Vol.2019-GI-41 No.5
2019/3/8

ターン制戦略ゲームへの深層学習の適用

木村 富宏[1,a)]

概要：深層学習のゲームへの適用と研究が進んでいるが, ディープマインド社が発表した DQN では,Atari 2600 のゲームが題材として使われた．これらのゲームの操作部のインターフェイス仕様は、ジョイスティックの方向とボタン入力のみとなっていた．さらに深層学習の研究を進めるうえで，より複雑な入力出力を行うゲームへの適用を考える際には，使用するニューラルネットワークの構成が複雑になるため困難になることが多く，これからの課題である。本研究ではターン制戦略ゲームを題材として複雑なデータ構造を持つゲームへの深層学習の適用についての提案を行う．この提案では，8 × 8 サイズのマップを例として，ニューラルネットワークの出力を分割することで，通常では複雑で大きなサイズになりがちな出力ニューロン数を抑え，単純で簡潔な出力表現ができることを示す．検証用マップとしては追跡・逃走マップと経路探索マップを用いて比較的高速で学習が行える強化学習手法も紹介する．

Simple data representation method with Deep Learning for turn-based strategy game

1. はじめに

深層学習の研究の題材としてゲームがよくとりあげられ，ディープマインド社は DQN[1] で Atari 2600 のゲーム群を，AlphaGo[2] と AlphaGoZero[3] では囲碁，さらに AlphaZero[4] では囲碁とチェスと将棋を扱っている．ゲーム入力の操作部分に関しては Atari 2600 のゲーム群ではジョイスティックの方向入力とボタンの押下情報とすべてのゲームについて標準化された仕様であり，AlphaGo と AlphaZero の仕様は囲碁の盤面のマス目にごとに一つの出力ニューロンを割り当てていた．AlphaZero のチェスと将棋では出力のレイヤー数が増加しているが，基本的には盤面上の駒の配置と対応した設計となっている．ターン制戦略ゲームといったさらに複雑な構造を持つゲームでニューラルネットワークの適用を考える場合，ニューラルネットワークのインタフェース仕様もデータ構造にあわせて複雑化していくことになるが，ニューラルネットワークが入れ子構造のようなデータ出力をする場合にはデータの相互干渉問題が存在する．

本研究ではターン制戦略ゲームの複雑なデータ構造をニューラルネットワークに出力させる場合に相互関連性を持つ課題を解決する簡単なデータ表現手法を提案する．

さらに DQN では大量のデータを必要としたり学習時間が長くなるなどの課題があったが，Profit sharing 法による高速でデータ量が比較的少ない量でスパースな報酬における学習ができるようになる手法の試みを行った．

[1] 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
[a)] kt499887@gmail.com

2. ターン制戦略ゲーム TUBSTAP

本研究で題材として使用する TUBSTAP について説明する．

2.1 TUBSTAP

ターン制戦略ゲームは PC ゲームの一分野として人気が高いが次のような特徴がある．

- 将棋や囲碁を越える巨大な探索空間
- 初期画面が固定ではなくさまざまなものがある（マップ）
- 駒の要素としてさまざまな特性がある（駒特性）
- 地形には何種類のものがある（地形特性）
- 一つのターンで複数の駒が動作する（1 ターンマルチユニット動作）

このような特徴のあるターン制戦略ゲームであるが学術的な研究としてはゲームが持つ複雑性のためもありあまり進展していなかった．そのようななか「大戦略」や「ファミコンウォーズ」といった既存のターン制戦略ゲームを分析し

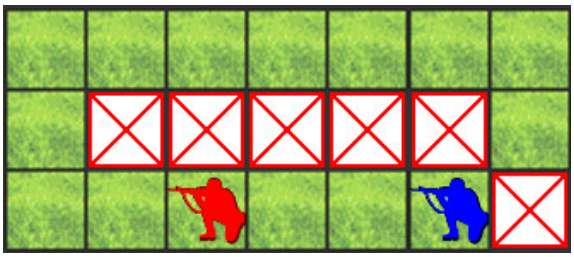


図 1　TUBSTAP のマップの例 1（run 7×3)

つつ，ターン制戦略ゲームに向けた学術研究用基盤プラットフォームとして使用できる「ターン制戦略ゲーム　学術用基盤プロジェクト：TUBSTAP」[5] が提唱された．また，TUBSTAP におけるマップの種類の少なさを補い AI の思考レベルの判定に使用することのできるベンチマーク問題も提案されている [6].

TUBSTAP のような巨大な探索空間を持つターン制戦略ゲームに適用できるゲームアルゴリズムとしては現在の所，モンテカルロ法 [7] や局面分割して探索する手法 [8] など限られたものしか知られていない．

2.2　TUBSTAP のゲームの流れ

TUBSTAP の初期画面はマップごとに異なり様々なものがあるが一例としてマップを示す (図 1)．ゲームは各ターンごとに進行し RED 軍と BLUE 軍の交代で各駒を動かす．各軍で攻撃することにより相手の駒の HP を減らすことができるが同時に反撃を受けて自分の駒の HP も減少する．駒の HP がゼロになったらその駒は取り除かれる．

例として図 1 において RED が先攻であって HP=10 で BLUE の HP= 1 であった場合，RED は BLUE の隣のマスまで移動して攻撃し全滅することができるがもし HP の設定が逆であった場合は BLUE には負けるため攻撃を回避（逃走）する方向へ移動する必要がある．マップ上で赤の X のマスは壁を表現しており侵入禁止マスで駒は入ることができない．緑のマスは草原を表現し駒は移動することができる．ゲームの終了条件はマップで規定されたターン数に達するかどちらかの軍が全滅した場合である．ゲームでの駒の行動表現は（駒のインデックス，移動元アドレス，移動先アドレス，攻撃先アドレス）のタプル（情報の組合わせ）となる．自軍に複数の駒がある場合は，これらのデータを駒の数に応じて並べたタプルのリストによる表現になる．

図 2 は RED 軍が二個，BLUE 軍も二個の駒が配置されている例である．RED 軍の一個が BLUE 軍の二個に近いためもし BLUE の HP が 10 と 10 などとなっていれば RED 軍の一個は HP が 10 だとしても数の差で負けてしまう．RED 軍側は中央に配置された一個と連携すれば負けないかもしれないが，この中央の一個は途中にある壁を回り込んで応援に駆けつけないといけない．

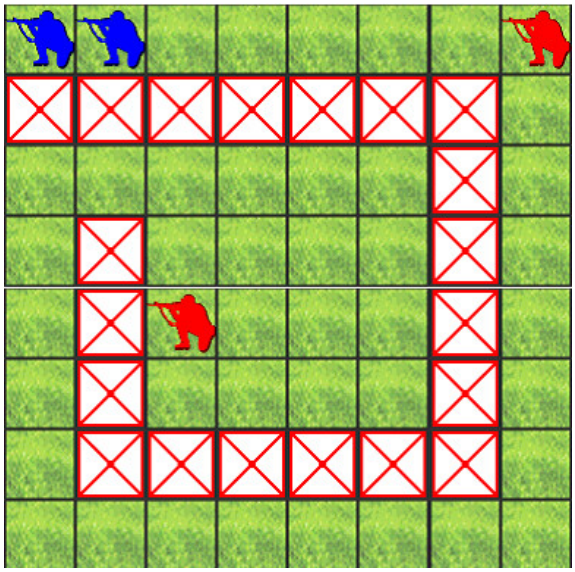


図 2　TUBSTAP のマップの例 2（pathfind05）

3.　深層学習と強化学習

3.1　強化学習

強化学習とは以下の評価関数 V_t を最大化するように，各状態での適切な行動選択の方策 π を学習することである．

$$V_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} + \cdots, \quad (1)$$

ここで γ は割引率（discount rate） r_t は時刻 t で得られる報酬（reward）である．

3.1.1　Q-learning

強化学習アルゴリズムのひとつに Q-learning がある．Q-learning では状態と行動のペアである個々のルールの評価値として Q 値と呼ばれる値を使用する．さらにさまざまな状態で実際に行動を実行して得られる報酬を基に Q 値を更新していく．Q 値の更新式は次のようになる．

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma V(s_{t+1})), \quad (2)$$

$$V(s) = \max_{b \in A} Q(s, b) \quad (3)$$

この式は時刻 t において状態が s_t であって行動 a_t を実行した結果，状態は s_{t+1} に遷移し，報酬 r_t が得られた時に適用され，α は学習率（learning rate）である．Q-learning は特定の方策モデルを必要としないため方策オフ型学習と呼ばれている．Q-learning はエージェントの行動選択において，全ての行動を十分な回数選択し，かつ学習率 α が $\sum_{t=0}^{\infty} \alpha(t) \to \infty$　かつ　$\sum_{t=0}^{\infty} \alpha(t)^2 < \infty$　をを満たす時間 t の関数となっているとき，Q-learning のアルゴリズムで得る Q 値は確率 1 で最適な Q 値に収束する（概収束）[9]．ただし環境はエルゴード性を有する離散有限マルコフ決定過程であることを仮定する．

3.1.2　DQN

Q-learning における関数近似を深層学習によって行うの

が DQN(Deep Q-network)[1] であり，深層学習ではディープニューラルネットワークが使用されている．DQN はディープマインド社によって発表され Atari 2600 の 46 個のゲームのうち 29 個のゲームで人間プレイヤーの成績を上回るという良い成績を示した．高次元で非線形な Q 関数を高精度で近似するために深層学習が導入されている．しかし，ニューラルネットワークの学習のためにデータと学習時間を大量に必要とする課題があり，このような DQN の課題を解決するために後継アルゴリズムが多数開発・研究されている．

3.2 Profit Sharing 法

3.2.1 Profit Sharing 法

Profit Sharinig 法（利益共有法）[10], [11], [12] では，実行されたルールの履歴を保存しておき，報酬が得られるたびに過去にさかのぼってルールの強さを更新する．更新はある時間ステップ t において報酬 r_t が得られるときエピソードに参加したルール R_{t-i} の強さ S_i は次式で行う．

$$S_i(t) \leftarrow (1-\alpha)S_i(t) + \alpha f_i(r) \tag{4}$$

$f_i(r)$ はエピソードの最後のステップ t から数えて i ステップ前のルールに分配する報酬の大きさを決定する関数で強化関数と呼ばれる．

Profit Sharing 法は少ないサンプルデータで高速で学習ができる長所があるが，短所としては探索性が弱く経験に固執して性能が上がりにくくなることがある点と，問題環境が大きくなり必要な行動選択回数が増えると学習が進まなくなることがある点が挙げられる [13].

Profit Sharing 法においても強化関数の方策の局所合理性を保証する合理性定理が知られている [14] .

3.2.2 先行研究

DQN への Profit Sharing 法の適用には DQNwithPS[15] があるが，良いゲームシミュレータを使用し価値の高い経験を用意することが重要であることが述べられている．この手法は各エピソードの終わりにエージェントが報酬が得られた場合にすべてのルールに報酬を規定の式に応じて分配しリプレイメモリに保存する．しかるのちにリプレイメモリから規定のサンプルをランダムに取り出し、通常の DQN と同様に勾配を学習する．通常の DQN との違いは報酬が得られた時は Profit Sharing 方式の学習を行い，報酬が得られなかった場合には通常の DQN の学習を行うところである．どちらの場合もペナルティ（罰）は使用している．

3.3 ニューラルネットワークの出力部のデータ表現

複雑なデータ構造を扱うようになってニューラルネットワークの設計は難易度が増している．

3.3.1 ニューラルネットワークの出力構成の問題

通常のニューラルネットワークの出力の形式は二値分類形式か多値クラス分類形式に分けられる．複雑な構造があったり階層構造や入れ子構造になっているデータの表現には二値分類形式か多値クラス分類形式を組み合わせて表現するか，テータをすべて展開して多数のニューロンで表現しなければならない．複数の形式を組み合わせて表現することはマルチラベル問題 [16] ではしばしば用いられ，そのための手法を参考にすることもできると考える．

TUBSTAP に深層学習を導入し駒の動作を表現するためには相互に依存するデータを同時に表現しなければならないためマルチラベル問題とまったく同一ではないものの似たような表現上の課題がある．

3.3.2 ゲームでのニューラルネットワークの出力表現

ゲームにおけるニューラルネットワークの出力部の構成について述べる．DQN の Atari 2600 のゲーム群は統一された操作部を想定しており，ジョイスティックによる 8 方向の方向入力とボタン入力のみに限定されていた [1].

AlphaZero[4] において囲碁の出力構成は $19 \times 19 + 1$ となっており，囲碁の盤面の 19×19 マスとパスを表示する一個で合計で $19 \times 19 + 1 = 362$ 個が policy（方策）ネットワークの確率分布を表す出力となっている．

AlphaZero Shogi のニューラルネットワークの出力は $9 \times 9 \times 139 = 11259$ 種類の指し手を表現している．この内訳は駒の種類（王，金，銀，桂馬，香車，飛車，角，歩）を表現するパラメータが 8 種類必要で，さらに方向を表現するパラメータが 8 方向（上下左右と右上，左上，右下，左下）必要なので 8×8 種類で、これに桂馬飛びを表現するのに 2 種類必要となり，$8 \times 8 + 2$ となる．さらに駒の成りを表現するパラメータが 2 種類必要なので $(8 \times 8 + 2) \times 2$ となる．これに持ち駒から盤面に打つ手を表現するのに王の駒を除いた 7 種類の駒を表現する必要があるのでトータルで $(8 \times 8 + 2) \times 2 + 7 = 139$ のレイヤーがそれぞれ 9×9 の将棋の盤面について必要となると予想される．

同様に AlphaZero Chess は $8 \times 8 \times 73 = 4672$ 種類の指し手を表現している．

このようにゲームで扱われているニューラルネットワークの出力構成は将棋については複雑化してきているものの，まだ比較的に単純化されたものが多いといえる．

3.4 RNN

RNN(Recurrent Neural Network: 再帰型ニューラルネットワーク) とは内部的に閉路を持つことで情報を一時的に記憶することができるニューラルネットワークである（図 3）．旧来の RNN では深いネットワークでの誤差逆伝搬アルゴリズムでの勾配消失問題や長い系列での関連性をとらえられないといった問題があった [17]．これに対応するためゲート構造を内蔵した RNN が LSTM(Long Short Term

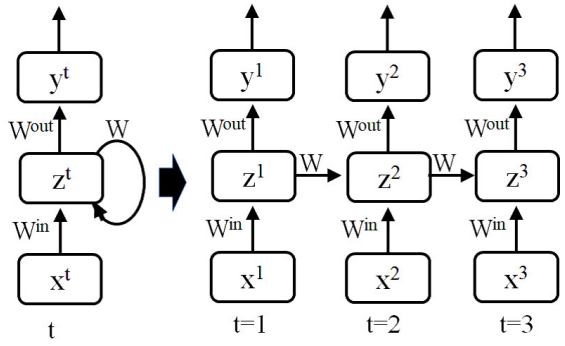


図 3　RNN と時間方向への展開

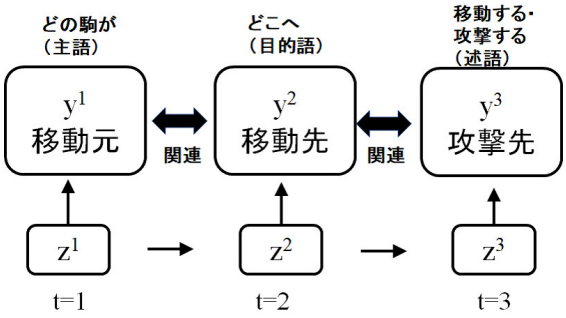


図 4　出力段の RNN による構成

Memory) である．RNN，特に LSTM は時系列データの解析とりわけ自然言語処理で多く使われているが近年ゲーム分野での活用も進んでいる．また LSTM の内部構造を簡略化し高速化を図った GRU(Gated recurrent unit) もある [18].

4. 提案システム

複雑なデータ構造でワンホット出力に適していないニューラルネットワークの出力段を適切な分割をすることで扱いやすくし，さらに各出力の関連性を RNN を採用することで高めたシステム提案を行う．さらに，TUBSTAP において強化学習を行うには使用できるマップの数やデータ量に限りがあり，何が価値のある行動であるか判断するのが困難な場合が多いため Profit Sharing による少ないサンプル数による効率的な学習を試みる．そして，経験を重視するのが Profit Sharing であるが深層学習による汎化能力による未知のマップへの対応能力の向上を図る．

4.1 ニューラルネットワークの出力段の設計の課題

TUBSTAP のようなターン制戦略ゲームでは 1 ターンで複数の駒が動作することを想定した行動の表現が必要となる．具体的には 1 個分の行動について

- 移動元：どの駒についての行動であるかを示す
- 移動先：どの位置への移動であるかを示す
- 攻撃先：どの駒を攻撃するかを示すかまたは攻撃しない

のような情報を含んでいなければならない．ニューラルネットワークにこれらの情報を出力させる設計を考える際に必要なニューロン数を検討する．この場合，8×8 マスのマップをそのままエンコードするとすると移動元で 64 個，移動先で 64 個，攻撃先で 65 個のニューロンが必要となる．これをそのまま通常のクラス分類問題としての出力段と考えると $64 \times 64 \times 65 = 266240$　ものニューロンを用意しなければならない．これは AlphaZero Shogi の 11259 や AlphaZero Chess の 4672　と比較しても非常に多い．これだけの数の設計では通常 GPU のメモリにはのらないので動作しないか，動作しても学習がきわめて遅くなってしまう．今回は出力段を移動元，移動先，攻撃先として三つに分割することで必要なニューロン数を低減させる．

4.2 RNN の TUBSTAP への適用

出力段のニューロン数を抑えることだけを考えるのであれば，出力を上記の項目ごとに分割してニューラルネットワークを多出力の構成にすることで実現できるはずである．しかしながら，多出力のニューラルネットワークの構成では学習において特定の出力段の学習のみ改善されてしまうことで学習が偏ったり相互の出力の関連性を持たせるのが難しくなったりする問題がある．このような各出力の相関を取り学習の偏りを防止するために 図 4 のようにリカレントネットワークを使用することができると考えて提案する．ここでは t=1 で駒の移動元の出力 z^1 が RNN のフィードバックによって次の段に入力され，z^2 の出力に影響を与えこれが移動先となる．さらに z^2 の出力が次の段に入力され z^3 の出力に影響を与えこれが攻撃先となる．実際に使用したユニットは LSTM ではなく実行速度を考慮して GRU ユニットを採用した．

4.3 Profit Sharing の TUBSTAP への適用

提案手法では図 5 のように DQN システムについて報酬の与え方を Profit Sharing 法に基づいたものに変更し報酬が与えられた時点で報酬を 1 としそのエピソードを Experiance Memory へと格納する．TUBSTAP の 1 つのマップのゲーム開始から終了までを 1 エピソードとみなすことができる．エピソードが終了した際に Profit Sharing の強化関数に基づいて報酬を分配する（図 6）．TUBSTAP の各ターンにおける局面と駒の特性が状態として入力されルールによって行動が出力される．ニューラルネットワークの Q-network には局面と駒の情報が与えられ行動を出力しているものとみなす．したがって Q-network は各局面におけるルールを経験に応じて個別に学習していくことになる．

4.3.1 採用した強化関数

TUBSTAP における各ステップはどの局面においても

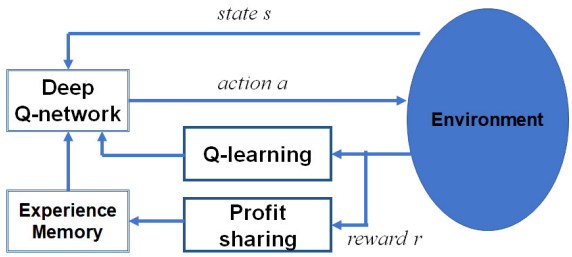


図 5　ブロック図

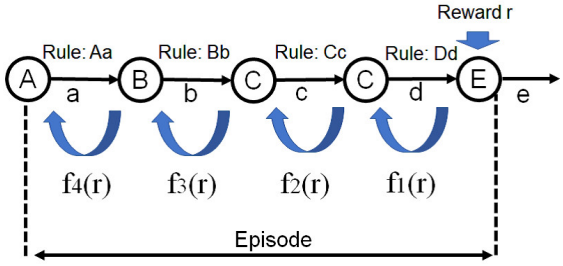


図 6　Profit Sharing の報酬の分配

価値はほぼ同じであると今回は仮定して次式のように設定した．

$$f_i(r) = 1 \tag{5}$$

5. 実験設定と結果

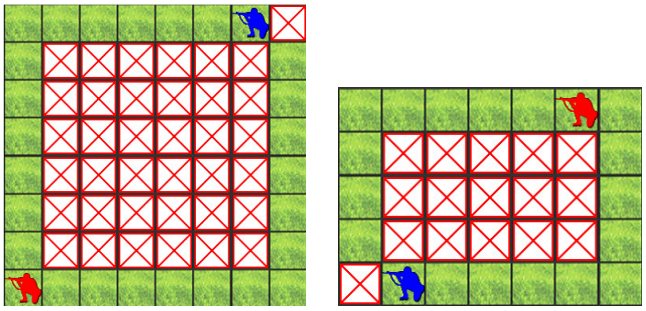
ここでは提案したニューラルネットワークの出力段の設計手法について確認するための教師あり学習の数値実験と上記に加えて Profit Sharing 法を適用した場合の強化学習の性能について DQN と比較した数値実験を行う．

5.1 環境（Environment）の設計

ここで環境プログラムの入力となるのはシステムの状態 s_t ，次の行動 a_t であり，出力となるのは次のシステムの状態 s_{t+1} と報酬 r_t である．s_t はマップの地形情報と駒の位置情報や HP の情報が，a_t には駒の現在位置情報と次の移動先と攻撃先情報が入っている．ここでは 1 つの状態から次の状態に至るまでで 1 ステップと呼ぶことにする．ひとつのマップにつきマップごとに決められたターン数に達するかどちらかの駒が全滅することでゲーム終了となる．これを 1 エピソードとしている．

5.1.1 対戦相手側の方策設計

学習する側と対戦する側の駒の動作は環境に含まれている．今回使用するマップは簡単であるため容易に追跡・逃走プログラムを設計できる．これを最適方策 $\pi*$ として採用し相手側の駒の動作として常に使用している．この方策はマップ設定で相手の HP と自らの HP を比較して敗北する場合は逃走し，勝利する場合は追跡する．逃走する場合はできる限りターン数が長くなる方向で道のり距離が遠い方向へと移動し，追跡する場合は道のり距離が最短の方向に移動する．複数の最適経路がある場合はランダムに選択する．



(a) run 8 × 8　　(b) run 7 × 5

図 7　追跡・逃走マップの例

5.1.2 報酬の与え方

エピソードに対して報酬を与えるのはマップにより規定されたターン数の中にゲームが終了した場合で

- HP の比較から戦力的に勝てるマップ設定であってかつ，相手を全滅させた
- HP の比較から戦力的に負けるマップ設定であってかつ，自軍が全滅しなかった（逃げ切った）

与えらえる報酬は 1 となる．上記以外の場合は報酬はゼロとなる．

5.2 評価に使用したマップ

本研究で使用したマップは駒の挙動や正解を確認しやすくするため，すべて 8 × 8 のサイズとし，使用する駒は RED が歩兵一個，BLUE が歩兵一個に限定した．駒の数を一個に制限することで解析しやすくなるがゲームの持つ複雑性は制限されるので限定的な条件での解析となる．なお図のマップではすべて RED 側が学習側であるが，RED と BLUE を入れ替えたマップも生成して学習する．

5.2.1 追跡・逃走マップ（run マップ）

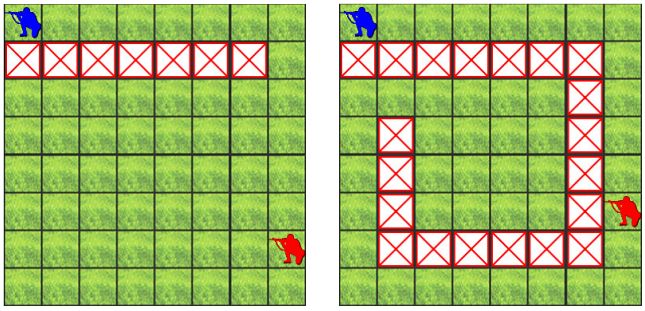
追跡・逃走問題は人工知能の研究課題として重要である．ここで用意したマップは図 7 のように回転型の経路上を追跡または逃走する設定のマップである．マップの地形は外周が角 1 つを除いて全て平地で四角形の経路のみである．ここでは run マップと呼び，run 8 × 8 のようにマップの横と縦のマス目のサイズに応じて数値で名称を付けている．例えば図 7 (a) は横 8 マス × 縦 8 マスのマップでる．

5.2.2 経路探索マップ（pathfind マップ）

経路探索もまた人工知能やロボット研究などでよく扱われる課題である．ここで用意したマップは図 8 のように障がい物となる壁を避けた経路を探索して相手に到達できるかどうかを課題としている．ここでは壁の形を変えた種類を用意した．

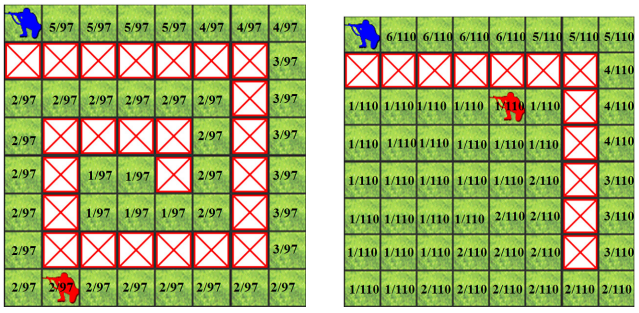
5.2.3 学習用マップと検証用マップ

用意したマップをまとめたものが表 1 である．ここでは学習用（train）に使用するマップと検証用（test）に使用



(a) pathfind01　(b) pathfind06

図 8　経路探索マップの例



(a) pathfind02　(b) pathfind03

図 9　駒出現確率分配データ

表 1　評価に使用したマップ

学習用（train）	検証用（test）
8 × 8, 8 × 7, 8 × 6, 8 × 5, 8 × 4, 8 × 3 7 × 7, 7 × 6, 7 × 5, 7 × 4, 6 × 6, 6 × 5, 6 × 4, 6 × 3, 5 × 5, 5 × 3, 4 × 4, 4 × 3, pathfind01, pathfind02, pathfind03, pathfind05	7 × 3, 5 × 4, pathfind06

するマップを明確に区別して使用する．各グラフ上で使用したマップは学習用が _train，検証用が _test を付記したものになる．学習と検証とも選択されるマップはランダムに選択される．追跡・逃走マップについては経路長が長いほど出現確率が高くなるよう調整している．

5.2.4　駒の配置について

相手側の駒は経路上の始点に固定して配置される．経路探索マップにおいては学習する側のエージェントの駒は図 9 のようにマップごとに決められた確率に応じてランダムに配置される．基本的には近いほど重要度が高いとして高い確率に定め，さらに重要性の高いマス目に多く駒を出現させることで学習を速めている．追跡・逃走マップの駒の配置確率はマス目ごとに等確率である．

5.2.5　駒の HP と勝敗について

駒の HP は自分も相手の駒も 1 から 10 までのランダムな整数値が選ばれる．つまり HP の組み合わせは 10 × 10 の 100 通りとなる．駒が戦闘した場合は TUBSTAP のルールに基づいて確定的に決定される．

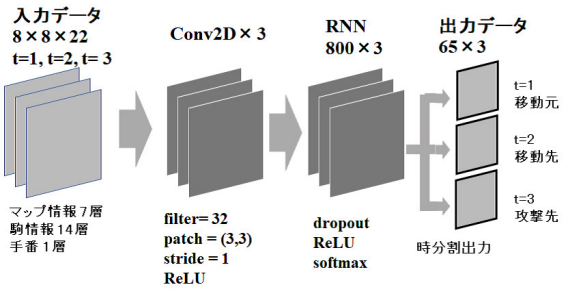


図 10　使用したニューラルネットワーク構成

5.2.6　Data Augmentation

マップデータの多様性を確保するため，マップを生成する過程においてデータ拡張（Data Augmentation）と呼ばれる手法によってデータ量を増加させている．具体的にはマップを生成時に鏡映および回転をランダムに行ってマップと駒配置を変化させている．

5.3　使用したニューラルネットワーク

今回使用したニューラルネットワークの内部構造を図 10 に示す．入力データはマップ情報の地形情報 7 種類と駒情報（兵種六種類＋移動済みフラグ）を RED と BLUE で 7 × 2，これ手番情報が 1 で，トータルで 7 + 7 × 2 + 1= 22 の情報を 8 × 8 のマス目状に変換したエンコードした 22 チャンネルのニューラルネットワーク入力用データである．駒の HP は 1 に正規化され浮動小数点の形式で入力される．入力層には通常の自然言語処理用のものは Embedding 層が採用されるが今回は畳み込み層を採用し三層積層している．ついでリカレントネットワーク層が積層されて三層あり，各層のニューロン数が 800 となっている．入力データは t=1, t=2, t= 3 で同一データを入力する．出力は駒の行動情報を時分割 (t=1 移動元, t=2 移動先, t= 3 攻撃先) で出力している．

5.4　使用したソフトウェアとハードウェア

今回の研究に使用したソフトウェアとハードウエアは以下である．

- ソフトウェア：Python 3.6, Keras 2.2
- ハードウェア：CPU Intel i7 3.4GHz, GPU NVIDIA GTX1050Ti

また学習に使用したプログラムは OpenAI Gym[19] における DQN プログラムをもとに改変を加えたものである．

5.5　学習の進展の評価法について

学習の進捗程度の評価法を考える．今回の実験設定では出力された行動を正しいかどうかをただちには評価するのが困難でありまた，可能な行動のうち複数の行動が同様に正しいといえる場面があるため正解ラベルを用意すること

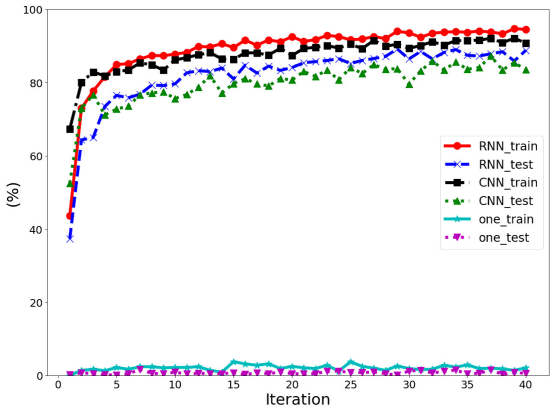


図 11　合法手出力割合（教師あり学習）

ができない．そのためまずは合法手を出力しているか？という点 (合法手出力割合) での評価を行う．しかし出力手が合法手であるからといって，それが各局面で正しい行動であるとは限らない．そこでエピソードが成功するしたかどうか？という点（エピソード成功割合）でも評価を行う．以上の二点で学習の進展を測定した．

5.6　実験結果

5.6.1　教師あり学習

設計したニューラルネットワークの性能を比較するために図 10 の RNN を使用したもの（RNN_train/ RNN_test）と，これから RNN の部分を三層の全結合層に置き換えた CNN + Fully Connected Layer を測定する．ただし，CNN の場合（CNN_train/ CNN_test）は出力を三つに分割した場合と 1 つに集約した場合（one_train/ one_test）とする．一つに集約した場合は本来なら 64 × 64 × 65 のサイズのニューロンが必要だがこのサイズでは GPU にのらないため移動元の出力を省略したタイプに変更し 64 × 65 のサイズで測定した．本来なら機能が異なるものであるが比較のために掲載する．

使用した方策は学習する側も対戦相手と同様に最適方策 $\pi*$ として双方が最適方策の状態でデータを生成して学習した．結果的に教師あり学習と同等な条件となる．実験結果が合法手出力割合が図 11 でエピソードの成功割合が図 12 である． 1 イタレーションあたりに 4000 ステップ実行している．

5.6.2　強化学習

DQN のシステムに Profit Sharing を組み込み，学習をさせて Profit Sharing 法と DQN の比較をする．探索方策は ε-グリーディ法であり，これは通常は Q-network の Q 値の最大値に基づいた行動選択を行い，確率 ε でランダムな合法手を打つものである．ここでは $\varepsilon = 0.25$ とした．もし Q-network の行動出力が合法手ではなく TUBSTAP のルール上実現できないものであった場合，報酬は-1 となり，ただちにエピソードを終了してそこまでの情報は

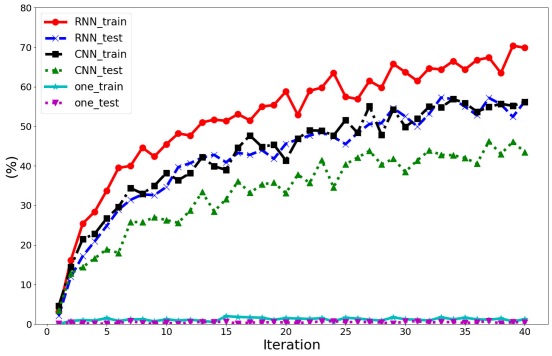


図 12　エピソード成功割合（教師あり学習）

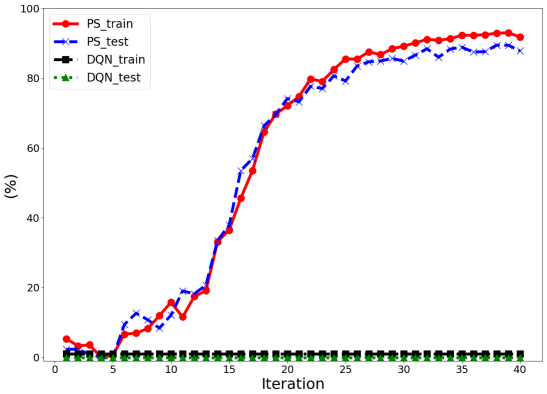


図 13　合法手出力割合（強化学習）

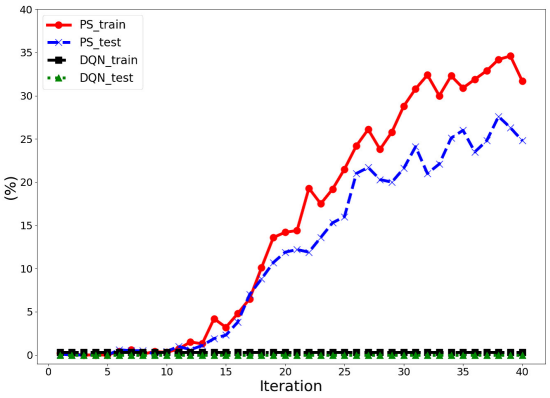


図 14　エピソード成功割合（強化学習）

Experience Memory へと格納されない．報酬が 0 の場合も Experience Memory へと情報は格納されない．ε-グリーディ法は学習時にのみ動作し，検証時には動作しない．検証時では Q 値のみによって行動決定をしている．実験結果は合法手出力割合が，図 13 エピソード成功割合が図 14 であり，Profit Sharing が PS_train と PS_test，DQN が DQN_train と DQN_test でそれぞれ学習と検証データの精度を表している．ここでは 1 イタレーションあたり 20000 ステップ実行している．

6.　考察

ニューラルネットワークの設計においては RNN で設計

した場合が CNN で設計した場合に比較して成績が上回った．また，RNN と CNN の双方のネットワーク設計のどちらについても，検証用に使用されたマップは学習用マップと似ているものではあるが、学習中にはまったく使用されていないにもかかわらず，検証用のテストでも高い合法手出力率とエピソード成功率を示した．これは深層学習による高い汎化性能による未知のマップへの対応能力によるものと言える．

出力をひとつに集約した場合は学習速度的が遅く精度的にも低い成績となった．これは出力のニューロン数が多くなりすぎたために学習の効率が低下しているためと思われる．RNN の設計が CNN をより良いのは，RNN において時系列出力として t=1 の出力を t=2 にて活用し，t=2 の出力を t=3 で活用するというように相互に関連性を持たせたことで学習の精度が高くなったものと推定される．

強化学習のデータについては DQN の設計のままでは学習が進まなかった．これはニューラルネットワークが初期状態からでは報酬が得られるケースが少なすぎてスパースな報酬の状態となり学習が進まないものと思われる．

7. おわりに

複雑なデータ構造を持つゲームに深層学習を適用する上でのニューラルネットワークの出力段についての新しい提案を行った．同時に少ないデータ量でスパースな報酬における強化学習ができる手法について検討した．将来的には複数の駒があるマップや複雑なマップのケースについて本手法を適用し，さらに自己対戦による強化学習を検討していく予定である．

参考文献

[1] Mnih, V., et al., Human-level control through deep reinforcement learning, Nature, 518, 529–533, (2015).
[2] Silver, D., et al., Mastering the game of Go with deep neural networks and tree search Nature 529, 484–489, (2016).
[3] Silver, D., et al., Mastering the game of Go without human knowledge, Nature 550, 354–359(2017)
[4] Silver, D., et al., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play Science, Vol. 362, Issue 6419, pp. 1140-1144(2018).
[5] 村山公志朗，藤木翼，池田心，学術研究用プラットフォームとしての大戦略系ゲームのルール提案，ゲームプログラミングワークショップ 2013 論文集，pp. 146-153
[6] 木村富宏，池田心，ターン制戦略ゲームにおけるベンチマークマップの提案，ゲームプログラミングワークショップ 2016 論文集，pp. 36-43
[7] 加藤千裕，三輪誠，鶴岡慶雅，近山隆，ターン制ストラテジーゲームにおける戦術決定のための UCT 探索とその効率化ゲームプログラミングワークショップ 2013 論文集，pp. 138-145
[8] 佐藤直之，藤木翼，池田心，ターン制戦略ゲームにおける局面評価値構成のための局面分割および単純化ゲームのオフライン木探索，ゲームプログラミングワークショップ 2015 論文集，pp. 61-68
[9] 木村元，宮崎和光，小林重信，強化学習システムの設計指針，計測と制御，Vol. 38, No. 10, pp. 618-623(1999).
[10] 堀内匡，藤野昭典，片井修，椹木哲夫，経験強化を考慮した Q-Learning の提案とその応用, 計測自動制御学会論文集，Vol. 35，No.5，pp. 645-653(1999).
[11] 荒井幸代，宮崎和光，小林重信，マルチエージェント強化学習の方法論：Q-Learning と Profit Sharing による接近人工知能学会誌，Vol. 13, No. 4, pp. 609-618(1998)
[12] 宮崎和光，木村元，小林, 重信，Profit Sharing に基づく強化学習の理論と応用，人工知能学会誌，Vol.14，No.5，pp. 800-807(1999).
[13] 植村渉，上野 敦志，辰巳昭治，経験に固執しない Profit Sharing 法，人工知能学会論文誌， Vol. 21, No. 1, pp. 81-93(2006)
[14] 宮崎和光，山村雅幸，小林重信，強化学習における報酬割り当ての理論的考察，人工知能学会誌，Vol.9，No.4，pp. 580-587(1994).
[15] Miyazaki K., Exploitation-Oriented Learning with Deep Learning - Introducing Profit Sharing to a Deep Q-Network - Journal of Advanced Computational Intelligence and Intelligent Informatics Vol. 21, No.5, pp. 849-855, (2017).
[16] Herrera, F. et. al. : Multilabel Classification Problem Analysis, Metrics and Techniques, Springer, 2016.
[17] 岡谷貴之：深層学習，講談社 (2015)
[18] Cho, K. et. al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, arXiv preprint, arXiv :1406.1078(2014)
[19] OpenAI Gym, `https://gym.openai.com/`