CF-004

ガイスターを用いたR-Nadアルゴリズムの不完全情報ゲーム適用の実験的検証

Experimental verification of the application of the R-Nad algorithm to incomplete information games with Geister

甲斐 陸斗[1)]　河原 吉伸[1) 2)]
Rikuto Kai　Yoshinobu Kawahara

1 序論

1.1 研究の背景

昨今，この世界には多種多様のゲームが溢れている．例えばチェス，将棋，囲碁といった古くからあるボードゲーム，オセロやカタンの開拓者といった近代に生み出されたボードゲーム，ポケモンのようなビデオゲームなどがあり，その種類は多岐にわたる．これらのゲームは，大きく 2 つの種類に分けることができる．全てのプレイヤーに，ゲーム上の全ての情報が共有されている完全情報ゲームと，各プレイヤーしか知り得ない情報が，存在する不完全情報ゲームの 2 つである．例えばチェス，将棋，囲碁はお互いのプレイヤーが，盤面の全情報を確認することができるため完全情報ゲームであり，ポーカーや麻雀は相手の手札や牌が分からないため，不完全情報ゲームである．

人工知能分野において，これらのゲームを攻略しようとする研究は盛んに行われている．特に，深層学習が機械学習の主流になってからの発展は目まぐるしく，2016 年には囲碁でプロを完封した AlphaGo[1] が話題となった．現在ではチェス，将棋，囲碁といった完全情報ゲームは，AI が生み出した定石をプロが用いるなど，完全にプロを上回るほどの実力を持っている．一方でポーカーや麻雀などの不完全情報ゲームの研究では，プロレベルに至っている AI は少なく，未だに研究が盛んに行われている．

1.2 研究の目的

本研究では，Google DeepMind が提唱したアルゴリズム，R-Nad[2] に二人零和不完全情報ゲームであるガイスターを適用して，実験的に検証する．検証によって，R-Nad が他のアルゴリズムと比べて，どの程度の性能を示すかを確かめる．また，R-Nad を用いた AI が論文内で示した程度の性能を，他のゲームで再現できるかどうかを確かめ，アルゴリズムの問題点や改善点を探る．

1.3 構成

本稿の構成は以下のとおりである．2 章で本研究の関連研究について述べ，3 章で評価実験について述べる．そして，4 章で実験結果について述べる．最後に，5 章で結果や対戦戦績についての考察について述べ，6 章で本研究の結論を述べる．

1) 大阪大学 大学院 情報科学研究科
Graduate School of Information Science and Technology, Osaka University
2) 理化学研究所 革新知能統合研究センター
Institute of Physical and Chemical Research Center for Advanced Intelligence Project

2 関連研究

この章では，本研究に関連する研究について概説する．まず，2.1 節，2.2 節で本研究で使用されているアルゴリズムである，Deep Q Network と R-Nad とについてそれぞれ概説する．

2.1 Deep Q Network

Deep Q Network(DQN)[3] は，Q 関数を深層畳み込みニューラルネットワークで近似して，学習するアルゴリズムである．

Q 関数とは最適行動価値関数のことであり，以下のように定義される．

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

ここで s は状態，a は行動であり，r は報酬である．また，γ は割引率であり，この値が高いほど未来で得られる報酬を重視する．

この式はベルマン最適方程式ともよばれ，直感的には状態 s の次の状態 s' において，最適行動関数価値 $Q^*(s',a')$ が全ての行動 a' について既知であるとき，最適戦略は $r + \gamma \max_{a'} Q^*(s',a')$ の期待値を最大にする行動 a' を選ぶこと，と定義することにより得られる定式化である．

Q 学習では以下のような反復更新によって，行動価値関数を推定する．

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

ここで，i は反復回数である．Q_i は $i \to \infty$ のときに，Q^* に収束することが示されている．

しかし，この推定は状態 s が連続値もしくはそれに近い値のときに，状態数が無限になるため学習することができない．そこで，$Q(s,a;\theta) \approx Q^*(s,a)$ となるようなニューラルネットワークを用いた関数近似によって，行動価値関数を推定する．重み θ をもつ上記の関数近似器を Q-network と呼ぶ．Q-network は，以下のように定義される損失関数 $L_i(\theta_i)$ を最小化することで，学習することができる．

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

ただし，

$$y_i = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q(s',a';\theta_i^-)\right]$$

である．θ_i^- は一定間隔ごとに $\theta_i^- \leftarrow \theta_{i-1}$ で更新される．このようにパラーメーターの更新の際に古いパラメーターを用いることで，パラメーターの更新が影響を与えるまでに遅延が入り，発散や振動が起こりにくくなる．

2.2 R-Nad

R-Nad[2] とは二人零和ゲームに適用できる，モデルフリーの強化学習アルゴリズムである．このアルゴリズムは以下の 3 つのステップから成る．

1. Reward transformation step
2. Dynamics step
3. Update step

R-Nad はこの 3 つのステップを繰り返すことで，学習を進めていく．

Reward transformation step では正則化方策 π_{reg} に基づいて，以下のように定義された式によって報酬を変形する．

$$r^i_{\pi_{\text{reg}}}\left(\pi^i, \pi^{-i}, a^i, a^{-i}\right) = r^i\left(a^i, a^{-i}\right) - \eta \log\left(\frac{\pi^i\left(a^i\right)}{\pi^i_{\text{reg}}\left(a^i\right)}\right) + \eta \log\left(\frac{\pi^{-i}\left(a^{-i}\right)}{\pi^{-i}_{\text{reg}}\left(a^{-i}\right)}\right)$$

このとき，π は方策，r は報酬，η は正則化パラメーターである．また i はそれぞれのプレイヤー番号であり，$-i$ は i ではない方のプレイヤーのことを意味する．初期の正則化方策は，行動をとる確率が全て 0 となるような方策以外であれば任意の値でよい．

Dynamics step では以下で定義される the replicator dynamics system[4, 5, 6, 7] を繰り返し実行する．

$$\frac{d}{d\tau}\pi^i_\tau\left(a^i\right) = \pi^i_\tau\left(a^i\right)\left[Q^i_{\pi_\tau}\left(a^i\right) - \sum_{b^i}\pi^i_\tau\left(b^i\right)Q^i_{\pi_\tau}\left(b^i\right)\right.$$

ただし，$Q^i_{\pi_\tau}\left(a^i\right) = \mathbb{E}_{a^{-i}\sim\pi^{-i}_\tau}\left[r^i\left(\pi^i_\tau, \pi^{-i}_\tau, a^i, a^{-i}\right)\right]$ である．また，τ は時間を表す．$Q^i_{\pi_\tau}\left(a^i\right)$ はあるプレイヤー i の対戦相手の方策が π^{-i}_τ であるときに，行う行動 a^i の価値を表す．すなわち Dynamics step では，行動価値 $Q^i_{\pi_\tau}(a^i)$ が行動価値の期待値 $\sum_{b^i}\pi^i_\tau\left(b^i\right)Q^i_{\pi_\tau}\left(b^i\right)$ を上回っている場合は，行動 a^i が選択される確率を上昇させ，期待値を下回っている場合は，行動 a^i が選択される確率を減少させるように，方策 π を更新する．この微分方程式は Reward transformation step での報酬変換により，方策 π がある不動点 π_{fix} に収束することが，リアプノフ関数 $H_{\pi_{\text{fix}}}(\pi) = \sum_{i=1}^{2}\sum_{a^i\in A^i}\pi^i_{\text{fix}}\left(a^i\right)\log\left(\frac{\pi^i_{\text{fix}}\left(a^i\right)}{\pi^i\left(a^i\right)}\right)$[8] から示される．ここで A^i はプレイヤー i が取り得る行動の集合である．

Update step では Dynamics step で求めた方策 π_{fix} を，次の繰り返しの正則化方策 π_{reg} とする．すなわち，Dynamics step の n 回目の繰り返しで求められる π_{fix} を $\pi_{n,\text{fix}}$ と表すと，$n+1$ 回目の Reward transformation step での正則化方策 $\pi_{n+1,\text{reg}} = \pi_{n,\text{fix}}$ として更新する．

これらの 3 つのステップを繰り返すことによって，n が十分大きい時，方策 $\pi_{n,\text{fix}}$ とのナッシュ均衡 π_{nash} の KL 情報量は 0 へと近づく．

3 評価実験

この章では，R-Nad を実験的に検証した際の，評価実験の詳細について述べる．まず，3.1 節で実験手法について，3.2 節で使用した環境について，3.3 節で使用したアルゴリズムについて説明する．

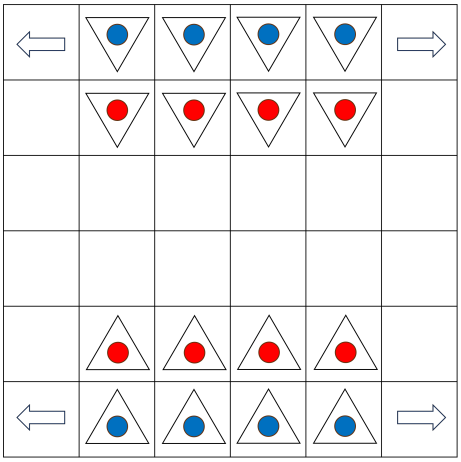

図 1 ガイスターの初期盤面例

3.1 実験手法

実験には，オープンソースフレームワークである OpenSpiel[9] を用いて，R-Nad に 3.2.1 項で説明するガイスターというゲームを適用して学習させた．また比較のために，Deep Q Network(DQN)[10] にもガイスターを適用して学習させた．R-Nad および DQN を用いて学習したモデルを，それぞれランダム bot や人間と対戦させるとともに，両モデル間の対戦も行い，その戦績を記録した．

3.2 使用環境

3.2.1 ガイスター

ガイスターとは，2 種類の駒を用いて遊ぶ二人零和確定不完全ゲームである．二人零和確定ゲームとは，二人でプレイするゲームのうち片方が利得を得たときに，対戦相手にそれと等価な損失が生じるようなゲームであり，対戦にランダム要素が絡まないゲームのことである．

ガイスターの盤面は 6 × 6 で構成されており，各角のマスが脱出マスと呼ばれるマスである．駒は赤いオバケと青いオバケの 2 種類で，各プレイヤーは図 1 のようにゲーム開始前に，それぞれ 4 つずつの駒を相手に配置場所を知られないように，手前 2 列の両端を除いた 8 マスに自由に初期配置する．各駒は 1 度の移動で前後左右に 1 マスのみ動くことができ，移動先に相手の駒が存在する場合はその駒を捕まえることができる．捕まえた駒はゲームから除外され，再利用することはできない．勝利条件は相手の青いオバケを全て捕まえること，相手が自分の赤いオバケを全て捕まえること，相手側の脱出マスから青いオバケを 1 つ盤外に出すことの 3 つである．

このゲームの行動数は，盤面のどのマス (36 マス) がどの方向 (4 方向) に動くか，で表現することができるので 36 × 4 = 144 通り存在する．

今回の実験では簡易化のため，駒を自由に配置するのではなく，ランダムに配置することとする．

3.3 アルゴリズム

3.3.1 R-Nad

実験に使用する R-Nad のアルゴリズムは，基本的には 2.2 節で説明したものであるが，R-Nad のアルゴリズム

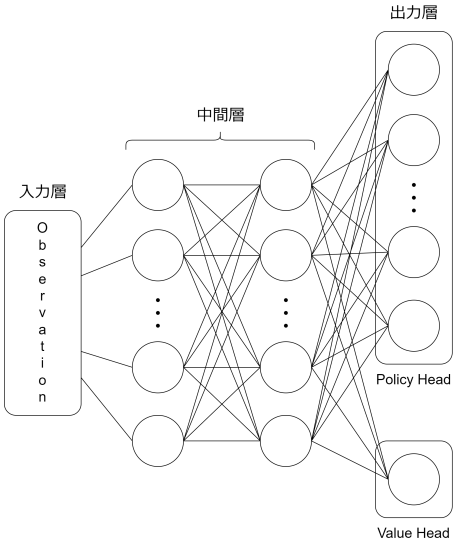

図 2　R-Nad の学習に使用した MLP

表 1　状態 s と盤面の状態の対応

s の値	状態の説明
0	マスに自分の青いオバケがおいてある状態
1	マスに自分の赤いオバケがおいてある状態
2	マスに敵の青いオバケがおいてある状態
3	マスに敵の赤いオバケがおいてある状態
4	マスに何も駒がおかれていない状態

を規模の大きなゲームで適用できるように，方策の推定に MLP を使用している．

使用した MLP の構造を図 2 に示す．ただし 2 層の中間層のノード数は 256 であり，中間層 2 層目の出力には活性化関数として，ReLU が用いられている．

入力は $8 \times 6 \times 6$ の観測情報のテンソルであり，テンソルの各値を $[s,r,c]$ で表すとすると，テンソルの各値 $[s,r,c]$ は盤面 (r,c) 上の状態が，s である確率を表す．ただし r は行を表し c は列を表す．また，状態 s と盤面の状態の対応を表 1 にまとめた．

出力層は Policy Head と Value Head の 2 つに分かれる．Policy Head は行動に対するロジットを，取り得る行動数の次元である 144 次元のベクトルで，Value Head は方策の評価値を 1 次元のベクトルで出力する．

4　実験結果

この章では 3 章で説明した評価実験の実験結果について述べる．まず，4.1 節で学習した際の損失関数と報酬の例を示す．次に 4.2 節で R-Nad で学習したモデルを対戦させた際の戦績を示す．

4.1　学習結果

R-Nad で，30 万 step 学習をしたときの損失関数の例を図 3 に示す．

図 3 を見ると，損失関数が収束し正しく学習できていることがわかる．

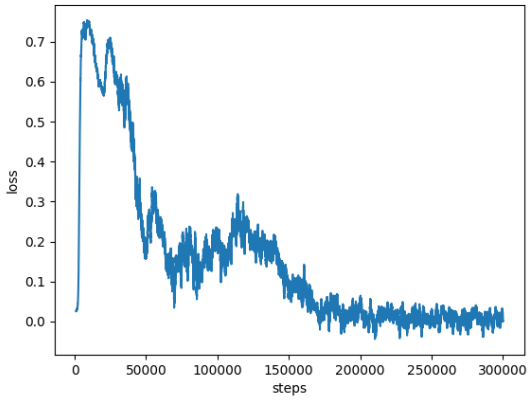

図 3　R-Nad で学習した際の損失関数 (移動平均) (Window Size = 1000)

表 2　ランダム bot との対戦戦績

学習アルゴリズム	試合数	勝ち	引き分け	負け
R-Nad	1000	89.2%	0.0%	10.8%
DQN	1000	97.8%	0.0%	2.2%

4.2　対戦戦績

R-Nad で学習したモデルの性能評価をするために，DQN で学習したモデル，ランダム bot[1)]，人間とそれぞれ対戦させその結果を記録した．また，比較のため DQN で学習したモデルとランダム bot も対戦させ，その結果を記録した．このゲームにはルール上引き分けが存在しないが，対局中お互いが 1000 手以上打って，決着がつかない場合は引き分けとした．

R-Nad と DQN でそれぞれ学習させたモデルと，ランダム bot との対戦戦績を表 2 に示す．

また，R-Nad と DQN との対戦戦績を表 3 に示す．

人間との対局では，ガイスターを過去にプレイしたことのある人物 1 名と 10 試合以上対戦したが，R-Nad は一度も勝利することができなかった．また，過去にガイスターをプレイしたことがなく，評価実験の際に初めてガイスターのルールを説明した人物 2 名と，計 4 試合対戦したが，同様に R-Nad は一度も勝利することができなかった．

R-Nad が打つ手の傾向として，対戦初期は人間と非常に酷似した手を打つ．例えば序盤では，相手の陣地にある脱出マスへ向けて駒を動かしたり，逆に相手が自分の脱出マスに近づくとその駒を取る，といった手を打つことができる．しかし，戦局が進むにつれ次第に攻めてきた対戦相手の駒を取るという保守的な動きのみを取るようになった．特に盤面の駒が少なくなるほど，この傾向は顕著になった．

表 3　DQN との対戦戦績

試合数	勝ち	引き分け	負け
1000	86.9%	2.8%	10.3%

1)　取り得る有効手の中から，行動を等確率で選択するエージェントのこと．

5 考察

表 2 より，R-Nad と DQN はともにランダム bot よりも，強いと考えることができる．また，表 3 より，R-Nad は DQN よりも強いと考えることができる．これにより，R-Nad は不完全情報ゲームに対して DQN などの既存アルゴリズムよりも，不完全情報ゲームに適していると考える．

しかし，R-Nad のランダム bot に対する勝率は，DQN のランダム bot に対する勝率よりもわずかに低い．これは，R-Nad の学習アルゴリズムがナッシュ均衡に陥るように，すなわち相手もその盤面における，最善手を打ってくることを前提としているのに，相手がランダムに行動するため，学習した結果を正しく利用できないからであると考える．

同じアルゴリズムを使用している DeepNash[2] は，人間のプロ相手に高い勝率を記録していたが，本研究では人間ほどの性能には達していない．これは学習に使用しているニューラルネットワークが，DeepNash が使用しているものと異なるためであると考える．具体的には，使用するニューラルネットワークを MLP からもう少し層が深いものや，CNN に変更すれば性能は向上すると考える．

6 結論

本研究では，R-Nad に対して不完全情報ゲームであるガイスターを適用し，その評価実験を行った．その結果，取り得る有効手の中から行動を等確率で選択する相手や，既存のアルゴリズムよりも性能が高いことが示された．しかし，その性能はまだアマチュアレベルの人間にも遠く及ばないことも分かった．

本研究には課題がいくつかある．本研究では学習の簡易化のために，ガイスターの駒の初期配置は互いにランダムとしたが，本来のルールではお互いに駒を指定の 8 コマに自由に配置することができる．これによりさらに戦略の幅が生まれるため，このルールを採用したうえで R-Nad を適用した際に性能がどのようになるかを調べる必要がある．

また R-Nad は，方策のナッシュ均衡を求めるアルゴリズムであるが，本研究では求めた方策が本当にナッシュ均衡に収束しているかを調べていない．これは，規模の大きなゲームではナッシュ均衡解を求めるのが難解であるからである．これを解決するための研究はいくつかあり，例えば近似的な最適反応と近似的ナッシュ収束度を定義して，求めた方策がどの程度ナッシュ均衡に近づいているかを確かめる手法 [11] がある．この手法を使うなどで，本研究で求めた方策が本当にナッシュ均衡に収束しているかを調べる必要がある．

さらに，本研究では学習の際に MLP を用いているが，これを U-net のような CNN に変更した際に，どの程度性能が向上するかを調べる必要がある．

アルゴリズムそのものの改善も今後の課題である．現在の R-Nad アルゴリズムでは，学習の際に自分の観測情報のみを与えて学習しているが，相手の観測情報も同時に与えて学習させた場合や，ルールベースの手法を組み合わせた際に，更なる性能の改善があるかどうかを調べることも必要である．

参考文献

[1] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, Vol. 529, No. 7587, pp. 484–489, 2016.

[2] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, Vol. 378, No. 6623, p. 990–996, December 2022.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. 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. *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

[4] E. C. Zeeman. Population dynamics from game theory. In Zbigniew Nitecki and Clark Robinson, editors, *Global Theory of Dynamical Systems*, pp. 471–497, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg.

[5] E. Christopher Zeeman. Dynamics of the evolution of animal conflicts. *Journal of Theoretical Biology*, Vol. 89, pp. 249–270, 1981.

[6] J.M. Smith. *On Evolution*. Edinburgh University Press, 1972.

[7] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, Vol. 53, pp. 659–697, 08 2015.

[8] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139 of *Proceedings of Machine Learning Research*, pp. 8525–8535. PMLR, 18–24 Jul 2021.

[9] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. Openspiel: A framework for reinforcement learning in games, 2020.

[10] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, USA, 1992. UMI Order No. GAX93-22750.

[11] Finbarr Timbers, Nolan Bard, Edward Lockhart, Marc Lanctot, Martin Schmid, Neil Burch, Julian Schrittwieser, Thomas Hubert, and Michael Bowling. Approximate exploitability: Learning a best response in large games, 2022.