内部報酬とHybrid Reward Architectureを用いたローグライクゲームの強化学習

加納 由希夫[1,a)]　鶴岡 慶雅[1]

概要：近年，深層学習と強化学習を組み合わせた深層強化学習を用いたゲーム AI の研究が盛んに行われているが，複雑な状態を取るようなゲーム環境を扱う強化学習は，学習が遅く安定しない傾向にある．このようなゲームの一つに，コンピュータ RPG の一種であるローグライクゲーム (Rogue-like games) がある．世界での人気が非常に高いゲームであり，難易度の高いものが多く，プレイングに高度に知的な能力を要求される．そのため，当然このゲームの AI に求められる能力も高度なものになり，学習は容易ではない．本研究の目的は，このローグライクゲームを対象とした強化学習を行い，ゲームを自動攻略できるような AI を作ることである．強化学習の効率化・安定化を実現するためのアプローチとして，本研究では，強化学習における「報酬」の部分に着目した手法を用いる．本稿では，簡単なローグライクゲームの環境を用意し，強化学習のアルゴリズムである A3C に，内部報酬を生成する ICM を組み合せた学習を行い，ICM が学習に与える効果を検証した．

Deep Reinforcement Learning of Roguelike Games Using Internal Rewards and Hybrid Reward Architecture

Yukio Kano[1,a)]　Yoshimasa Tsuruoka[1]

Abstract: In recent years, research on game AI using deep reinforcement learning that combines deep learning and reinforcement learning has been actively conducted. However, reinforcement learning dealing with a complex game environment tends to be slow and unstable. One such game is Rogue-like games, a type of computer RPG. It is a game with a very high popularity in the world, many of them have high difficulty, and highly intelligent ability is required for playing. Therefore, the ability required for the AI of this game naturally becomes sophisticated, it is not easy to learn it. The purpose of this research is to learn for this roguelike game and make AI that can capture the game automatically. As an approach to realize efficiency and stabilization of reinforcement learning, this research focuses on the "reward" part of reinforcement learning. In this paper, we prepare an environment of simple and easy roguelike games, and conducted reinforcement learning by combining ICM generating internal compensation to A3C. Then, we verified the effect.

1. はじめに

近年，ゲーム AI に関する研究において，エージェントがゲーム環境との相互作用の中で自律的に学習を行う強化学習を用いる手法が注目を集めている．特に，深層学習と強化学習を組み合わせた深層強化学習を用いたゲーム AI の研究が盛んに行われている．深層強化学習の代表的な手法の一つに，強化学習の典型的なアルゴリズムである Q 学習と深層学習を組み合わせた Deep Q Network (DQN) があり，ゲームの画面を直接学習に利用する方法で，Atari 2600 のゲームの一部では人間を上回るスコアを記録している [1]．しかし，複雑な状態を取るようなゲームでは，DQN による学習が遅く，不安定になりやすいという問題が存在する．

[1] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] kano@logos.t.u-tokyo.ac.jp

DQN に限らず，複雑な環境を扱う強化学習は学習が遅く安定しない傾向がある．このような問題を解決するアプローチの一つに，強化学習の「報酬」部分に着目するものがある．

Pathak らは，Intrinsic Curiosity Module (ICM) と呼ばれる，環境から得られる既存の報酬 (外部報酬) とは別に，内部報酬として，エージェントが観測する環境の状態がそれまでの観測結果から予測しにくいものであるほど大きくなるような報酬を自動的に生成する手法を提案した [2]．内部報酬を用いることによって，外部報酬に至るまでの学習がより効率的に行われることが期待される．Pathak らは，VizDoom と Super Mario Bros というゲームに対して，Mnih らによる深層強化学習の手法である A3C [3] に，報酬として ICM の生成する内部報酬を用いた強化学習を行い，高い学習結果を示した．Burda らは，ICM の性能の向上を実現するとともに，外部報酬を用いずに内部報酬のみを利用する学習の実験を Atari 2600 をはじめとした大量のゲームで行っており，内部報酬のみで学習を行っても多くのゲームで高い学習結果が得られることを示した [4]．

また，Seijen らは，Hybrid Reward Architecture (HRA) と呼ばれる，問題固有の知識を用いて報酬空間を分割して学習を行う手法を提案した [5]．これは，環境をから得られる報酬を，問題固有の知識に基づいていくつかに分割した後，そのそれぞれの報酬に対して別々のモデルを独立して学習するものである．各モデルがそれに対応する報酬を得るための方策を別々に学習するので，1 つ 1 つのモデルの学習が比較的容易になって全体的に学習の効率がよくなることが期待される．また，各モデルがそれぞれの報酬獲得タスクに特化していくので，環境の状態に応じて最適なモデルを選択して利用するような使い方も考えられる．HRA では，DQN では学習が難しかった Atari 2600 のゲームである Ms. Pac-Man. において高い学習効率を示し，人間を上回るスコアも記録した．

本研究の目的は，コンピュータ RPG の一種であるローグライクゲームを対象とした強化学習を行い，ゲームを自動攻略できるような AI を作ることである．ローグライクゲームとは，1980 年の Rogue から派生したゲームの総称であり，日本では不思議のダンジョンシリーズなどがよく知られている．多くのローグライクゲームでは，プレイヤーが主人公を動かすことで，毎回ランダムに生成されるダンジョンを探索して階段を見つけ出し，より奥へと進むことが主な目的となる．道中では敵としてモンスターが出現するので，ダンジョンで収集できるアイテムなどを利用しながら対処していくのが基本である．ローグライクゲームの攻略には，レベル上げやアイテム回収などを見据えた長期的な戦略を立てる能力や，モンスターなどに対処するための戦術をその場で考える能力，どの程度まで危険を冒してアイテムを回収するべきなのかを判断する能力など様々なものが要求され，人間がプレイしても手応えがあるようなものが多く，ゲーム AI が扱うには非常に難易度の高いタスクであると言える [6]．

ローグライクゲーム攻略の基本となるダンジョン探索については，より予測のつかない未知な場所へと進もうとする ICM の性質と相性が良いと考えられるが，一方で，Burda らは，確率的な環境の状態遷移があるゲームに対しては，内部報酬のみを用いる強化学習がうまく進まなくなる問題を挙げている．これは，ICM が状態の予測誤差を報酬として用いているために，確率的な状態遷移にはうまく対応できないからである [4]．ローグライクゲームは，ダンジョンの生成やモンスターの挙動などが確率的である場合が多く，内部報酬のみで学習を行うのは困難であると予想される．さらに，Burda らは，強化学習がうまく進むような well-shaped な報酬関数を設計するのは非常に困難だとも主張しており [4]，実際，これはローグライクゲームの報酬の設計にも当てはまる問題である．HRA はこの問題を解決するアプローチとなりうると考えられる．なぜなら，HRA では複雑な報酬をいくつかの単純な報酬に分割することで学習の効率化を実現しており，ローグライクゲームの報酬も，階段や食料などのアイテム，モンスターやレベルなど，ほとんどの要素について分割して考えることができるからだ．

本稿では，簡単なローグライクゲームの環境を用意し，A3C と ICM を組み合わせた学習について実験を行った．

2. 関連研究

2.1 強化学習

機械学習の手法の一つに強化学習 [7] がある．

標準的な強化学習のモデルでは，学習の対象とする環境と，実際に学習と意思決定を行うエージェントが存在する．エージェントと環境は離散的な時間ステップ t で連続して相互作用を行う．ある時刻 t において，エージェントは環境から状態 s_t を受け取り，それに基づいて行動 a_t を選択する．これにより，エージェントは環境から報酬 r_{t+1} を受け取り，環境の状態が s_{t+1} に変化したことを知る．この相互作用の繰り返しが基本となる．図 1 にこの相互作用を説明したものを示す．エージェントには環境の状態に応じて何らかの行動を返す関数が実装されており，この関数を方策 (policy) と呼ぶ．一連の行動の結果として受け取る報酬を最大化するような方策を見つけることが強化学習の主な目的となる．[8]

同じ機械学習の手法の一つである教師あり学習と比較すると，強化学習は環境との相互作用で学習を行うので教師データを用意する必要がないという利点がある．一方で，強化学習では明確な指示がない状態で自ら試行錯誤して学

[2] 出典:『強化学習』[8]，p.56. (一部修正)

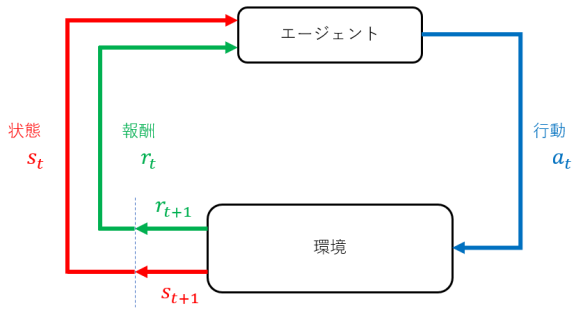

図 1 強化学習におけるエージェントと環境間の相互作用 [1]

習を行うので，学習に必要な時間が長くなることが多い．

また，教師あり学習では，多くの場合，教師データに当てはまるような解に学習が近づいていくが，強化学習では，教師データのように人間が最適だと判断していた解よりも優れた解が学習によって発見される可能性も考えられる．一方で，強化学習によって望ましくない解が得られてしまう可能性も考えられるため，強化学習によって得られた解の妥当性は適切に判断しなければならない．

2.2 Q 学習

代表的な強化学習の手法に Q 学習 [9] がある．Q 学習では，まず環境を有限マルコフ決定過程 (finite Markov decision process, finite MDP) でモデル化する．有限マルコフ決定過程とは，以下の 4 つの要素で表されるモデルである．

- 状態の有限集合：$S = \{s_1, s_2, ..., s_m\}$
- 行動の有限集合：$A = \{a_1, a_2, ..., a_n\}$
- 遷移関数：状態 s で行動 a を選択したときに状態 s' に遷移する確率 $P(s, a, s')$
- 報酬関数：状態 s で行動 a をとったときの報酬 $R(s, a)$

強化学習の目的は，一連の行動の結果として受け取る報酬の総和を最大化するような方策を見つけることである．そこで，各状態 s に対して行動 a を与える関数 $\pi(s) = a$ を方策として表現すると，Q 学習における目的関数は次のように定義できる．

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]_{s_0=s} \tag{1}$$

ただし，γ $(0 < \gamma < 1)$ は割引率と呼ばれる定数である．時間が経つほど割引率によって報酬にペナルティがかけられエージェントが受け取ることのできる報酬が減少していくため，報酬を手に入れるまでの時間を短くするように学習させる狙いがある．式 1 は，状態 s から始めて方策 π に従って行動を選択していった場合に，エージェントが受け取る割引された報酬の和を示している．

ここで，最適化された方策を π^* とすれば，式 1 は次式のように書き直せる．

$$Q^{\pi^*}(s,a) = R(s,\pi^*(s)) + \gamma \max \sum_{s'} P(s,a,s') Q^{\pi^*}(s', \pi^*(s')) \tag{2}$$

式 2 で表されるのは最適化された方策をとったときの行動価値関数であり，この行動価値関数 $Q^{\pi^*}(s,a)$ を，関数 $Q(s,a)$ で近似することによって学習を行うのが Q 学習である．Q 学習では，次のようなアルゴリズムで関数 $Q(s,a)$ を更新し，学習を行っていく．

$$Q(s_t,a_t) \longleftarrow Q(s_t,a_t) + \alpha_t \left(R(s_t,a_t) + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right) \tag{3}$$

ここで，α_t は学習率とよばれ，この学習率 α_t が以下の 2 つの式を満たすとき，関数 $Q(s,a)$ は必ず $Q^{\pi^*}(s,a)$ に収束することが証明されている．

$$\sum_{t=0}^{\infty} \alpha_t = \infty \ , \quad \sum_{t=0}^{\infty} {\alpha_t}^2 < \infty \tag{4}$$

2.3 Deep Q-Network (DQN)

Deep Q-Network (DQN) [1] は，2015 年に発表された Q 学習における行動価値関数 $Q(s,a)$ をニューラルネットワークを用いて学習・近似する手法である．Q 学習で扱うのが難しい状態数の大きな複雑な環境においても，ニューラルネットワークを用いることによって効率的に学習を行えるようになった．Atari 2600 シリーズのクラシックなゲーム環境を 49 個も用いて行われた DQN の実験では，その過半数のゲームで人間のレベルを超える学習結果が得られており，DQN の高い学習能力が示されている．

DQN では，ニューラルネットワークの重みを θ_i とし行動価値関数を $Q_{\theta_i}(s,a)$ と表現する．この行動価値関数を学習するために，誤差関数を Q 学習の式 3 をベースに次のように定義する．

$$L_i(\theta_i) = E\left[\left(R(s,a) + \gamma \max_{a'} Q_{\theta_{i-1}}(s',a') - Q_{\theta_i}(s,a)\right)^2\right] \tag{5}$$

ただし，$Q_{\theta_{i-1}}(s',a')$ の θ_{i-1} とは，一つ前の重み θ を使って計算することを意味する．この誤差関数を微分して得られる次の式を用いて誤差逆伝播法を行うことで，ニューラルネットワークを学習していく．

$$\nabla_{\theta_i} L_i(\theta_i) = E\left[\left(R(s,a) + \gamma \max_{a'} Q_{\theta_{i-1}}(s',a') - Q_{\theta_i}(s,a)\right) \nabla_{\theta_i} Q_{\theta_i}(s,a)\right] \tag{6}$$

DQN を用いた強化学習では，学習を安定化させるための工夫がいくつか存在する [1]．ここでは，その中でも代表的なものを 2 つ紹介する．

2.3.1 Experience Replay

強化学習においてエージェントに与えられるデータは，学習の性質上，時系列的に連続したものになる場合がほとんどである．しかし，このままだとデータとデータの間に相関が生じてしまい，このデータの依存関係が学習に影響を及ぼしうる．

Experience Replay [10] とは，強化学習においてエージェントに与えられる様々なデータ (経験) を一旦メモリーに蓄積し，学習を行う際はメモリーに蓄積されたデータからランダムにサンプリングして利用するというものである．DQN では，この Experience Replay に従ってデータをいくつかランダムサンプリングしてミニバッチを作成して学習する．

2.3.2 Q Network と Target Network

DQN では，教師信号 $R(s,a) + \gamma \max_{a'} Q_{\theta_{i-1}}(s',a')$ を計算するための Target Network と，$Q_{\theta_i}(s,a)$ を計算する Q Network の 2 つのニューラルネットワークを用いる．Q Network のパラメータは定期的に Target Network にコピーされ，次の更新まで固定される．これにより，教師信号と予測する Q 関数の相関を低くすることで，学習の安定を図っている．

2.4 Asynchronous Advantage Actor-Critic (A3C)

Asynchronous Advantage Actor-Critic (A3C) [3] とは，2016 年に発表されたニューラルネットワークを用いた強化学習を並列に行う手法である．A3C のアルゴリズムは，Asynchronous，Advantage，Actor-Critic の 3 つの考え方を組み合わせて設計されており，ここではその考え方について，DQN との比較も交えながら説明する．

2.4.1 Asynchronous

A3C における Asynchronous とは，非同期的にマルチエージェントによる分散学習を行う考え方である．マルチスレッドで複数の学習環境を用意し，各環境のエージェントがそれぞれ独立して経験を積み重ねる．各スレッドのエージェントは，経験を十分に溜めると，自分のスレッドが持つネットワークを用いてネットワークを更新するための勾配を計算する．この勾配の値を用いて，共有ネットワークと呼ばれる全スレッドで共有するネットワークの更新が行われる．勾配を計算したスレッドは，更新された共有ネットワークの重みを自分のスレッドにコピーして，次の経験を積み重ねる．この操作の繰り返しを各スレッドが非同期的に行う．

A3C には DQN とは異なり，時系列的に連続した経験を学習に用いるオンライン学習ができるという利点がある．これは，マルチスレッドで経験を集めることにより，各スレッドにおいては時系列的に連続した経験でも，全体的には Experience Replay でランダムサンプリングしたような状態に近づけられるからである．

2.4.2 Advantage

DQN では，行動価値関数 $Q(s,a)$ の更新を，式 3 のように 1 ステップ先の報酬を用いて行っている．

A3C における Advantage とは，次状態の価値の推定値 $V(s_{t+1})$ を更新するのに，2 ステップ以上先の報酬まで用いて行う考え方である．n ステップ先までの報酬を用いたときに，A3C の推定値の更新に使われる advantage と呼ばれる量は，次の式のように定義される．

$$\begin{aligned} nstep &= \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}) \\ advantage &= nstep - V(s_t) \end{aligned} \tag{7}$$

2.4.3 Actor-Critic

DQN では，状態の入力 s に対して行動 a の価値の推定値 $Q(s,a)$ を出力しこれを学習していた．

A3C における Actor-Critic とは，状態の入力 s に対して，各行動を起こす確率 $\pi(s,a)$ (Actor) と，その状態の価値の推定値 $V(s)$ (Critic) をそれぞれ独立に推定しながら学習を行う考え方である．

2.5 Intrinsic Curiosity Module (ICM)

Intrinsic Curiosity Module (ICM) [2] とは，強化学習の際に用いる報酬をエージェント内部で自動生成する手法である．この手法は，人間の好奇心からアイデアを取っており，結果がどのようなものになるか予測がつかないような行動をするほど高い報酬が生成されるものである．

ICM は主に 2 つのニューラルネットワークのモデルで構成される．

一つは，逆モデル (inverse dynamics model) である．エージェントは環境から 2 つの離散的な時間軸上で隣り合う状態である s_t, s_{t+1} を観測する．次に，これらの状態をニューラルネットワークの畳み込み層などを利用して，特徴ベクトル $\phi(s_t), \phi(s_{t+1})$ へ変換する．状態 s_t から s_{t+1} には，行動 a_t によって遷移するが，この行動 a_t を 2 つの状態の入力から予想するのがこの逆モデルの役割である．このモデルは次のようにかける．

$$\hat{a}_t = g\left(s_t, s_{t+1}; \theta_I\right) \tag{8}$$

ただし，g はニューラルネットワークの学習機能 (learning function) であり，θ_I はニューラルネットワークのパラメータ，$\hat{a}_t$ が予想した行動の確率分布である．このニューラルネットワークは，次のような最適化を行うように学習する．

$$\min_{\theta_I} \mathrm{L}_I\left(\hat{a}_t, a_t\right) \tag{9}$$

L_I は，実際の行動と予想した行動の確率分布との不一致度を示す損失関数である．

もう一つは，順モデル (forward dynamics model) であ

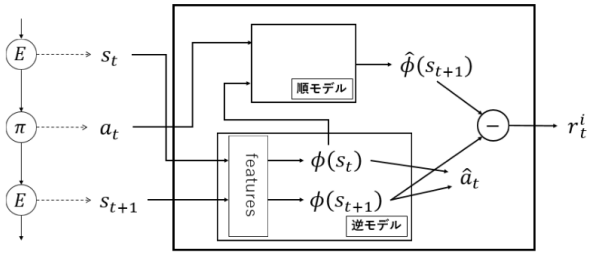

図 2 ICM の構造

る．これは，エージェントが観測した状態 $\phi(s_t)$ と，その時点で選択した行動 a_t から，次状態 $\phi(s_{t+1})$ を予測するモデルである．順モデルの計算は，2 つの状態が逆モデルの中で特徴ベクトルに変換されたものを用いて行われる．このニューラルネットワークのモデルは次のようにかける．

$$\hat{\phi}(s_{t+1}) = f\left(\phi(s_t), a_t; \theta_F\right) \tag{10}$$

逆モデルと同様に，f はニューラルネットワークの学習機能 (learning function) であり，θ_F はニューラルネットワークのパラメータ，$\hat{\phi}(s_{t+1})$ が予測された次状態の特徴ベクトルである．このニューラルネットワークは，次のような最適化を行うように学習する．

$$\min_{\theta_F} \mathrm{L}_F\left(\phi(s_t), \hat{\phi}(s_{t+1})\right) = \min_{\theta_F} \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|^2 \tag{11}$$

この時，この順モデルは内部報酬の信号として，次式のように計算される r_t^i を発する．

$$r_t^i = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|^2 \tag{12}$$

ただし，$\eta > 0$ である．これは，次状態が予測できないほどエージェントが受け取る内部報酬の量が多くなることを意味している．

ICM では，これら 2 つのモデルを同時に学習させる．次式のように，L_I と L_F のそれぞれに重みを付けた和を目的関数とし，この和を最小化させるように学習を行う．

$$\min_{\theta_I, \theta_F}[(1-\beta)L_I + \beta L_F] \tag{13}$$

図 2 に，ICM の構造の簡略図を示す．内部報酬を生成する上でもっとも重要なのは順モデルの学習である．しかし，順モデルが扱うのは次状態を予測するような計算コストの高いタスクなので，順モデル単体で学習させるのは難しい．そこで，逆モデルの学習を順モデルの学習に組み込むことで，取った行動を予測するという比較的容易なタスクの学習を通して状態の特徴変換法 ($s_t \to \phi(s_t)$) の学習速度を向上させることができ，これによって順モデルの学習効率も向上させることができる．

2.6 Hybrid Reward Architecture (HRA)

Hybrid Reward Architecture (HRA) [5] とは，2017 年に発表された，環境の報酬関数をいくつかの報酬関数に分割して，分割されたそれぞれの報酬関数に対してそれぞれ独立したモデルの学習を行う手法である．

$$R_{env}(s, a, s') = \sum_{k=1}^{n} R_k(s, a, s') \tag{14}$$

$$Q_{HRA}(s, a; \theta) = \sum_{k=1}^{n} Q_k(s, a; \theta) \tag{15}$$

報酬関数を n 個に分割する HRA のモデルを数式で表すと式 14, 15 のようになる．分割された報酬関数 $R_k(s, a, s')$ を用いて，それぞれ独立に $Q_k(s, a; \theta)$ を計算する．ただし，Q-Network の低レイヤーの部分では n 個のネットワークは重みを共有してもよく，その重みを式中では θ で表現している．

3. 提案手法

本研究の目的は，コンピュータ RPG の一種であるローグライクゲームを対象とした強化学習を行い，ゲームを自動攻略できるような AI を作ることである．本稿では，簡単なローグライクゲームの環境を用意し，このゲームを攻略する AI を学習するために，強化学習のエージェントには A3C [3] のアルゴリズムを用いて実験を行う．ゲーム環境の報酬としては，内部報酬は ICM [2] によって自動生成し，複数の外部報酬が存在する場合は HRA [5] によって報酬関数を分割して，学習の効率化を目指す．

4. 実験

4.1 用意したゲーム環境

図 3，4 に，今回実装したローグライクゲームのもっとも簡単な環境を示す．図 3 は，ランダムに生成されるダンジョンの例であり，黒いマスが壁，白いマスが移動可能なエリア，緑色のマスがそのダンジョンのゴール (階段)，赤いマスが操作するエージェントを意味している．エージェントは，1 回の行動として上下左右のうち壁がない方向に 1 マス進むことができる．図 4 には，実際にゲームプレイするときにダンジョンがどのように展開されているのかを示している．ゲーム開始時にダンジョンの全体図を把握できるわけではなく，ダンジョンを探索するほど，見たことのあるマスがマップ上に展開されていく．本実験では，全体サイズ 20×20 のマップ上にダンジョンをランダムに生成した．

4.2 学習に用いたモデル

4.2.1 A3C

本実験のために実装した A3C は CNN2 層と全結合層 2 層からなる．表 4.2.1 にモデルの設定を示す．最適化手法

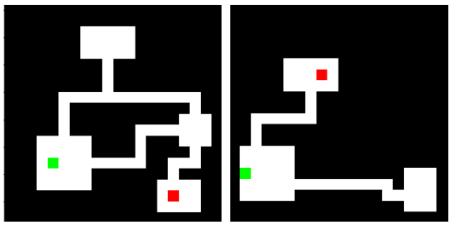

図 3　ランダムに生成されるダンジョン

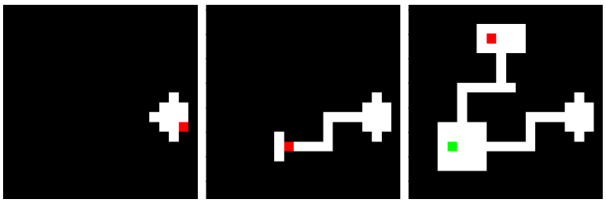

図 4　ダンジョンの展開

表 1　A3C のモデルの設定

ネットワーク層	説明	値
入力層	入力の次元	20 × 20 × 3
CNN 第 1 層	フィルタ数	32
	フィルタサイズ	3 × 3
	ストライド	2 × 2
	活性化関数	ReLU
CNN 第 2 層	フィルタ数	32
	フィルタサイズ	3 × 3
	ストライド	1 × 1
	活性化関数	ReLU
全結合層	出力の次元	32
	活性化関数	ReLU
出力層	Actor のサイズ	4
	Critic のサイズ	1
その他パラメータ	Optimizer	Adam
	学習率	1e-4

には Adam [11] を用いた．本実験で用いた Adam のパラメータの値は，Adam の論文で提案されている値をそのまま用いている．CNN 層におけるゼロパディングは行っていない．

4.2.2　ICM

本実験のために実装した ICM のモデルを図 5 に示す．逆モデルの 2 層の CNN 層で，状態の入力を特徴ベクトルに落とし込み，その特徴ベクトルを用いて，順モデルで次状態の特徴ベクトルを予測，逆モデルで行動を予測する．損失関数は，順モデルでは出力と正解の平均二乗誤差を，逆モデルでは出力として得られる行動の確率分布に対して正解の行動とのクロスエントロピーを取ったものを用いている．本実験で用いた詳細なモデルの設定を表 4.2.2 に示す．順モデルの出力次元 (1568) は，この設定における特徴ベクトルの次元数である．最適化手法には Adam [11] を用いた．重要な点として，CNN の各層の出力に対して Batch-Normalization [12] を適応することで，特徴ベクトルを正規化している [4].

表 2　ICM のモデルの設定

ネットワーク層	説明	値
入力層	状態の次元	20 × 20 × 2
	行動の次元	4
CNN 第 1 層	フィルタ数	32
	フィルタサイズ	3 × 3
	ストライド	2 × 2
	活性化関数	ReLU
CNN 第 2 層	フィルタ数	32
	フィルタサイズ	3 × 3
	ストライド	1 × 1
	活性化関数	ReLU
逆モデルの全結合層	出力の次元	32
	活性化関数	ReLU
逆モデルの出力層	出力の次元	4
	活性化関数	Softmax
順モデルの全結合層	出力の次元	32
	活性化関数	ReLU
順モデルの出力層	出力の次元	1568
	活性化関数	なし
その他パラメータ	Optimizer	Adam
	学習率	1e-4

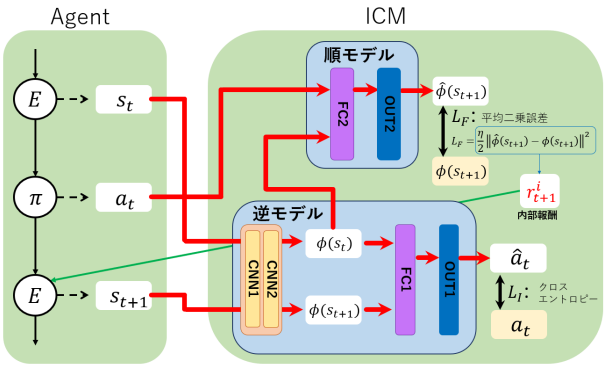

図 5　実装した ICM のモデル

4.3　実験: 階段へ向かうタスク

4.3.1　ゲーム設定

- まず，実際に学習を行う d 個のダンジョンを生成する．プレイヤーの初期位置と階段の位置は，それぞれのダンジョンについて固定である．
- プレイヤーは，次の (1)〜(7) の動作を繰り返す．
  (1) d 個のダンジョンのうち 1 つのダンジョンをランダムに選択する．
  (2) 選んだダンジョンのマップ情報 (図 4 参照) は初期化されている．
  (3) プレイヤーは 1 ステップにつき 1 回，上下左右の 4 方向のうち，どの方向に進むかを決定する．
  (4) 選んだ方向に壁が無かった場合は，プレイヤーはその方向に 1 マス進める．
  (5) プレイヤーは 500 ステップが経過するまで，ダン

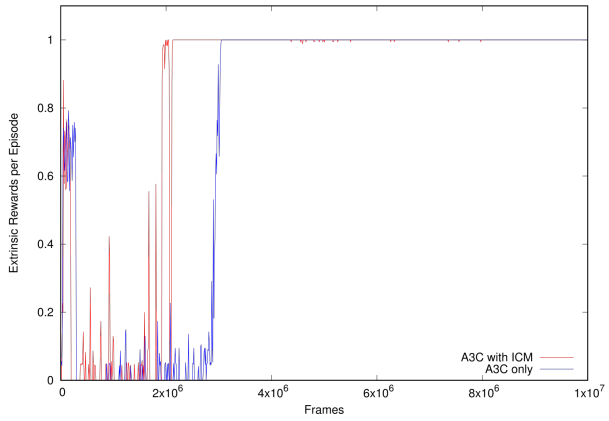

図 6 1 エピソードあたりに獲得する外部報酬の推移 ($d = 1$)

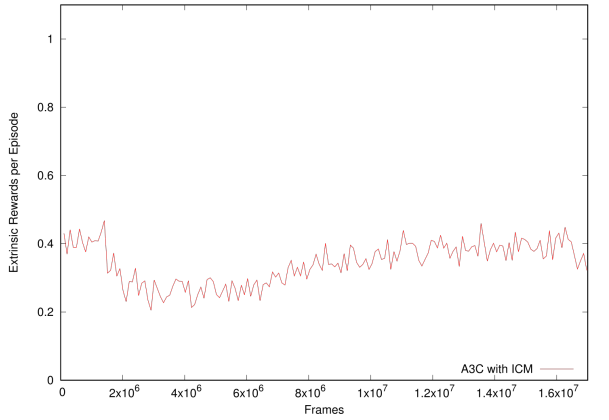

図 7 1 エピソードあたりに獲得する外部報酬の推移 ($d = 50$)

ジョン内を動くことができる．

(6) プレイヤーが途中で階段のマスに到達するか，500 ステップ経過するまでを 1 エピソードと定義する．

(7) エピソードが終了したら，プレイヤーは次のダンジョンに移動する．

このゲームのルールに従って，$d = 1$ の場合と $d = 50$ の場合について学習を行った．エージェントは，階段に到達したときに環境から一定値の報酬を得る．外部報酬とは別に，ICM が生成する内部報酬も利用して学習を行う．

4.3.2 実験結果

図 6 と図 7 に，$d = 1$ の場合と $d = 50$ の場合における 1 エピソードあたりに得られる外部報酬の値の平均値の推移を示す．階段に到達したときの報酬を 1 としているので，このグラフは実質的には階段に到達したエピソードの割合の推移を示している．横軸はフレーム数で，A3C の全てのスレッドのエージェントが経験したステップの総数である．

4.3.3 考察

図 6 には，1 個のダンジョンで繰り返し学習を行ったときの結果が，A3C で外部報酬のみを用いたものと，外部報酬に加えて ICM の生成する内部報酬を用いたものとでそれぞれ示されている．この図を見ると，外部報酬のみの場合よりも，ICM の内部報酬を用いた場合のほうが，1×10^6 ステップ早く学習が完了していることが分かり，ICM の生成する内部報酬の効果を実感することができる．

図 7 には，50 個ものダンジョンで，A3C で外部報酬に加えて ICM の生成する内部報酬を用いた学習を行った結果を示されている．この図を見ると，縦軸の値が 2×10^6 フレーム付近で減少したものの，それ以降はゆるやかに増加していく様子が観察できる．ここで図 6 を見てみると，1 個のダンジョンで行った学習においても，学習の序盤で縦軸の値が大きく減少している．

この原因としては，筆者のミスにより A3C の学習に ε-greedy 法を用いてしまったことが挙げられる．A3C は本来，A3C の生成する Actor(各行動を取る確率) に従って行動を選択しないと理想的に振る舞わないものであり，ε の確率でランダムに行動を選択してしまった影響が学習に現れてしまったと考えられる．Actor に従う行動選択を行う条件で学習をやり直すことにしたが，時間の問題で，本稿ではその結果を紹介できなかった．

5. おわりに

本稿では，ローグライクゲームを対象として，強化学習のアルゴリズムである A3C に，内部報酬を生成する ICM を用いた学習を行ったが，A3C を用いて実験を行う際に ε-greedy 法を誤って用いてしまい，実験をやり直す必要が生じてしまった．目下の課題は，新しく実験結果を得て，その結果を検証・考察することである．

本稿の実験で用いたローグライクゲームは「毎回異なるダンジョンを探索する」「ダンジョンを進むほどその形状が明らかになっていく」という 2 つの重要な要素を持つように意図したものであったが，敵やアイテムの存在，体力や空腹などの概念，複数階層での連続的なプレイングなどの基本的な要素が欠けており，ゲームとしては簡単すぎるものとなっているという問題もある．今後の研究では，ゲームの要素を更に追加した環境を用意し，今度は HRA によって報酬関数を分割するなどして，より発展的な学習方法を検討したい．

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, pp. 529–533 (2015).

[2] Precup, D. and Teh, Y. W.(eds.): *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, Vol. 70, PMLR (2017).

[3] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D. and Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning, *CoRR*, Vol. abs/1602.01783 (online), available from ⟨http://arxiv.org/abs/1602.01783⟩ (2016).

[4] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T. and Efros, A. A.: Large-Scale Study of Curiosity-Driven Learning, *arXiv:1808.04355* (2018).

[5] Van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T. and Tsang, J.: Hybrid Reward Architecture for Reinforcement Learning, *Advances in Neural Information Processing Systems 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Curran Associates, Inc., pp. 5392–5402 (online), available from ⟨http://papers.nips.cc/paper/7123-hybrid-reward-architecture-for-reinforcement-learning.pdf⟩ (2017).

[6] 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，ゲームプログラミングワークショップ 2017 論文集，Vol. 2017, pp. 19–25 (2017).

[7] Kaelbling, L. P., Littman, M. L. and Moore, A. W.: Reinforcement learning: A survey, *Journal of artificial intelligence research*, Vol. 4, pp. 237–285 (1996).

[8] Andrew G.Barto, R. S.: 強化学習，森北出版株式会社 (2000 年 12 月 27 日).

[9] Watkins, C. J. and Dayan, P.: Q-learning, *Machine learning*, Vol. 8, No. 3-4, pp. 279–292 (1992).

[10] Lin, L.-J.: Reinforcement Learning for Robots Using Neural Networks, PhD Thesis, Pittsburgh, PA, USA (1992).

[11] Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

[12] Ioffe, S. and Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *CoRR*, Vol. abs/1502.03167 (online), available from ⟨http://arxiv.org/abs/1502.03167⟩ (2015).

