確率的ゲーム 2048 の強化学習の研究

山下修平

2023 年 12 月 14 日

目次

第 1 章

はじめに

第 2 章

2048

2.1　2048 のルールと用語説明

2048 は， Gabrirle Cirulli によって公開された 1 人用のパズルゲームである [1]．ゲームは 16 マスからランダムに選ばれた 2 マスに 2 か 4 の数字タイルが置かれた盤面から始まる．プレイヤが行うことは上下左右いずれかの方向を選択することである．プレイヤがある 1 つの方向を選ぶと， 盤面上のすべての数字タイルは選択した方向に向かってスライドして移動する．スライドする数字タイルは空きマスを通過し， 異なる数字タイルの直前か盤面の端で停止する．スライドして移動する際に 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字の 1 つのタイルへ変化し， プレイヤはその数値を得点として獲得する．そのため， ゲームには 2 の累乗の数字タイルしか現れない．図 2.1 にある盤面から上下左右を選択したときの， 数字タイルのスライドの仕方の具体例を示す．

数字タイルのスライド後， 空きマスから等確率に選択されたある 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．ゲームはプレイヤの行動による数字タイルのスライドと新たな数字タイルの出現を交互に繰り返して進行する．盤面上の数字タイルが市松模様のようになると， プレイヤが選択可能な行動がなくなったときにゲームは終了する (図 2.2 を参照)．

ここでプレイヤが行動を選択する盤面を**状態**， 行動を選択して新たな数字タイルが出現する直前の盤面を *afterstate* と呼ぶ．またプレイヤがゲームを開始する盤面を初期状態， ゲームが終了した盤面を終了状態と呼ぶ．図 2.4 に状態 s から afterstate s' を経由して， 次の状態 s_{next} に遷移する例を示す．

プレイヤの一般的な目標はゲームのタイトルが示す $2^{11} = 2048$ のタイルを完成させることだが， それ以降もゲームを続けることができる．

2.2　ゲームの進行上の性質

2048 はゲームの性質上， 状態から afterstate への遷移において盤面上の数字タイルの合計値は不変である (図 2.1 を参照)．盤面上の数字タイルの合計値は afterstate から次の状態への遷移においてのみ変

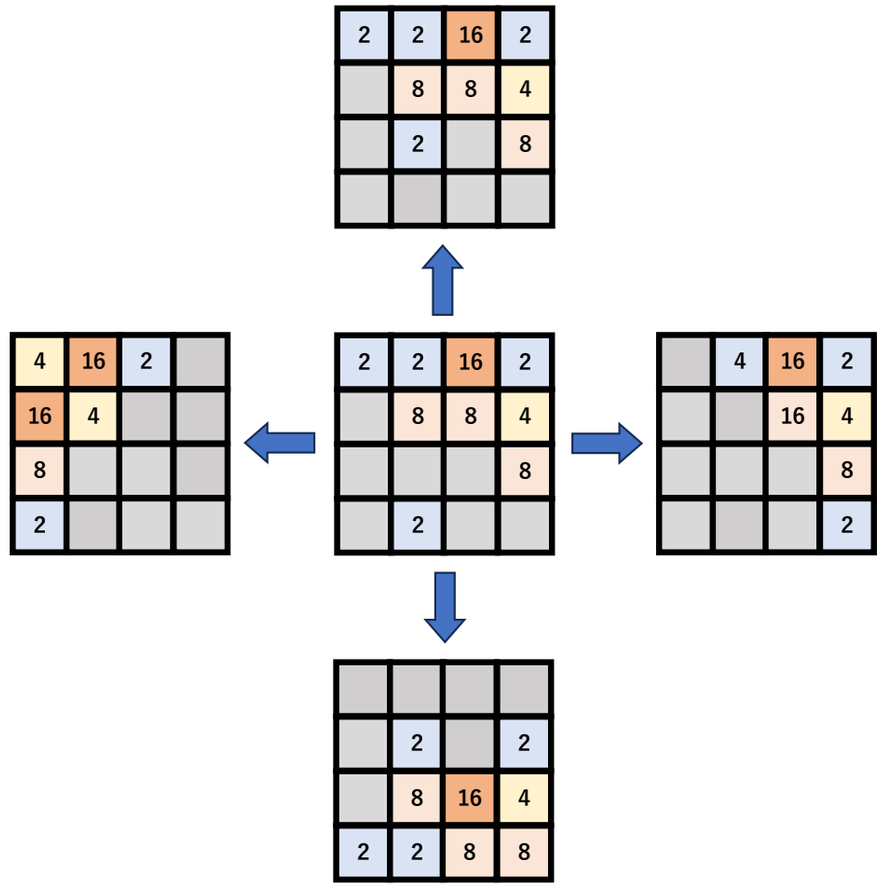


図 2.1: 上下左右それぞれへのスライドの例

図 2.2: 終了状態の例

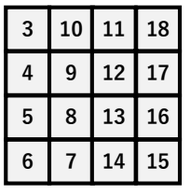


図 2.3: 最大到達盤面の例 (数字は指数部のみ)

化する．新しい数字タイルとして 2 か 4 のタイルが出現することで， 数字タイルの合計値はその値の分だけ必ず増加する．すなわちプレイヤが 1 回行動するたびに， 盤面上の数字タイルの合計値は 2 か 4 ずつ単調に増加する．

よって盤面上の数字タイルの合計値をゲームの進行度合いとして用いることができる．以降これを**時刻**と呼ぶ．例えば図 2.4 の遷移では時刻 $2 \times 4 + 4 + 8 \times 3 + 16 = 52$ の状態 s が時刻 $52 + 2 = 54$ の状態 s_{next} に遷移している．ゲームの時刻はプレイヤが行動するたびに必ず増加するため， 2048 はサイクル

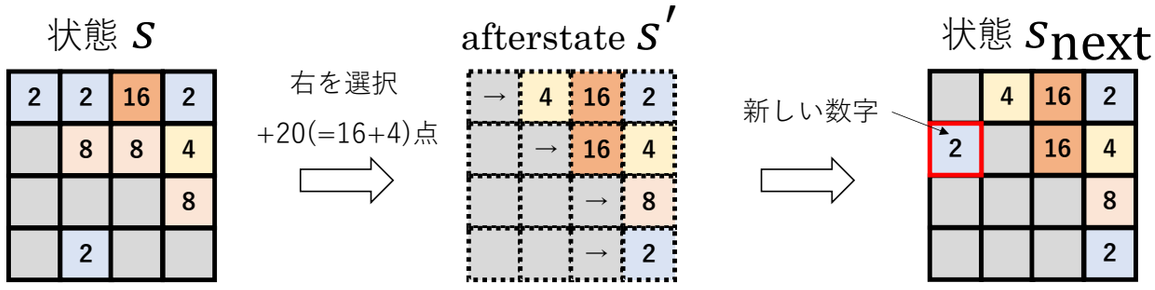


図 2.4: 状態遷移の例

の出現しないゲームであることがわかる．

また 2048 は必ず終了するゲームである．2^n のタイルを 1 つ完成させるには，盤面上に 2 つの 2^{n-1} のタイルを同時に存在させる必要がある．2 つの 2^{n-1} のタイルを同時に存在させるには，1 つの 2^{n-1} のタイルと 2 つの 2^{n-2} を同時に存在させる必要がある．新しい数字タイルとして 2 と 4 が出現することを踏まえると，帰納的に考えて 2^n のタイルを完成させるには最小でも $n-2$ マスを必要とすることが分かる．よって 16 マスの 2048 では最高でも 2^{18} のタイルまでしか完成させられず，運や選択手に関わらずゲームを永久に続けることはできないことが分かる．

第 3 章

2048 と強化学習

これまでに 2048 を対象とした強化学習の研究は数多くなされてきた．本章では強化学習の概要，および 2048 に対する強化学習の先行研究について記述する．

3.1 強化学習の概要

まず本節では 2048 との関係を踏まえつつ，一般的な強化学習の概要について記述する．なお本節の内容は全体に文献 [2] を参照して書かれた．

3.1.1 マルコフ決定過程

強化学習は与えられた環境において試行錯誤することを通して，目標を達成するための戦略や意思決定を学習するための手法である．学習や意思決定を行う主体はエージェントと呼ばれる．エージェントは離散タイムステップに従って行動を選択し続け，環境とやり取りを行う．

このような問題設定はマルコフ決定過程 (MDP) というモデルによって定式化されている．MDP は以下の 4 つの要素で構成される．

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

エージェントはステップ t で状態 $S_t \in \mathcal{S}$ から行動 $A_t \in \mathcal{A}$ を選択する．そして確率 $p(S_{t+1}|S_t, A_t)$ で次の状態 S_{t+1} に遷移し，$R_{t+1} = r(S_t, A_t, S_{t+1})$ を即時報酬として獲得する．状態遷移関数と報酬関数は環境のダイナミクスと呼ばれることがある．図 3.1 に MDP の概念図を示す．

状態集合と行動集合が有限である MDP を有限 MDP と呼ぶ．2048 は有限 MDP にそのまま当てはま

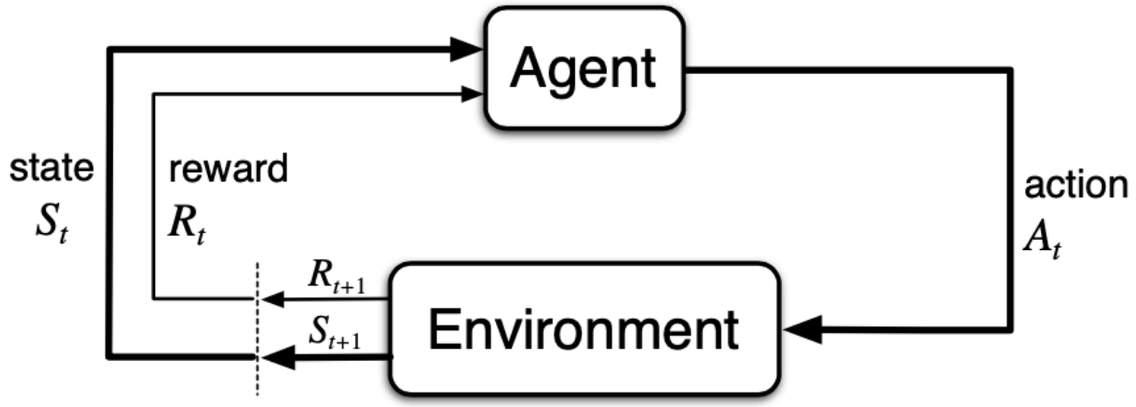


図 3.1: MDP の模式図 (文献 [2] より引用)

るゲームである．行動集合 A はプレイが選ぶ上下左右に対応し， 報酬はプレイヤが獲得する得点に直接対応する．

一般に強化学習で扱う問題には, エージェントと環境のやり取りが終わる終了状態が存在する episodic task と終了状態が存在しない continuing task が存在する．episodic task ではエージェントと環境のやり取りを初期状態から終了状態までのエピソードと呼ばれる単位で分割することができる．2.2 節で説明したように 2048 は必ず終了するゲームであるため， 以降 episodic task での定義を確認する．

3.1.2　方策と価値関数

エージェントがある状態において行動を決定する際の戦略， すなわち確率分布 $\pi : S \times A \to [0, 1]$ を方策と呼ぶ．状態価値関数 $v_\pi(s)$ は状態 s から方策 π に従って行動を選択し続けた場合の累積報酬和の期待値であり， 次のように定義される．

$$v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s \right] \tag{3.1}$$

同様に状態 s から行動 a を選択し， その後方策 π に従って行動を選択し続けた場合の累積報酬和の期待値である行動価値関数 $q_\pi(s, a)$ の定義は以下のようになる．

$$q_\pi(s, a) \stackrel{\text{def}}{=} \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{3.2}$$

強化学習の目標は多くの報酬を獲得できるような良い方策を見つけることである．価値関数の定義より 2 つの方策 π と π' があるとすると， すべての状態 $s \in S$ について $v_\pi(s) \geq v_{\pi'}(s)$ が成り立つならば π は π' と等価か π' よりも良い方策だと言える．ここで他のすべての方策と比べて等価であるか， それよ

りも良い方策が少なくとも 1 つ存在する．これは最適方策 π_* と呼ばれる方策である．π_* に従うときの状態価値関数は最適状態価値関数と呼ばれ， v_* で表される．同様に π_* に従うときの行動価値関数は最適行動価値関数と呼ばれ， q_* で表される．それぞれの具体的な定義を式 3.3， 3.4 に示す．

$$v_*(s) = \max_\pi v_\pi(s) \quad \text{for all } s \in S \tag{3.3}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a) \quad \text{for all } s \in S \text{ and } a \in A(s) \tag{3.4}$$

このとき $v_*(s) = \max_{a \in A(s)} q_*(s,a)$ であるから， 以下の式が導かれる (図 3.2 を参照) ．

$$v_*(s) = \max_{a \in A(s)} q_*(s,a) \tag{3.5}$$

$$= \max_{a \in A(s)} \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \tag{3.6}$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \tag{3.7}$$

$$= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \tag{3.8}$$

式 3.6， 3.8 はベルマン最適方程式と呼ばれる．

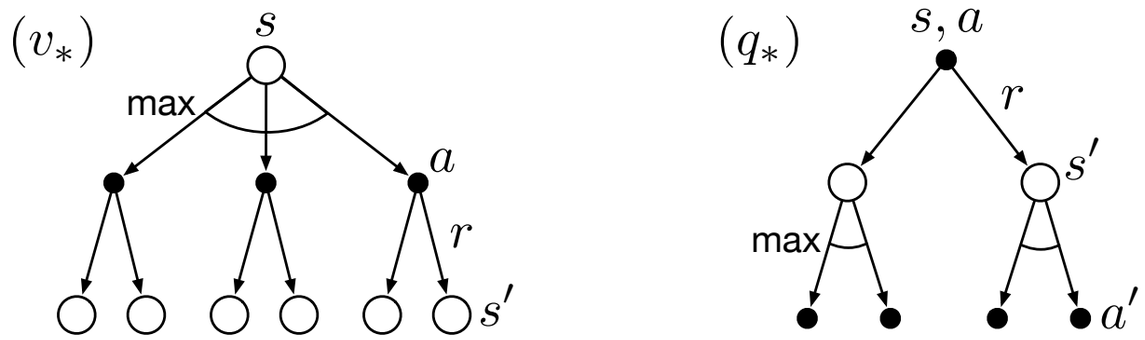


図 3.2: 最適価値関数のバックアップ図 (文献 [2] より引用)

3.1.3 価値ベースな手法

状態 s における最適方策に従った具体的な行動は $\arg\max \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$ と， 1 ステップ先の状態を探索してそれらの最適状態価値関数を参照することで計算できる．最適行動価値関数が参照できれば， 1 ステップ先の状態を探索する必要すらなく， 単に $\arg\max q_*(s,a)$ を選択すればよい．最適方策を得るために価値関数を学習する手法は価値ベースな手法と呼ばれる．

Q 学習は最も基本的な価値ベースな手法の 1 つで， 最適行動価値関数 q_* の推定値 Q を得られた経験から学習する．状態 S_t から行動 A_t を選択し， 報酬 R_{t+1} を獲得して次の状態 S_{t+1} に遷移したとする．

このとき Q 学習は以下の更新式 3.9 に従って $Q(S_t, A_t)$ を更新する．

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3.9}$$

式 3.9 は現在の推定値 $Q(S_t, A_t)$ を，ベルマン最適方程式 3.8 の右辺の推定値である $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$ に近づけていると解釈できる．任意の $(s, a) \in \mathcal{S} \times \mathcal{A}$ について $Q(s, a)$ が更新され続けるという条件の下で，Q 学習は収束が保証されている．

状態集合や行動集合が大きい場合や有限でない場合には，テーブル形式で Q 値を保持することができないため何らかの関数近似を行う必要がある．近年では深層学習 [3] の研究の発展により，関数近似の方法としてニューラルネットワークが用いられることが多い．価値関数や方策をニューラルネットワークで近似する手法は，まとめて深層強化学習と呼ばれる．Q 値をニューラルネットワークで近似する，Deep Q Network (DQN) [4] は深層強化学習の先駆けとして有名な手法である．DQN の発表以降，様々な工夫が提案され続けている．詳細は文献 [5] の 4 節などを参照されたい．

3.1.4　方策ベースな手法

方策を直接パラメタライズされた関数で近似し，目的関数との誤差を最小化するように勾配を

3.2　AlphaZero

Silver らが提案した AlphaZero [6] は，二人ゼロ和完全確定情報ゲームを対象とした有力な深層強化学習手法である．AlphaZero は盤面の特徴量を入力として方策と価値を出力するニューラルネットワークを，自己対戦を通して得たデータから学習する．自己対戦において，AlphaZero はモンテカルロ木探索 (MCTS) というアルゴリズムを使用して指し手を選択する．

3.2.1　モンテカルロ木探索

ここでは AlphaZero が使用したモンテカルロ木探索 (MCTS) について説明する．MCTS は探索で発見した各状態 (囲碁や将棋では盤面) に対応するノードから成る探索木を構築する．探索木中のそれぞれの状態と行動の対 (s, a) について，$N(s, a), W(s, a), Q(s, a), P(s, a)$ という 4 つの統計量を管理する．

- $N(s, a) \cdots$ 探索中に s から a を選択した回数
- $W(s, a) \cdots$ 最適行動価値 $q_*(s, a)$ の推定値の累計
- $Q(s, a) \cdots$ 最適行動価値 $q_*(s, a)$ の推定値の平均 $(W(s, a)/N(s, a))$
- $P(s, a) \cdots$ ニューラルネットワークが評価する s から a を選択する確率

探索木は最初，現在の状態に対応するノードである根ノードのみから成る．MCTS は選択，評価，逆

伝播， 展開の 4 つのステップを繰り返すことで， 探索木を大きくする．この 4 つのステップはまとめてシミュレーションと呼ばれる．すなわちシミュレーションを繰り返すことで深い探索を行い， 良い行動を選択することができる．以下ではそれぞれのステップについて説明する．

選択 (図 3.3a)

現在のゲーム木の根ノード s_0 から有望な行動を選択し， 葉ノード s_L に至るまで辿り続ける．具体的にはノード s_t において，以下の式 3.10 で表される a_t を選択する．

$$a_t = \arg\max_a (Q(s_t, a) + C(s_t)P(s_t, a)\sqrt{N(s_t)}/(1 + N(s_t, a))) \tag{3.10}$$

ただし $C(s_t)$ は探索率， $N(s_t)$ はノード s_t の訪問回数を表す．これは現段階での価値の推定値 $Q(s_t, a)$ と探索項 $C(s_t)P(s_t, a)\sqrt{N(s_t)}/(1 + N(s_t, a))$ を合計したものである．$N(s_t, a)$ が小さい行動ほど探索項は大きな値を示す．

評価 (図 3.3b)

選択のステップの葉ノード s_L をニューラルネットワーク f_θ により評価する．f_θ が評価した s_L における方策を $\pi(\cdot|s_L)$， s_L の価値を v_L とする．

逆伝播 (図 3.3c)

s_L から選択可能なそれぞれの行動 a について，$N(s_L, a) = 0, W(s_L, a) = 0, Q(s_L, a) = 0, P(s_L, a) = \pi(a|s_L)$ と初期化する．さらに s_0 から s_L に至る各エッジ (s_t, a_t) について， 以下のように各統計量の値を更新する．

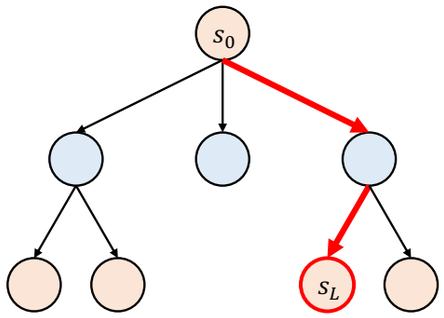
$$\begin{aligned} N(s_t, a_t) &= N(s_t, a_t) + 1 \\ W(s_t, a_t) &= W(s_t, a_t) + v_L \\ Q(s_t, a_t) &= \frac{W(s_t, a_t)}{N(s_t, a_t)} \end{aligned}$$

展開 (図 3.3d)

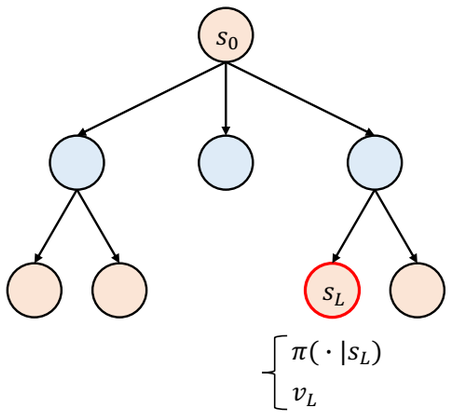
s_L から遷移しうる次の状態を新たなノードとして探索木に追加する．

行動の決定方法

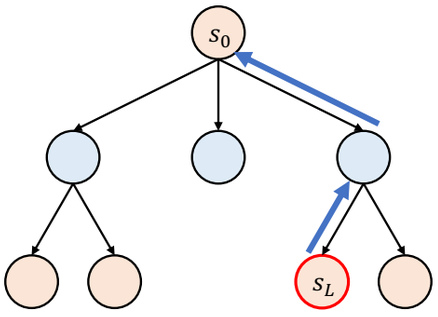
一定回数のシミュレーションを行った後に， 実際に選ぶ行動を決定する．根ノード s_0 から選択可能な各行動 a について， $N(s_0, a)^{1/\tau}$ に比例した確率で行動を選択する．τ は温度パラメータと呼ばれ， 0 に近づくほど訪問回数に関して貪欲な選択をするようになる．多様な学習データを集めるために学習の序盤では τ を大きく設定し， 学習が進むについて 0 に近づける．



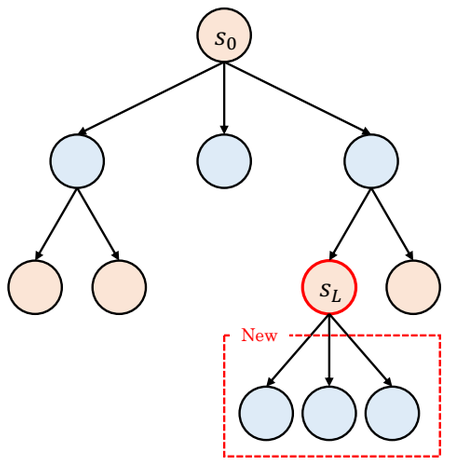
(a) 選択



(b) 評価



(c) 逆伝播



(d) 展開

図 3.3: MCTS の各ステップ (オレンジ色のノードは自分の手番，青色のノードは相手の手番)

3.2.2　自己対戦とニューラルネットワークの学習

AlphaZero は人間の棋譜ではなく，自己対戦によって得た学習データを使ってニューラルネットワークを訓練する．自己対戦は，

3.2.3　AlphaZero の評価と MuZero

囲碁，将棋，チェスにおいて当時の有力なプログラムを上回る強さを示して注目を集めた．AlphaZero が行う MCTS は，ゲームのルールや環境のダイナミクスが既知であることを前提としている．すなわち現在の状態 s_t から行動 a_t を選択した後の状態 s_{t+1} を，s_t のみ観測可能な状況から知ることができる必

要がある．そのため AlphaZero は複雑なダイナミクスを持つ Atari などのゲームには適用することが難しい．Schrittwieser らが提案した MuZero [7] は， ニューラルネットワークで環境のダイナミクスをモデル化し， 学習することを提案した．

3.3 2048 への強化学習の応用

3.1 節では強化学習一般の概要について説明した．これを踏まえて本節では， 2048 を対象とした強化学習研究の先行研究について概説する．

3.3.1 TD afterstate 学習

Szubert ら [8] は TD afterstate 学習と呼ばれる価値ベースの強化学習手法を提案した．2048 はゲームの性質上， 状態 s から行動 a をとって獲得する得点 $r(s,a)$ と遷移する afterstate s' は決定的である．よって afterstate s' の最適価値を $v'_*(s')$ と表すと， 行動価値 $q_*(s,a)$ は $q_*(s,a) = r(s,a) + v'_*(s')$ と分解できる．そこで TD afterstate 学習は， Q 値ではなく afterstate の価値 v'_* の推定値 V' を式 3.11 に従って更新する．

$$V'(S'_t) \leftarrow V'(S'_t) + \alpha[R_{t+1} + V'(S'_{t+1}) - V'(S'_t)] \tag{3.11}$$

式 3.11 は Q 学習の更新式 3.9 において， $Q(S_t, A_t) = R(S_t, A_t) + V'(S'_t)$ として書き直したものと見ることができる．

Szubert らは N-tuple ネットワークというヒューリスティックな盤面評価関数を用いて V' を表現し，2048 のタイルを 98% の確率で到達させることに成功した．Jaskowski [9] はこれに様々な工夫を加えた学習方法を提案し， 最終的に expectimax 探索を 1 手 1 秒の制限で行うことで平均 609,104 点を獲得するプレイヤを開発した．松崎 [10] はニューラルネットワークによって表現した V' を学習させ， 3-ply expectimax 探索を行うことで平均 406,927 点を達成した．

Szubert らは文献 [8] で N-tuple ネットワークを用いた Q 学習の実験も行ったが， TD afterstate 学習の性能を大きく下回る結果になったことを報告している．TD afterstate 学習は s から afterstate s' への遷移までをシミュレートする， いわば 0.5 手読みの探索を行っているため， 0 手読みの Q 学習を上回ると考えられる．

3.3.2 Afterstate PPO

価値ベースな手法である TD afterstate 学習が一方で方策ベースな強化学習手法で 2048 を山下ら [11] は

3.3.3 Stochastic MuZero

Antonoglou らが提案した Stochastic MuZero [12] は， 2048 において．Stochastic MuZero は 3.2.3 節で述べた MuZero を， 確率的な環境にも対応できるように拡張した手法である．

第 4 章

提案手法

これまでに 2048 を対象とした強化学習の研究は数多くなされてきた．ゲームを完全解析

4.1　2048 のミニゲームの完全解析

3 章で述べた強化学習は環境 (ゲーム) と何度もやり取りすることで， 最適な方策を学習するための手法である．一方で小さなゲームであれば， 力ずくの計算によってゲームを完全に解くこともできる．本章では 2048 を解析的なアプローチによって解くことについて考察する．

4.1.1　2048 の完全解析とは

2048 は 1 人用のゲームであるため， 勝敗のようなプレイヤの明確な目標は存在しない．そのためプレイヤが何を目標とするかによって， プレイヤの最善手の定義は変化する．また 2.1 節で述べたようにゲームはランダム性を伴うため， 同じ状態から毎回同じ手を選んでも結果は確率的に変動する．

そこで本稿ではある状態 s における最善手を「s から獲得できる得点の合計の期待値が最も高くなるような手」と定義する．これは 3 節で述べた強化学習の最適状態価値と等価なものである．よって状態 s から最善手を選び続けて獲得できる得点の合計の期待値を状態 s の最適価値と呼び， $v_*(s)$ で表すことにする．

このとき $v_*(s)$ は式 4.1 のように再帰的な形式で書くことができる．

$$v_*(s) = \begin{cases} 0 & (s \text{ が終了状態}) \\ \max_a \left(r(s,a) + \mathbb{E}_{s_{\text{next}} \in \mathcal{T}(s,a)} v_*(s_{\text{next}}) \right) & (\text{otherwise}) \end{cases} \tag{4.1}$$

ただし $r(s,a)$ は状態 s から行動 a をとって獲得する得点， $s_{\text{next}} \in \mathcal{T}(s,a)$ は状態 s から行動 a をとって遷移しうる次の状態の集合を表す (図 4.1 を参照)．式 4.1 の $r(s,a) + \mathbb{E}_{s_{\text{next}} \in \mathcal{T}(s,a)} v_*(s_{\text{next}})$ は， 強化学習における最適行動価値 $q_*(s,a)$ に対応する．また $\mathbb{E}_{s_{\text{next}} \in \mathcal{T}(s,a)} v_*(s_{\text{next}})$ は， s から a をとって遷移する afterstate s' の価値といえる．

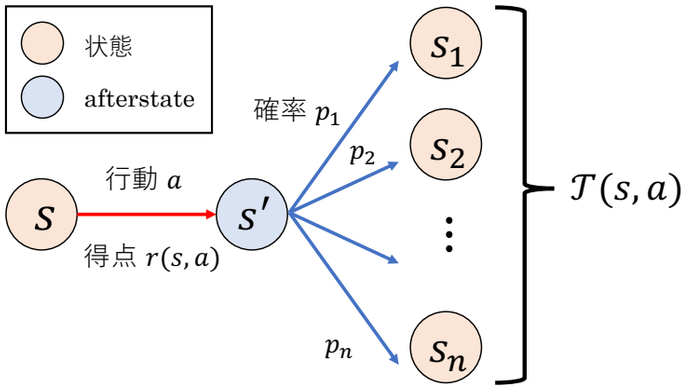


図 4.1: 式 4.1 の補足図

ゲームに現れうるすべての状態の最適価値を計算すれば， 任意の状態において最善手を選ぶことができる．本稿ではこれを 2048 の完全解析ということにする．

完全解析をすることで， ゲームの初期状態の最適価値を明かすだけでなく， 最善手を選び続けるプレイヤの戦略を解析することができる．また 2048 を対象とした強化学習手法の良し悪しを定量的に評価し， より良い手法を提案することができると考えられる．一方で， 2048 を完全解析することはそのゲーム木の大きさから現状難しいと考えられる．そこで以降では本来 4×4 盤面上で行われる 2048 のミニゲームとして， 盤面サイズを縮小した 2048 を完全解析することを考える．

4.1.2　盤面サイズが小さな 2048 の完全解析

基本的なルールは 2048 と同じで盤面サイズを 4×4 から縮小したゲームを完全解析することを考える．盤面サイズに関わらず， 以下の 2 つのステップを順番に行うことで 2048 を完全解析することができる．

1. ゲームに現れうるすべての状態の列挙
2. 列挙した状態の価値の計算

4.1.2 節と 4.1.2 節で具体的な方法について述べる．なお本節の内容は文献 [13] および文献 [14] を元に執筆された．

幅優先探索によるすべての状態の列挙

完全解析の第 1 ステップとして幅優先探索によってゲームに現れうるすべての状態を列挙する．まず初期状態をキューに詰めて探索を開始する．キューの先頭の状態 s を取り出し， s から遷移可能な次の状態 $s_{\text{next}} \in \mathcal{T}(s)$ をキューに追加する．これをキューが空になるまで繰り返すことで， すべての状態を列挙することができる．

Algorithm 1 幅優先探索によるすべての状態の列挙

```
function ENUMERATION
    for t = 4 to t_max do
        for all s_t ∈ queue_t do
            for all s_{t+2} ∈ 𝒯(s_t) do
                queue_{t+2}.push(s_{t+2})
            end for
            for all s_{t+4} ∈ 𝒯(s_t) do
                queue_{t+4}.push(s_{t+4})
            end for
        end for
    end for
end function
```

ここで $s_{\text{next}} \in \mathcal{T}(s)$ がすでに発見済みであるか確認するために， これまでに発見した状態を管理する集合が必要である．素朴な方法ではメモリでこれまでに発見した全状態を管理することで行えるが， 状態数が非常に大きな場合にはメモリの容量を超えてしまう．

そこで 2.2 節で説明した時刻によってゲーム木を整理する．時刻 t の状態は時刻 $t+2$ か $t+4$ の状態にしか遷移しないため， 時刻 $t+2$ と $t+4$ の発見した状態をメモリで管理すれば十分である．よって時刻が最小の 4 の状態から時刻 2 刻みで順番に列挙を行うことで， ディスクを効率的に活用することができる．以上を踏まえた疑似コードを Algorithm 1 に示す．

後退解析による状態の価値の計算

4.1.2 節で列挙した状態の価値を式 4.1 に従って計算する．時刻 t の状態の価値は， 時刻 $t+2$ と $t+4$ の状態の価値が計算済みであれば計算できる．よって時刻が最大の状態から順番に走査することで， 効率的にすべての状態の価値を計算できる．疑似コードを Algorithm 2 に示す．

4.1.3　実験結果

2×2 盤面から 4×3 盤面までの 2048 を完全解析した結果を表 4.1 に示す．

参考として， 2×2 盤面の 2048 のゲーム木全体を図 4.2 に示す．オレンジ色のノードは初期状態， 青色のノードは終了状態を表している．

4.2　3 × 3 盤面の 2048 の完全解析と強化学習

Algorithm 2 後退解析による価値計算

```
1: function CALCULATION(t)
2:     for t = t_max to 4 do
3:         v(s) = max_a (r(s,a) + E_{s_next ∈ T(s,a)} v_*(s_next))
4:     end for
5:     for all s_t ∈ queue_t do
6:         for all s_{t+2} ← s_t do
7:             queue_{t+2}.push(s_{t+2})
8:         end for
9:         if element > max then
10:            max ← element
11:        end if
12:    end for
13:    return max
14: end function
```

盤面サイズ	状態数	初期状態の価値
$2 \times 2 = 4$	110	67.6
$3 \times 2 = 6$	$21,752$	480.9
$4 \times 2 = 8$	$4,980,767$	$2,642.6$
$3 \times 3 = 9$	$48,713,519$	$5,469.2$
$4 \times 3 = 12$	$1,152,817,492,752$	$50,724.2$

表 4.1: 盤面の大きさと解析結果に関する表

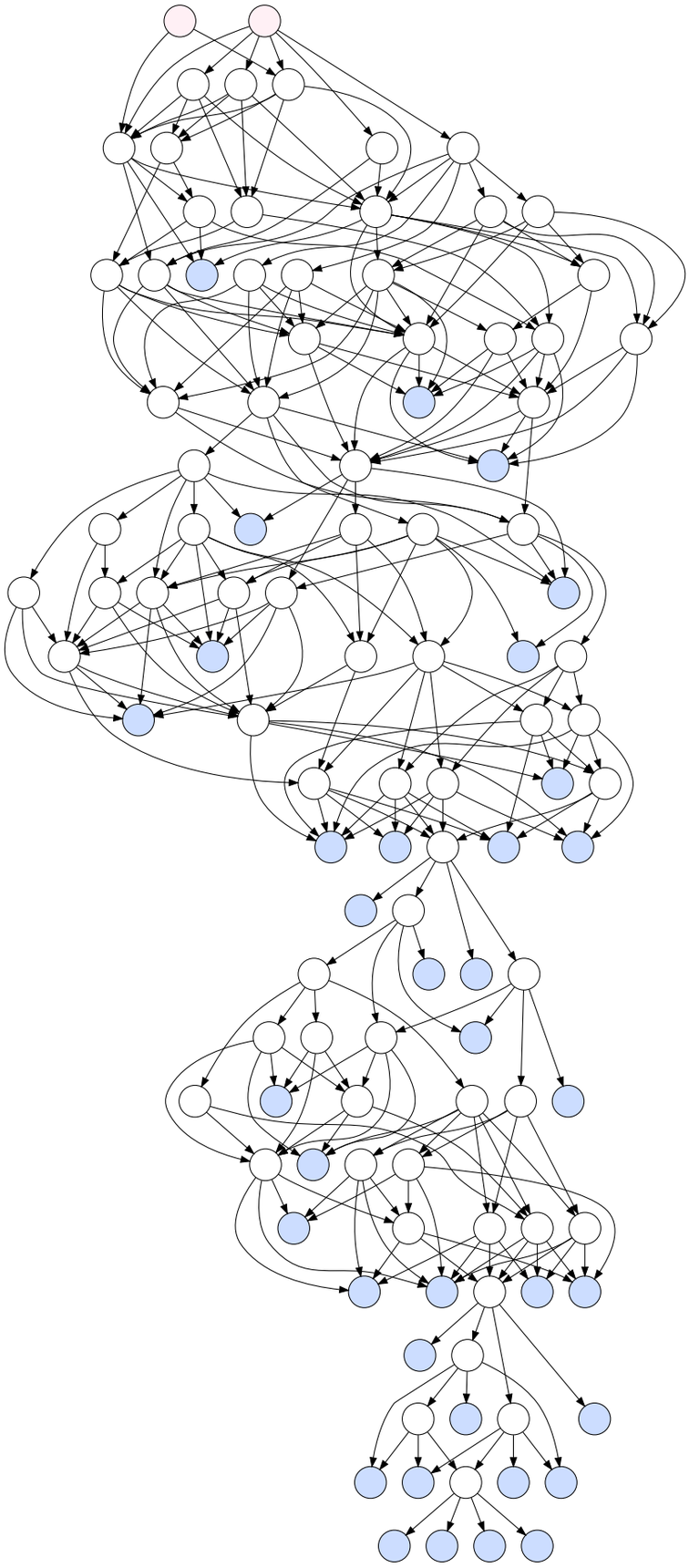


図 4.2: 2 × 2 盤面の 2048 のゲーム木

付録 A

実装の詳細

A.1　ゲーム環境の実装

2048 は状態から afterstate への遷移において，各行 (列) の変化は独立に考えることができる．また回転と反転を考慮することで上下左右は等価な盤面変化を起こす．よって 1 行の全パターンについて，ある一方向を選択したときの遷移先を前もって計算することで，全方向に対する盤面全体の遷移を高速に行える．

A.2　完全解析の実装

A.3　強化学習の実装

参考文献

[1] G Cirulli. 2048, available from ⟨http://gabrielecirulli.github.io/2048/⟩, 2014.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[3] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, Vol. 518, No. 7540, pp. 529–533, 2015.

[5] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *CoRR*, Vol. abs/1811.12560, , 2018.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, Vol. 362, No. 6419, pp. 1140–1144, 2018.

[7] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, Vol. 588, No. 7839, pp. 604–609, Dec 2020.

[8] Marcin Szubert and Wojciech Jaśkowski. Temporal difference learning of n-tuple networks for the game 2048. In *2014 IEEE CIG*, pp. 1–8, 2014.

[9] Wojciech Jaskowski. Mastering 2048 with delayed temporal coherence learning, multi-state weight promotion, redundant encoding and carousel shaping. *CoRR*, Vol. abs/1604.05085, , 2016.

[10] Kiminori Matsuzaki. Developing value networks for game 2048 with reinforcement learning.

Journal of Information Processing, Vol. 29, pp. 336–346, 2021.

[11] 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, 第 2021 巻, pp. 179–185, nov 2021.

[12] Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2022.

[13] 山下修平, 金子知適, 中屋敷太一. 3 × 3 盤面の 2048 の完全解析と強化学習の研究. ゲームプログラミングワークショップ 2022 論文集, 第 2022 巻, pp. 1–8, nov 2022.

[14] 山下修平, 金子知適. 4 × 3 盤面の 2048 の完全解析. ゲームプログラミングワークショップ 2023 論文集, 第 2023 巻, pp. 1–5, nov 2023.