知能情報実験 III（データマイニング班）
DQN を用いた 3 次元リバーシ AI の特徴の考察

185732B 船附海斗, 185734H 高良隼,
185743G 大城祐介, 185751H 長濱北斗

提出日：2021 年 2 月 15 日

目次

概要

本文書は強化学習の中でも DQN（Deep Q-Network）法を用いたリバーシの強化学習を行った報告書である．各局面において置ける箇所にランダムに手を打つプレイヤーと，DQN 手法を用いて手を打つプレイヤーを対戦させ，ハイパーパラメータの変化によって勝率がどう変化するか実験を行った．実験に用いる DQN のエージェントの学習環境となるリバーシは，一般的な平面のリバーシに高さを追加した 3 次元のリバーシである．ハイパーパラメータには，バッチサイズ，e の下限値を変化させて，適当なバッチサイズで学習させた場合に，e の下限値を減らすことで勝率が伸びた．

1　はじめに

近年，強化学習を行った AI が囲碁の世界チャンピオンを倒したことで話題になった [1]．そこでリバーシゲームを用意して，強化学習を用いた AI の開発と手法を検討することにした．実験を通して強化学習と深層学習を学ぶために，両方を組み合わせたアルゴリズムである DQN を取り組むことにした．

1.1　本実験の位置づけ

リバーシや囲碁といったゲームの強化学習は多くの先行研究が見られる [2]．深層学習をする上でメジャーな 2D リバーシを 3D に拡張することで方策や戦略の幅を増やすことで，DQN による自律的な学習がモンテカルロ木探索などに代表されるヒューリスティックな探索より，多様で十分に探索が困難な学習環境に対して強いという印象があった．そこで，実際にその二つを検討してみようと試みたが，モンテカルロ木探索の実験が未完遂であるので，DQN による学習の手法について各ハイパーパラメータの値を変えながら当該ハイパーパラメータの特性を実験にて調査した．

2　実験方法

2.1　実験目的

DQN のパラメータの変化によって勝率がどう変化するか，DQN の特徴を考察する．

2.2　モデル選定

実験に用いるアルゴリズムは，DQN を選択した．理由として，先行研究 [3] が多数あり，情報も豊富なため参考にしやすい点から採用に至った．DQN のアルゴリズムのコードは， [4, 5] を参考にした．

2.3 パラメータ調整

今回，実験で行うハイパーパラメーターの調整はバッチサイズと e の下限値である．

- バッチサイズ
  今回バッチサイズとして用いた値は MinibatchSDG におけるバッチサイズとは異なり，今回は学習する観測データのキューの最大サイズを 10,000 としており，この中からランダムに学習するデータを取り出す．一回の学習で用いるデータの数をバッチサイズとし，ミニバッチを形成しないため，ネットワークの更新は一回である．また学習及びネットワークの更新は手を一手打つたびに行う．本実験では，バッチサイズを 64，128，256 の幅で調整した．
- e の下限値
  e とは，ε-greedy 法において，エージェントが行動選択を行う際に利用される値である．強化学習は探索を行いながら学習していくが，探索のみを行っていては最適な判断ができるようにならない．どこかで経験を「活用」する必要がある．ε-greedy 法は，確立 e で環境の探索を行い，それ以外では今までの経験を活用し，より高い報酬を得ることを期待してその時点で最も価値が高いと判断した行動を取る．実際の ε-greedy 法では，序盤は e を大きい値に設定して積極的に環境の探索を行って経験を蓄積し，徐々に減じて最も行動価値の高い手を打つ割合を増やしていくようにする方式がよく用いられている．全く探索しなくなって，局所解に陥ってしまうことを防ぐために，e に下限値を設定し，一定の割合で探索を行わせることが必要である．本実験では，e の下限値を 0.05，0.20，0.50 の幅で調整した．

2.4 実験手順

DQN を適用したプレイヤーとランダムに手を打つプレイヤーを対戦させる実験を行った．ゲームを用意するために 3D リバーシを実装し，ランダムで合法手を打つ CPU を用意した．CPU は，盤面から全ての合法手を取得し，その中からランダム関数を用いて打つ手を決める動きを行う．エージェントとなる DQN の実装に伴い，ニューラルネットワークの実装も行った．今回は，MLP(多層パーセプトロン) でモデルを作成した．その構造を，次に示す．以下に示す層は全て全結合層である．

- 入力層 (入力サイズ 128, 出力サイズ 256)
- 中間層 1(入力サイズ 256, 出力サイズ 256)
- 中間層 2 (入力サイズ 128, 出力サイズ 256)
- 出力層 (入力サイズ 256, 出力サイズ 64)

以下，図 1 にネットワークの概要図を示す．

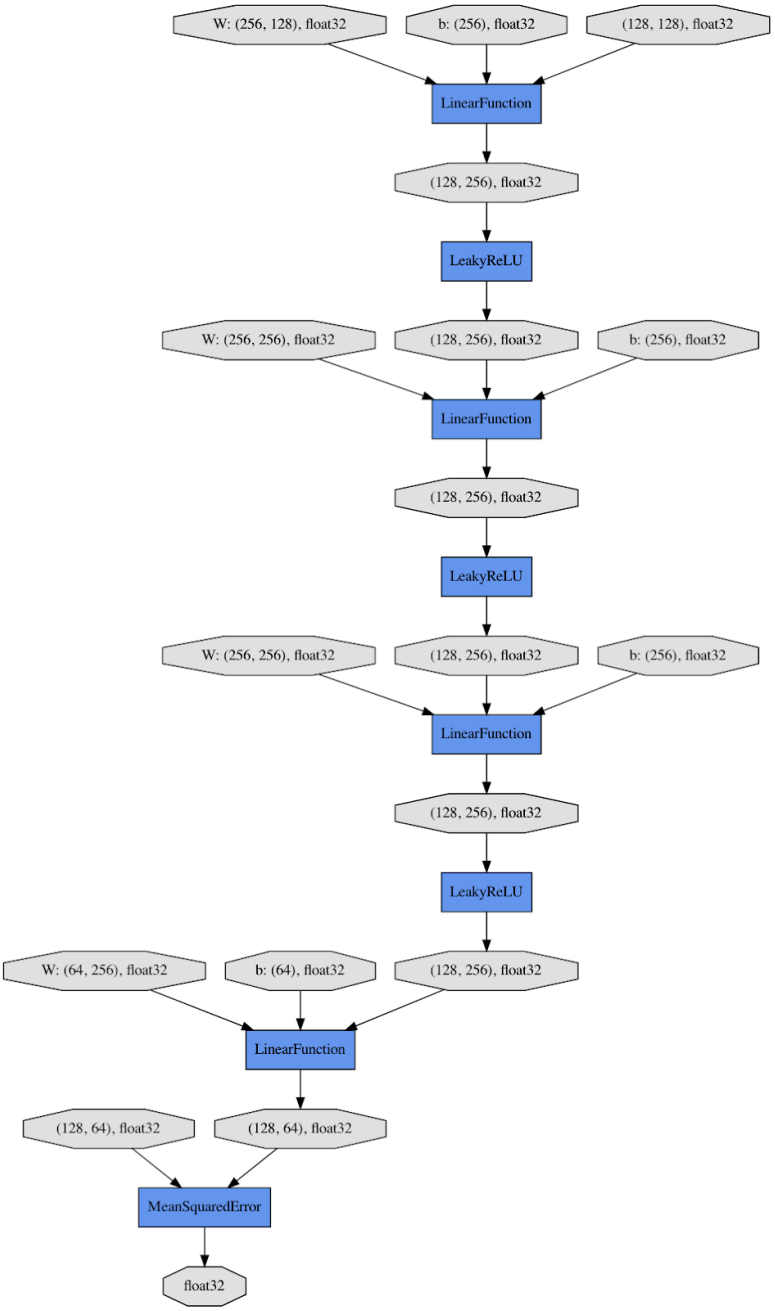

図１　ネットワークの概要図

盤面状態データを入力として与え，教師データには盤面状態データに対して石をおいた際の行動価値を与えるようにした．盤面状態を s, s に対して行う行動を a, 次の盤面状態を s’, s’ に対して行う行動を a’ とすると，行動価値 $Q(s,a)$ は，次のような式で表される．

$$Q(s,a) = r + (1-t) + \gamma max Q(s', a')$$

t は，状態 s に対して行動 a を行った時にゲームが終了する場合のみ $t=1$ になり，それ以外では $t=0$ である．また，γ は行動価値 Q を更新する際に，次状態の評価の期待値をどれだけ考慮するかを定めるパラメータであり，γ の値は 0.95 とした．

3　実験結果

各バッチサイズで e の下限値を変化させた，実験結果を以下に示す．また，各図のグラフは 1000 試合ごとの勝率をプロットしており，縦軸に勝率，横軸に試合数を表している．

表 1 から，バッチサイズが 64 で e の下限値が 0.05，0.20，0.50 のとき勝率と引き分け率の和は約 48.8%，約 54.3%，約 53.0% となった．図 2 は，e の下限値ごとの平均勝率の変化である．

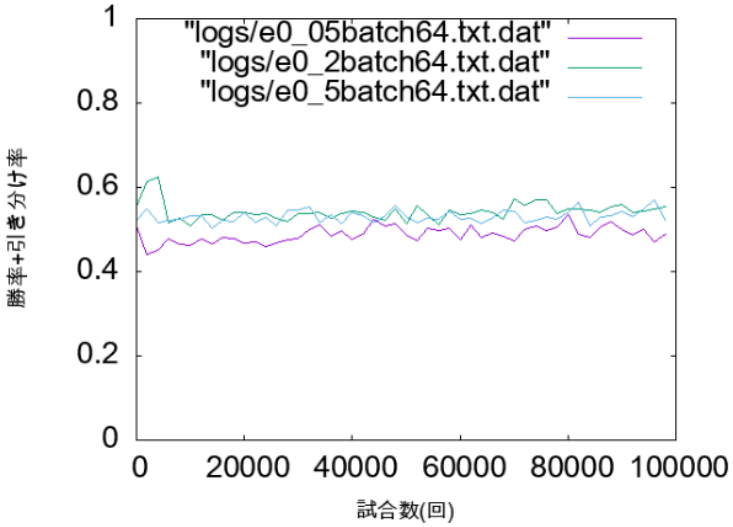

図 2　バッチサイズが 64 の時の 1000 試合毎の勝率の変化

表 1 から，バッチサイズが 128 で e の下限値が 0.05，0.20，0.50 のとき勝率と引き分け率の和は約 67.8%，約 58.9%，約 57.8% となった．図 3 は，e の下限値ごとの平均勝率の変化である．

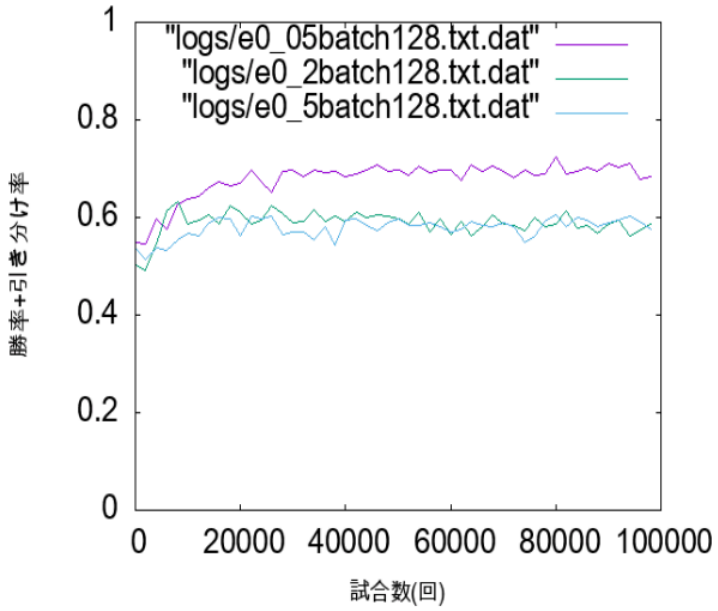

図 3　バッチサイズが 128 の時の 1000 試合毎の勝率の変化

表 1 から，バッチサイズが 256 で e の下限値が 0.05，0.20，0.50 のとき勝率と引き分け率の和は約 56.9%，約 48.1%，約 49.7% となった．図 4 は，e の下限値ごとの平均勝率の変化である．

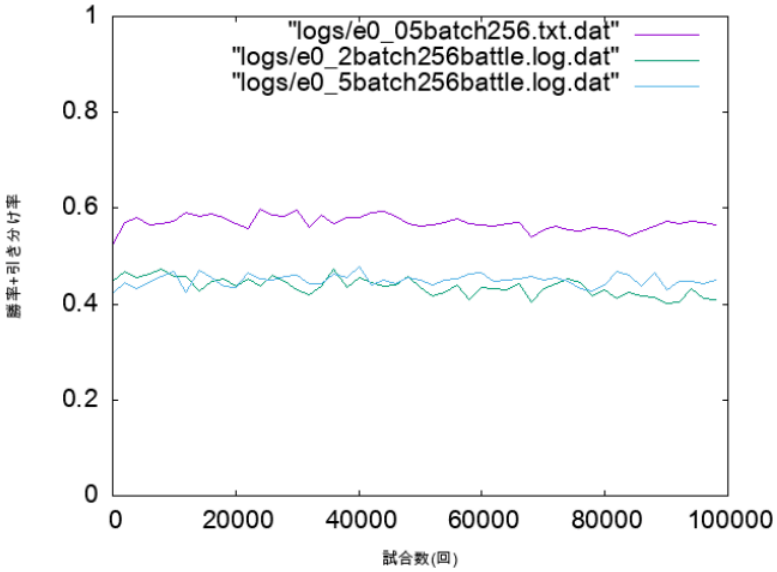

図 4　バッチサイズが 256 の時の 1000 試合毎の勝率の変化

表 1　各バッチサイズの e の下限値と勝率・引き分け率の和の表

e の下限値 ＼ バッチサイズ	64	128	256
0.50	0.530	0.578	0.497
0.20	0.543	0.589	0.481
0.05	0.488	0.678	0.569

4　考察

バッチサイズは特に e の最も低い $e = 0.05$ で比較した時，128 が最も高い勝率を示した．これはバッチサイズが大き過ぎると対応できる状況が限られてしまい，小さ過ぎると十分に学習結果をネットワークに反映することが困難になるためだと考えられる．バッチサイズ 128 のとき e の下限値が低いほど勝率が上がったのは，行動価値の高い手を選択しやすいためだと考えられる．バッチサイズが 64,256 の時は，学習のどの段階をみても十分な勝率の上昇が見れないのでそもそも観測データからの学習に失敗していると考えられる．

5　意図していた実験計画との違い

意図していた実験計画との違いとして，以下の 3 つが挙げられる．

- 環境の用意（3D リバーシの完成）に手間がかかった．
- アルゴリズムを 4 つ比較するはずが，1 つしか完成に間に合わなかったため他のアルゴリズムとの比較ができなかった．
- 1 つのアルゴリズムを使用するとなった時，ハイパーパラメータの変化による勝率をみる方向性にした．

アルゴリズムを比較する件に関しては，各アルゴリズムを下調べして実装の難易度を把握し，自分たちの力量を見定めた計画を練る必要があったと考えられる．

6　まとめ

今回実験で調整したハイパーパラメーターは 2 種類だけだったが，他のハイパーパラメーターも存在する．今回の実験だけで最終的な勝率を判断することは難しいため，より良い実験結果を得るためには，他のハイパーパラメーターの調整やランダムプレイヤー以外に他のアルゴリズム同士，実際にプレイヤーとの対戦を交えるなど検証項目を増やすことが良いと考えられる．

7　振り返り

タスクの難易度にあった人数の割り振りをする．学習する観測データの収集にはランダムプレイヤーとの対戦を用いたが，そうではない別の戦略を持った相手から集めた方がより高度な戦略を備えた DQN ができた．また，一辺が 4 の盤面だと十分な戦略の幅が生まれないので実装段階ではそれでいいが，実験に使うなら一辺が 6 や 8 の盤面を用いるとより高度に学習した AI が勝ちやすい環境になるので実験に適している．

活性化関数に関しては LeakyReLU を用いたが，本当にマイナスの値を軽減して反映させるべきかを検討し，その他の活性化関数にも目を向けてるべきだった．バッチサイズやミニバッチ勾配降下法についての理解が浅く，一般的な定義と実装内容で乖離があった．観測データのうち学習データをランダムに抽出しているが，学習データをバッチサイズの数だけのミニバッチに分割してその数だけネットワークを更新するのが一般的なやり方であった．

実験で途中や最終の性能評価に使った DQN プレイヤーは実際のところ，性能評価相手であるランダムプレイヤーから学習している．これをとりのぞかないと適正な性能評価とはいえない．

参考文献

[1] Wikipedia AlphaGo, https://ja.wikipedia.org/wiki/AlphaGo, 2021/1/20.

[2] 小松泰士, 山内ゆかり, 動的強化学習ネットワークにおける特徴抽出と強化学習の融合, 2020-12-12, http://www.cit.nihon-u.ac.jp/laboratorydata/kenkyu/kouennkai/reference/No.53/pdf/2-18.pdf, 2021/2/12.

[3] 山岡勇太, CFR 法と強化学習法の格闘ゲーム人工知能への適用, 2020-3-18, https://uec.repo.nii.ac.jp/?action=repository_uri&item_id=9532&file_id=20&file_no=1, 2021/2/12.

[4] GitHub RL_reversi, https://github.com/ryogrid/RL_reversi, 2020/12/8.

[5] GitHub Learning-Machine-Learning, https://github.com/Kumapapa2012/Learning-Machine-Learning, 2021/1/19.