ローグライクゲームによる強化学習ベンチマーク環境 Rogue-Gymの提案

金川裕司[1,a)]　金子 知適[2,3,b)]

概要：深層強化学習研究では Arcade Learning Envinronment をはじめ、ゲームを題材にした様々な実験環境が使われている。しかし、報酬の遅延等があり行動計画が必要なドメインは RTS など大規模な計算資源を必要とするものが多い。そこで本論文では、報酬の遅延や部分観測性を持つゲームとしてローグに着目し、ローグをもとに様々な規模の実験に対応できるよう柔軟な設定機能を持たせた実験環境 Rogue-Gym を提案する。更に、実際にこれを使用した学習実験を行い、ベンチマーク環境としての有用性を示す。

A new reinforcement learning platform based on Rogue: Rogue-Gym

Yuji Kanagawa[1,a)]　Tomoyuki Kaneko[2,3,b)]

Abstract: In deep reinforcement learning research, many games are used as experiment environment, such as Arcade Learning Environment. However, environments which involve planning, like RTS, often need massive computation resources. So we focus on Rogue as a game with delayed rewards and partial observation. We propose Rogue-Gym, which is an experiment environment based on Rogue and also highly configurable. In addition, we demonstrate its usefulness by showing the results of standard reinforcement learning algorithms in Rogue-Gym.

1. はじめに

強化学習は状態・行動に対する確率的遷移と報酬が定まった系において、実際にエージェントを行動させ、得られたサンプル報酬から最適方策を予測する手法である。

これは、環境に関する事前知識を用いずエージェントを学習させられる有望な方法であるが、学習に要する時間が状態数に応じて指数的に増加するため、状態数の多い問題への適用は長年の研究課題だった。しかし、DQN [1] とそれに続く一連の深層強化学習研究は、ニューラルネットワークを関数近似器として用いることでビデオゲームなど高次元入力をとるドメインへの適用に成功し、ブレイクスルーとなった。

しかし、その理論的説明は未だ困難であり、多くの論文がベンチマーク課題の実験結果に基づいてアルゴリズムの有用性を主張している。そのため適切なベンチマーク環境を使用することは強化学習研究にとって重要だが、Arcade Learning Environment [2] など標準的なベンチマーク環境では、後述する部分観測性や報酬の遅延など、実験に必要な性質を後から調整することが困難なことが多い。

そこで、本研究では、現在のベンチマーク環境を補うものとして、ローグをもとにしたゲーム上で手軽に実験ができるベンチマーク環境 Rogue-Gym を提案する。更に、Rogue-Gym 上で深層強化学習の標準的手法を用いた実験を行い、ローグのどのような性質が強化学習手法を適用する上で課題になるかを考察した上で、今後の課題を提案する。

2. 問題設定

強化学習における環境の定式化として、割引マルコフ決

[1] 東京大学教養学部学際科学科
[2] 東京大学大学院情報学環
Interfaculty Initiative in Information Studies, the University of Tokyo
[3] 国立研究開発法人科学技術振興機構さきがけ
JST, PRESTO
[a)] kanagawa-yuji968@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org

定過程および割引部分観測マルコフ決定過程を紹介する。また、価値ベース強化学習手法の代表的手法である Q 学習と DQN を紹介する。

2.1 割引マルコフ決定過程 (MDP)

系の全体は状態集合 $s \in \mathcal{S}$ により与えられる。エージェントは各時間 t においてある状態 s_t で 行動 $a_t \in \mathcal{A}(s_t)$ を選択し、状態遷移確率 $P(s_{t+1}|s_t, a_t)$ に従って次の状態 s_{t+1} に遷移する。状態遷移に伴い、エージェントは報酬 $r_t = R(s_{t+1}, s_t, a_t)$ を受けとる。エージェントはこの環境で割引報酬和 $G_t = \sum_{t=0}^{\infty} \gamma^t r_t$($\gamma$ は $0 < \gamma < 1$ を満たす定数) を最大にすることを目標に行動し、これを実現する方策 $\pi(a_t|s_t)$ を最適方策と呼ぶ。状態遷移確率と報酬がマルコフ性を満たすことから、これを割引マルコフ決定過程と呼ぶ。

2.2 割引部分観測マルコフ決定過程 (POMDP)

上記の設定ではエージェントが状態 s を直接観測できたのに対し、ある状態に対するエージェントの観測値 $o_t \in \omega$ が確率 $P(o|s, a)$ により定まる場合を考える。これを部分観測マルコフ決定過程 [3] と呼び、通常のマルコフ決定過程より難しくなる。

2.3 Q 学習と DQN

ある方策 π について、状態行動価値関数 Q を、状態 s で行動 a を選択した時の割引報酬和の期待値

$$Q^{\pi}(s, a) := \mathbb{E}\left(\sum_{t} = t_0^{\infty} \gamma^t r_t\right) \quad (s_0 = s, a_0 = a)$$

として定める。このとき、最適方策 π' における関数 Q は以下のベルマン方程式

$$Q^{\pi'}(s, a) = \mathbb{E}_s \pi(s'|s, a)\left(R(s', s, a) + \mathbb{E}_{a \in \mathcal{A}(s)} \gamma \pi(a'|s') Q^{\pi'}(s', a')\right)$$

により定まる。これは、環境の完全なモデル (全状態と遷移確率) が与えられた時、反復法により解くことができる [4]。これを、エージェントが実際に行動して得たサンプル報酬からモンテカルロ近似して求める手法が Q 学習 [5] であり、更新式は学習率を α_t として

$$Q(s, a) = Q(s, a) + \alpha_t\big(r_t + \max_{a_{t+1}} \gamma Q(s_{t+1}, a_{t+1}) - Q(s, a)\big)$$

となる。これは関数 Q を全ての状態・行動のペアに対応するテーブルで表現すれば収束が保証される。しかし、状態数に対し指数的に学習時間が増大するため、よりパラメタの少ない関数を用い近似的に表現することが望ましい。そこで、DQN [1] では 2 つのニューラルネットワークを Q 値を近似する関数 $Q_{\text{target}}, Q_{\text{train}}$ として用い、損失関数を

$$\left(r + \gamma \max_{a'} Q_{\text{target}}(s', a') - Q_{\text{train}}(s, a)\right)^2$$

として誤差逆伝播法で Q_{train} を学習させた。Q_{target} には過去の Q_{train} の値を用い、適宜同期させる。時系列相関を軽減させるため、r, s, a, s' は過去の経験からランダムサンプリングしたものを使う。

3. 既存のベンチマーク環境

3.1 Arcade Learning Envinronment

現在強化学習研究において広く使用されているベンチマーク環境に ArcadeLearning Envinronmet [2](ALE) がある。これはファミリーコンピュータ登場前の第二世代にあたる家庭用据置きゲーム機 Atari2600 のエミュレータと、その上で動作する 76 種類のゲームに対する AI 用のラッパーを提供する。このベンチマーク環境には、実験用のゲームではなく実際に遊ばれていたゲームで実験できる、ゲームの種類が多く様々な環境でアルゴリズムの性能を検証できるといったメリットがある。

その一方で、ALE はある入力 (行動) に対する状態変化を得るのに適当な頻度 (文献 [1] で用いられた 4 フレームごとが一般的) を決めてサンプリングしなければならない、実験したい内容に合わせてゲームの内容を細かく変更できない等の制限がある。文献 [6] では POMDP 環境を実現するために確率 0.5 でゲームの画面を隠すなど、様々な工夫がなされているが、思い通りにゲームを制御することは難しい。

3.2 StarCraft 2, Dota 2

近年有力な研究機関が使用し注目されているベンチマーク環境として、Star Craft2 [7] や dota2 といったリアルタイムシュミレーション (RTS) がある。一般に、強化学習では報酬とそれを誘発した行動との間のステップ数の隔たりが大きければ大きいほど、学習が困難になるとされている [1]。これは、Q 学習の場合は Q 値が一度に 1 ステップぶんしか更新されないため、伝播が難しくなるためだと説明できる。RTS はこの報酬の遅延がある環境でのベンチマークに向いているとされているが、これらは非常に大規模かつ複雑なオンラインゲームであり、学習を収束させることは困難である。OpenAI Five [8] では 1024 もの LSTM units から成る巨大なネットワークをクラウド上で高速に実験を回すことで学習させたが、このような手法を個人研究者や小規模な研究機関が採用することは難しい。

4. 提案する環境

本研究では、上に挙げたようなベンチマークの欠点をふまえ、これらを補うものローグライクゲームを題材として用いたベンチマーク環境 rogue-gym を提案する。

4.1 ローグ

ローグ[*1]は 80 年代に BSD UNIX 用に開発された CUI 表示のターン制コマンド選択型 RPG で、プレイヤはランダムに生成されるダンジョンでモンスターとの戦闘やアイテム収集・戦闘を行いながら、ダンジョンの地下 25 階にあるイェンダーの書を持ち帰ることを目指す。ローグは直系の子孫である Nethack、Angband など多くのフォロワーを生み、今日ではダンジョンがランダム生成されるゲーム全般をローグライクと呼びならわすほど大きな影響を与えた。

4.2 実験環境としての特徴

ローグが強化学習の実験環境として有用である理由を、ローグが持つ三つの特徴と関連づけて説明する。

4.2.1 事前知識

ローグを上手にプレイするためには、アイテム、敵などに関する大量の事前知識を持っている必要がある。これはモデルベース探索手法の適用を著しく困難にする。第一著者は実際にオリジナル版ローグを用いた探索ベースのエージェントを作成した経験がある[*2]が、敵の強さなど非常に多くの情報をプログラム上に埋めこむ必要があり、作成は難しかった。先行研究 [9] としてエキスパートシステムとしてローグのエージェントを実現した例があるが、これは探索と人間が考えた戦略を組み合わせるもので、やはり人間の知識に負う部分が著しく多い。

このようなドメインで事前知識を仮定せずに強化学習が成功すれば、人間が探索を書く場合と比べ大幅に負担が軽減される。そのため、強化学習の研究として、挑戦する価値の高いドメインである。

4.2.2 行動の階層性

ローグでは、移動・アイテムの使用といったミクロな行動が、しばしば「敵を到す」「HP を回復する」といったより上位の行動の一部となっている。このように行動が階層性を持つゲームでは、報酬の遅延がより顕著な形で現れるため、強化学習をそのまま適用することは難しい。文献 [1] では、ALE に含まれるモンテズマ・リベンジという「鍵をとって、部屋から出る」という単純な階層行動を持つゲームで学習がうまくいかなかったという報告がなされている。

一方、この階層性を単なる報酬の遅延ではなく、学習を成功させるための鍵とみなすアイデアもある。文献 [10] では、サブゴールを設定することで、上位の行動をマルコフ決定過程上での行動列として扱う表現する手法を提案している。この手法をベースとするものをはじめ、階層構造を考慮した学習法には様々な手法が提案されているが、いまだ発展途上にある。そのため、階層構造を考慮した強化学習の研究のベンチマーク環境という点からも、ローグを用いることには有用性がある。

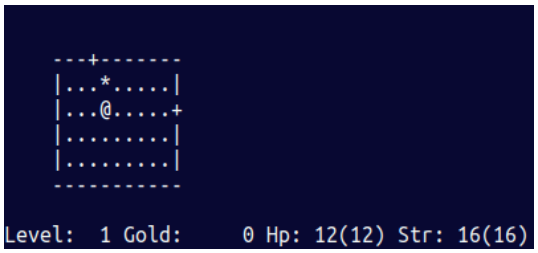

図 1 初期状態のダンジョン

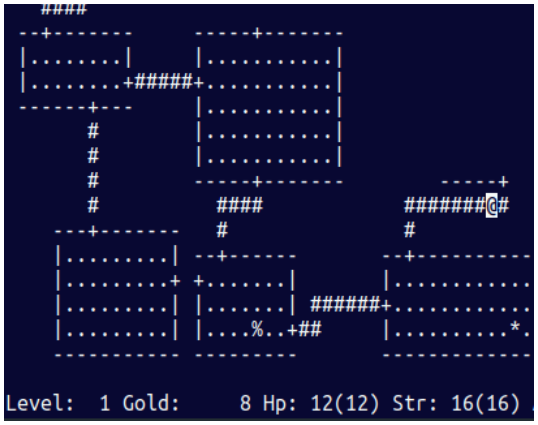

図 2 ある程度探索したダンジョン

4.2.3 部分観測性

ローグでは一度も入ったことがない部屋は表示されないようになっており (図 1, 図 2)、わかりやすいかたちで部分観測性が顕れている。そのため、エージェントが直面する困難として自然な設定で部分観測問題に対するベンチマークを行うことができる。

4.3 実験環境の実装

オリジナル版ローグのソースコードを実験に用いることも検討したが、タイムアウトを使わないとターンの終了を検知できない、ダンジョンの大きさ等パラメタがハードコーディングされていて改造が難しいなど、AI 実験用に用いるには不便な部分が多く見うけられた。そこで、独自のプログラムを実装した。

先行研究 [11] では、ダンジョンの種類が 5 種類に固定されている独自のローグライクゲームを実装した。本研究では、アルゴリズムの汎化性能を見る上でランダム性が重要だと考えたため、ダンジョンが無数に生成される BSD 版ローグとほぼ同じ動作をするゲーム Rogue-Gym を作成した。大きな特徴として、軽量マークアップ言語の JSON を用いてダンジョンの大きさ、隠れ部屋の有無、部屋の数などの設定を詳細に記述できる。これにより、VizDoom [12] など MOD による拡張が可能なベンチマーク環境と比較してもより簡便に、上述した学習を困難にする要素の有無

[*1] https://en.wikipedia.org/wiki/Rogue_(video_game)(Accessed: 2018-10-16)
[*2] https://github.com/kngwyu/rogue-ai-2nd(Accessed: 2018-10-16)

を切りかえ、比較しながら実験することができる。なお、指定しない要素についてはデフォルト設定が適用されるため、設定が過度に面倒になることはない。

現在、敵・アイテムを除き BSD 版ローグ 5.4.4 とほぼ同じ機能を持つ段階までゲームが完成しており、MIT と Apache2.0 のデュアルライセンスのもと第一著者による実装を公開している [*3]。ゲーム本体は五千行程度の Rust 言語用ライブラリとして実装した。AI 用のインターフェースは様々な言語向けにあることが望ましいが、今回は数値計算・機械学習用途で人気がある Python 言語へのバインディングを実装した。これは、ALE など多数のベンチマーク環境をサポートする標準的な強化学習用の環境ラッパーである OpenAI gym [13] の gym.Env クラスを継承しており、既存のコードベースからも利用しやすい。また、curses 相当のライブラリにより人間用の UI も実装した。

5. 実験

実際に Rogue-Gym がベンチマーク環境として使用できることを示すため、Rogue-Gym で深層強化学習の標準的アルゴリズムの一つである Double DQN の性能評価を行った。ニューラルネットワークの実装には PyTorch [*4]0.4.0 を使用した。

5.1 Double DQN

Double DQN [14] は DQN を改良したアルゴリズムで、DQN の損失関数を

$$\left(r + \gamma Q_{\text{target}}\left(s', \max_{a'} Q_{\text{train}}(s', a')\right) - Q_{\text{train}}(s, a)\right)^2$$

に変えて学習する。Q 値の過大評価を抑える効果があり、多くの問題で DQN より性能がいいとされる。

ネットワークの最適化式は RMSProp [15] を用いた。Exploration には ϵ -greedy を用い、ϵ は初期値 1、下限 0.1 としてステップ数に比例させて小さくした。 ハイパーパラメータは付録に記載した。

評価スコアは、ネットワークの出力する Action Value を使用して貪欲法で行動した結果とした。

5.2 ゲームの設定

実験は、全て敵がおらず、アイテムがゴールドしかない簡素化された設定で行った。

ローグのスコア画面では、ダンジョンで拾得したゴールドの数が報酬となる。今回はこれに加え、下の階へ降りた時に疑似報酬として 1000 を与えた。また、2 階に到達したらゲームが終了するという設定にした。

エージェントがとれる行動は、周囲 8 マスへの移動など 10 種類存在する。これは、付録に詳しく示した。ダンジョンは、

- 盤面サイズ 16×32 シード 5 (以下、サイズ 32 と呼ぶ)
- 盤面サイズ 24×48 シード 0 (以下、サイズ 48 と呼ぶ)

の 2 種類を使った。なお、シードには 128bit 整数を指定できる。

終了までのターン数は、サイズ 32 の場合は 1000 ステップ、サイズ 48 の場合は 5000 ステップとした。

5.3 ニューラルネットワークへの入力

DQN など多くの深層学習アルゴリズムは、入力画像を畳みこみニューラルネットワークで処理することを前提としている。そこで、以下に示すような情報をチャンネル数×ダンジョンの幅×ダンジョンの高さの三次元画像として重ねることで入力を構成した。ただし、盤面以外のヒストリについては、一部の実験においてのみ使用した。プレイヤのステータスは、今回の実験では入力としなかった。

5.3.1 ダンジョン

ローグをはじめ curses による UI を持つ多くのローグライクゲームでは、プレイヤが@、壁が#など、特定の意味を割りあてた ASCII 文字を並べることで盤面を表現しており、Rogue-Gym もこれを用いている。そこで、ゲーム中に現れるブロックに 0-index な番号づけを施した上で、以下の 2 種類の盤面表現を用い、結果を比較した。

- $$\text{image}[0][j][k] = \frac{\text{block}[j][k]}{\max_{j',k'} \text{block}[j'][k']}$$
  として構成した 1 チャネルのグレー画像
- $$\text{image}[i][j][k] = \begin{cases} 1 & (\text{block}[j][k] = i) \\ 0 & (\text{block}[j][k] \neq i) \end{cases}$$
  として構成したブロック数分チャネルを持つシンボル画像 (i はブロックの番号)

後者については、二次元のシンボル表現を畳みこみニューラルネットワークの入力とする研究として AlphaGo [16] を参照した。今回使用した敵がいない設定では、ブロック数は 17 になった。

5.3.2 エージェントの行動履歴

一部の実験では、メタデータとしてダンジョンのうちエージェントが通ったことのある部分を 1、通ったことのない部分を 0 とする二次元画像を与えた。これはゲームの内部情報を利用しているため、理想的にはない方が良い。

5.4 実験の結果と考察

5.4.1 グレー画像とシンボル画像

まず、

- サイズ 32
- ダンジョンを隠さない

[*3] https://github.com/kngwyu/rogue-gym(Ac cessed: 2018-10-16)

[*4] https://pytorch.org(Accessed: 2018-10-16)

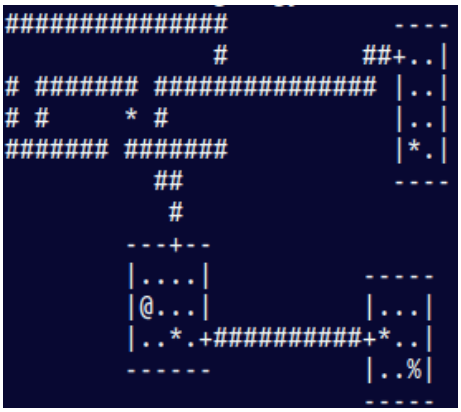

図 3　サイズ 32 のダンジョン
Fig. 3　Size32 dungeon

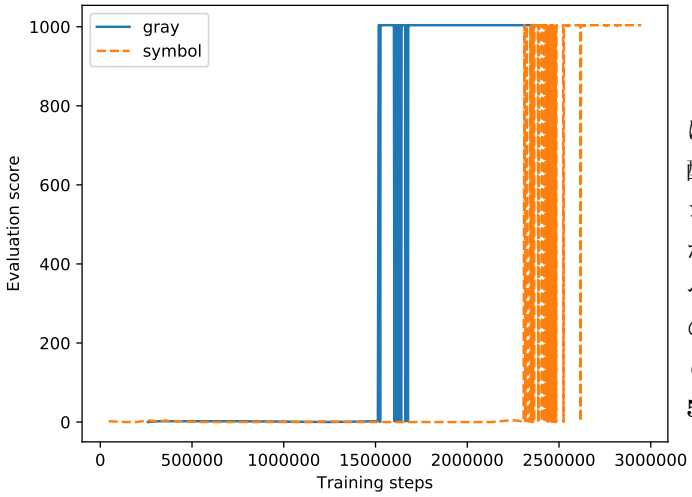

図 4　グレー画像とシンボル画像
Fig. 4　gray scaled image and symbol image

という設定で、グレー画像とシンボル画像を入力として用いた場合の結果を比較した。このダンジョンの全体を図 3 に示した。@がプレイヤ、. が部屋の床、#が通路、+ がドア、-|が壁、%が階段、*が金貨を示している。x 軸を学習に使ったステップ数、y 軸を評価スコアとしたグラフを図 4 に示した。図の通り、グレー画像を入力とした方が速く学習が収束している。これは、ネットワークのパラメタが少ないためだと考えられる。

ただし、学習したエージェントの評価スコアはどちらの場合も 1004 となったが、これは最大スコアではない。図 3 のダンジョンにはスコア 2 の金貨が 4 つあり、これら全てと地下へ降りた時の疑似報酬 1000 の合計 1008 が最大スコアである。しかし、学習させたエージェントは、まっすぐ階段に向かうという行動をとった。今回使用した Double DQN をはじめ、価値ベースの強化学習アルゴリズムではエージェントがランダム行動して収集した報酬・行動のサンプルを使って学習する。そのため図 3 のダンジョンで

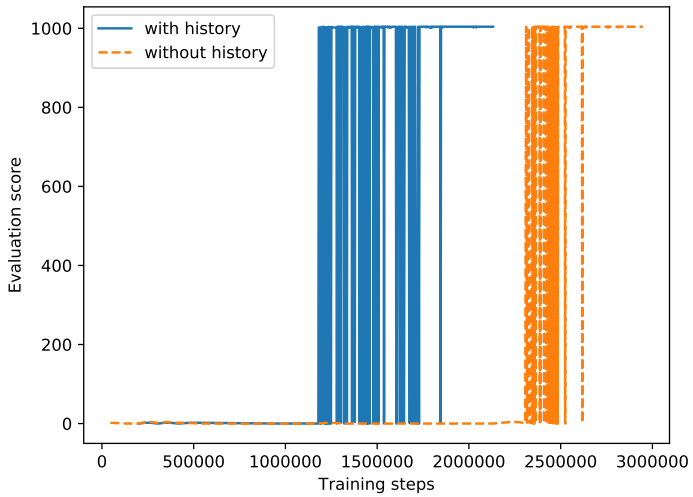

図 5　ヒストリありとヒストリなし
Fig. 5　with history and without history

は、ダンジョン上部で金貨を回収したのち階段へ向かい報酬 1000 を得る、という行動サンプルを得ない限り、ダンジョン上部に向かう行動に高い Q 値はつかない。ところが、ランダム行動からそのような行動列を得るのは、 階段へまっすぐ向かうサンプルを得るよりもずっと難しい。そのため、ダンジョン中を探索するようなエージェントにならなかったと考えられる。

5.4.2　ヒストリ

5.4.1 節と同じ設定で、入力を

- シンボル画像
- シンボル画像+ヒストリ

の 2 種類で実験を行い、結果を図 5 に示した。収束時のぶれが非常に大きくでているが、ヒストリを入力に加えた方が速く学習が収束している。なお、収束したスコアは 5.4.1 節と同じ 1004 である。

次に、

- サイズ 32
- ダンジョンを隠す (入った部屋のみ表示される)

という設定で実験を行い、結果を図 6 に示した。図から、ヒストリなしの場合は全く学習できていない一方、ヒストリありのものは学習できており、ヒストリの有無が学習に大きな影響を与えていることがわかる。図 7 に示したとおり、このダンジョンでは初期位置が「暗い部屋」になるためダンジョンの大部分が隠れる。しかし、階段に辿りつくため必要な行動は全く変わらないため、この結果は予想できなかった。ヒストリなしの場合、エージェントの動き方によって画像が全く違ったものになるため、学習が安定しなかったと考えられる。

5.4.3　ダンジョンの大きさ

追加実験として、サイズ 48 のダンジョン (図 8) で、性

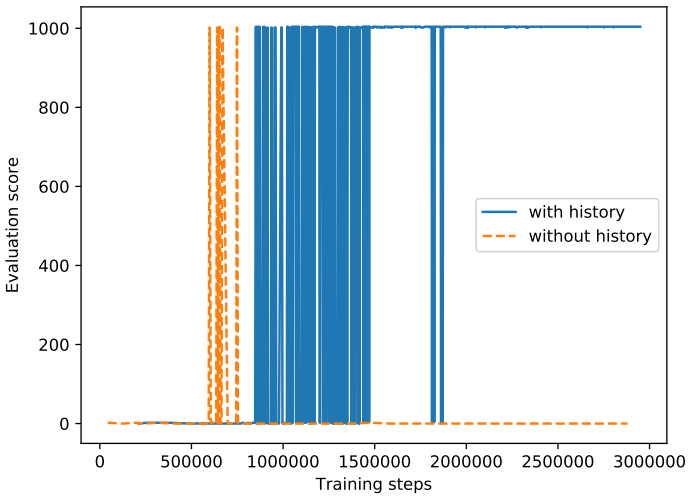

図 6　ヒストリありとヒストリなし ダンジョンを隠す場合
Fig. 6　with history and without history hide dungeon

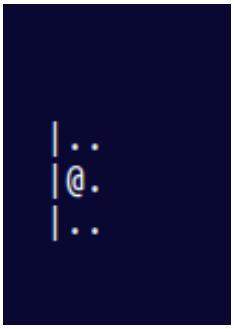

図 7　サイズ 32 のダンジョン ダンジョンを隠す場合
Fig. 7　Size32 hidden dungeon

能が良かったグレー画像とシンボル+ヒストリーの学習結果を比較した。執筆時点では十分な学習ステップが確保できていないが、途中経過を図 9 に示した。次の階に下りる行動をとるのに、サイズ 32 のものより多くの学習ステップ数がかかっていることがわかる。

6.　まとめ

本稿では、既存の強化学習用実験環境を補うものとして、ローグを題材にした実験環境 Rogue-Gym を提案した。更に、これを使って様々な設定で実験を行い、実験環境としての有用性を示した。

実験からは、階を下りる時に疑似報酬を与える設定でそのまま学習させると、ダンジョンを探索せず階を下りるようにエージェントを学習させてしまうことがわかった。今後の課題の一つとして、これについて詳しく調べるため、疑似報酬を与えない設定での実験や、ダンジョン全体を探索するような Exploration の工夫が挙げられる。

また、部分観測設定ではヒストリの有無が学習に大きく影響を与えることがわかった。これについては、再帰的

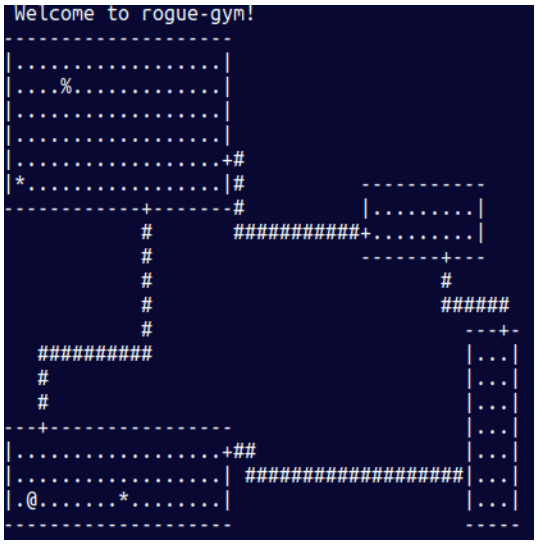

図 8　サイズ 48 のダンジョン
Fig. 8　Size48 dungeon

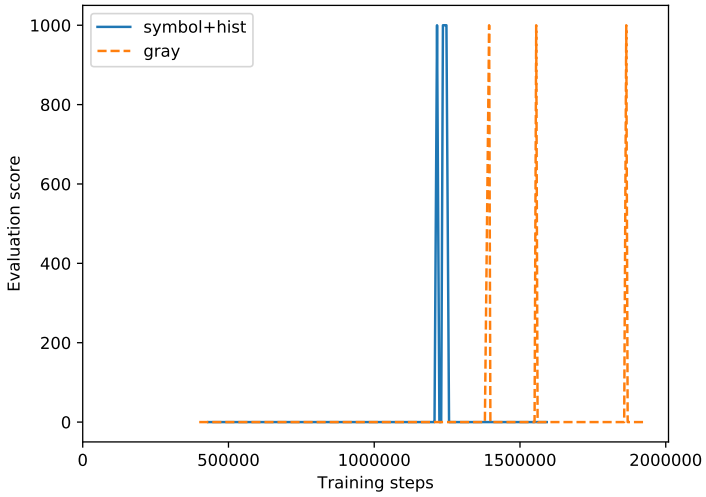

図 9　サイズ 48 のダンジョンでの実験 シンボル+ヒストリとグレー画像
Fig. 9　Size48 dungeon symbol+history and gray scale image

ニューラルネットワークなど記憶を保持するようなアルゴリズムを用いヒストリを使わずに学習させられるかどうかが課題となる。また Rogue-Gym について、チュートリアルなど利用者に向けた文書は本論文執筆時点では書かれていないが、今後速やかに整備される予定である。

謝辞

この研究の一部は，JSPS 科研費 16H02927 と JST さきがけの支援を受けています。

参考文献

[1]　Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller,

M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, pp. 529–533 (online), DOI: 10.1038/nature14236 (2015).

[2] Bellemare, M. G., Naddaf, Y., Veness, J. and Bowling, M.: The Arcade Learning Environment: An Evaluation Platform for General Agents, *CoRR*, Vol. abs/1207.4708 (online), available from ⟨http://arxiv.org/abs/1207.4708⟩ (2012).

[3] Kaelbling, L. P., Littman, M. L. and Cassandra, A. R.: Planning and Acting in Partially Observable Stochastic Domains, *Artif. Intell.*, Vol. 101, No. 1-2, pp. 99–134 (online), DOI: 10.1016/S0004-3702(98)00023-X (1998).

[4] Bellman, R. and Kalaba, R.: On the role of dynamic programming in statistical communication theory, *IRE Trans. Information Theory*, Vol. 3, No. 3, pp. 197–203 (online), DOI: 10.1109/TIT.1957.1057416 (1957).

[5] Watkins, C. J. C. H. and Dayan, P.: Q-learning, *Machine Learning*, Vol. 8, No. 3, pp. 279–292 (online), DOI: 10.1007/BF00992698 (1992).

[6] Hausknecht, M. J. and Stone, P.: Deep Recurrent Q-Learning for Partially Observable MDPs, *CoRR*, Vol. abs/1507.06527 (2015).

[7] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T. P., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J. and Tsing, R.: StarCraft II: A New Challenge for Reinforcement Learning, *CoRR*, Vol. abs/1708.04782 (online), available from ⟨http://arxiv.org/abs/1708.04782⟩ (2017).

[8] Petrov, J. R. M., Pachocki, J., Brockman, G., Wolski, F., Zhang, S., Chan, B., Józefowicz, R., Pondé, H., Dębiak, P., Dennison, C., Farhi, D., Sidor, S., Tang, J., Yoon, D., Sigler, E., Christiano, P., Luan, D., Sutskever, I., Hashme, S., Radford, A., Hesse, C., Schneider, J., Berner, C., Schulman, J., Schiavo, L., Clark, J., Fischer, Q. and Gray, S.: OpenAI Five, `https://blog.openai.com/openai-five` (2018).

[9] Dewdney, A.: An expert system outperforms mere mortals as it conquers the feared dungeons of doom, *SCIENTIFIC AMERICAN*, Vol. 252, No. 2, p. 18 (1985).

[10] Sutton, R. S., Precup, D. and Singh, S. P.: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning, *Artif. Intell.*, Vol. 112, No. 1-2, pp. 181–211 (online), DOI: 10.1016/S0004-3702(99)00052-1 (1999).

[11] 高橋一幸，Temsiririrkkul, S.，池田　心：ローグライクゲームの研究用ルール提案とモンテカルロ法の適用，GPW2017 論文集，Vol. 2017, pp. 19–25 (2017).

[12] Kempka, M., Wydmuch, M., Runc, G., Toczek, J. and Jaśkowski, W.: ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning, *IEEE Conference on Computational Intelligence and Games*, Santorini, Greece, IEEE, pp. 341–348 (2016).

[13] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym (2016).

[14] van Hasselt, H., Guez, A. and Silver, D.: Deep Reinforcement Learning with Double Q-learning, *CoRR*, Vol. abs/1509.06461 (online), available from ⟨http://arxiv.org/abs/1509.06461⟩ (2015).

[15] Tieleman, T. and Hinton, G.: Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning (2012).

[16] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7587, pp. 484–489 (online), DOI: 10.1038/nature16961 (2016).

付　　録

学習率	0.00025
α	0.95
ϵ	0.01

表 A·1　RMSProp のパラメータ

バッチサイズ	32
学習を開始するまでのステップ数	50000
リプレイバッファの大きさ	1000000
ネットワーク同期のスパン	10000

表 A·2　学習パラメータ

層の種類	パラメータ
畳みこみ	カーネルサイズ 8, ストライド 1
ReLU	
畳みこみ	カーネルサイズ 4, ストライド 1
ReLU	
畳みこみ	カーネルサイズ 3, ストライド 1
ReLU	
全結合	出力サイズ 512
全結合	出力サイズ 10

表 A·3　ネットワーク構造

コマンド	意味
`h`	左移動
`j`	上移動
`k`	下移動
`l`	右移動
`n`	右下移動
`b`	左下移動
`u`	右上移動
`y`	左下移動
`>`	階段を降りる
`s`	サーチ (隠し扉・通路が周囲 8 マスに存在した場合運が良ければそれを発見できる)

表 A·4　エージェントがとれるアクション