

卒業論文

ゲーム「2048」のプレイヤーについて

08-152021 金澤望生

指導教員 山口和紀 教授

2018 年 1 月

東京大学教養学部学際科学科総合情報学コース

概要

インターネットブラウザやスマートフォン上で遊ぶことのできるパズルゲーム「2048」をプレイする AI の改良を行った．改良には盤面上で最も大きな数のタイルが隅にあることを重視する独自のヒューリスティック「corner bonus」を使用した．(仮)

キーワード ゲーム AI , 機械学習

目次

第 1 章	導入	1
第 2 章	先行研究の紹介	2
2.1	Szubert & Jaskowski (2014)	2
2.2	Wu et al. (2014)	3
2.3	Oka & Matsuzaki (2016)	6
2.4	Yeh et al. (2016)	9
第 3 章	本研究のアイデア	11
第 4 章	提案と実装	12
第 5 章	実験	13
第 6 章	考察と結論	14
	謝辞	15
	参考文献	16
	付録 A	17

第 1 章

導入

モチベーションや 2048 の基本ルール・指標について説明します。

第 2 章

先行研究の紹介

既存研究が使用している手法とプレイヤーの成績について説明します。

2.1 Szubert & Jaskowski (2014)

Szubert & Jaskowski は、TD 学習を用いたプレイヤーの訓練と n タプルネットワークを用いた価値関数の表現を組み合わせることによって、人間の知識やゲーム木探索を使用しないで十分強い 2048 プレイヤを実装することに成功した。

2.1.1 TD 学習

TD 学習の「TD」とは temporal difference の略であり、すなわち状態間における価値の差分を学習することによって学習器の訓練を行う手法である。2048 にあてはめると、とある盤面 s' の価値と、その盤面の 1 プレイ後の盤面 s'_{next} の価値の差分を取り、これを現状定まっている s' に足し込んでいくことで訓練を行うことになる。TD 学習にはさまざまな派生があるが、Szubert & Jaskowski が使用している TD(0) 学習は以下の式によって表現される：

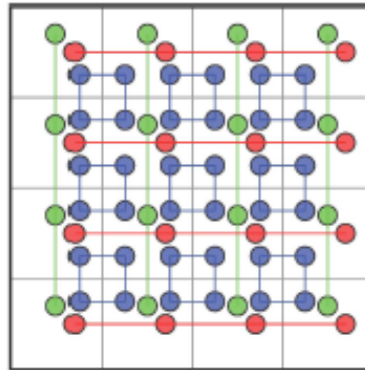
$$V(s) \leftarrow V(s) + \alpha(r + V(s') - V(s))$$

この式において、 V は価値関数、 α は学習率、 r は報酬である。学習率は計算された差分を価値関数の更新にどれほど反映するかを決定するパラメータである。

TD 学習は Tesauro によるバックギャモンへの適用でよく知られるようになり、碁やオセロ、チェスにおけるゲーム AI の方策決定の手法として用いられるようになった。

2.1.2 n タプルネットワーク

TD 学習によって盤面の評価とその学習を行うことができるが、盤面と評価値をどのように結びつけるかが問題になる。まず、2048 で有り得るすべての盤面に対して評価値を与える 1 対 1 対応のルックアップテーブル (LUT) を作成することを考えると、2048 で有り得る盤面の数は $(4 \times 4)^{18} \approx 4.7 \times 10^{21}$ と膨大な数になり、このような LUT を計算機上で実装するこ

図 2.1. n タプルネットワークの例

とは現実的に不可能である。

そこで、一部のマスの組み合わせによる「タプル」というクラスターを作成し、さらに複数のタプルを組み合わせることで盤面を表現する手法「 n タプルネットワーク」を 2048 に導入することが、Szubert & Jaskowski によって提案された。たとえば、図 2.1 のような n タプルネットワークを実装した場合、1 つのゲーム内で保持すべき重みの数は 860625 であり、全ての有り得る盤面に対する LUT を保持するのに対して非常に少なく済む。

n タプルネットワークは Bledsoe & Browning (1959) によりパターン認識に用いられたのが最初の採用例である。ゲーム AI の分野では Jaskowski (2014) によってオセロに適用され、一定の成果が得られた。

2.1.3 結果と課題

本手法をもとに行った実験のうち、最も良い勝率を達成したプレイヤーを用いて 10 万ゲーム中の成績を検証したところ、勝率は 0.9781 であり、平均スコアは 100,178 であった。1 ゲーム中に達成されたスコアで最も良かったのは 261,526 であった。Szubert & Jaskowski による新たな手法は探索ベースの手法よりも大幅に高速で、かつ成績が良かった。

しかしながら、この手法では「常に 2048-tile を生成すること」よりも「時々 16384-tile を生成すること」を重視しているため、勝率は必ずしも 100% を達成できていない。また、人間の知識を一切導入していないため、最も大きな数のタイルが盤面上の端に配置されないなど、人間の直感的な戦略とは反しているといったデメリットがあった。

2.2 Wu et al. (2014)

Wu は、Szubert & Jaskowski の手法を改良し、木探索を用いた先読みと組み合わせることによってさらに良いプレイヤーを実装することに成功した。

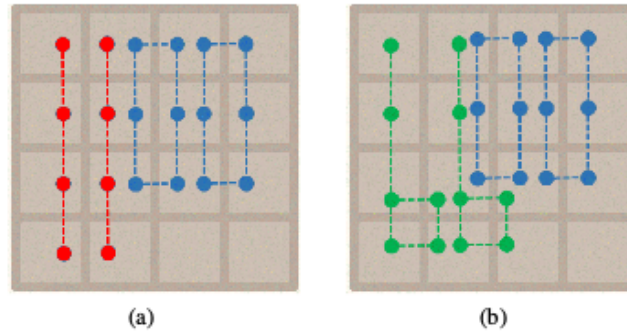


図 2.2. Szubert & Jaskowski (a) と Wu (b) が提唱した n タプルネットワーク

2.2.1 n タプルネットワークの配置の改善

Wu は, Szubert & Jaskowski が考案した n タプルネットワークのうち, 直線型で 4 タプルとして配置していたタプルを, 図 2.2 (b) のように柄杓型の 6 タプルに変更した. これによって増える重みの数は約 2 倍程度であったが, この変更によって Szubert & Jaskowski のものよりも飛躍的に良い成績を得ることができた. なお, なぜこのようなタプルの配置が最善だと判断したのかについて, Wu は論文において言及していない.

2.2.2 Multi-Stage TD 学習の導入

Multi-Stage TD 学習 (MS-TD 学習) とは, ゲームの局面に応じて異なる価値関数を保持することによって, よりそれぞれの局面に対して適切な重みを学習させることを目的とした手法である. Wu は学習のプロセスを 3 つのステージに分割し, ゲームプレイも同様に 3 つのステージに分割して行うようにした. Wu の提唱した 2048 における MS-TD 学習は, 以下のような手順で学習を行う. なお, 「 T_{16k} 」とは「そのゲーム中で初めて 16384-tile を生成することに成功した時」, 「 T_{16+8k} 」とは「そのゲーム中で初めて 16384-tile を生成した後に, 初めて 8192-tile を生成することに成功した時」のことを示す.

1. 第 1 ステージにおいては, 初期盤面からゲームを始めて, 価値関数が十分飽和するまで学習を行う. このステージで学習された価値関数の重みのことを「Stage-1 価値関数」と呼ぶことにする. また, 学習ゲーム中に T_{16k} を達成したなら, その時の盤面を全て保存しておく.
2. 第 2 ステージにおいては, 第 1 ステージで保存した盤面からゲームを始めて, TD 学習を行う. このステージで学習された価値関数の重みのことを「Stage-2 価値関数」と呼ぶことにする. また, 学習ゲーム中に T_{16+8k} を達成したなら, その時の盤面を全て保存しておく.
3. 第 3 ステージにおいては, 第 2 ステージで保存した盤面からゲームを始めて, TD 学習

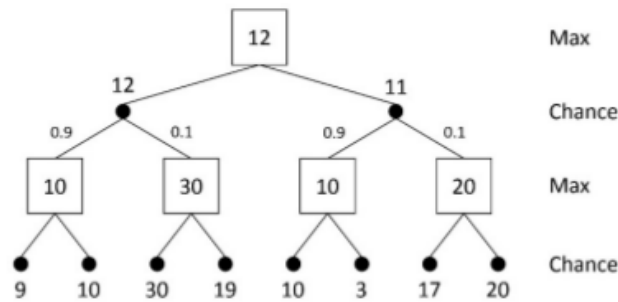


図 2.3. Expectimax 木の例 (Wu et al. (2015))

を行う．このステージで学習された価値関数の重みのことを「Stage-3 価値関数」と呼ぶことにする．

その後，以下のような手順でゲームプレイを行う．

1. 盤面が T_{16k} を達成するまでは，Stage-1 価値関数を用いてゲームプレイを行う．
2. 盤面が T_{16k} を達成してから T_{16+8k} を達成するまでは，Stage-2 価値関数を用いてゲームプレイを行う．
3. 盤面が T_{16+8k} を達成してからは，Stage-3 価値関数を用いてゲームプレイを行う．

2.2.3 Expectimax 木探索

Szubert & Jaskowski (2014) によって，木探索ベースの 2048 プレイヤは強化学習で訓練したプレイヤに劣ることが示されたが，Wu は強化学習によるプレイヤに対して Expectimax 木探索を補助的に組み合わせることによって，さらにプレイヤの性能を高めようと試みた．

Expectimax 木探索には，Max ノードと Chance ノードという 2 種類のノードがあり，それぞれのノードの値は子ノードから決定される．Max ノードの値は，子ノードのうち最も大きな値のノードの値となる．例えば図 2.3 の根ノードの値は 12 であるが，これは子ノードの「12」と「11」のうち最大の値である 12 を取ったものである．一方で Chance ノードの値は子ノードの期待値となる．例えば図 2.3 の根ノードの子ノードの 1 つである「12」というノードは，0.9 の確率で 10 となるノードと 0.1 の確率で 3 となるノードの期待値，すなわち $0.9 \times 10 + 0.1 \times 3 = 12$ によって 12 という値が決定する．

Wu の提案においては，Max ノードの値はプレイヤがアクションを選択して遷移を行った後の盤面，Chance ノードは遷移を行った後にランダムタイルを発生させた後の盤面が与える評価値となる．例えば，ある盤面 s (アクション選択と遷移が終わった直後の盤面とする) の評価値を深さ 3 の Expectimax 木探索を用いて求めたい時，図 hogehoge のような探索木が考えられる． s の盤面が分かれば，その子ノードであるランダムタイル生成後の盤面，さらにその子ノードである遷移後の盤面を求めることができる．さらに，葉ノードにあたる Chance ノー

ドの値は価値関数が与える評価値とすることで、各ノードの値を求めることができ、最終的に根ノード、すなわち評価値を求めたい盤面 s の評価値も求められるということになる。

(図 2.4. 何かいい感じの 2048 の探索木を自分で描画して貼る)

Expectimax 木探索を用いて先読みをすることで、将来の盤面を予想してより良いアクションを選択できるようになった。これはスコアの面での成績が良くなることに繋がる一方で、maxTile=2048 を達成する割合、すなわち勝率の向上にも大きな影響をもたらした。

2.2.4 結果と課題

Wu によるプレイヤーは Szubert & Jaskowski のものに比べて著しく良い成績を達成した。まず Szubert & Jaskowski が達成できなかった 32768-tile の生成に成功し、10.9% の確率で 32768-tile を生成できるようになった。勝率に関しては 1 を達成、すなわち 2048-tile は 100% の確率で生成できるようになり、平均スコアは 328,946、最大スコアは 605,752 を記録した。これは当時としては 1 つの例外^{*1}を除き、計算機による 2048 プレイヤの中で最も優れた成績であった。

一方で、 n タプルネットワークの形状変更や MS-TD 学習のステージングについては、この研究で行なわれた調整についてこれといった根拠が述べられておらず、依然として改良の余地は残していた。特に n タプルネットワークの形状変更については、次の Oka & Matsuzaki の研究で詳しく検討されることとなった。

2.3 Oka & Matsuzaki (2016)

Szubert & Jaskowski は当初図 2.1 のような n タプルネットワークを考案していたが、この n タプルネットワークによる学習器はあまり性能が高くなく、平均スコアは 10 万ゲーム学習した時点で 5 万～6 万程度にとどまっていた。そこで、図 2.2 (a) のような n タプルネットワークへの改良が行われ、その結果平均スコア・勝率ともに改善することができた。さらに Wu は Szubert & Jaskowski の考案した n タプルネットワークを改良し、成績を伸ばすことができた。

しかしながら、ここまでは「タプルのサイズを大きくすると学習器の成績も良くなる」という大まかな関係しかわかっておらず、成績を最善にする n タプルネットワークはどのような形状になるのか厳密には検討されていなかった。これを厳密に検討したのが Oka & Matsuzaki である。

^{*1} Xiao による深深度先読みと人間による調整を行った評価関数を用いたプレイヤーがこれにあたるが、Wu のものよりも 100 倍遅い：<https://www.youtube.com/watch?v=JQut67u8LIg>

表 2.1. 形成されうる n タブルの数

N	3	4	5	6	7	8	9	10	11	12	13
All	77	252	567	1051	1465	1674	1465	1051	567	252	77
Connected	8	17	33	68	119	195	261	300	257	169	66

















N	4 best tuples				4 worst tuples			
	$\langle 6-01 \rangle$	$\langle 6-02 \rangle$	$\langle 6-03 \rangle$	$\langle 6-04 \rangle$	$\langle 6-65 \rangle$	$\langle 6-66 \rangle$	$\langle 6-67 \rangle$	$\langle 6-68 \rangle$
6								
	31,161	24,530	22,207	20,576	9,644	9,642	9,563	9,052
	$\langle 7-001 \rangle$	$\langle 7-002 \rangle$	$\langle 7-003 \rangle$	$\langle 7-004 \rangle$	$\langle 7-116 \rangle$	$\langle 7-117 \rangle$	$\langle 7-118 \rangle$	$\langle 7-119 \rangle$
7								
	32,900	23,504	23,483	23,338	8,543	8,204	7,918	7,683

図 2.4. 最も優れている / 劣っている上位 4 つの 6 タブル・7 タブル (Oka & Matsuzaki (2016))

2.3.1 n タブル単体の性能の評価

Oka & Matsuzaki は、まず n タブル単体の性能を網羅的に評価することを目指した。2048 の盤面上で n 個のタイルを選んで形成されうる n タブルの数は、表 2.1 の通りとなっている。

ここで、Connected とはタブル上の全てのタイルが 1 個以上の他のタブルの接していることを指す。Oka & Matsuzaki は Connected タブルのうち、十分性能が良く、なおかつ現実的に n タブルネットワークとして運用することができる $N = 6$ と $N = 7$ の場合のみ検討することとした。彼らは次に、形成されうる Connected な 6 タブル 68 個の中からランダムに 10 個のタブルを選んで 100 万ゲームの学習を行う実験を 680 回^{*2}繰り返し、各タブルの性能を比較可能な数値として算出した。7 タブルについても同様な実験を行った。

その結果、6 タブルと 7 タブルのうち最も優れた 4 つのタブルと最も劣った 4 つのタブルが、図 2.4 の通り明らかになった。

2.3.2 n タブルネットワークに組み込む n タブルの数の検討

6 タブルおよび 7 タブルの性能が判明したので、性能が良い任意の数のタブルを組み合わせることで n タブルネットワークを作成することが可能になった。Szubert & Jaskowski および Wu は 4 個のタブルを組み合わせることで n タブルネットワークを作成していたが、Oka & Matsuzaki は 5 個以上のタブルを組み合わせることを検討した。すなわち、 n タブルの数が多くなればな

^{*2} $680 \times 10 = 6800$ より、全てのタブルが 100 回は選ばれるようにするため

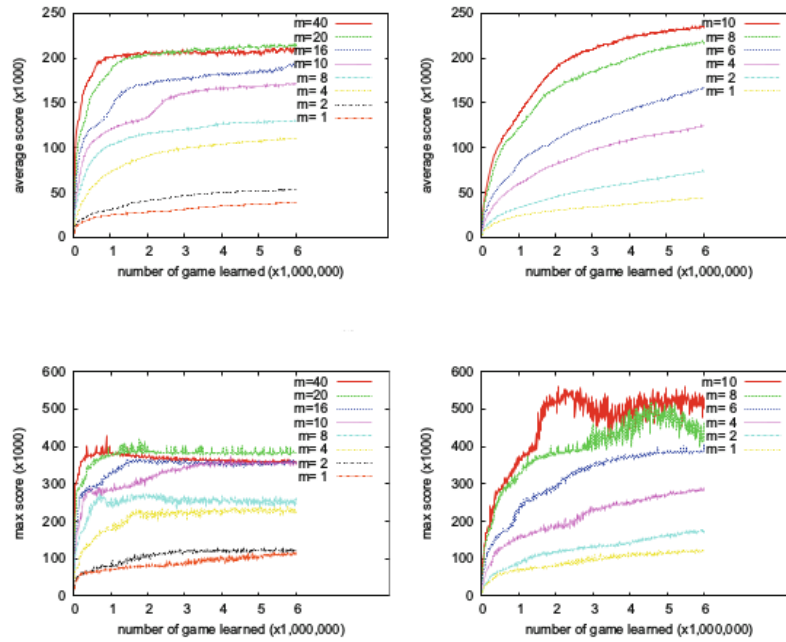


図 2.5. 成績上位タプル m 個を組み合わせたタプルネットワークの実験 (Oka & Matsuzaki (2016))

るほど n タプルネットワーク全体としての性能も上がると思われるが、最も性能が良くなるタプルの数は何個かを明らかにすることである。

実験においては、6 タプルの場合最大で上位 45 個、7 タプルの場合最大で上位 10 個のタプルから n タプルネットワークを作成することとした。各 n タプルネットワークに対して、600 万ゲーム学習を行わせ、10,000 ゲームごとに平均スコアと最大スコアを記録した。その結果が図 2.5 である。なお、 m は実験を行うにあたって上位 m 個の n タプルを採用して n タプルネットワークを生成したことを示す。

上記の通り、実験群の中では 10 個の 7 タプルを組み合わせたタプルネットワークが最も良い成績を収めることがわかった。なお、グラフからも読み取れるが、7 タプルによるネットワークに関しては 600 万ゲーム学習しても平均スコア・最高スコアが収束しない。そのためさらに組み合わせるタプルを増やすとさらに成績が良くなることが考えられるが、技術上の制約により $m = 11$ 以上は実現できなかった*3。

*3 7 タプルを組み合わせてタプルネットワークを作成する場合、重みを保持するために 1 個のタプルにつき 1GB のメモリを消費しなければならない。Oka & Matsuzaki が実験を行った環境はメモリが 12GB しか確保できなかったため、 $m = 11$ 以上は実験を行うことが困難だったのではないかと考えられる。また、この研究においてゲーム木探索を組み込むことができなかったのもメモリの制約が原因ではないかと考えられる。

2.3.3 結果と課題

Oka & Matsuzaki の研究における最も良い n タプルネットワークは、勝率が 0.9850、平均スコアが 234,136、最高スコアが 504,660 という成績を残した。これはゲーム木探索を用いない計算機を用いた 2048 プレイヤとしては最も良い成績であった。しかしながら、ゲーム木探索を用いなかったことによりゲーム序盤では安定性を欠いてしまい、 W_u が達成した勝率 1 を割り込む形となってしまったし、単純に 2048 プレイヤとしての成績を比べると W_u のものよりも平均スコア・最大スコアともに低くなっている。一方で、ゲーム木探索を採用しなかったことはプログラムの高速化という点では利があり、1 秒あたり 88000 手の遷移を行うことができるプログラムとなった。これは W_u の研究と比べて約 290 倍高速である。

2.4 Yeh et al. (2016)

Yeh は Wu et al. (2014) の研究にも参画していた共同研究者で、 W_u の研究の流れを引き継ぐような形で 2048 プレイヤの改良を行った。 W_u の研究で用いた MS-TD 学習の考え方、改良された n タプルネットワーク、Expectimax 木探索は引き続き用いられているため、この研究で新たに加わった要素のみを述べる。

2.4.1 新たな特徴の追加

Yeh は、 n タプルネットワークによる学習の他に、盤面に現れた特徴を評価関数として用いることとした。盤面の評価値には従来の n タプルネットワークによる評価値と新たに採用した特徴による評価関数の和を用い、これを TD 学習で訓練した。なお、採用した特徴は以下のつである。

1. 大きな数が書かれたタイルの個数
2. 何も書かれていないタイルの個数
3. 異なる数が書かれた (distinct) タイルの個数
4. マージすることができるタイルのペアの組数
5. 書かれているタイルの数が $(n, 2n)$ の形になっているタイルのペアの個数

2.4.2 MS-TD 学習のステージングの改良

W_u は MS-TD 学習を行うにあたり 3 ステージに分けて訓練・ゲームプレイを行っていたが、このステージの分け方をさらに細かくした。Yeh によってテストされたステージングは以下の通りである。なお、 T_{x+y+zk} という表記は、「盤面に初めて $1000x, 1000y, 1000z$ のタイルが同時に現れた時」のことを指す。各タイルの数は百の位で切り捨てられて表現されている。

1. 4 ステージ： $T_{8k}, T_{16k}, T_{16+8k}$ を境界としてステージを分ける

2. 4 ステージ : $T_{16k}, T_{16+8k}, T_{16+8+4k}$ を境界としてステージを分ける
3. 5 ステージ : $T_{16k}, T_{16+8k}, T_{16+8+4k}, T_{16+8+4+2k}$ を境界としてステージを分ける
4. 6 ステージ : $T_{16k}, T_{16+8k}, T_{16+8+4k}, T_{16+8+4+2k}, T_{16+8+4+2+1k}$ を境界としてステージを分ける

これらのステージングを、上から順に戦略 1 から戦略 4 とする．まず戦略 1 をテストした際、 T_{8k} を境界としてステージを分けることは意味がないことがわかったため、戦略 2 以降では T_{16k} 以降のステージングのみを考えることとした．戦略 2 から戦略 4 をテストした結果、平均スコアでは戦略 4 が最も優秀であることがわかったが、最高スコアと 32678-tile の到達率では戦略 3 が最も優秀であった．各戦略ごとの成績は表 hoge hoge の通りである．

(表 hoge hoge をここに書きます)

以上の結果より、Yeh は戦略 3 が最も優れたステージングだと判断し、これ以降は戦略 3 を用いて実験を行った．

2.4.3 学習率の調整

TD 学習をより正確に行うために、まず学習率 $\alpha = 0.0025$ で TD 学習を行った後、価値関数に大きな改善が見られなくなったと判断したら、学習率 $\alpha = 0.00025$ に変更して学習を続行させるようにした．

2.4.4 $TD(\lambda)$ の使用

Szuber & Jaskowski および Wu は TD 学習において $TD(0)$ を使用していたが、Yeh は $TD(0.5)$ を用いて 5 ステップの報酬を平均化することとした．すなわち、

(ここに数式を書きます)

で表現される R_t^λ を用いて TD 学習を行うようにした．

2.4.5 結果と課題

上記の改良を重ねた結果、Yeh は平均スコア 443, 526, 最大スコア 793, 835 と非常に良い成績の学習器を訓練することに成功した．なお、32678-tile への到達率は 31.75%, 1 秒あたりの遷移速度は 500 手であった．さらに、検証ゲーム中のある 1 ゲームにおいて 65536-tile を生成することにも成功した．これは強化学習ベースの 2048 プレイヤでは当時として最も強く、Xiao のよる深深度探索ベースのものよりも平均スコアでは優っており、Yeh の方が 125 倍速いプログラムであった．

第 3 章

本研究のアイデア

本研究で導入しようとしている手法のアイデアについて説明します．

第 4 章

提案と実装

前章で説明したアイデアの具体的な提案とその実装方法を説明します。

第 5 章

実験

提案したアイデアの実験結果と既存研究の実験結果を比較します。

第 6 章

考察と結論

実験結果をもとに，結果の考察を行い，本研究をまとめます．

謝辞

謝辞を書きます。

参考文献

[1]

[2]

付録 A

表やプログラムリストの掲載が必要になったらここに掲載します。