2022 年 12 月 6 日 B3 西村昭賢

進捗報告

1 今週やったこと

- 学習ステップ数を増やした DQN の結果
- DQN の実験の結果を踏まえた改善
- DQN の再実験
- コスト, HP , 特殊効果追加した ver の環境の作成

2 学習ステップを増やした DQN の実験結果

先週, DQN で 200000 ステップ (約 5000 エピソード) 学習した結果先手が勝率 5 割を切るという結果となった. 一方, 新たに実装したモンテカルロ探索では 1000000 エピソード学習し先手で 0.8011 という勝率を残した.

この結果を受け, DQN のステップ数を増やして学習が行われるか実験した. パラメータは表 1 に示す.

表 1: DQN のパラメータ

方策	ε -greedy
ε	0.1
全結合層の活性化関数	ReLU
全結合層の次元	64
最適化アルゴリズム	Adam
Target Network 更新重み	1e-2
Experience Replay のメモリ量	1000000

結果, 10000000 ステップ先手で学習し, 勝率は 0.3631 となった. 学習の過程を図 1 に示す. 全く学習できてないので環境側の問題, モデル側の問題の 2 つに分けて改善を図った.

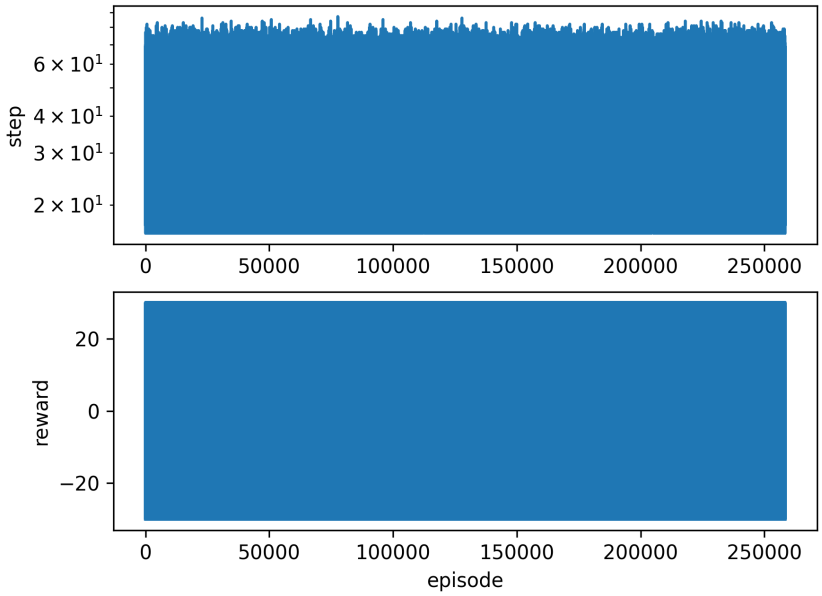

図 1: 10000000 ステップの学習結果

3 DQN の実験の結果を踏まえた改善

3.1 環境側の改善

以下のような改善を図った.

- ライブラリを変更
- 行動空間の定義の変更
- 終了条件の変更
- 報酬の変更
- 状態空間の定義の変更

3.1.1 ライブラリ変更

学習プレイヤーと敵プレイヤーのライブラリを表 2 のようにして同じライブラリにした.

表 2: ライブラリ () 内の数字は (攻撃力, HP) を表す

カード	枚数
(3,3)	5
(2,4)	5
(2,3)	5

3.1.2 行動空間の変更

以前は「自盤面のターン中に行動可能なカードが全て行動すると自動的にターンエンド」としていた. 上手く行かない学習過程を見てみると, 盤面にカードが存在しない時にこの条件では 1 枚カードをプレイして強制的にターンエンドとなってしまっていることに気づいた.

この問題を解決するために手札のカードに対しても表 3 に示すように「プレイしない」という選択肢をつけて,「自手札, 自盤面全てのカードについてターン中に何をするか決定したら自動的にターンエンド」と条件を変更した.

ゲームのイメージは図 2 の通り変更はない.

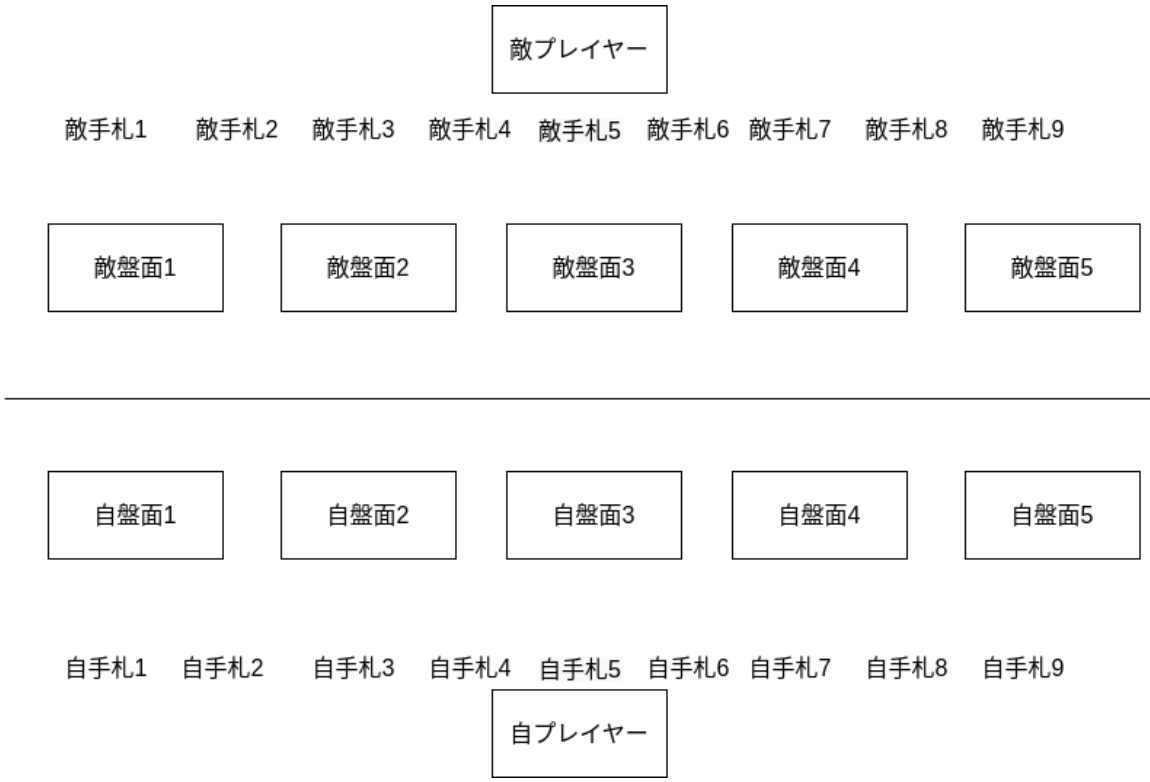

図 2: 作成した環境のイメージ

表 3: 定義した行動空間 (太字は今回新規に追加したパラメータ)

行動説明	次元数
手札 1~9 を自盤面に出す	9
手札 1~9 を地盤面に出さない	**9**
自盤面 1 が敵盤面 1~5 に攻撃 or 何もしない	6
自盤面 2 が敵盤面 1~5 に攻撃 or 何もしない	6
自盤面 3 が敵盤面 1~5 に攻撃 or 何もしない	6
自盤面 4 が敵盤面 1~5 に攻撃 or 何もしない	6
自盤面 5 が敵盤面 1~5 に攻撃 or 何もしない	6

3.1.3 終了条件の変更

単純にライブラリ切れで終了すればよい旨のアドバイスを頂いたので反映した.

「どちらかのプレイヤーにおいて盤面に盤面にカードが無いかつ手札とデッキにカードが無い場合に終了」
⇓
どちらかのプレイヤーが, ライブラリにカードがない時にカードを引こうとした時に終了（ライブラリ切れの時終了)」

3.1.4 報酬の変更

現段階ではプレイヤーがターンが回ってくるたびに 1 枚ドローするルールとなっているためライブラリ切れを勝敗条件に含むと勝率によって学習できているか判断できなくなってしまう. そのため以下のように盤面の残り枚数で設定した.

$$reward = 0.0, \quad 1\text{エピソード終了後}\, reward = \begin{cases} 1.0 & (\text{自盤面のカード枚数}) > (\text{敵盤面のカード枚数}) \\ -1.0 & (\text{自盤面のカード枚数}) \leq (\text{敵盤面のカード枚数}) \end{cases}$$

以前まではエピソード中に報酬を与える検討をしていたため reward を 30 としたままだったが,DQN の学習が安定しないという結果を受けて reward cliping として 1.0 と -1.0 と報酬を設定した.

3.1.5 実験 1

上記に示した 3 つの変更を施して実験した. モンテカルロ探索で先手, 後手ともに 700000 エピソード学習し, 10000 エピソードモデルを検証して勝率を計算した. 結果を表 4 に, 学習の際の reward の平均の推移を図 3 , 4 に示す.

表 4: 実験 1 の結果

学習プレイヤー	勝率
先手	0.9344
後手	0.8562

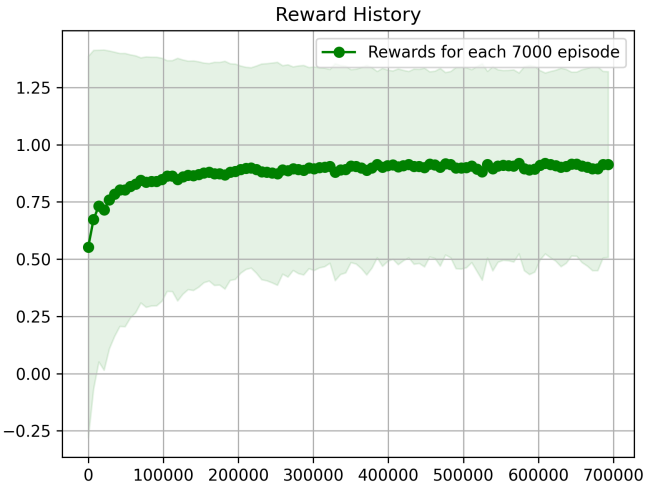

図 3: (実験 1) 先手の学習過程における報酬の平均の推移

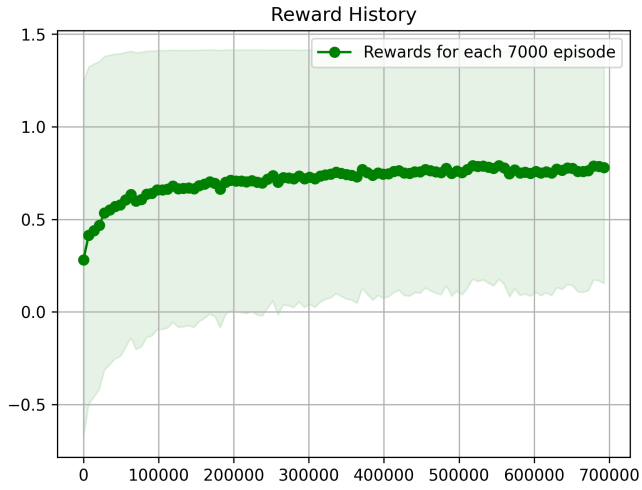

図 4: (実験 1) 後手の学習過程における報酬の平均の推移

3.1.6　行動空間の定義の変更

学習する際にライブラリの残り枚数も見れたらゲームエンドまで後何ターンかかるかという状態を含めて学習してくれるのではと考えて行動空間を表 5 のように再定義した. カードが存在しない領域は 0 で Padding している.

表 5: 定義した状態空間 (太字は新しく追加したパラメータ)

状態説明	次元数	最小値	最大値
手札 1 ~9 の HP と攻撃力	18	0	20
自盤面 1 ~5 の HP と攻撃力	10	0	20
敵盤面 1 ~5 の HP と攻撃力	10	0	20
自盤面 1 ~5 がターン中行動可能かどうか	5	0	1
お互いのライブラリの残り枚数	**2**	**0**	**15**

3.1.7　実験 2

3.1.5 節で述べた行動空間の変更を踏まえて実験 1 と同じ条件で実験した. 結果を表 6 , 実験中の報酬の平均の推移を図 5, 6 に示す.

表 6: 実験 1 の結果

学習プレイヤー	勝率
先手	0.9435
後手	0.8589

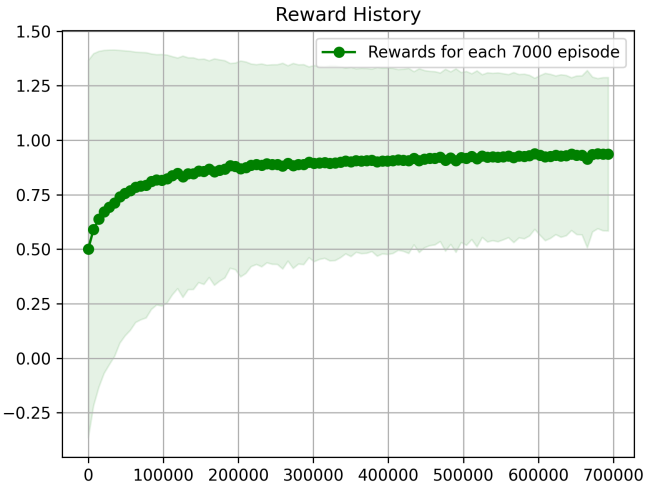

図 5: (実験 2) 先手の学習過程における報酬の平均の推移

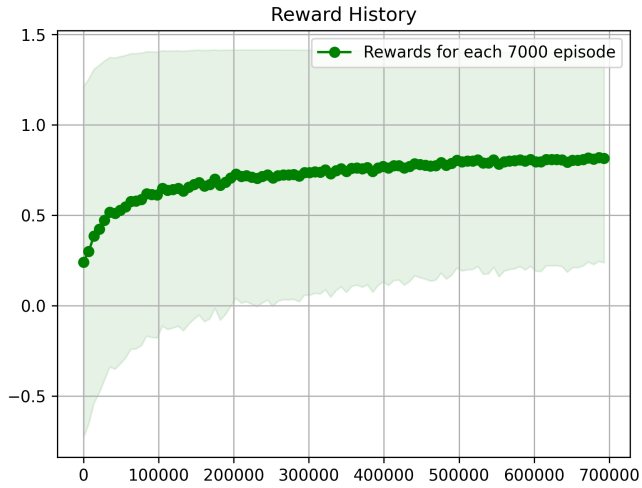

図 6: (実験 2) 後手の学習過程における報酬の平均の推移

ランダム性を考慮すると有意な勝率の増加とは言えない程度であるが, 勝率は向上している. 採用しない理由もないため, 状態空間の定義にライブラリの残り枚数も加えることとする.

3.2 モデル側の改善

DQN は keras のライブラリを使用して実装していた. そのため細かいチューニングは難しいが, 提供されているパラメータを調整し表 7 のように改善を試みた [1].

表 7: 実験 1 の結果

変更したパラメータ	変化前	変化後
Experience Memory への書き込み開始ステップ数	1000	10000
Target Network の更新重み	0.01	0.5

3.2.1 Exprience Memory への書き込み開始 Step の調整

keras-rl には nb_steps_warmup というパラメータが存在し, Exprience Memory への書き込みを nb_steps_warmup ステップ終わってから始めることで学習が進んで無い場合の不安定な状態を Exprience Memory に書き込むことを防いでいる. 標準は 1000 ステップとなっていたため 10000 ステップへと変更した. また,Exprience Memory のメモリ量を 1000000 から 50000 に変更した.

3.2.2 Target Network の更新の重み調整

keras-rl は 1 ステップごとに target_model_update というパラメータで,
target_model = target_model_update * target_model + (1 - target_model_update) * model
という式で Target Network のパラメータの更新を行っている. DQN において Q 値の更新式の $maxQ(s_{t+1}, a_{t+1})$ の計算で Target Network が用いられている [2]. 標準では $1e-2$ となっていたが, DQN がステップを重ねても学習が進まなかったことを考え 0.5 とした.

4 実験 3

DQN で 5000000 ステップ学習を行い, 学習したモデルを 10000 回検証し勝率を検証した. その後, 同程度エピソード数モンテカルロ探索で学習・検証を行い勝率を比較した. なお, 学習プレイヤーは後手とした.DQN のパラメータを表 8 に示す. 結果を表 9 , 学習過程での 100 エピソードの報酬の平均の推移を図 7, 8 に示す.

表 8: DQN のパラメータ

方策	ε -greedy
ε	0.1
全結合層の活性化関数	ReLU
全結合層の次元	64
最適化アルゴリズム	Adam
Target Network 更新重み	0.5
Exprience Memory への書き込み開始 step	10000
Experience Replay のメモリ量	50000

表 9: 実験 3 の結果

method	勝率
DQN	0.9069
MCS	0.7257

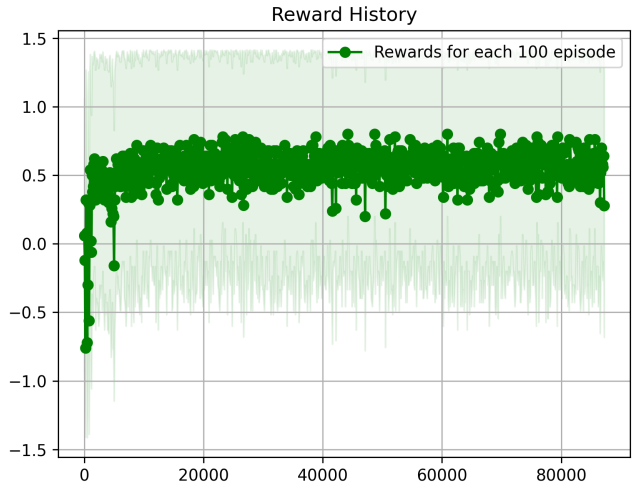

図 7: (実験 3)DQN における報酬の平均の推移

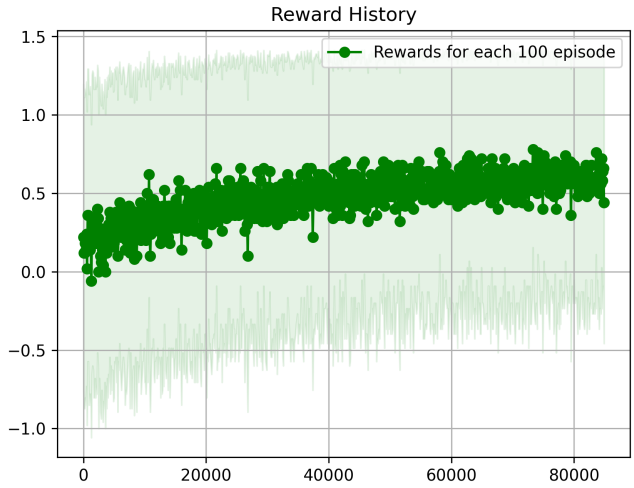

図 8: (実験 3)MCS における報酬の平均の推移

100 エピソードでの報酬平均をとってしまったので見辛いグラフとなったが, DQN が学習できていることが分かる.

5 HP, コスト, 特殊効果をつけた ver の環境作成

DQN の学習時間がかなり長かったので今後の環境の拡張も考え実装した. 作成した特殊効果は以下の通りである.

- 盤面に出したら (攻撃力 , HP) = (1 , 1) のユニット追加で出す.
- 盤面に出したら自プレイヤーの HP を 2 回復
- 盤面に出したら敵プレイヤーの HP を 2 削る
- 盤面に出したら自プレイヤーは 1 枚カードをドロー
- 盤面に出たターンに攻撃できる

6 今後の課題

- 来週の発表練習, 再来週の発表について
  テーマは「自作カードゲーム環境における強化学習手法の適用の検討」と考えています. ですが, 今まで改善⇔実験の繰り返ししかしてないためどのような発表にすればよいか悩んでいます. 本資料のように「こういう問題が生まれたのでこういう対処をしたら上手く行った」といった形式で発表になるのでしょうか.

- これからの方針について

  先週も同じこと言ってたんですが, ここからどのように卒研に落とし込むかの方針が立っていません. 自作環境で既存手法試すことが卒研のハードルを満たしているなら,AlphaZero や Rainbow といった現在実装済みのアルゴリズムの上位互換, また ReBeL[3], MuZero[4], DreamerV2[5] といった最近考案された不完全情報ゲームに対する手法を実装して検証してみたいと思っているのですが難易度が高いかつ新規性が薄いのではないかと感じています. 恐縮ですがお時間があれば相談したいです.

- 学習したモデル同士を戦わせるようにする

  勝率の比較だとモデルの優劣を厳密に示せないため作成したい

- 可視化シュミレータ

  必要性は薄いかもしれないが, 実験の結果を可視化できるようになれば考察や発表の際役に立ちそう.PyGame は触ったこと無いので Unity でログ食わせる方法になりそう.

参考文献

[1] goodclues. Python の強化学習ライブラリ keras-rl のパラメータ設定, 2019,08,19. https://qiita.com/goodclues/items/9b2b618ac5ba4c3be1c5.

[2] たぬきねこ. 【深層強化学習】【dqn】target network, 2020.09.18. https://www.tcom242242.net/entry/ai-2/%E5%BC%B7%E5%8C%96%E5%AD%A6%E7%BF%92/target-network/.

[3] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. *arXiv e-prints*, p. arXiv:2007.13544, July 2020.

[4] Jim Kynvin Adam Cain and Aleksandrs Polozuns. Muzero: Mastering go, chess, shogi and atari without rules, 2020,12,23. https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules.

[5] Danijar Hafner. Mastering atari with discrete world models, 2021,02,18. https://ai.googleblog.com/2021/02/mastering-atari-with-discrete-world.html.