Vol.2019-GI-42 No.11
2019/7/20

ターン制コマンドバトルにおける強化学習効率化

辻本 貴昭[1,a)]　尾崎 嘉彦[1,b)]　森田 想平[1,c)]

概要：フリーミアムモデルを採用した典型的なスマートフォン向けゲームは，継続的にアップデートを行うことが多く，アップデートの都度必要となるゲームバランス調整作業はより一層の効率化が求められている．ゲームバランス調整効率化の有効な手段としてゲームプレイ AI の利用が考えられ，高い実績を持つゲームプレイ AI 学習手法として強化学習がある．しかし，強化学習は一般に膨大な学習時間を必要とするため，アップデート間隔が短いフリーミアムモデルを採用したゲームに適用することは難しい．本研究では，テキストベースのゲームで学習効率化の有効性が確認されている，行動の価値を個別に評価するネットワーク構造を，ターン制コマンドバトルゲームに導入し，計算実験により同様に有効であることを示した．また，ゲームのマスタデータから行動の埋め込み表現を事前に獲得することで，さらなる学習効率化が可能であることを示した．

Efficient Reinforcement Learning in a Turn-Based Combat Game

Abstract: Typical mobile games adopt a free-to-play business model. Mobile game developers update the games every day and an efficient way of adjusting the parameters on game balancing are required. Currently, the automated playtesting with AI agents is considered as the most promised way to efficiently adjust these parameters. Deep reinforcement learning-based agents have great success in playing games but are not suitable for iterative playtesting in a short period of time because a long training time is required. In this study, we apply the network that evaluates the values for each action succeeded in playing a text-based game to a turn-based combat game. Our experimental results indicate the proposed approach is efficient. We also show that getting the embedding vectors of actions from the master data of the game enables us to efficiently train the agents.

1. はじめに

フリーミアムモデルが採用されたゲームは，有料の追加コンテンツなどを購入するプレイヤに継続利用されることで収益を上げる．そのため，コンテンツの追加やそれに伴うゲームバランス調整を継続して行う必要がある．ゲームバランスを調整するためには，敵の強さや味方の強さなどを決定する膨大な数のパラメータを変更しながらテストプレイを繰り返す必要があり，多くの作業工数が費やされる．現在のバランス調整作業は作業者の属人的な対応に依存しており，バランス調整作業の効率化が求められている．

近年，ゲームのバランス調整を効率化するために，AI によるテストプレイを利用した研究が行われている [1], [2], [3]．これらの研究では，理想的なタイミングから少しずれて行動するなど，人間を模したヒューリスティックに基づく AI や，人間のプレイヤの行動ログから教師あり学習させた AI などにゲームをプレイさせ，その結果を用いてゲームの難易度推定や重要なパラメータの発見などを行っている．

ヒューリスティックに基づく行動のモデル化が困難な場合や，プレイヤの行動ログが利用できない場合にゲームプレイ AI を学習する方法として強化学習 [4] が考えられる．強化学習とは，ある環境においてエージェントが現在の状態を観測し行動を選択，そして行動に応じて環境から得られた報酬に基づいて報酬を最大化する行動を学習するアルゴリズムである．近年では，強化学習は多くのゲームタスクに適用され高い性能を達成している [5]．

深層学習を利用した深層強化学習によって，状態空間が大きいタスクでも状態価値関数の近似が可能となり，複雑なゲームでも強化学習が有効であることが示されている [6], [7], [8], [9]．しかし，深層強化学習には膨大な時間と

[1] グリー株式会社
GREE, Inc.
[a)] takaaki.tsujimoto@gree.net
[b)] yoshihiko.ozaki@gree.net
[c)] sohei.morita@gree.net

ⓒ 2019 Information Processing Society of Japan

計算リソースが必要である．そのため，学習を効率化する研究もよく行われている．特に行動の選択肢が多い場合に行動を埋め込むことで学習を効率化する手法 [10], [11], [12] があり，高い効果を上げている．行動の埋め込みを用いる手法には，連続値を取る行動を離散化したものを扱うもの [10] や，自然言語で表される行動を扱うもの [11], [12] が提案されている．

本研究では，行動が技の性能などの表形式のデータで表現される環境において，行動を適切に埋め込むことでゲームプレイ AI の学習を効率化することを目的とする．行動が表形式のデータで表現されるゲームである Pokémon Showdown[13] において，キャラクタの技の属性や威力などを埋め込むことで類似した技を一般化し，異なる技同士の関係性を有効活用することで学習を効率化できることを示す．

2. 関連研究

2.1 AI を用いたテストプレイ

一般に，ゲームの難易度を把握することは非常に困難である．ゲームには，難易度に影響を与えるパラメータが多数存在し，敵の体力，攻撃力，防御力などがある．さらに，パラメータの相互作用によっても難易度が変化する．人間によるテストプレイには多くの時間が必要となるのでプレイ回数が制限され，正確な難易度評価や幅広いパラメータ探索が困難である．そこで，難易度評価やパラメータ探索のために，AI を用いて大量にテストプレイを行う手法が提案されている．

Isaksen ら [1] は人間を模した AI と生存分析を用いて難易度の評価を行っている．この手法では，人間の反応の遅延などをモデル化した AI にゲームを十分な回数プレイさせ，スコアの分布に基づいて難易度を数値化している．Stefan ら [2] はプレイヤのプレイログから AI を学習させ，開発中のコンテンツでのプレイヤのスコアを推測している．Fernando ら [3] は A*アルゴリズムによってゲームをプレイさせ，ゲームの各パラメータが難易度にどのような影響を与えるのか調査している．

2.2 ゲームプレイ AI

ゲームプレイ AI の単純な実現方法として効用関数を用いた方法がある [14]．この方法は意思決定をするために現在の状態に対する各行動の効用を計算し，一番効用が高い行動を選択する．ゲーム内の Non Player Character などの行動を設計するには，体力や想定されるダメージ量などを用いて行動毎の効用関数を定義する．そして，効用関数の値が最も高い行動を選択する．Ms. Pac-Man では，ゴーストの位置に基づいて計算された現在の状態を 3 つの効用関数に代入して，一番効用の高い行動を選択している（**図 1**）．

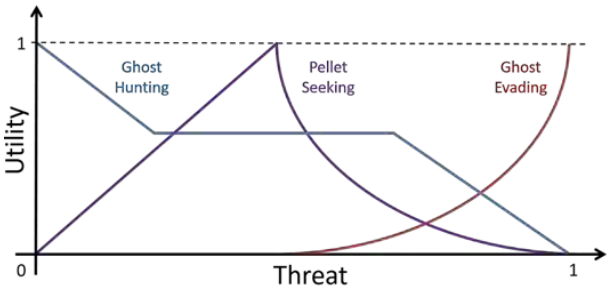

図 1 Ms. Pac-Man での効用関数
Artificial Intelligence and Games[14] P.38 より
Fig. 1 The utility function of Ms. Pac-Man [14]

計算リソースの増大にともない，シミュレーションを何度も繰り返し，シミュレーション結果が最も良かった行動を選択する手法も一般的となっている [7], [15].

2.3 強化学習

機械学習手法の一つに強化学習 [4] がある．強化学習は環境から得られる最終的な報酬を最大化するような方策を学習する．

強化学習によって複雑な問題を扱うために，状態空間を近似する手法が提案されている．2013 年には Deep Q-Network（DQN）が提案され Atari のゲームに適用された [5]．DQN は Q 学習 [16] の Q 関数を畳み込みニューラルネットワークで表現し，画像のような巨大な状態空間における価値関数を近似することで，様々なゲームで人間レベルのプレイができるようになった．現在までに，DQN の様々な改良が提案されている [17], [18], [19].

2.4 行動の埋め込み

強化学習を効率化するため，画像からの特徴抽出に注力する手法 [20], [21] や，ニューラルネットワークで環境を近似する手法 [22], [23] などが考えられてきた．従来，学習効率化手法の多くは状態を対象としていたが，近年では行動を対象として行動を埋め込むことのメリットを示す手法が提案されている．

Dulac ら [10] は連続値を取る行動を離散化したタスクやレコメンドタスクのような，行動空間が大きいタスクを解くために，行動を一般化した埋め込み表現に変換して学習する方法を提案している．この手法は Actor-Critic 手法を拡張したもので，既知とした埋め込み表現を活用することで，価値関数の計算時に考慮する行動の数を削減した．埋め込み表現上で価値関数を更新するので，個別の行動ではなく一般化された行動の価値関数を更新することになり，学習効率も向上した．

He ら [11] は状態と行動が自然言語で表現されるテキストベースのゲームを解くために埋め込み表現を活用している．2 つのニューラルネットワークを用いて状態と行動を

ⓒ 2019 Information Processing Society of Japan

個別に埋め込み表現に変換することで，テキストベースのゲームにおいて性能が向上することを示した．

Fulda[12] らもテキストベースのゲームを解くために，Wikipedia のデータから word2vec[24] によって事前学習された埋め込み表現を利用して行動の探索空間を制限することで，考慮する行動の削減と学習の効率化を行っている．

2.5 Deep Reinforcement Relevance Network

He ら [11] はテキストベースのゲームにおいて，状態によって選択可能な行動数が変化し，なおかつその上限が予測困難な問題に DQN を対応させた Per-Action DQN（PA-DQN）を基に，Deep Reinforcement Relevance Network （DRRN）と呼ばれる手法を提案した．PA-DQN は状態と行動のペアを入力として行動ごとに価値を評価する．DRRN は PA-DQN を基に，報酬が多く得られるときに状態と行動の埋め込み表現が似たものとなるように，それぞれの埋め込み表現を学習する．その際，状態を記述する文章は長く複雑な文法が含まれるのに対し，行動を記述する文章はとても簡潔であるという特徴に着目し，状態を埋め込むネットワークと行動を埋め込むネットワークを個別に用いる．He らは，計算実験によって PA-DQN を用いることで学習効率化できることを示し，DRRN を用いることで学習効率化だけでなく性能も向上することを示した．

2.6 Pokémon Showdown

Pokémon Showdown は 2017 年の Computational Intelligence and Games で大会 [13] が開催されたターン制コマンドバトルゲームである．このゲームは二人零和有限不確定不完全情報ゲームに分類され，場に出ているポケモンと呼ばれるキャラクタによって取りうる行動が変化する．それぞれの行動は技の性能として表形式のデータで表現されている．行動を決定する際は，技を選択するだけでなくポケモンを交代することもできる．この場合，行動は交代ポケモンの性能として表形式のデータで表現される．ゲームの特徴として，状態が確率的に遷移するので遷移確率を考慮しなければならない，チェスや将棋と異なりお互いのプレイヤが同時に行動を決定する，最初は対戦相手の情報がほとんど隠されているといった点が挙げられる．

3. 提案手法

Pokémon Showdown では，2 人のプレイヤがそれぞれ 6 体のポケモンでチームを構築し，ターン毎に 4 つの技のうち 1 つ，もしくは残り 5 体のポケモンから交代するポケモン 1 体を選択する．すなわち，計 9 つの行動から 1 つを選択する．そして，全ての相手ポケモンの体力を 0 にしたプレイヤが勝者となる．ただし，倒されたポケモンには交代できず，また戦闘中ポケモンの状態によっては一部の技が選択できない場合があるため，常に 9 つの行動が選べるわけではない．ターン毎に選択可能な技はそれぞれ属性，攻撃力，命中率などの表形式のデータで表現することができ，交代ポケモンもそれぞれ属性，体力，攻撃力，防御力などの表形式のデータで表現することができる．

本研究では，表形式のデータで表現される行動を事前に埋め込むことでターン制コマンドバトルの学習を効率化する手法を提案する．提案手法は状態毎に行動の選択肢数が異なる問題に対応するために，行動ごとに価値を評価する PA-DQN に基づいている．そして，技と交代ポケモンという異なった特徴を持つ行動を埋め込むために DRRN と同様に 2 つのニューラルネットワークを用いる．ただし，DRRN は強化学習と同時に埋め込みを学習していたのに対し，提案手法はマスタデータを使って埋め込みを事前に学習する．

提案手法は 2 つのステップに分けられる．1 つ目は，AutoEncoder[25] によって行動を埋め込むステップである．このステップは強化学習を実行する前に行う．2 つ目は，行動の埋め込み表現を利用して強化学習を行うステップである．

3.1 行動の埋め込み

強化学習の際に，技の威力や属性など行動を表す表形式のデータをそのまま利用すると，方策の学習と同時に行動の特徴抽出を行うことになる．そのため，ゲームのステップを進めなければ行動の特徴抽出ができず，すべての技や交代ポケモンの特徴を学習するためには技とポケモンの膨大な組み合わせでシミュレーションを実行しなければならない．

技や交代ポケモンの特徴はゲーム中の状態に関係なく，マスタデータとして定義されているので，シミュレータを実行しなくても学習ができる．そこで，シミュレーションを実行する前に AutoEncoder を使って次元削減することで強化学習を効率化する．活性化関数を f，技や交代ポケモンの特徴を入力 x，エンコーダを $y = f(Wx+b)$，デコーダを $x' = f(W'y+b')$ として $x = x'$ となるよう学習させる．そして，入力 x に対応する隠れ層の値 y を埋め込み表現として利用する．

Pokémon Showdown でステップごとに選択できる行動は，技の選択と交代ポケモンの選択という 2 種類に分類できる．それぞれのデータの特性は全く異なると考えられるので，行動を埋め込む際は技と交代ポケモンそれぞれ別の AutoEncoder を用いる．まず，入力として技を表現する 42 次元のデータ（**表 1**）を AutoEncoder に与え，技の埋め込み表現 y_{move} を得る．そして，入力として交代ポケモンを表現する 42 次元のデータ （**表 2**）を別の AutoEncoder に与え，交代ポケモンの埋め込み表現 y_{switch} を得る．

ⓒ 2019 Information Processing Society of Japan

表 1　技を埋め込む AutoEncoder への入力
Table 1　Inputs for the AutoEncoder (move)

名称	概要
属性	18 種類の属性を表現したダミー変数
分類	3 種類の分類を表現したダミー変数
状態異常	7 種類の状態異常を表現したダミー変数
状態変化	12 種類の状態変化を表現したダミー変数
威力	1 次元の実数
命中率	1 次元の実数

表 2　交代ポケモンを埋め込む AutoEncoder への入力
Table 2　Inputs for the AutoEncoder (switch)

名称	概要
攻撃力	1 次元の実数
防御力	1 次元の実数
特殊攻撃力	1 次元の実数
特殊防御力	1 次元の実数
すばやさ	1 次元の実数
属性	18 種類の属性を表現したダミー変数
状態異常	7 種類の状態異常を表現したダミー変数
状態変化	12 種類の状態変化を表現したダミー変数

3.2　行動の選択

$\mathcal{S}$ を状態の集合，$\mathcal{A}$ を行動の集合としたとき，一般的な DQN では入力 $s \in S$ に対して $|\mathcal{A}|$ 個の値を出力して，それぞれの値を行動価値 $Q(s,a)$ $(a \in \mathcal{A})$ とみなす（**図 2**）．そして，一番価値が高い行動 a を選択する．

Pokémon Showdown では倒されたポケモンには交代できず，戦闘中ポケモンの状態異常により 4 つの技のうち一部が選択できない場合もある．そのため，行動の選択肢数は固定ではなく，バトルの状態 s によって行動の選択肢数 $|\mathcal{A}_s|$ が異なる．そのため，状態毎に行動の選択肢数が異なるテキストベースのゲームに対して，学習効率化の有効性が確認されている PA-DQN[11] を用いて，行動の価値を 1 つずつ個別に評価する．具体的には，状態 s と行動 a に対応する埋め込み表現のペアをネットワークに入力して，出力は行動価値 $Q(s,a)$ の 1 つだけを出力する．これを状態 s での全ての行動の選択肢 $a \in \mathcal{A}_s$ について行い，一番価値が高い行動 a を選択する．これにより，状態毎に変化する選択肢の数を考慮する必要がなくなる．He ら [11] の PA-DQN はネットワークを 1 つしか持たないが，本研究が対象としている Pokémon Showdown では技と交代ポケモンではデータの性質が異なるので，DRRN のアイデアを一部取り入れ，技の価値を評価するネットワークと交代ポケモンの価値を評価するネットワークを用いる（**図 3**）．

各ターンで選択可能な技は i 個 $(0 \leq i \leq 4)$ で，交代可能なポケモンは j 体 $(0 \leq j \leq 5)$ である．状態 s のときに行動を選択する際は，選択可能な技の埋め込み表現 y^i_{move} と選択可能な交代ポケモンの埋め込み表現 y^j_{switch} の価値をそれぞれ評価して，価値 $Q(s, y^i_{\text{move}})$ または $Q(s, y^j_{\text{switch}})$ が最大となる行動を選択する．

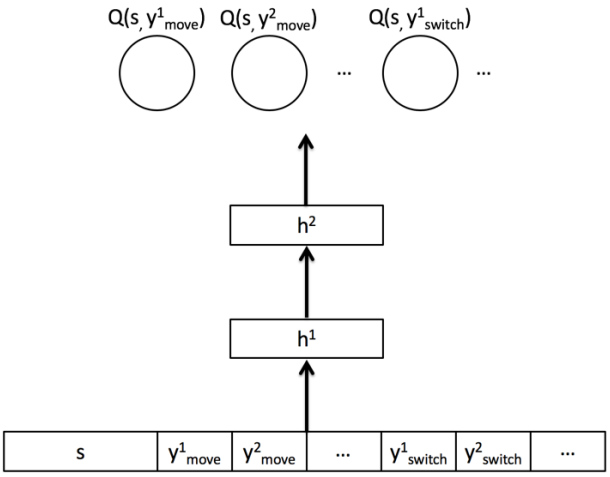

図 2　一般的な DQN のネットワーク概要
Fig. 2　The typical network architecture of DQN

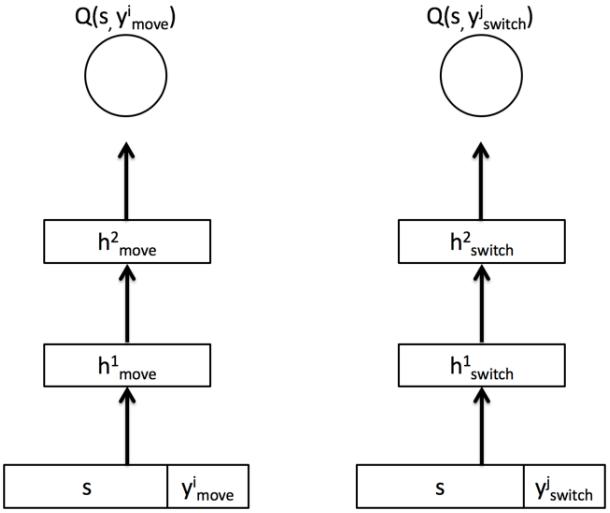

図 3　提案手法のネットワーク概要
Fig. 3　The proposed network architectures

提案手法のアルゴリズムを Algorithm 1 に示す．

4.　実験

行動の表形式のデータを埋め込んだことによる効率化，技・交代ポケモンそれぞれの価値を個別に評価したことによる効率化を検証するために計算実験を行った．

本実験は，第 1 世代と呼ばれるポケモン 151 種を使用し，それぞれのチームのポケモンはランダムに決定される設定を用いた．対戦相手は Pokémon Showdown Competition で提供されている多層パーセプトロンによる実装を用いた．100 ステップ学習する毎に 100 回対戦を行い，勝率で評価を行った．

提案手法による学習効率化の効果を検証するために，DQN，AutoEncoder のみを取り入れた提案手法 （AE），行動の個別評価のみの手法（PA-DQN），AutoEncoder と行動の個別評価を取り入れた提案手法 （AE+PA-DQN），

ⓒ 2019 Information Processing Society of Japan

Algorithm 1 提案手法

master data of moves m_{move}
master data of switches m_{switch}
activation function f
for epoch = 1, ..., B **do**
  $y_{move} = f(W_{move}m_{move} + b_{move})$
  $x'_{move} = f(W'_{move}y_{move} + b'_{move})$
  $y_{switch} = f(W_{switch}m_{switch} + b_{switch})$
  $x'_{switch} = f(W'_{switch}y_{switch} + b'_{switch})$
  Perform a gradient descent step on $(m_{move} - x'_{move})^2$ with respect to the network parameters
  Perform a gradient descent step on $(m_{switch} - x'_{switch})^2$ with respect to the network parameters
end for
replay memory $\mathcal{D}$
for episode = 1, ..., M **do**
  restart game
  set s_1 = state
  for t = 1, ..., T **do**
    if with probability ϵ **then**
      select a random action a_t
    else
      compute each action value $Q(s, y^i_{move}), Q(s, y^j_{switch})$
      select an action a_t with max value
    end if
    execute action a_t
    observe reward r_t and state s_{t+1}
    set A_{t+1} = next list of actions
    store transition $(s_t, a_t, r_t, s_{t+1}, A_{t+1})$ in $\mathcal{D}$
    sample mini batch of transitions $(s_k, a_k, r_k, s_{k+1}, A_{k+1})$ from $\mathcal{D}$
    if s_{k+1} is terminal **then**
      set $y_k = r_k$
    else
      set $y_k = r_k + \gamma max_{a \in A_{k+1}} Q(s_{k+1}, a)$
    end if
    Perform a gradient descent step on $(y_k - Q(s_k, a_k))^2$ with respect to the network parameters
  end for
end for

それぞれで学習させた際の学習効率を比較する．

DQN では**表 3** に示した 464 次元のデータをゲームの状態とした．選択可能な技が 4 個に満たない場合と選択可能な交代ポケモンが 5 体に満たない場合は，状態の対応する要素を 0 埋めした．

AE では，DQN に入力する状態のうち選択肢のデータを埋め込み表現に置き換え，**表 4** に示した 230 次元のデータをゲームの状態とした．DQN と同様に選択可能な行動が少ない場合は対応する要素を 0 埋めした．

PA-DQN では**表 5** に示した 86 次元のデータをゲームの状態とし，行動に対応する入力は技と交代ポケモンのマスタデータ表現（表 1，表 2）をそのまま利用した．

AE+PA-DQN では表 5 に示した 86 次元のデータをゲームの状態とし，行動に対応する入力は技と交代ポケモンの 16 次元の埋め込み表現（y_{move}，y_{switch}）を利用した．

埋め込み表現獲得のための AutoEncoder は 5000 エポックの学習を行い，全ての手法で同じ埋め込み表現を使っている．

各手法で実験を 5 回繰り返して，**図 4** に各手法の学習ステップ毎の勝率を示した．実線は勝率の平均を表し，塗りつぶされた領域は標準偏差を表している．

行動の価値を個別に評価していない DQN と AE では，性能が向上し始めるには約 60,000 ステップの学習が必要となり，AutoEncoder による埋め込み表現だけでは学習効率化の効果が得られなかった．一方，行動の価値を個別に評価した PA-DQN と AE+PA-DQN では学習開始直後か

表 3 DQN で用いた状態
Table 3 States used by DQN

状態	次元
敵戦闘中ポケモンの性能	42
敵戦闘中ポケモンの残り体力	1
味方戦闘中ポケモンの性能	42
味方戦闘中ポケモンの残り体力	1
技 1 の性能	42
技 2 の性能	42
技 3 の性能	42
技 4 の性能	42
交代ポケモン 1 の性能	42
交代ポケモン 2 の性能	42
交代ポケモン 3 の性能	42
交代ポケモン 4 の性能	42
交代ポケモン 5 の性能	42

表 4 AE で用いた状態
Table 4 States used by AE

状態	次元
敵戦闘中ポケモンの性能	42
敵戦闘中ポケモンの残り体力	1
味方戦闘中ポケモンの性能	42
味方戦闘中ポケモンの残り体力	1
技 1 の埋め込み表現	16
技 2 の埋め込み表現	16
技 3 の埋め込み表現	16
技 4 の埋め込み表現	16
交代ポケモン 1 の埋め込み表現	16
交代ポケモン 2 の埋め込み表現	16
交代ポケモン 3 の埋め込み表現	16
交代ポケモン 4 の埋め込み表現	16
交代ポケモン 5 の埋め込み表現	16

表 5 PA-DQN，AE+PA-DQN で用いた状態
Table 5 States used by PA-DQN and AE+PA-DQN

状態	次元
敵戦闘中ポケモンの性能	42
敵戦闘中ポケモンの残り体力	1
味方戦闘中ポケモンの性能	42
味方戦闘中ポケモンの残り体力	1

ⓒ 2019 Information Processing Society of Japan

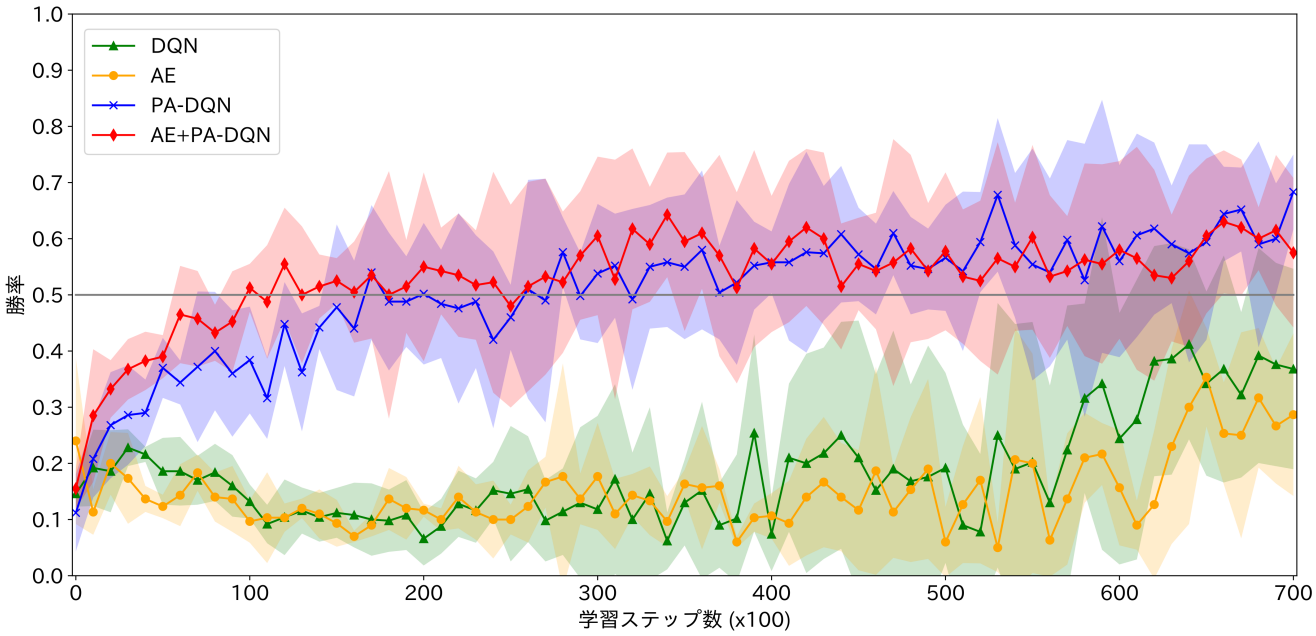

図 4　学習ステップ毎の勝率および標準偏差
Fig. 4　Learning steps v.s. winning percentages with stds

ら性能の向上が始まっている．

PA-DQN と AE+PA-DQN の両手法とも最終的に約 60%の勝率を達成した．しかし，勝率 50%を達成するために PA-DQN では約 17,000 ステップの学習が必要だったのに対し，埋め込み表現を用いた AE+PA-DQN では勝率 50%を達成するのに必要な学習ステップは約 10,000 ステップに削減された．埋め込み表現だけでは学習効率化の効果が得られなかったが，行動の価値を個別に評価するネットワークと事前に獲得した埋め込み表現を組み合わせることにより，強化学習が効率化できている．

5. 考察

全ての行動を状態として扱い全ての行動の価値を同時に評価する標準的な DQN のネットワーク構造では，性能が向上するまでに必要な学習ステップが非常に多かった．これは，ネットワークのパラメータ数が増えるだけでなく，行動を状態に含めることで状態の多様性が高くなり，シミュレーション中に同じような状態が出現する頻度が少なくなることが原因だと考えられる．技やポケモンの並び順によって状態の多様性は指数関数的に増えていくため，埋め込み表現によって状態の次元を削減するだけでは学習効率化の効果が得られなかったと思われる．一方，行動の価値を個別に評価するネットワーク構造では状態の多様性が低くなったため学習できたと考えられる．

深層強化学習は方策の学習と同時に状態からの特徴抽出も行っていると見なせる．しかし，事前に埋め込み表現を獲得しておくことで強化学習の実行時に特徴抽出の必要性が減り，学習効率が向上したと考えられる．

埋め込み表現を用いた場合は，用いなかった場合と比べて最終的な性能が劣っている．AutoEncoder による埋め込みではすべての特徴を同等に扱うため，ゲーム中で重要な特徴を考慮できなかったからではないかと推測される．

6. まとめ

本研究では，ターン制コマンドバトルゲームに行動の価値を個別に評価するネットワークを適用し，AutoEncoder によって獲得した行動の埋め込み表現を用いる手法を提案した．計算実験によって，行動の価値を個別に評価することでゲームプレイ AI の学習が効率化され，行動の埋め込み表現を用いることでさらなる学習効率化が可能であることを示した．

性能向上のためのドメイン知識を活用した埋め込み表現の適用や，多数のキャラクタの行動を同時に選択するゲームへの適用は今後の課題としたい．

参考文献

[1] Isaksen, A., Gopstein, D. and Nealen, A.: Exploring Game Space Using Survival Analysis., FDG (2015).
[2] Gudmundsson, S., Eisen, P., Poromaa, E., Nodet, A., Purmonen, S., Kozakowski, B., Meurling, R. and Cao, L.: Human-Like Playtesting with Deep Learning, pp. 1–8 (online), DOI: 10.1109/CIG.2018.8490442 (2018).
[3] Silva, F. D. M., Borovikov, I., Kolen, J., Aghdaie, N. and Zaman, K.: Exploring gameplay with AI agents, *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference* (2018).
[4] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, The MIT Press, second edition (2018).

ⓒ 2019 Information Processing Society of Japan

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari With Deep Reinforcement Learning, *NIPS Deep Learning Workshop* (2013).
[6] Lample, G. and Chaplot, D. S.: Playing FPS games with deep reinforcement learning, *Thirty-First AAAI Conference on Artificial Intelligence* (2017).
[7] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al.: Mastering the game of go without human knowledge, *Nature*, Vol. 550, No. 7676, p. 354 (2017).
[8] OpenAI: OpenAI Five, https://blog.openai.com/openai-five/ (2018).
[9] Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D. and Silver, D.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).
[10] Dulac-Arnold, G., Evans, R., Sunehag, P. and Coppin, B.: Reinforcement Learning in Large Discrete Action Spaces, *CoRR*, Vol. abs/1512.07679 (online), available from ⟨http://arxiv.org/abs/1512.07679⟩ (2015).
[11] He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L. and Ostendorf, M.: Deep Reinforcement Learning with a Natural Language Action Space, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, pp. 1621–1630 (2016).
[12] Fulda, N., Ricks, D., Murdoch, B. and Wingate, D.: What can you do with a rock? affordance extraction viaword embeddings, pp. 1039–1045 (2017).
[13] Lee, S. and Togelius, J.: Showdown AI competition, *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 191–198 (online), DOI: 10.1109/CIG.2017.8080435 (2017).
[14] Yannakakis, G. N. and Togelius, J.: *Artificial Intelligence and Games*, Springer (2018).
[15] Świechowski, M., Tajmajer, T. and Janusz, A.: Improving Hearthstone AI by Combining MCTS and Supervised Learning Algorithms, *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, pp. 1–8 (2018).
[16] Watkins, C. J. C. H.: Learning from delayed rewards, *PhD thesis, Cambridge University*.
[17] Van Hasselt, H., Guez, A. and Silver, D.: Deep reinforcement learning with double q-learning, *Thirtieth AAAI Conference on Artificial Intelligence* (2016).
[18] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M. and Freitas, N.: Dueling Network Architectures for Deep Reinforcement Learning, *International Conference on Machine Learning*, pp. 1995–2003 (2016).
[19] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D.: Rainbow: Combining improvements in deep reinforcement learning, *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
[20] Ha, D. and Schmidhuber, J.: World Models, *CoRR*, Vol. abs/1803.10122 (2018).
[21] Cuccu, G., Togelius, J. and Cudré-Mauroux, P.: Playing atari with six neurons, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 998–1006 (2019).
[22] Nagabandi, A., Kahn, G., Fearing, R. S. and Levine, S.: Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 7559–7566 (2018).
[23] Chua, K., Calandra, R., McAllister, R. and Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models, *Advances in Neural Information Processing Systems*, pp. 4754–4765 (2018).
[24] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, *Neural and Information Processing System (NIPS)*, (online), available from ⟨https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf⟩ (2013).
[25] Hinton, G. E. and Salakhutdinov, R. R.: Reducing the dimensionality of data with neural networks, *science*, Vol. 313, No. 5786, pp. 504–507 (2006).

ⓒ 2019 Information Processing Society of Japan