# ゲーム「2048」への Monte-Carlo Softmax 探索の適用

1265112　渡邊翔太　【 高度プログラミング研究室 】

# An Application of Monte-Carlo Softmax Search to the 2048 Game

1265112　WATANABE, Shota　【 High-Level Programming Lab. 】

## 1　はじめに

本研究では，G.Cirulli が 2014 年に公開した確率的一人ゲーム 2048 を対象とする．2048 に対するコンピュータプレイヤの多くは，評価関数を貪欲なプレイのもとで TD 学習により学習させたものである．

近年，AlphaZero の登場により，モンテカルロ木探索（MCTS）とニューラルネットワーク（評価関数）を併用した強化学習が注目されている．AlphaZero の手法は様々なゲームに適用可能な反面，膨大な計算資源が必要であることは大きな問題点である．類似の木探索と評価関数を併用する強化学習手法として，評価関数にポリシー関数を必要としない Monte-Carlo Softmax 探索 (MCSS) [1] が提案されている．本研究では，MCSS と強化学習を併用した軽量なアルゴリズムの設計を目指す．まず，MCSS を使用する複数のプレイヤを用いた学習なしでの比較実験を行い，MCSS における選択とバックアップの有効な規則を発見した．また，探索木中の内部ノードを評価関数の学習データとする強化学習の実装を試みた．結果は，過大評価の影響により評価値が安定せず，既存の TD 学習の派生アルゴリズムによる強化学習を越える結果は得られなかった．

## 2　Monte-Carlo Softmax 探索と 2048 に適用する上での問題点

MCSS[1] は，他のモンテカルロ木探索と同様に選択→展開→評価→バックアップを繰り返す．古典的なモンテカルロ木探索との違いは，選択，評価，バックアップにおいて評価関数を用いること，木探索の結果をサンプリングして評価関数の学習データを得ることである．

選択フェーズでは，子ノードの評価値による Boltzmann 分布に従う確率で子を選択する．評価フェーズでは，プレイアウトを行う代わりに評価関数の値をそのまま葉の値とする．バックアップフェーズでは，子ノードの評価値による Boltzmann 分布を重みとして，子ノードの値の重み付き平均を親ノードの値とする．Boltzmann 分布における温度パラメータを調整することで，探索における「探索と活用」をコントロールできる．2048 に MCSS を適用する上で，考慮すべき点は以下の 2 つである：

- 評価値：2048 における評価値は，その状態から将来得られると期待される報酬の総和を表しており，その範囲は 0〜数十万である．
- 過大評価：強化学習の実装に関して，内部ノードの評価値をそのまま学習させると，評価値が無限大に発散してしまう問題が発生した．

これらの点から，2048 に MCSS を適用する際には専用のアルゴリズムを開発する必要がある．まず，評価値の基準が異なるため，選択・バックアップフェーズでの規則についても適切なものを設計する必要がある．また，強化学習の実装に関して，過大評価の影響を抑える学習手法を考える必要がある．

## 3　探索における規則の比較

本実験のために，候補となる規則を 6 種類考案した．これらを選択規則とバックアップ規則それぞれに適用したプレイヤを比較する．1 手あたりのシミュレーション回数 N = 100 を固定し，36 種類のプレイヤを用いた徹底的な比較実験を行った．評価関数には，Matsuzaki による TD 学習を 240 時間行った CNN22 ネットワーク [2] を用いた．本稿では，本実験において優れていた 2 つの規則のみを以下に示す：

**L-max** 子ノードの最大値 $v_{\max}$ を用いて $v' = v/v_{\max}$ とマッピングした値を評価値とし，対象ノードの訪問回数を $n$ として，温度パラメータを $T = 2.0/2^{\log_{10} n}$ とする．

**best** 温度を $T = 0$ と固定する．（最大値を選択）

また，共通してバックアップ規則には best（温度 $T = 0$, 最大値を選択）を用いたプレイヤが優れていた．

次に、 1 手あたりのシミュレーション回数を 25 回から 2000 回の間で変更し，より長い実験を行った。図 1 に，それぞれの平均スコアを示す．用いた CNN22 ネットワークは，1 手先読みで貪欲にプレイした場合の平均スコアが 206 802 であり，MCSS では 50 回のシミュレーションで性能が上回っている．2000 回のシミュレーションのもとでの平均スコア 533 542 は，同じ評価関数を用いた expectimax 3-ply search の結果 (平均スコア 394 632[2]) を大きく上回るものであった．

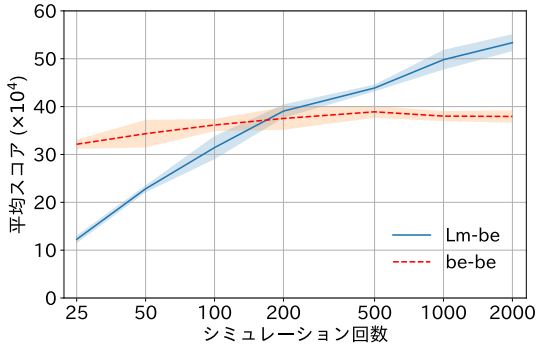


図 1　シミュレーション回数に対する平均スコア

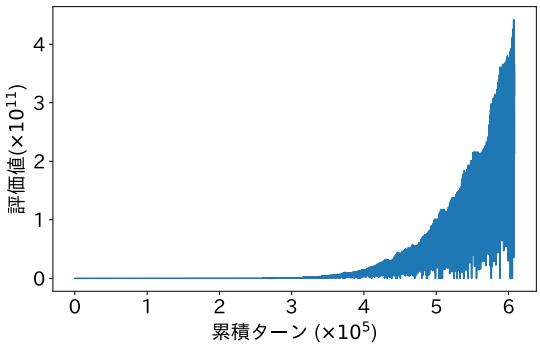


図 2　累積ターンに対する現局面の評価値の推移

## 4　過大評価を避ける強化学習

過大評価の発生については，評価関数には誤差が少なからずあるため，その影響が学習によって修正されないまま大きくなり続けることが原因であると考えられる．過大評価は行動と価値の評価に同一の関数を用いる Q 学習では起こりうることであり，対策として，独立な分布を持つ 2 つの評価関数を用いる Double Q-learning という手法が提案されている [3]．次の実験では， Double Q-learning をベースとした強化学習の実装方法について検討する．以下の方法で探索と学習を行うプレイヤを設計した：

**2 つの評価関数と評価値**　葉ノードの評価において，パラメータの異なる評価関数を 2 つ用いる．それぞれネットワーク A, B とする．それに伴い，探索木中のノードは評価値を 2 つ持つ．それぞれ $evA$, $evB$ とする．

**木の展開**　シミュレーション中，ノードを選択する際の基準となる評価値は，エピソード毎に 50 %の確率で決定する．

**バックアップ**　以下式で評価値の更新を行う．

$$\begin{aligned} i_{\max} &= \mathrm{argmax}_i(children_i.r + children_i.evB) \\ evA &\leftarrow children_{i_{\max}}.r + children_{i_{\max}}.evA \end{aligned}$$

なお，evB の更新方法は式中の $evA$ と $evB$ を逆に読み替えたものである．

**サンプリング**　ターン終了時に学習データのサンプリングを行う．このとき，学習対象となるネットワークは木を展開した際の基準となった評価値とする．また，ネットワーク A には $evB$ を，ネットワーク B には $evA$ を学習させる．

実験で用いるプレイヤは，シミュレーション数 $N = 25$ とし，学習対象となるノードは訪問回数が木の中で上位 5 以内であるものとした． 図 2 に評価値の推移を，図 3 にエピソード毎の平均スコアを示す．

結果として，本実験の方法では値の発散を十分に抑えることはできなかった．

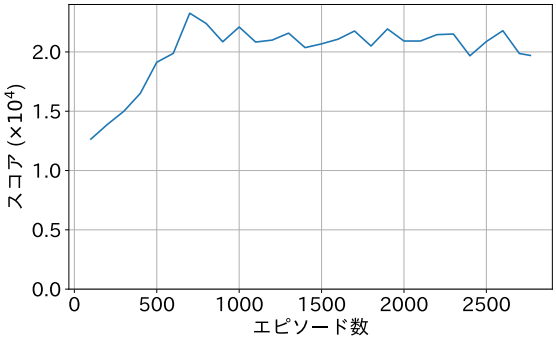


図 3　エピソード数に対する平均スコアの推移

## 5　まとめ

本研究では，2048 に MCSS を実装し，2 つの実験を行った．1 つ目の実験では，選択とバックアップに用いる規則をそれぞれ 6 種類ずつ考案し，それらを組み合わせたプレイヤでの比較実験により，2048 における MCSS での有効な規則を得た．2 つ目の実験では， Double Q-learning の手法を参考に内部ノードの学習による性能向上を目指した．結果は，評価値が安定せず，既存の TD 学習の派生アルゴリズムによる強化学習を越える結果は得られなかった．しかし，探索の有効性が示されており，かつノードが評価値と同等の値を持っている以上，学習データとして活用することが望ましい．そのため，学習方法を改良することは今後の課題である．

## 参考文献

[1] 五十嵐治一，森岡祐一，山本一将，MC Softmax 探索における局面評価関数の学習, ゲームプログラミングワークショップ 2018 論文集, pp. 212–219, 2018.

[2] K. Matsuzaki, Developing Value Networks for Game 2048 with Reinforcement Learning, *Journal of Information Processing*, Vol. 29, pp. 336–346, 2021.

[3] H. van Hasselt, Double Q-learning, *Advances in Neural Information Processing Systems*, 23:2613-2621, 2010.

[4] S. Watanabe and K. Matsuzaki, Enhancement of CNN-based 2048 Player with Monte-Carlo Tree Search, *presented at the 27th Int. Conf. Technol. Appl. Artif. Intell. (TAAI2022)*, Tainan, Taiwan, Dec. 1–3, 2022.