，2016 年優勝プレイヤの Glicine (グリーチネ) に用いられている線形関数による方策関数の機械学習法の研究を行った [4].

他のゲームでは，ニューラルネットワークと強化学習を組み合わせて強いプレイヤを作成することに成功している．バックギャモン [6]，ヘッズアップノーリミットテキサスホールデム [7]，チェス/囲碁/将棋 [8] のいずれもニューラルネットワークを用いた価値関数の近似を行っている．これらの研究と本研究の価値と関数近似に用いたニューラルネットワークの構成を表 1 に示す．

表 1 他のゲームの価値とニューラルネットワークの構成

ゲーム	価値	ネットワーク構成
バックギャモン	期待勝ち点	全結合
テキサスホールデム	期待獲得チップ	全結合
チェス/囲碁/将棋	勝率	畳込み
本研究	期待獲得点	畳込み

4. 目的

大貧民における本研究の目的は，強化学習と深層学習を適用し，強いコンピュータ大貧民プレイヤを作成することである．現在，大貧民では方策関数の機械学習とモンテカルロ法を組み合わせる手法が有効とされているが，バックギャモンやポーカー，チェス/囲碁/将棋等のように行動・状態価値を強化学習により推定する手法はまだ適用されていない．本研究では，状態価値に基づいたグリーディ方策を 2 つのアプローチから作成し，性能を評価する．

1 つめは，大渡らが [4] で行ったように，過去の UECda 優勝プレイヤのプレイログを学習データに用いるアプローチである．本研究では，UECda 優勝プレイヤ同士の対戦によるログから，そのプレイヤの方策に対する状態価値を推定し，方策改善により良い方策を得ることを目指す．

2 つめは，自己対戦によるアプローチである．ランダムプレイヤから開始し，自己対戦によって方策評価と方策改善を繰り返すこと (一般化方策反復) によって状態価値を獲得することを目指す．

一般化方策反復は，あたかも 5 人ゲームを有限 MDP であるかのように近似的に扱って行う．ここでは，1 つのゲームプレイは 5 つの収益が異なるエピソード的タスクとして扱われる．

また，状態価値関数はニューラルネットワークを用いて近似する．状態価値はそのプレイにおける期待獲得点とし，ニューラルネットワークで事後状態に対する順位予測を行うことでこれを実現する．ネットワークは畳込みニューラルネットワークと全結合ニューラルネットワークを用いて性能を比較する．さらに，畳込みニューラルネットワークではプレイ履歴を入力に加えることで性能が向上するかを調査する．

5. 大貧民の事後状態と行動

事後状態とは，行動後のゲーム状況のことである．行動は，手番プレイヤにおける以下の一連のカード提出とする．

(1) 再提出可能であるカード提出 (パスを含む) 0 回以上

(2) 再提出可能であるとは限らないカード提出 (パスを含む) 1 回

あるカード提出が，再提出可能である条件は以下とした．

(1) 8 切り

(2) ♠3 切り
(3) 相手プレイヤの持ちうるカード集合と，パスしていない相手プレイヤの残り手札枚数から，必ず場が流れることが分かるカードの提出．例えば，相手プレイヤが 2 以上のカードを持ってないことが判明しているときに A を提出したり，パスしていない相手プレイヤの残り手札枚数が全員 3 枚以下であるときに 4 枚出しをする場合は必ず場が流れることが分かる．
(4) 相手プレイヤが全員パスしている状態でのパス．ただし，最後に提出したプレイヤが上がっている状態で全員がパスした場合，次の手番プレイヤは最後に提出したプレイヤの次の席のプレイヤとなることに注意する．

例えば，[[♣8 ♢8], [♣3]] という一連のカード提出であれば 1 つめの提出が条件 1 を満たしている 1 つの行動，[[♣2 ♢2], [♠8], [♣4]] であれば 1 つめの提出が条件 3，2 つめの提出が条件 1 を満たしている 1 つの行動とみなすことができる．

カード提出アルゴリズムは，事後状態の価値が最大となる事後状態に遷移する行動をグリーディに選択するというものである．全体の流れは以下の通りである．

(1) 残り 2 人であれば，snowl (UECda2010 優勝プレイヤ) による必勝手探索を行う．必勝手が見つかればそれを選択し，そうでなければ 2 へ進む．
(2) 深さ優先探索で行動を列挙しながら，各事後状態の価値の算出を行う．もし，あがりとなる行動を見つけたらそれを選択し，そうでなければ価値最大となる行動を選択する．

行動の列挙は，カード提出を枝とした木を深さ優先探索することで行う．再提出可能なカード提出を行った後のゲーム状況は内部節点となり，事後状態は終端節点となる．

カード提出前の状況を根節点とし，次のようにして行動を列挙する．

- 提出可能なカード集合を列挙する．
- 各カード提出に対して，事後状態かどうかを判定する．
  - カード提出が再提出可能である条件 1,2,4 のいずれかを満たしていたら，場が流れた内部節点を生成する．
  - カード提出が再提出可能である条件 3 を満たしていたら，相手全員がパスした内部節点を生成する (自分もパスをして場が流れたこととしないのは，自分が出したカードに対してさらにカードを提出することが可能であるため)．
  - 1 から 4 の条件のいずれも満たしていなかったら，その節点を事後状態とする．

また，ゾブリストハッシュ法 [9] で訪れた節点に対するハッシュ値をメモすることで探索木を DAG として扱い深さ優先探索を効率化した．

事後状態の価値は，プレイの期待獲得点である．ニューラルネットワークに事後状態 (または事後状態とプレイ履歴) を入力として与え，そのプレイの順位の確率分布を出力する．1 位から 5 位までの確率を出力し，期待獲得点を算出する．

6. ニューラルネットワークの構成

本研究では，3 つのニューラルネットワーク構成を用いる．1 つめは事後状態を入力とする畳込みニューラルネットワーク (BaseCNN)，2 つめは事後状態とプレイ履歴を入力とする畳込みニューラルネットワーク (HistoryCNN)，3 つめは事後状態を入力とする全結合ニューラルネットワーク (FC-NN) である．これら 3 つのネットワーク構成について説明する．

BaseCNN

BaseCNN は，事後状態を入力とする畳込みニューラルネットワークである．このネットワーク構成を用いたグリーディ方策は，UECda の大会環境で制限時間内で動作することを想定したネットワーク規模となっている．

ネットワーク構成を表 2 に示す．ただし，対戦実験では畳込み層の出力チャネル数を変えた構成も作成した．以降，出力チャネルの違うネットワーク構成を区別して明記する場合は BaseCNNx とする (x は畳込み層 ×10 の出力チャネル数)．

表 2 BaseCNN のネットワーク構成

層	詳細
入力層	出力チャネル：151，画像サイズ：縦 1 横 15
畳込み層 ×10	出力チャネル：128，フィルタサイズ：縦 1 横 3 パディング：縦 0 横 1， 活性化関数：ReLU
畳込み層	出力チャネル：1，フィルタサイズ：縦 1 横 3 パディング：縦 0 横 1， 活性化関数：なし
全結合層	出力ユニット：5
ソフトマックス層	

BaseCNN におけるゲーム状況は，縦幅が 1，横幅が 15 の画像 151 チャネルで表現する．詳細を以下に示す．

手札情報
　自分の手札 (5 チャネル)
　相手の手札 (5 チャネル)
場の情報
　場に出ているカード (5 チャネル)
　場の強さ (1 チャネル)
　階段 (5 チャネル)
　ペア (5 チャネル)
　革命 (1 チャネル)
各プレイヤの情報
　提出したカード (5×5 チャネル)
　残り手札枚数 (12×5 チャネル)
　パスまたはあがり (1×5 チャネル)
　手札交換で判明している確定手札 (5×4 チャネル)
その他 (定数)

カードの位置 (13 チャネル)

HistoryCNN

HistoryCNN は，事後状態とプレイ履歴を入力とする畳込みニューラルネットワークである．このネットワーク構成を用いたグリーディ方策は UECda の時間制限内で動作することは想定しておらず，BaseCNN の約 1/20 の計算速度である．

プレイ履歴は，誰がどの状況でどの行動を取ったかという情報，つまり手番プレイヤ (プレイヤ番号ではなく席の位置)・行動前のゲーム状況・提出したカードの組で表され，各プレイヤがそれぞれ保持する．プレイ履歴の追加は，手番プレイヤが行動した後に行う．ただし，相手プレイヤがパスしか選択出来ない状況でパスした場合は追加はしないこととする．

ネットワーク構成を表 3 に示す．HistoryCNN では，ゲーム状況を事後状態と直近 63 個のプレイ履歴で表し，縦幅 64，横幅 15 の画像 161 チャネルで表現する．1 つのプレイ履歴は縦幅 1，横幅 15 の画像 161 チャネルで表現され，画像の縦方向を時間方向とみなして直近 63 個のプレイ履歴を表した画像を各チャネルごとに縦に連結する．さらに事後状態を表す画像と連結したものが入力となる．つまり，入力の各チャネルについて 1 行目が事後状態，2 行目が最も直近のプレイ履歴，3 行目が 2 番目に直近のプレイ履歴…となる．

表 3 HistoryCNN のネットワーク構成

層	詳細
入力層	出力チャネル：161，画像サイズ：縦 64 横 15
(畳込み層 ×2 + max プーリング層)×3	畳込み層 出力チャネル：128，フィルタサイズ：縦 3 横 3 パディング：縦 1 横 1， 活性化関数：ReLU max プーリング層 フィルタサイズ：縦 2 横 1，ストライド：縦 2 横 1
畳込み層 + max プーリング層	同上
畳込み層	出力チャネル：1，フィルタサイズ：縦 3 横 3 パディング：縦 1 横 1， 活性化関数：なし
全結合層	出力ユニット：5
ソフトマックス層	

FC-NN

FC-NN は，事後状態のみを入力とする全結合ニューラルネットワークである．畳込みニューラルネットワークと全結合ニューラルネットワークの性能を比較するため，重みパラメータの総数が BaseCNN と近いネットワーク構成となるようにした．

ネットワーク構成を表 4 に示す．全結合ニューラルネットワークにおけるゲーム状況は 750 ユニットで表現する．

表 4 FC-NN のネットワーク構成

層	詳細
入力層	出力ユニット：750
全結合層	出力ユニット：700，活性化関数：ReLU
全結合層	出力ユニット：5， 活性化関数：なし
ソフトマックス層	

7. アプローチ 1：UECda 優勝プレイヤを用いた強化学習

Glicine の方策評価を方策オフ型モンテカルロ法に基づき行い，それによって得られた状態価値関数に対するグリーディ方策を得ることを目指す．本研究では，2 パターンの方策を挙動方策として用いる．1 つめは，デフォルト設定の Glicine (以下 GlicineNormal)，2 つめは，一定確率でランダム行動を行う Glicine (以下 Glicine-ϵ) である．本研究ではランダム行動確率を 3%とした．

GlicineNormal と Glicine-ϵ でそれぞれ約 10,000,000 プレイ分のログを生成した．CPU Intel Xeon X5690x2 12 コア，メモリ 24GB マシンでサーバープログラムを並列して 5 つ実行してログの生成を行った．ログの生成スピードは，1 つのサーバープログラムで 10,000 プレイあたり約 250 分であった．

学習データは，対戦ログを用いて以下の手順で生成する．

(1) プレイ開始から終了までログを再生し，手番プレイヤの行動後の事後状態 (手番プレイヤを視点としたもののみ) を生成する．ただし，残りプレイヤが 2 人であるときの事後状態は学習データとする必要がないため除く．また，Glicine-ϵ の対戦ログにおいてはランダム行動以降の事後状態のみを学習データとする．つまり，ランダム行動を観測したとき，それまでに保持していた事後状態をすべて捨てる．さらに，HistoryCNN の学習データを作成する場合は各プレイヤごとにプレイ履歴も保持する．

(2) プレイ終了後、各プレイヤの各事後状態 (HistoryCNN では事後状態とプレイ履歴の組) に対してそのプレイヤの順位をラベル付けする．ラベルは，1 位は 0，2 位は 1，3 位は 2，4 位は 3，5 位は 4 とする．

1 プレイあたりのデータの数は，GlicineNormal の対戦では約 25，Glicine-ϵ の対戦では約 14 であった．

重みベクトルの調整は，順位を正解ラベルとする 5 クラス分類問題とみなし，交差エントロピー誤差を損失関数としたミニバッチによる確率的最急降下法 (SGD) を用いて行った．同じプレイの状態や収益はそれぞれ相関があるため，データは 100,000 プレイ分読み込んでシャッフルしてから与えた．バッチサイズは，読み込んだ 100,000 プレイから生成したデータの総数を 100,000 で割った数とした (GlicineNormal では約 25，Glicine-ϵ では約 14)．総データ数は，GlicineNormal では約 250,000,000，Glicine-ϵ

では約 140,000,000 である．学習率と慣性係数は，ネットワークの構成にかかわらずそれぞれ 0.0003，0.9 とし，重みパラメータの初期化は Xavier Initialization [10] を用いて行った．ニューラルネットワークの重みベクトルの調整は，Caffe(バージョン 1.0.0) [11] を用いて行った．GPU は GeForce GTX 980 Ti または GeForce GTX 1080 を用いた．バッチ処理速度は，ネットワークが BaseCNN128，ログ生成に使用した GlicineNormal の場合，GeForce GTX 1080 を用いて 1 時間あたり約 1,000,000 バッチであった (ただし，ログを読み込んで学習データを生成する時間も含む)．

方策オフ型モンテカルロ法では，推定方策と挙動方策に対する相対生起確率 w を計算する必要があるが，本研究では簡単のため，w を常に 1 とした．関数近似を行った場合に w を計算する利点は不明である．

学習アルゴリズムは以下の通りである．

アプローチ 1 の学習アルゴリズム

$\boldsymbol{\theta} \leftarrow$ ランダムに初期化 (ニューラルネットワークの重みベクトル)
$\mathcal{D} \leftarrow$ 空集合 (学習データの集合)
$\pi \leftarrow$GlicineNormal の方策
$\pi' \leftarrow$GlicineNormal の方策または Glicine-ϵ の方策
繰り返し：
　π' による 5 人対戦で生成された 5 つのエピソードの内，各プレイヤ $i(1 \le i \le 5)$ のエピソード $s_1^i, \cdots, s_{T-1}^i, s_T^i, r_t^i$ について：
　　$\tau \leftarrow$ どのプレイヤも π に従っていた終端までの部分列が $s_\tau, \cdots, s_T$ となるような時刻 τ
　　時刻 τ あるいはそれ以降にエピソード中に出現した各 s_j^i について：
　　$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_j^i, r_t^i)\}$
　100,000 プレイごとに
　　$\mathcal{D}$ の要素をそれぞれ 1 回だけ用いて SGD で $\boldsymbol{\theta}$ を調整
　　$\mathcal{D} \leftarrow$ 空集合

8. アプローチ 2：自己対戦による強化学習

既存プレイヤを用いずにランダムな方策から開始した自己対戦により一般化方策反復を行う．BaseCNN の重みはランダムに初期化したものからスタートし，ログの生成と学習を並列に行う．ここでは，反復回数が $5,000,000 \times N$ であるネットワークを世代 N のネットワークと呼ぶこととし，ランダム初期化したネットワークは世代 0 とする．

ログは，直近 200 世代までからランダムに 5 つの選出して対戦を行い生成した．対戦は 10,000 プレイごとに組み合わせの変更を行うようにした．CPU Intel Xeon X5690x2 12 コア，メモリ 24GB のマシンでサーバープログラムを並列に 5 つ実行してログの生成を行った．ログの生成スピードは，1 つのサーバープログラムで 10,000 プレイあたり約 350 分であった．本研究では 9 台のマシンを用いて行った．

学習データの生成は，アプローチ 1 の GlicineNormal における方法と同様である．重みベクトルの調整は直近 5,000,000 プレイのログから生成し，SGD を行うことを繰り返した．また，この手法も方策オフ型モンテカルロ法に基づくが，挙動方策 π' は推定方策と等しいとした．

学習アルゴリズムは以下の通りである．

アプローチ 2 の学習アルゴリズム

$\boldsymbol{\theta} \leftarrow$ ランダムに初期化したニューラルネットワークの重みベクトル
$\mathcal{P} \leftarrow \{\boldsymbol{\theta}\}$(重みベクトルの集合)
(a) と (b) を並列に実行
(a) 繰り返し：
　5,000,000 プレイ分のログがたまるまで待つ
　$\mathcal{D} \leftarrow$ 空集合 (学習データの集合)
　直近 5,000,000 プレイでの $k(1 \le k \le 5,000,000)$ 回目について：
　　各プレイヤ $i(1 \le i \le 5)$ に関するエピソード $s_1^i, \cdots, s_{T-1}^i, s_T^i, r_t^i$ について：
　　　エピソード中に出現した各 s_j^i について：
　　　$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_j^i, r_t)\}$
　$\mathcal{D}$ を用いて SGD で $\boldsymbol{\theta}$ を調整 (5,000,000 反復行う)
　$\mathcal{P} \leftarrow \mathcal{P} \cup \{\boldsymbol{\theta}\}$
(b) 繰り返し：
　$\mathcal{P}$ の直近 200 世代までから重複を許してランダムに 5 つ重みベクトル $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_5$ を選出
　$\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_5$ を用いて 10,000 プレイ行い，ログを生成する

9. 実験

9.1 節では，アプローチ 1 と 2 で作成したグリーディ方策と過去の UECda 優勝プレイヤで対戦実験を行い，ネットワーク構成やログを生成したプレイヤが性能に与える影響を見る．9.2 節では，アプローチ 2 で作成した世代の異なるグリーディ方策同士で対戦実験を行い，世代と強さの関係を観察する．どちらの実験でもグリーディ方策だけは snowl に準拠した手札交換でプレイ開始することとする．

プレイヤの性能比較は，UECda ルールにおいて階級リセットが起こるタイミングである 100 プレイを 1 セットとし，1 セットの期待総獲得点で行う．また，プレイした事後状態の価値と実際の獲得点の平均二乗誤差平方根 (root mean sqared error,RMSE) を計算し，比較する．すなわち，事後状態の価値と実際の獲得点の組 $(\tilde{R}, R)$ が N 組あったとき，平均二乗誤差平方根 $RMSE$ は，

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\tilde{R}_j - R_j)^2}$$

で計算される．以降では，この獲得点に関する値を $RMSE$ と呼ぶこととする．

9.1 実験 1

アプローチ 1 とアプローチ 2 で作成したグリーディ方策

表 5 グリーディ方策の一覧．推定方策は, Self は Self, 他全ては Glicine.

プレイヤ名	ネットワーク	重み総数	学習反復回数	挙動方策
BaseCNN16	BaseCNN16	14,417	580,000,000	GlicineNormal
BaseCNN32	BaseCNN32	42,577	600,000,000	GlicineNormal
BaseCNN64	BaseCNN64	140,369	860,000,000	GlicineNormal
BaseCNN128	BaseCNN128	501,841	700,000,000	GlicineNormal
BaseCNN128eps	BaseCNN128	501,841	880,000,000	Glicine-ϵ
FC-NN	FC-NN	529,205	850,000,000	GlicineNormal
HistoryCNN	HistoryCNN	1,072,562	150,000,000	GlicineNormal
HistoryCNNeps	HistoryCNN	1,072,562	190,000,000	Glicine-ϵ
Self	BaseCNN128	501,841	1,280,000,000	Self

表 6 実験に用いた UECda 優勝プレイヤ一覧

snowl	UECda2010 優勝
beersong	UECda2013 無差別級優勝
kou2	UECda2015 ライト級優勝
Glicine	UECda2016 無差別級優勝

表 7 実験 1 の結果

	期待総獲得点	$RMSE$
BaseCNN16	288 ± 2	0.809 ± 0.003
BaseCNN32	300 ± 2	0.796 ± 0.003
BaseCNN64	309 ± 2	0.792 ± 0.008
BaseCNN128	311 ± 2	0.786 ± 0.003
BaseCNN128eps	308 ± 2	0.770 ± 0.002
FC-NN	285 ± 2	0.823 ± 0.002
HistoryCNN	312 ± 2	0.794 ± 0.004
HistoryCNNeps	307 ± 2	0.808 ± 0.002
Self	326 ± 2	0.778 ± 0.004
snowl	268 ± 2	
beersong	298 ± 2	
kou2	262 ± 2	
Glicine	336 ± 2	

9 種と過去の UECda 優勝プレイヤ 4 種の計 13 種のプレイヤを用いて対戦を行い，性能を比較する．

グリーディ方策を表 5，UECda 優勝プレイヤを表 6 に示す． 対戦は，13 種のプレイヤから重複を許さずランダムに 5 つ選出し，10 セット行うことを繰り返して行った．対戦の組み合わせを変更する間隔が 1 セットでない理由は，snowl や Glicine のようなプレイ中にパラメータの調整を行っているプレイヤがいるため，パラメータのリセットのタイミングを念のため遅らせるためである．各セットは互いに独立とは言えないが，独立と扱って期待総獲得点の標準誤差を見積もった．

結果を表 7 に示す．BaseCNN128 や Self などのグリーディ方策は，Glicine 以外の優勝プレイヤの性能を上回った．Glicine を用いて学習したグリーディ方策はいずれも Glicine の性能を上回ることができず，Glicine の方策に対する方策改善はなされなかった．その原因としては，畳込みニューラルネットワークの表現能力が足りていないこと，学習データがすべての事後状態を網羅していないことが挙げられる．

本研究で作成したグリーディ方策の中では，Self が一番性能が高い結果となり，アプローチ 1 と 2 ではアプローチ 2 の方がより性能の高いプレイヤを作成できたという結果となった．その要因としては，アプローチ 2 の方が多様なデータが獲得できていることが考えられる．

以上から，2 つのアプローチで，UECda2016 無差別級優勝プレイヤの Glicine に準ずる強さのグリーディ方策を作成することができたと結論づける．

また，$RMSE$ を見ると，期待総獲得点が高いプレイヤほど $RMSE$ が小さい傾向が見られる．したがって，価値関数が期待獲得点を正確に推定できればできるほど強いプレイヤとなると考えられる．

BaseCNN と FC-NN の比較

BaseCNN16 ~ 64 では，BaseCNN の畳込み層の出力チャネル数を増やすことで期待総獲得点が増加していることがわかる．しかし，BaseCNN64 と BaseCNN128 では有意な差が見られなかった．チャネル数 64 で強さが頭打ちになってしまっている可能性があるが，BaseCNN128 の学習反復回数が足りていなかったり, 学習データが足りていなかったりする可能性もある．

BaseCNN と FC-NN を比較すると，パラメータ数が同等である BaseCNN128 は FC-NN よりも圧倒的に性能が上であり，さらに FC-NN の約 1/37 のパラメータ数である BaseCNN16 が FC-NN と同等の性能であった．このことから, 本実験においては, 全結合ニューラルネットワークよりも畳込みニューラルネットワークの方がより良い状態価値の近似ができていると言える．

GlicineNormal と Glicine-ϵ の比較/BaseCNN と HistoryCNN の比較

BaseCNN128 と BaseCNN128eps の間, HistoryCNN, HistoryCNNeps の間で比較しても，大きな性能の差は見られず，GlicineNormal を用いた学習と Glicine-ϵ を用いた学習では差は現れなかった．Glicine-ϵ を用いることで，GlicineNormal では訪れない事後状態に対する学習を行うことを期待したが，プレイ開始時の手札配布のばらつき具合から GlicineNormal でも事後の探査が十分であったこと，データ量が少なく強さに差が現れるほどデータが採取できていないことが考えられる．

BaseCNN と HistoryCNN を比較しても，大きな差は見られなかった．過去のプレイヤの行動履歴を用いることでより良い状態価値関数を作成できることを期待したが，BaseCNN でも各プレイヤが提出したカードからある程度行動履歴が考慮できていたり，行動履歴を扱えるほどニューラルネットワークの表現能力が足りていなかったり，HistoryCNN はネットワークの規模も大きく学習に時間がかかるために学習反復回数が足りていなかったりしたことが考えられる．

表 8　実験 2 の結果

	学習反復回数	期待総獲得点	RMSE
世代 0	0	134.7 ± 0.3	1.8280 ± 0.0001
世代 1	5,000,000	270 ± 1	0.9772 ± 0.0003
世代 2	10,000,000	275 ± 1	0.9644 ± 0.0003
世代 4	20,000,000	277 ± 1	0.9671 ± 0.0003
世代 8	40,000,000	283 ± 1	0.9372 ± 0.0003
世代 16	80,000,000	303 ± 1	0.8453 ± 0.0003
世代 32	160,000,000	358 ± 1	0.7999 ± 0.0003
世代 64	320,000,000	367 ± 1	0.7913 ± 0.0003
世代 128	640,000,000	364 ± 1	0.7884 ± 0.0003
世代 256	1,280,000,000	369 ± 1	0.8355 ± 0.0003

9.2 実験 2

この実験では，世代の異なる 10 種の Self を用いて対戦を行い，性能を比較する．対戦は，10 種のプレイヤから重複を許さずランダムに 5 つ選出し，1 セット行うことを繰り返した．

結果を表 8 に示す．世代を重ねるごとに期待総獲得点がおおむね単調増加しているが，世代 64 から世代 128 のように減少してしまっているところも見られる．学習データに対する誤差を見たところ，世代 64 から世代 128 の間で誤差が跳ね上がる現象が見られており，それが原因であると考えられる (この現象の原因は特定できていない)．まだ強さの向上の頭打ちが起きている様子は見られないため，学習反復回数を増やすことでより強くなる可能性がある．

$RMSE$ を見ると，期待総獲得点が高いプレイヤほど $RMSE$ が小さい傾向が見られるが，世代 256 では増加している．したがって, 順位予測精度と期待総得点にはある程度の相関はあることが分かる．

10. まとめ

UECda2016 優勝プレイヤ Glicine を用いた強化学習と，ランダムプレイヤから開始した自己対戦による強化学習の 2 つのアプローチで，深層学習を適用して状態価値を学習し，グリーディ方策を作成した．

本研究で作成したグリーディ方策はいずれも Glicine を超える強さにはならなかったが，期待総得点差は大きくは開いておらず，Glicine に準ずる強さのグリーディ方策を作成できたと結論づける．なお，本研究の成果を用いたプレイヤは，UECda2017 の無差別級で 2 位の成績を収めた [*2]．

状態価値の推定に畳込みニューラルネットワークと全結合ニューラルネットワークを用いて比較した結果，畳込みニューラルネットワークの方が高い性能を示した．また畳込みニューラルネットワークの規模を大きくするほど強くなる傾向が見られた．

2 つのアプローチを比較した場合，自己対戦によるアプローチの方が強くなった．また，プレイヤの行動履歴をニューラルネットワークの入力に加える場合とそうでない場合では強さに差は見られず，Glicine を用いたアプローチでは Glicine に小さな確率でランダム行動を取らせる場合とそうでない場合でも差が見られなかった．これらはいずれも学習データが十分でないことが原因の一つと考えられる．

自己対戦による強化学習のアプローチでは，世代を重ねるごとに強さが向上していった．さらに，まだ強さ向上の頭打ちが起きている様子は見られないため，学習反復回数を増やすことでより強くなる可能性がある．

本研究では，価値関数の学習と価値に基づくグリーディ方策のみで Glicine に準ずる強さを獲得できたことから，他ゲームの先行研究のように別の探索手法 (例えばモンテカルロ法) や方策の学習，相手の手札推定と組み合わせたり手札交換アルゴリズムの改良を行うことで Glicine 以上の強さのプレイヤを作成することが期待できる．

[*2] UECda2017 大会結果 http://www.tnlab.inf.uec.ac.jp/daihinmin/2017/result2017.html (last access, 2018)

参考文献

[1] 須藤郁弥，　篠原歩：モンテカルロ法を用いたコンピュータ大貧民の思考ルーチン設計，第 1 回 UEC コンピュータ大貧民シンポジウム (2009).
[2] 須藤郁弥，成澤和志，　篠原歩：UEC コンピュータ大貧民大会向けクライアント「snowl」 の開発，第 2 回 UEC コンピュータ大貧民シンポジウム (2010).
[3] 小沼啓，本多武尊，保木邦仁，西野哲朗：コンピュータ大貧民に対する差分学習法の応用，研究報告ゲーム情報学 (GI)，Vol. 2012, No. 1, pp. 1–4 (2012).
[4] 大渡勝己，田中哲朗：方策勾配を用いた教師有り学習によるコンピュータ大貧民の方策関数の学習とモンテカルロシミュレーションへの利用，研究報告ゲーム情報学 (GI), Vol. 2016, No. 10, pp. 1–8 (2016).
[5] Sutton, R. S.，Barto, A. G.，貞芳三上，雅章皆川：強化学習，森北出版 (2000).
[6] Tesauro., G.: Temporal difference learning and TD-Gammon, *Communications of the ACM*, Vol. 38, No. 3, pp. 58–68 (1995).
[7] Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M. and Bowling, M.: DeepStack: Expert-level artificial intelligence in heads-up no-limit poker, *Science*, Vol. 356, No. 6337, pp. 508–513 (2017).
[8] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. and Hassabis, D.: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (2017).
[9] Zobrist, A. L.: A new hashing method with application for game playing, *ICCA Journal*, Vol. 13, No. 2, pp. 69–73 (1970).
[10] Glorot, X. and Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256 (2010).
[11] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding, *arXiv:1408.5093* (2014).