3 × 3盤面の2048の完全解析と強化学習の研究

山下修平[1,a)]　金子知適[1,b)]　中屋敷太一[1,c)]

概要：1 人用ゲーム 2048 は強化学習手法の性能の評価の題材として適している．これまで行われてきた得点による評価に加えて，最適方策との比較を定量的に行えるとより良い．そこで本研究では 2048 を 3 × 3 盤面に縮小したゲームであるミニ 2048 を考案した．ミニ 2048 はオリジナルの 2048 の興味深い性質を受け継ぎつつ，盤面の小ささから完全解析を行うことができる．完全解析ではミニ 2048 の状態数，最適方策に従ったときに得られる得点などの指標に加えて遷移モデルを少し変更したときの変化についても調査する．さらにミニ 2048 において Stochastic MuZero を簡略化した手法でエージェントを学習させ，その性能を最適方策との一致率で評価した．最後に最善手と最悪手の期待得点の差が大きい盤面を集中的に学習することがエージェントの学習に重要なことを示した．

Strongly Solving 2048 on 3 × 3 Board and Performance Evaluation of Reinforcement Learning Agents

Shuhei Yamashita[1,a)]　Tomoyuki Kaneko[1,b)]　Taichi Nakayashiki[1,c)]

Abstract: The single-player game 2048 is an interesting target for the evaluation of reinforcement learning methods. While one usually measures the average scores to show the learning efficiency of a method, it would be beneficial if one could additionally show the distance to an optimal policy. Toward this end, this paper presents *mini2048*, a small variant of 2048 with a 3x3 board. While mini2048 inherits interesting properties from the original 2048, we can strongly solve the game thanks to its smaller board size. We report the statistics of the game and the score achieved by the optimal strategies, including their changes along with a slight modification of transition dynamics. Moreover, we trained agents with a simplified version of Stochastic MuZero in mini2048 and evaluated its effectiveness by the rate of agreement with optimal strategies. Finally, we showed it is important for agents to intensively learn such a state that the difference in the expected return between the best and worst actions is limited.

1. はじめに

2048 というゲームは， ルールは単純だが高得点を獲得することは難しいパズルゲームで多くの研究の対象となってきた．最近では Stochastic MuZero [1] が先行研究 [2][3] を上回る得点を獲得するプレイヤの学習に成功した．これらの研究ではプレイヤの獲得する得点を最大化することに注目している．一方でそれらのプレイヤがゲームのどういった場面で悪手を打つのか，強いプレイヤと弱いプレイヤで判断が分かれるのはどのような盤面なのかといった，プレイヤの動作とゲームに関する詳細な研究は知られていない．仮にすべての盤面における最善手が分かっていれば，プレイヤの指し手と比較することでプレイヤの強さを定量的に評価することができる．しかし 2048 は状態空間の大きさからすべての盤面で最善手を解析し計算することは難しい．そこで本研究では 4 × 4 盤面上で行われている 2048 を 3 × 3 に盤面を縮小したゲーム (ミニ 2048) の完全解析を行った．完全解析ではミニ 2048 の状態数，最適方策に従ったときに得られる得点などの指標に加えて遷移モデルを少し変更したときの変化についても調査した．さらにミニ 2048 において Stochastic MuZero を簡略化した手

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[a)] yamashita-shuhei@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org
[c)] tnakayashiki@g.ecc.u-tokyo.ac.jp

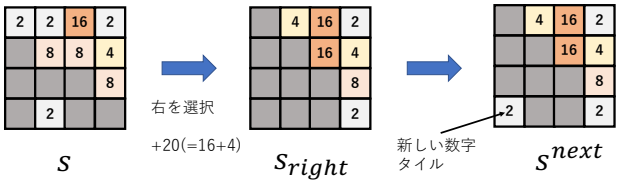

図 1　状態遷移の例

法でエージェントを学習させ，その性能を最適方策との一致率で評価した．最後に最善手と最悪手の期待得点の差が大きい盤面を集中的に学習することがエージェントの学習に重要なことを示した．

2. 2048

2048 は 4×4 の 16 マスの盤面上で遊ばれるゲームである． 初期盤面は 16 マスの内どこか 2 マスに 2 か 4 の数字タイルが置かれた盤面である．プレイヤは上下左右いずれかの方向を選択する． すると各数字タイルはその方向に移動する．移動した結果，2 つの同じ数字のタイルが衝突するとこれらは合体してその合計の数のタイルへ変化し， プレイヤはその数値を報酬として得る．その後，空白のマスから等確率で選択されたある 1 マスに 90% の確率で 2 のタイルが， 10% の確率で 4 のタイルが置かれる．プレイヤはいずれかのタイルが移動または衝突することで，盤面が変化するような方向しか選択することができない．いずれの方向も選択できなくなるとゲームは終了する．

これ以降プレイヤが行動を選択する盤面の状態のことを *state* と呼ぶ．また行動を選択し盤面上のタイルが移動した後，新たなタイルが出現する直前の盤面のことを *afterstate* と呼ぶ．図 1 は状態遷移の例を示したものである．state s から右を選んだことで afterstate s_{right} に遷移する．このとき横に並ぶ 8 のタイルと 2 のタイルがそれぞれ合体することで $8 \times 2 + 2 \times 2 = 20$ 点を獲得する．さらに s_{right} から 2 のタイルが左下に新たに出現し，s^{next} に遷移した．ここで行動を決めた場合の state から afterstate への遷移と獲得する得点は決定的であり，afterstate から次の state への遷移が確率的であることを強調しておく．

3. マルコフ決定過程と強化学習

本節ではマルコフ決定過程 (MDP) と強化学習の問題設定について簡単に説明し，今後の議論のために 2048 と MDP の対応を説明する．詳細は文献 [4] などを参照されたい．MDP はエージェントと環境のやり取りを抽象的に記述するための枠組みで，以下の 4 つの要素から構成される．

- 状態集合 S
- 行動集合 A
- 状態遷移関数 $p : S \times A \times S \to [0, 1]$
- 報酬関数 $r : S \times A \times S \to \mathbb{R}$

エージェントは方策と呼ばれる，ある状態からそれぞれの

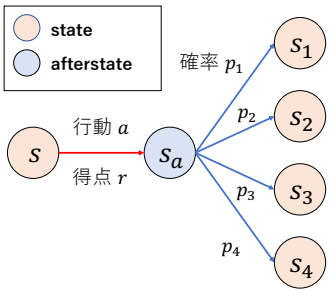

図 2　state と afterstate の遷移関係

行動を選択する確率の分布に従って行動を決定する．一般に方策を π と表記し，状態 s で行動 a を選択する確率は $\pi(a|s)$ と表せる．方策 π に従ったときの状態 s の価値は $v_\pi(s)$ で表され，$v_\pi(s) = \mathbb{E}_\pi[\Sigma_{k=0}^{T} R_{t+k+1} | S_t = s]$ として定義される．すなわち $v_\pi(s)$ とは方策 π に従ったときに状態 s から獲得する報酬の合計の期待値である．また状態 s から行動 a を選択して，その後方策 π に従い続けた場合に獲得する報酬の合計の期待値を行動価値と呼び，$q_\pi(s, a)$ と表される．

最適方策 π_* とはエージェントが獲得する報酬の合計の期待値を最大化するような方策のことである．π_* はすべての状態 $s \in S$ で $v_{\pi_*}(s) = \max_\pi v_\pi(s)$ を満たし，すべての状態 $s \in S$，行動 $a \in A(s)$ で $q_{\pi_*}(s, a) = \max_\pi q_\pi(s, a)$ を満たす．v_{π_*} と q_{π_*} はそれぞれ最適状態価値関数，最適状態行動価値関数と呼ばれ，以下に示す関係がある．

$$v_{\pi_*}(s) = \max_{a \in A(s)} q_{\pi_*}(s, a) \tag{1}$$

$$q_{\pi_*}(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \tag{2}$$

式 2 は状態 s から行動 a をとって獲得する報酬，遷移する次の状態が一般に確率的に決まるため期待値をとっている．また γ は割引率と呼ばれ，s から見て将来の価値をどれくらい考慮するかを調整するパラメータである．強化学習は最適方策 π_* を見つけるための手法である．

3.1 2048 と MDP

2048 は MDP の枠組みに沿ったゲームである．すなわち状態 (state) s から行動 (上下左右のいずれか) a をとると，報酬 (得点) r を獲得して次の状態 (state) s_{next} に遷移する．式 1 の最適状態価値 $v_{\pi_*}(s)$ は state s から最善手を選び続けた際に獲得する得点の総和の期待値のことである．また 2 節で説明したように，state s から行動 a を選択して遷移する afterstate s_a と獲得する得点は決定的である (図 2 参照)．さらに afterstate s_a から次の state s_{next} への遷移で得点は獲得できない．そこで方策 π に従ったときに state s から獲得できる得点の総和の期待値を $V_\pi^d(s)$，方策 π に従ったときに afterstate s_a から獲得できる得点の総和の期待値を $V_\pi^c(s_a)$ と表すことにする (d と c はそれぞれ decision，chance の頭文字．A.1 節を参照)．

これらを考慮して，式 1，2 は以下のように書ける．

$$v^d_{\pi_*}(s) = \max_{a \in A(s)} q_{\pi_*}(s,a) \tag{3}$$

$$q_{\pi_*}(s,a) = r(s,a) + v^c_{\pi_*}(s_a) \tag{4}$$

$$v^c_{\pi_*}(s_a) = \mathop{\mathbb{E}}_{s_{\text{next}} \in \mathcal{T}(s_a)} v^d_{\pi_*}(s_{\text{next}}) \tag{5}$$

ただし，$\mathcal{T}(s_a)$ は afterstate s_a から遷移可能な次の state の集合とする．また 2048 は有限ステップで必ず終了するゲームであるため，割引率 γ は 1 とした．

$v^d_{\pi_*}(s)$, $v^c_{\pi_*}(s_a)$ をそれぞれ state s の最適価値，afterstate s_a の最適価値と呼ぶことにする．state s における最善手 a_* は $\arg\max_{a \in A(s)} q_{\pi_*}(s,a)$ として決定される．よって 5 節ではすべての state，afterstate の最適価値 $v^d_{\pi_*}$ と $v^c_{\pi_*}$ を計算することで最善手を求める．

4. 先行研究

2048 で良い手を選ぶための探索手法としては Expectimax 探索やモンテカルロ木探索を使った手法などが考えられる．強化学習を用いた手法としては TD-afterstate 学習 [2] が有名であり， それに必要な盤面の評価関数として N-tuple networks を使う手法 [2] やニューラルネットワークを使う手法 [3] が提案されている．以下では探索と強化学習を組み合わせて 2048 の学習に成功した Stochastic MuZero を紹介する．

4.1 Stochastic MuZero

Antonoglou らが提案した Stochastic MuZero [1] は，2048 において既存手法を上回る得点を獲得することに成功した．Stochastic MuZero は最適方策 $\pi*$，state，afterstate の最適価値 $v^d_{\pi*}$，$v^c_{\pi*}$ を推定するニューラルネットワークを学習する．他にも環境の遷移モデルを推定するニューラルネットワークを学習するが，本稿では環境の遷移モデルは既知としてその学習は扱わないため説明は省略する．以下では環境の遷移モデルを既知として方策と価値のみを Stochastic MuZero と同様に学習する方法を説明する．本稿ではこれを便宜上 2048AlphaZero と呼ぶことにする．ここでは本稿の理解に必要な 2048AlphaZero の学習の概要を必要な範囲で簡単に説明する．詳細は文献 [1] を参照されたい．

2048AlphaZero は，他の強化学習の手法と同様に，ゲームをプレイし (以下 selfplay と呼ぶ)，その経験から方策と価値関数を学ぶ．方策と価値関数はニューラルネットワークで表現する．

まず方策ニューラルネットワーク $\hat{\pi}$ の学習方法を説明する．selfplay 中のそれぞれの state において，モンテカルロ木探索 (MCTS) が行う．MCTS は有望な盤面を深く探索することによって，正確な先読みを可能にする探索手法である．MCTS を state s から開始すると，s で選択可能な行動のうち有望な手を優先的に探索する．探索によって有望な手は何度も訪問される一方，有望でない手はあまり訪問されない．s で選択可能なそれぞれの行動が探索中に訪問された回数の分布を訪問分布と呼ぶ．方策ニューラルネットワーク $\hat{\pi}$ は s における行動選択の確率分布 $\hat{\pi}(\cdot|s)$ が，訪問分布を近似するように学習する (図 3 参照)．MCTS で探索することでそれぞれの行動の良さを正確に推定し，これを方策ニューラルネットワーク $\hat{\pi}$ の教師に用いている．

次に価値ニューラルネットワーク $\hat{v}^d$，$\hat{v}^c$ の学習方法を説明する．selfplay 中の state s_t について，s_t から MCTS による行動の選択を繰り返してゲームをプレイする．この結果，たとえば state s_t 以降に実際に獲得した得点 G_t を $\hat{v}^d(s_t)$ の学習ターゲットとして用いることができる (ただし実際には分散が大きいため n-step return [4] などを用いる)．afterstate の価値の学習についても同様である (図 4 参照)．ゲームオーバー付近の価値が 0 に近い state，afterstate から価値ニューラルネットワークは学習し，徐々にゲーム序盤の state，afterstate まで伝播すると考えられる．

6 節では MCTS の探索の深さとプレイヤの性能に関する調査を行うので，MCTS についていくつか補足しておく．MCTS はシミュレーションという単位で探索を行い，シミュレーション回数が大きくなるほど探索は深くなるため正確な評価が可能となる．よって一般にシミュレーション回数を大きくするほど，より良い手を選ぶことができると考えられる．MCTS は selfplay 時に学習データを生成するのに使用されるだけでなく，学習後のプレイヤの強さを評価する際にも使用されることが多い．この際 MCTS のシミュレーション回数を増減させることでプレイヤの強さを調整することができる．

また selfplay 中に選択する行動は MCTS による各行動への訪問分布から決める．たとえば行動 $a_1 \dots a_k$ についてそれぞれ訪問回数が $N_1 \dots N_k$ だったとする．このとき次に選ぶ行動は T を温度パラメータとして，$N_1^{\frac{1}{T}} \cdots N_k^{\frac{1}{T}}$ を正規化した確率分布に従って選択する．$T = 1$ のときは単に訪問回数を正規化した確率で行動が選ばれ，$T = 0$ のときは最も訪問回数が多い行動が選ばれる．学習初期はなるべく多くの盤面を学習するために $T = 1$ で行動を選び，学習が進むに従って徐々に 0 に近づける．

また Stochastic MuZero が提案した 2048 などの確率的な環境における MCTS が，囲碁や将棋などの確定ゲームで用いられる MCTS とは少し異なる点を A.1 節に記したので (必要に応じて) 参照されたい．

5. 3 × 3 盤面の 2048 の完全解析

4 節で説明した 2048 に関する先行研究は，高得点を獲得するプレイヤを作成することに焦点が置かれ，そのプレイヤの動作に関する詳細な研究は少なかった．もしある状態における最善手がわかっていれば，強化学習プレイヤの様々

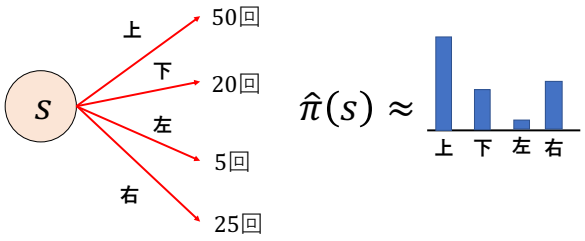

図 3　方策ニューラルネットワーク $\hat{\pi}$ の学習

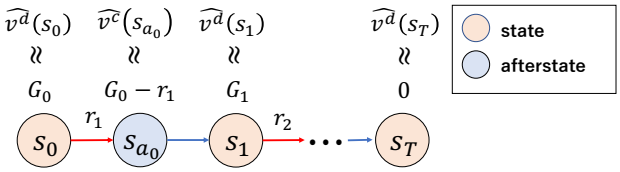

図 4　価値ニューラルネットワーク $\hat{v}$ の学習

な評価を行うことができる．たとえば中屋敷ら [5] は完全解析したどうぶつしょうぎ [6] を題材として AlphaZero [7] の手法の調査を行った．

そこで 2048 を簡単にしたゲームであるミニ 2048 を完全解析し，強化学習手法で訓練したプレイヤの手と完全解析結果を比較し調査することを考える．ミニ 2048 と通常の 2048 の違いは盤面のサイズのみで，それ以外のルールはすべて同じものとする．

解析はゲーム開始盤面である初期 state から到達可能な state の数え上げと，それらの state の最適価値の計算という 2 つのパートから成る．以下ではミニ 2048 の state の数え上げと最適価値の計算に関する具体的な方法と，その結果について通常の 4×4 盤面の 2048 との類似性を意識して考察する．さらに関連していくつか興味深い話題を取り上げる．

5.1　初期状態から到達可能な状態の列挙

ミニ 2048 は盤面が 3×3 であるため，作り得る最大の数字タイルは $2^{10} = 1,024$ のタイルである．よって実際に到達可能な state はかなり多く見積もっても 11^9 以下であり，さらに盤面の回転・反転に関して対称な状態を同一視すれば $11^9/8 \approx 3$ 億以下であることがわかる．これは計算時間・メモリの観点ともに十分容易に計算可能な量である．

まずキューを用いた幅優先探索で初期 state (ゲームの開始盤面) から到達可能なすべての state を列挙する．キューの先頭の state s を取り出し，s から 1 ステップ後に遷移可能な state s_{next} を列挙する． このうちまだ訪問済みでない state のみをキューに追加する．キューが空になったら探索を終了する．疑似コードを Algorithm 1 に示す．

この結果，初期 state から到達可能な state は全部で 48,713,519 通り，そのうち終端 state は 7,388,502 通りであることがわかった．また初期 state から到達可能な afterstate は全部で 31,431,370 通り見つかった．ただしいずれも対称な盤面を 1 つとして扱っている．多めに見積もった 3 億の上限に対して実際はその 5 分の 1 程度の state が到達可能だった．通常の 2048 の最大タイルが 2^{17} であることを踏まえて，同様に上限を見積もると $18^{16}/8 \approx 1.5 \times 10^{19}$ となる．ミニ 2048 の解析結果の類推から，上限の 10 分の 1 程度しか到達可能でないとしても，数え上げるのは計算時間・記憶容量の観点から難しいことが推測される．

Algorithm 1 すべての到達可能な state の列挙

```
que ← 空のキュー
初期 state s_0 を訪問済みとして記録する
que.push(s_0)

while que ≠ ∅ do
  s ← que.pop()
  for each s から遷移可能な次の状態 s_next do
    if s_next が非終端状態かつ s_next が未訪問 then
      s_next を訪問済みとする
      que.push(s_next)
```

表 1　ミニ 2048 の盤面上の最大数字タイルとその時の得点と手数の大まかな目安

タイル	得点	手数
64	320	30
128	750	60
256	1500	120
512	4000	240
1024	9000	480

5.2　状態価値の計算

5.1 節で列挙した state の最適価値を式 3，5 に従って計算する．終端 state s_T の最適価値は 0 であるから，終端 state から再帰的に計算することができる．疑似コードを Algorithm 2 に示す．

最適価値を計算した結果，初期盤面の価値は配置に関わらずいずれも約 5,570 点となった．この点数は最善手を選び続ければ平均的に 512 のタイルまでは到達できることを意味する (表 1 を参照)．512 を $2^{\text{盤面の大きさ}}$ と解釈して 2048 に当てはめると，$2^{16} = 65,536$ のタイルまで平均的に到達できることになる．65,536 のタイルに到達した時点で約 900,000 点の得点を獲得できるが，これは Stochastic MuZero が示した約 500,000 点を大きく上回る点数である．真の期待値がこれよりも低い可能性は大いにあるが，今後 Stochastic MuZero を上回る得点の獲得に成功する手法が提案されることも十分に考えられるだろう．

5.3　ゲームの状態遷移における確率を変更した場合の調査

すでに述べたように 2048 は state s から行動 a をとると afterstate s_a に決定的な遷移を行う．さらに s_a から次の状

Algorithm 2 到達可能な状態の価値の計算

que ← 空のキュー
V_d ← 状態価値を保持するハッシュテーブル
V_c ← *afterstate* の価値を保持するハッシュテーブル
que に 1 で列挙したすべての状態を追加

while que $\neq \emptyset$ **do**
  $s \leftarrow$ que.pop()
  if s が終端状態 **then**
    $V_d(s) = 0$
    continue
  s_has_unsettled_action ← false
  max_value $= -\inf$
  for each s から選択可能な行動 a **do**
    $s_a, r(s,a) \leftarrow$ do_action(s,a)
    s_a_has_unsettled_transition ← false
    for each $s_{\text{next}} \in \mathcal{T}(s_a)$ **do**
      if $V_d(s_{\text{next}})$ が未計算 **then**
        s_a_has_unsettled_transition ← true
    if s_a_has_unsettled_transition **then**
      s_has_unsettled_action ← true
    else
      $V_c(s_a) = \mathbb{E}_{s_{\text{next}} \in \mathcal{T}(s_a)} V_d(s_{\text{next}})$
      max_value $= \max(r(s,a) + V_c(s_a), \text{max\_value})$
  if s_has_unsettled_action **then**
    que.push(s)
  else
    $V_d(s) =$ max_value

態 s_{next} へは確率的な遷移をする．ミニ 2048 は afterstate から次の state に遷移する際に，本来の 2048 と同様に等確率で選ばれた空きマスに 90%の確率で 2 のタイルが，10%の確率で 4 のタイルが新たに出現するものとした．

プレイヤが高得点を獲得するには，afterstate s_a から state s_{next} への遷移確率を考慮して行動を選択する必要がある．Stochasic MuZero は環境の遷移モデルも学習することを 4.1 節でふれた．学習した遷移モデルが環境の真の遷移モデルと大きく異なると，方策や価値の学習も正しく進まないと考えられる．よってミニ 2048 の環境の遷移モデルが本来とは異なる場合に最適価値がどう変化するかを調べておくことは，今後 Stochastic MuZero の調査をする上で意義のあることだと考えられる．そこで本節では 2 と 4 の出現確率を変更した環境における最適価値を計算した．

表 2 に 4 の出現確率を 5%刻みで増減させて計算した結果を示す．初期配置の state の最適価値をそれぞれの配置でゲームが始まる確率で重み付け平均をとった値を，「ゲームの期待値」として表に記載している．2 と 4 の出現確率が偏っているほど期待値は高い傾向があるが，最も期待値が低いのが 50%の時というわけではないのが興味深い．

6. 2048 と強化学習

Stochastic MuZero は MCTS による先読み (探索) を学習に利用することで，2048 において既存手法の性能を上回る成果を出した．本節ではミニ 2048 においても同様に

表 2　4 の出現確率を増減させたときのゲームの期待値

4 の確率	ゲームの期待値	4 の確率	ゲームの期待値
0.00	7172.00	0.55	3206.00
0.05	6161.17	0.60	3171.24
0.10	**5468.49**	0.65	3165.36
0.15	4932.54	0.70	3194.44
0.20	4515.42	0.75	3269.18
0.25	4182.44	0.80	3399.20
0.30	3919.20	0.85	3607.78
0.35	3704.44	0.90	3993.30
0.40	3531.46	0.95	4938.20
0.45	3390.19	1.00	14344.00
0.50	3278.70		

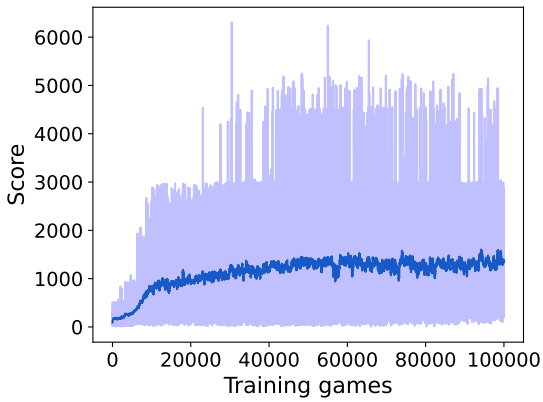

図 5　ミニ 2048 における Afterstate PPO の学習の様子．薄い線はゲームごとの得点，濃い線はその移動平均．

MCTS を利用した方策と価値関数の学習が，従来の探索を行わない手法に比べて有効であることを確認する．その際に学習時の MCTS のシミュレーション回数が学習に与える影響を調査した．また本研究では最後に 5 節のミニ 2048 の解析結果を元に，最善手と最悪手の価値の差が大きい state の学習が強いプレイヤの作成に重要であることを示す．

6.1 探索を行わない手法の評価

まず MCTS などの探索を行わない 2048 の既存の強化学習手法の 1 つをミニ 2048 で評価する．本稿では文献 [8] で提案された Afterstate PPO を実験した．Afterstate PPO は有力な方策ベースな強化学習手法の 1 つである Proximal Policy Optimization (PPO) [9] を 2048 においても学習が進むように改良したものである．詳細は論文を参照されたい．図 5 に Afterstate PPO を 100,000 ゲーム学習させた様子を示す．1000 点付近までは順調に学習が進むが，その後停滞している．表 1 より 128 のタイルを作るところまでは学習が進むが，次の 256 のタイルには届いていない．

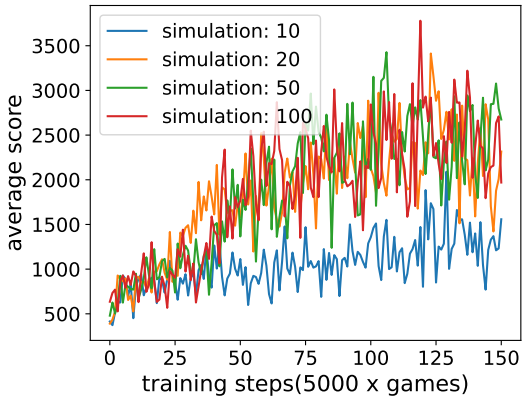

図 6　異なるシミュレーション回数の MCTS で学習させたプレイヤの学習曲線．ただし評価時はいずれもシミュレーション 50 回でプレイさせた．

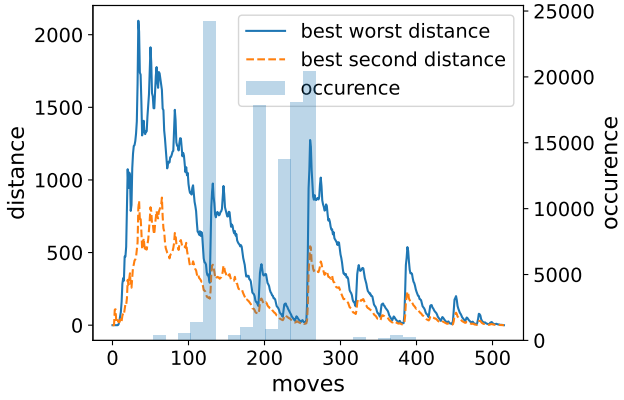

図 7　手数ごとの最善手と最悪手および次善手の価値の差の平均のグラフとパーフェクトプレイヤがゲームオーバーになる手数のヒストグラム

6.2　MCTS の探索による方策と価値の学習

MCTS の訪問分布を教師とした方策の学習，MCTS で選択した行動で生成したエピソードによる価値の学習をミニ 2048 を対象に行う．ここでは selfplay 時の MCTS のシミュレーション回数が，学習にどう影響を与えるか調査する．MCTS のシミュレーション回数が大きいほど，良質な学習データが集まり学習に良い影響を与えると考えられるがどうだろうか．

シミュレーション 10 回，20 回，50 回，100 回の MCTS で学習させたときの経過を図 6 に示す．ただし学習では 5,000 ゲームごとに生成した教師データを元にニューラルネットワークを更新した．図 6 はニューラルネットワークを更新するたびにシミュレーション 50 回，温度 0 の MCTS で 10 回ゲームをプレイさせ，その平均得点をプロットした．図から分かるように，シミュレーション回数が 10 回と少ないと学習は伸び悩んだ．一方で，シミュレーション 20 回以上では学習に大きな差はないことがうかがえる．Schrittwieser らは文献 [10] で Atari ゲームにおいて selfplay 時の MCTS のシミュレーション回数がプレイヤの学習に与える影響を調査した．この結果着手の選択肢が多いゲームである囲碁とは異なり，Atari ゲームではシミュレーションを一定以上増やしても学習の進みはあまり変わらないことを報告している．2048 は行動の選択肢が常に高々 4 つであるため，少ないシミュレーション回数の MCTS でも効率的に学習できると思われる．

次節ではこれらの結果とミニ 2048 の完全解析結果を踏まえて，効率的に学習を行うための工夫を考察する．

6.3　効率的な学習の考察と学習が難しい盤面

6.1 節，6.2 節の実験からミニ 2048 の学習において，ある程度の段階までは MCTS におけるシミュレーション数はそれほど必要ないと考えられる．2048 はゲームが進行するにつれて大きな数字のタイルが盤面を占めるようになり，判断を誤るとゲームオーバーになる盤面が増える．そのためゲーム序盤の簡単な盤面では MCTS による探索を節約し，ゲームが進み複雑化するにつれて精密な探索を行うことが考えられる．

本研究では最善手の判断が難しい state についてより詳細に探るため，最善手と最悪手の価値の差に注目した．すなわち state s の最善手を a^*，最悪手を a' として，$d(s) = q_{\pi_*}(s, a^*) - q_{\pi_*}(s, a')$ に注目する (任意の非終端 state について常に 2 つか 4 つの行動の選択肢がある)．state s は $d(s)$ が大きいほど，良い手と悪い手がはっきりしていると言える．逆に $d(s)$ が 0 に近いほど最善手と最悪手に大きな違いがなく，判断が難しい state と言える．

図 7 はそれを示したものである．すべての state を到達に必要な手数 (step) ごとに分類し，それぞれの手数で $d(s)$ の平均をプロットしたものを *distance* として折れ線グラフで示している (点線は最善手と次善手の差)．さらに解析で得られた常に最善手を選ぶプレイヤ (パーフェクトプレイヤ) を 100,000 回プレイさせて，どの手数でゲームオーバーになったかをヒストグラムにして重ねた．図からわかるように，ゲームオーバーになるのは $d(s)$ が 0 に近い手数の state であることが多い．またゲームが進むにつれて $d(s)$ は小さくなる傾向があるが，120 手付近や 260 手付近では大きく増加している．これは 512 や 1024 などの大きな数字タイルを作ることができると，一時的にゲームの難易度が下がることを示している．

6.2 節で訓練したそれぞれのニューラルネットワークで異なるシミュレーション回数の MCTS で 10 回ゲームをプレイさせた．プレイ中に現れたすべての state について，プレイヤの行動と最善手の一致率を調べたのが表 3 である．いずれのプレイヤも $d(s)$ が小さな state の正答率が悪いことがわかる．これらからミニ 2048 の学習では $d(s)$ が小さ

な state の方策や価値を，正確に推定できるようになることが重要だと考えられる．これを踏まえた上で 2048 の効率的な学習を提案することを今後の課題としたい．

7. まとめ

本稿では，まず 2048 を 3×3 に縮小したゲームであるミニ 2048 の完全解析を行った．ミニ 2048 の解析を通して，本来 4×4 盤面上で行われている 2048 にも共通するゲームの性質を明らかにした．さらに現在 2048 をプレイする AI 研究において有力な手法である Stochastic MuZero が用いる MCTS に関する調査をミニ 2048 を対象に行った．この結果，学習時の MCTS のシミュレーション回数を大きくすることが，必ずしも学習を大幅に加速させるわけではなく，少ないシミュレーション回数でも学習が進むことを実験で明らかにした．また完全解析により得られた最適方策の指し手と比較して，強いプレイヤの作成には最善手と最悪手の価値の差が大きな盤面で正しい判断を行えるように学習することが重要であることを示した．

付　　録

A.1 確率的な環境における MCTS

Stochastic MuZero で用いられた MCTS の探索木は decision ノードと chance ノードと呼ばれる 2 種類のノードから成る．decision ノードはプレイヤが行動の決定を行う state に対応し，chance ノードは afterstate に対応する．decision ノードと chance ノードは探索の深さに沿って交互に並んでおり，また根ノードは常に decision ノードである．あるノード s から選択可能な行動 a について対応するエッジ (s,a) が存在する．探索木 t のエッジ (s,a) について $[N(s,a),P(s,a),\hat{Q}(s,a),R(s,a)]_t$ という 4 つの値を保持する．それぞれエッジ (s,a) の訪問回数，事前確率，探索による価値の推定値，即時報酬に対応する．MCTS はこれらを用いて以下の 4 つのステップを繰り返す．

(1) 選択 : 根ノードから有望なエッジをたどり，探索木の末端の葉ノード s_l を見つける．decision ノード s_d は以下の PUCT 式が最大となる行動 a^k を選んで子 chance ノードを選択する．

$$a^k=\mathrm{argmax}\{\hat{Q}(s_d,a)+P(s_d,a)\frac{\sqrt{\Sigma_b N(s_d,b)}}{1+N(s_d,a)}[c_1+\log\left(\frac{\Sigma_b N(s_d,b)+c_2+1}{c_2}\right)]\} \tag{A.1}$$

ただし c_1，c_2 は定数である．PUCT 式は最も価値の推定値が高いノードと訪問回数が少ないノードに対して高い値を示す．一方 chance ノード s_c は $P(s_c,a)$ に基づいて確率的に子 decision ノードを選択する．すなわち chance ノードからは実際に遷移する確率の高い decision ノードが優先的に選択される．

(2) 展開 : 選択で選ばれた葉ノード s_l をゲームのルールに即して展開する．s_l が decision ノードだった場合は，s_l に対応する state から遷移可能な afterstate に対応する chance ノードを子に付け加える．s_l が chance ノードだった場合は，s_l に対応する afterstate から遷移可能な state に対応する decision ノードを子に decision ノードを付け加える．

(3) 評価 : 葉ノード s_l をニューラルネットワークで評価する．s_l が decision ノードだった場合は方策 $\pi(\cdot|s_l)$ と価値の推定値 $\hat{v_d}(s_l)$ を得る．s_l が chance ノードだった場合は $P(s_l,a)$ は環境の遷移確率として，価値の推定値 $\hat{v_c}(s_l)$ のみをニューラルネットワークから得る．

(4) 逆伝播 : $\hat{v_d}(s_l)$, $\hat{v_c}(s_l)$ を葉ノードから親ノードに至るまで伝播させる．すなわち葉ノードから親ノードに至る各ノード s について，s から s_l までに獲得する即時報酬の和と $\hat{v_d}(s_l)$ (もしくは $\hat{v_c}(s_l)$) の和を G_s とする．このとき，$\hat{Q}(s,a)=\frac{N(s,a)\cdot\hat{Q}(s,a)+G_s}{N(s,a)+1}$, $N(s,a)=N(s,a)+1$ と更新する．

(1) から (4) の手順をまとめてシミュレーションと呼ぶ．シミュレーションの回数を大きくするほど正確な推定が可能になり，プレイヤはより良い手を選ぶことができる．

A.2 ミニ 2048 の完全解析の補足

A.2.1 実装上の補足

ミニ 2048 においてそれぞれのマスには 11 通りの値が考えられ，これが 9 マスある．よって 11^9 通りの盤面を区別する必要があるが，これは 64 ビット整数で一意に指定することができる．解析にはメモリ 32GB でプロセッサは AMD Ryzen 5 のマシンを用いた．今回はすべての state，afterstate の情報がメモリに収まったため，記憶容量に関する実装上の工夫は特に行なっていない．到達可能な state，afterstate の列挙は約 2 分 30 秒でプログラムが終了した．また到達可能なすべての state，afterstate の価値計算には約 5 時間を要した．

A.2.2 ミニ 2048 におけるミニマックス価値の計算

通常 afterstate から次の state への遷移は確率的に行われる．この遷移が常にプレイヤの得点を最小化するように行われる環境を考える．すなわち得点を最大化したいプレイヤと，得点を最小化したい環境の対戦ゲームのようなものを考えるということである．この場合，ある状態の価値はミニマックス法により求めることができる．5.2 節で説明した通常の環境における価値計算との相違は，afterstate の最適価値 $v^c_{\pi_*}(s_a)$ の計算方法のみで，式 A.2 に示すように遷移可能な次の状態 s_{next} の最適価値の最小値となる．

$$v^c_{\pi_*}(s_a)=\min_{s_{\mathrm{next}}\in\mathcal{T}(s_a)} v^d_{\pi*}(s_{\mathrm{next}}) \tag{A.2}$$

Algorithm 2 と同様に価値を計算した結果ゲームの期待値

表 3　最善手と最悪手の価値の差とプレイヤの正答率

プレイヤ	$0 < d(s) <= 500$	$500 < d(s) <= 1000$	$1000 < d(s) <= 1500$	$1500 < d(s)$
agent(10, 10)	0.54	0.47	0.72	0.71
agent(10, 100)	0.49	0.57	0.62	0.67
agent(10, 1000)	0.50	0.65	0.70	0.77
agent(20, 10)	0.47	0.61	0.73	0.73
agent(20, 100)	0.49	0.61	0.68	0.77
agent(20, 1000)	0.47	0.56	0.59	0.75
agent(50, 10)	0.53	0.60	0.67	0.79
agent(50, 100)	0.54	0.64	0.73	0.80
agent(50, 1000)	0.52	0.70	0.72	0.83
agent(100, 10)	0.50	0.62	0.77	0.76
agent(100, 100)	0.50	0.62	0.76	0.76
agent(100, 1000)	0.50	0.62	0.77	0.81

は 178.4 と，通常の確率的な環境での価値と比較すると大きく異なる値が得られた．afterstate から次の state への遷移がプレイヤを最大限に妨げるルールだと，プレイヤはいかなる手を選んでも得点はかなり小さいものとなる．

ミニマックス法により求めた状態価値をもとに次の手を選ぶプレイヤをミニマックスプレイヤと呼ぶことにすると，ミニマックスプレイヤは常に環境がプレイヤの妨げとなるような遷移を行うことを想定して行動を選択する．このミニマックスプレイヤが通常の環境において獲得できる得点の期待値 (状態価値) をこれまでと同様の方法で計算したところ，ゲームの期待値は 1363.43 となった．慎重すぎるために本来獲得できる点数をも見逃してしまっていると解釈できるだろう．2048 は一般には 1 人用で遊ばれるゲームだが，環境が担う新しい数字タイルの配置を別のプレイヤが行うことで対戦型ゲームに拡張することができる．ミニマックスプレイヤのように慎重に手堅く点を稼ぐ戦略や，相手のミスに期待して高得点を狙う戦略などが考えられ，プレイヤ同士の戦略の組み合わせによって新たなゲーム性が生まれることが期待できるだろう．

A.3　ニューラルネットワークの構造

6.2 節の selfplay による方策と価値の学習の実験設定を述べる．3×3 の盤面は 3×3，11 チャネルのテンソルにエンコーディングしてニューラルネットワークに入力される．この際，n 番目のチャネルは元の盤面で 2^n が存在する位置が 1，それ以外は 0 となるようにする．ただし 0 番目のチャネルは $2^0 = 1$ ではなく元の盤面で空白の位置に 1 がたつ．入力はカーネルサイズ 3×3，パディング 1，128 チャネルの Residual ブロック [11]10 層の中間層で処理される．中間層の出力は 2 つに分かれる．それぞれ畳み込み層 (Residual ブロックと同じ構造) の次に全結合層を適用して，方策と価値を出力する．またいずれの畳み込み層にも Batch Normalization を施した．学習率は 0.0003 とし，Adam を用いて学習を行なった．

参考文献

[1] Antonoglou, I. et al.: Planning in Stochastic Environments with a Learned Model, *International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=X6D9bAHhBQ1⟩ (2022).

[2] Szubert, M. and Jaśkowski, W.: Temporal difference learning of N-tuple networks for the game 2048, *2014 IEEE CIG*, pp. 1–8 (online), DOI: 10.1109/CIG.2014.6932907 (2014).

[3] Matsuzaki, K.: Developing Value Networks for Game 2048 with Reinforcement Learning, *Journal of Information Processing*, Vol. 29, pp. 336–346 (online), DOI: 10.2197/ipsjjip.29.336 (2021).

[4] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, The MIT Press, second edition (2018).

[5] 中屋敷太一，金子知適：どうぶつしょうぎを用いた AlphaZero の手法の調査，ゲームプログラミングワークショップ 2019 論文集，Vol. 2019, pp. 86–93 (2019).

[6] 田中哲朗：「どうぶつしょうぎ」 の完全解析，研究報告ゲーム情報学（GI），Vol. 2009, No. 3, pp. 1–8 (2009).

[7] Silver, D. et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, Vol. 362, No. 6419, pp. 1140–1144 (online), DOI: 10.1126/science.aar6404 (2018).

[8] 山下修平，金子知適：2048 への方策勾配法の適用，ゲームプログラミングワークショップ 2021 論文集，Vol. 2021, pp. 179–185 (2021).

[9] Schulman, J. et al.: Proximal Policy Optimization Algorithms (2017).

[10] Schrittwieser, J. et al.: Mastering Atari, Go, chess and shogi by planning with a learned model, *Nature*, Vol. 588, No. 7839, pp. 604–609 (online), DOI: 10.1038/s41586-020-03051-4 (2020).

[11] He, K. et al.: Deep Residual Learning for Image Recognition, *CoRR*, Vol. abs/1512.03385 (online), available from ⟨http://arxiv.org/abs/1512.03385⟩ (2015).