補助学習を加えた階層化方策勾配法による強化学習の学習効率評価

吉田 雅也[1,a)]　相川 勝[2,b)]　井上 健太郎[3,c)]　山森 一人[3,d)]

概要：近年、様々な強化学習の手法が開発されており、その発展は目覚ましい。その中で、UNsupervised REinforcement and Auxiliary Learning (UNREAL) と Proximal Policy Optimization (PPO) に着目し、UNREAL 内で用いられている Asynchronous Advantage Actor-Critic (A3C) を PPO で置換した階層化方策勾配法を提案し、3D モデルの制御問題を用いて提案手法を評価する。

キーワード：強化学習、階層化方策勾配法、補助学習

Hierarchical policy gradient method with auxiliary learning

Masaya Yoshida[1,a)]　Masaru Aikawa[2,b)]　Kentaro Inoue[3,c)]　Kunihito Yamamori[3,d)]

Abstract: In recent years, various reinforcement learning algorithms have been developed and showed remarkable progress. Among them, we focus on UNsupervised REinforcement and Auxiliary Learning (UNREAL) and Proximal Policy Optimization (PPO). In this paper, we propose a hierarchical policy gradient method in which the part of Asynchronous Advantage Actor-Critic (A3C) in UNREAL is replaced by PPO. We evaluate the proposed method by controlling 3D models.

Keywords: Reinforcement Learning, Hierarchical Policy Gradient Method, Auxiliary Learning

1. はじめに

強化学習（Reinforcement Learning : RL）[1–3] とは、試行錯誤的に行動の選択を繰り返すことで環境に適応する、教師なし機械学習の 1 つである。RL は、環境の状態を知覚し、過去の経験から選択すべき行動を決定する学習エージェント（agent）の学習手段として用いられる。学習エージェントは、現在の状態と過去の経験を照らし合わせることで最善と思われる行動を選択する。学習エージェントが選択した行動は環境を遷移させ、遷移した環境は学習モデルに対して報酬（reward）を与える。学習エージェントは、行動の結果得られた報酬をもとに方策を評価することで学習を進めていく。RL の目的は、将来の報酬も踏まえた上で、最も多くの報酬を得る行動の選択方針を学習することである。RL は、ロボットの制御規則の獲得 [4,5] や、ゲーム AI のアルゴリズム [6–8] として研究されている。

近年、実世界の問題を機械学習や強化学習で解決しようという試みが活発になっている。例えば、Google DeepMind が強化学習で開発した ‘AlphaGo Zero’ [9] が、碁のプロ棋士にも勝利した ‘AlphaGo’ [10] をも圧倒したことが話題になった。他にも Preferred Networks Inc.(PNE) の分散強化学習を使った自動運転技術の獲得 [11] や、人工知能の自動設計を行う Google Brain の AutoML [12]、Google のデータセンターにおける強化学習を用いた空調の最適化 [13] 等が挙

[1] 宮崎大学 工学研究科
Graduate School of Engineering, University of Miyazaki, Japan
[2] 宮崎大学 工学部 教育研究支援技術センター
Technical Center, Faculty of Engineering, University of Miyazaki, Japan
[3] 宮崎大学 工学教育研究部
Faculty of Engineering, University of Miyazaki, Japan
[a)] masaya@taurus.cs.miyazaki-u.ac.jp
[b)] aikawa@cs.miyazaki-u.ac.jp
[c)] inoue@cs.miyazaki-u.ac.jp
[d)] yamamori@cs.miyazaki-u.ac.jp

© 2019 Information Processing Society of Japan

げられる。

このような試みの中で様々な強化学習の手法が開発されている [14]。その中でも本研究では、UNsupervised REinforcement and Auxiliary Learning (UNREAL) と Proximal Policy Optimization (PPO) に着目する。UNREAL は Jaderberg ら [15] が提案した階層化方策勾配法の一つであり、Asynchronous Advantage Actor-Critic(A3C) [16] と補助タスクを併用することで学習収束の高速化を図った手法である。PPO は Schulman ら [17] が提案した方策勾配法の一つで、方策の更新量を制限することで学習の安定化を図った手法である。UNREAL で用いられている A3C と PPO を比較した場合、PPO の方が学習収束の安定性という面において A3C よりも優れていることが、Stooke ら [18] において示されている。本論文では、補助タスクと PPO を併用した階層化方策勾配法を提案する。本研究の目的は、高い報酬を既存手法より早期に獲得し、強化学習の学習効率を改善することである。

本論文の構成は以下の通りである。第 2 章では強化学習について説明する。第 3 章では提案手法について具体的に解説する。第 4 章では実験により提案手法の評価を行う。第 5 章は本稿のまとめである。

2. 強化学習

2.1 強化学習の概要

強化学習 (RL：Reinforcement Learning) は、特定の状況下における行動選択の方策の学習を目的とした機械学習の 1 つである。RL では、行動選択の繰り返しによって望ましい結果が得られた場合に報酬が発生する。RL は、得られる報酬を最大化する行動を選択するための方策を学習することを目指す。

RL は Sutton ら [1] によって提案された手法である。RL は生物の大脳基底核に見られる、ドーパミンを報酬とした意思決定の学習の仕組みをもとにしている。RL の一般的な手法として Q 学習 [19] や SARSA [20] があり、特に Q 学習は学習結果が必ず収束することが保証されているためよく用いられる。また、学習対象とする問題が複雑な場合や大規模な場合に対応するために、マルチエージェントを用いた強化学習 [21] や、複数の学習環境を用意し並列的に学習を進める群強化学習 [22] が提案されている。近年では、深層学習と RL を組み合わせた深層強化学習が注目を集めている。深層強化学習には、DQN（Deep Q-Network）[23] を始め、GPU（Graphic Processing Unit）を用いずに現実的な時間での深層強化学習を実現した A3C（Asynchronous Advantage Actor-Critic）[16] や、並列化による DQN の高速を図った GORILA（General Reinforcement Learning Architecture）[24] がある。

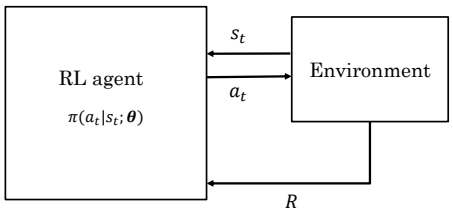

図 1 基本的な RL のフレームワーク.

2.2 強化学習の枠組み

基本的な強化学習の枠組みを図 1 に示し、以下に各構成要素について説明する。

- RL エージェント
  強化学習において、行動を選択し報酬を受け取り、方策の更新を行う学習の主体を強化学習エージェント（RL エージェント）と呼ぶ。RL エージェントは現在の環境 s_t 下で多くの報酬が得られる行動 a_t を、方策 $\pi(a_t|s_t)$ に従って選択する。RL エージェントは事前知識を持たず、学習初期はランダムに行動を決定する。ある程度学習が進むと、それ以前の経験をもとに得られる報酬を予測して、行動を選択できるようになる。ただし、行動選択の結果、すぐに報酬を得られるとは限らない（遅延報酬：delayed reward）ため、RL エージェントは行動価値（action-value）をもとに行動の選択を行う。行動価値は、ある行動 a を選択した場合に、将来的に得られる報酬期待値である。
- 環境（environment）
  環境は、RL エージェントが仮想的に行動を選択するための空間である。環境は RL エージェントの選択した行動によって自身の状態を変化させる。その際に行動の結果を評価し、環境は RL エージェントに報酬 R を与える。
- 状態（state）
  状態 s_t は、ある時刻 t において環境が現在どうなっているのかを表す変数であり、他の状態 $s_{t'}$ とは異なる必要がある。
- 行動（action）
  行動は RL エージェントが環境に働きかけるための手段である。ある時刻 t における行動 a_t は環境に作用し、環境を次の状態 s_{t+1} に変化させる。

2.3 RL の一般的な学習手順

基本的な強化学習の流れを図 2 に示し、以下で説明する。

(1) RL エージェントの学習パラメータを初期化する。

(2) 学習終了まで以下の処理を繰り返す。

2.1. 環境を初期状態にリセットする。

2.2. 1 episode 終了まで以下の処理を繰り返す。

(a) RL エージェントは現在の環境の状態 s_t を取得

© 2019 Information Processing Society of Japan

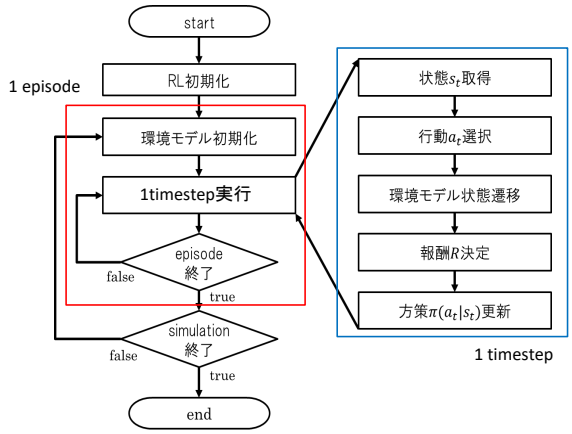

図 2　基本的な強化学習の流れ.

する。

(b) RL エージェントは、s_t における行動 a_t を、方策 $\pi(a_t|s_t)$ に基づいて 1 つ選択する。

(c) 環境は行動 a_t を実施し、状態を s_t から s_{t+1} へ遷移する。

(d) 環境は状態 s_{t+1} をもとに報酬 R を RL エージェントに与える。

(e) RL エージェントは報酬 R を用いて a_t を選択する確率を更新する。

状態取得・行動選択・方策更新の一連の処理を 1 timestep と呼ぶ。RL エージェントは、終了条件に達するまで上記 timestep の処理を繰り返し実施する。また、シミュレーションが 1 回終了するまでの一連の timestep の繰り返しを 1 episode と呼ぶ。強化学習では、学習終了条件に到達するまで episode を繰り返し実施し、より多くの報酬を得るための方策を学習する。

2.4 Actor-Critic

2.4.1 Actor-Critic の概要

Actor-Critic [25] は、強化学習のアルゴリズムの 1 つである。Actor-Critic のフレームワークを図 3 に示す。Actor-Critic は、行動選択を目的とする Actor と、選択した行動の価値の予測を目的とする Critic の 2 つのモデルを用いる。

Actor は得られる報酬を最大化するように、方策 $\pi(a|s;\boldsymbol{\theta})$ の学習を行う。$\boldsymbol{\theta}$ は行動の選択確率を調節するパラメータである。Actor 自身は選択した行動 a_t を評価できないため、パラメータ $\boldsymbol{\theta}$ の更新には Critic から行動の選択価値 $Q(a_t|s_t;\boldsymbol{w})$ を受け取る必要がある。

Critic は Actor の選択した行動を評価し、行動の選択価値 $Q(a_t|s_t;\boldsymbol{w})$ を計算して Actor へ提供する。Critic 自身も事前に知識を持たないため、報酬 R をもとにパラメータ $\boldsymbol{w}$ の値を更新して信頼できる選択価値を学習する必要がある。

2.4.2 行動選択

Actor-Critic での行動選択は Actor が行う。Actor は状態

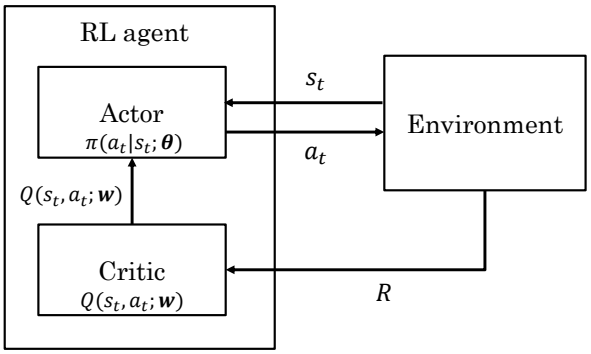

図 3　Actor-Critic のフレームワーク

s_t を環境から受け取り、行動 a_t を選択する。環境の状態 s_t 下で行動 a_t を選択する確率 $\pi(a_t|s_t;\boldsymbol{\theta})$ は式（1）に従う。

$$\pi(a_t|s_t;\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\phi}(s_t,a_t)^T\boldsymbol{\theta})}{\sum_{b\in A(s_t)}\exp(\boldsymbol{\phi}(s_t,b)^T\boldsymbol{\theta})}, \tag{1}$$

ここで、$\boldsymbol{\phi}(s_t,a_t)$ は環境の状態を表す特徴ベクトルである。また、$A(s_t)$ は状態 s_t において取り得る行動の集合を表す。

2.4.3 $\boldsymbol{\theta}$ と $\boldsymbol{w}$ の更新

Actor-Critic は、Actor のパラメータ $\boldsymbol{\theta}$ と Critic のパラメータ $\boldsymbol{w}$ の両方を並行して学習する。

まず、Critic は行動 a_t の選択価値 $Q(s_t,a_t;\boldsymbol{w})$ を計算し、Actor に提供する。選択価値 $Q(s_t,a_t;\boldsymbol{w})$ は式（2）により求める。

$$Q(s_t,a_t;\boldsymbol{w}) = \boldsymbol{\phi}(s_t,a_t)^T\boldsymbol{w}. \tag{2}$$

次に、Actor は選択価値 $Q(s_t,a_t;\boldsymbol{w_t})$ を用いた勾配法によりパラメータ $\boldsymbol{\theta}_t$ を更新する。パラメータ $\boldsymbol{\theta}_t$ の更新は式（3）に従う。

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\nabla_{\boldsymbol{\theta}}\log\pi(a_t|s_t;\boldsymbol{\theta}_t)Q(s_t,a_t;\boldsymbol{w_t}), \tag{3}$$

ここで、α は $0<\alpha\leq 1$ を満たす学習パラメータで、$\boldsymbol{\theta}$ の更新量を調節するのに用いられる。

加えて、Critic も環境から受け取った報酬 R をもとに、パラメータ $\boldsymbol{w}_t$ を式（4）に従って更新する。

$$\begin{aligned}\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \beta A_t\boldsymbol{\phi}(s_t,a_t), \\ A_t &= R + \gamma Q(s_{t+1},a_{t+1};\boldsymbol{w}_t) - Q(s_t,a_t;\boldsymbol{w}_t),\end{aligned} \tag{4}$$

ここで、β は $0<\beta\leq 1$ を満たす学習パラメータで、$\boldsymbol{w}$ の更新量 を調節するのに用いられる。γ は割引率と呼ばれる、$0\leq\gamma\leq 1$ を満たす学習パラメータであり、過去に得られた経験をどれだけ学習に反映させるかを調節する。また、A_t はアドバンテージ関数と呼び、期待した行動価値と実際に得られた行動価値の差分を表す。

2.5 PPO

2.5.1 PPO の概要

PPO(Proximal Policy Optimization) [17] は、2.4 節で説明

© 2019 Information Processing Society of Japan

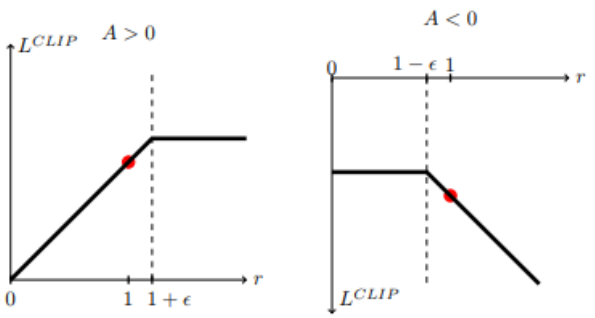

図 4　クリッピング操作の挙動 [17].

した Actor-Critic を応用した強化学習のアルゴリズムの一つである。PPO のフレームワークや行動選択法等は、Actor-Critic と同じである。PPO の特徴は、方策の更新量が大きい場合、変化量を一定にするクリッピングという操作を行う点にある。

2.5.2　PPO の方策の更新

クリッピングを行う際に、指標として更新前の方策 $\pi_{\theta_{old}}(a_t|s_t)$ と更新後の方策 $\pi_{\theta}(a_t|s_t)$ の比 $r_t(\theta)$ を用いる。$r_t(\theta)$ を式 (5) に示す。

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{5}$$

クリッピング操作の概念について図 4 に示す。図 4 の左はアドバンテージ関数 A_t の値が負の、図 4 の右は A_t の値が正の時のクリッピング操作である。

クリッピングの条件を式 (6) に示す。

$$clip(r_t(\theta), 1-\epsilon, 1+\epsilon) = \begin{cases} 1+\epsilon, & (A_t > 0 \wedge r_t(\theta) > 1+\epsilon), \\ 1-\epsilon, & (A_t < 0 \wedge r_t(\theta) < 1-\epsilon), \\ r_t(\theta), & (otherwise). \end{cases} \tag{6}$$

PPO で用いられる損失関数 $\mathcal{L}_{CLIP}$ を式 (7) に示す。まず、$r_t(\theta)$ にアドバンテージ関数 $\hat{A}_t$ を乗じた一項目と、式 (6) の条件のもと $r_t(\theta)$ をクリッピングした上でアドバンテージ関数 $\hat{A}_t$ を乗じた二項目を比較し、より小さい値を選択する。その後、期待値の平均 $\hat{\mathbb{E}}_t$ が最終的な $\mathcal{L}_{CLIP}$ となる。

$$\mathcal{L}_{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]. \tag{7}$$

2.6　UNREAL

2.6.1　UNREAL の概要

UNREAL(UNsupervised REinforcement and Auxiliary Learning) [15] は、2.4 節で説明した Actor-Critic を応用した A3C [16] をベースにした階層化方策勾配法 [26] の一つである。UNREAL の特徴として、A3C をメインの学習器、サブの学習器として後述する補助タスクを用いている点が挙げられる。

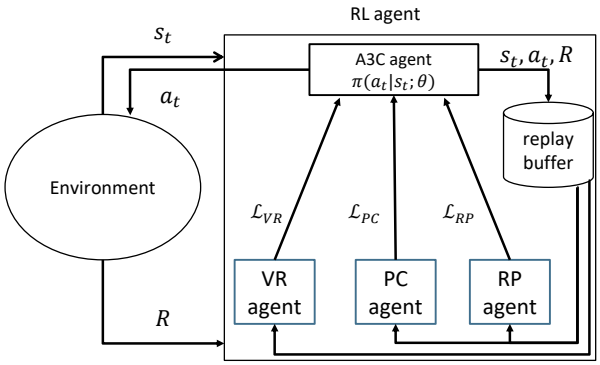

図 5　UNREAL のフレームワーク.

2.6.2　補助タスク

UNREAL で用いられている補助タスクは以下の 3 つである。

(1) Value Function Replay (VR) は、報酬の再計算が必要かどうか、過去の経験に基づいて決定する。

(2) Pixel Control (PC) は、環境が大きく変更された場合に、RL エージェントにより多くの $Q(s,a;\boldsymbol{w})$ を与える方策を学習する。これは、シーン切り替え等のイベントによって、環境が大きく変化することがあるためである。

(3) Reward Prediction (RP) は、現在の状態での選択可能な行動が、以前の経験に基づいて RL エージェントに正負または 0 の報酬を与えることが出来るかどうかを予測する。

2.6.3　UNREAL のフレームワーク

UNREAL のフレームワークを図 5 に示す。メインの学習器である A3C agent が行動を選択する。サブの学習器である各補助タスクエージェントは、リプレイバッファーから過去の経験を取り出して各々の処理を行い、各損失関数を A3C agent に渡して方策の更新を補助する。

2.6.4　UNREAL の方策の更新

UNREAL の方策更新に用いられる損失関数 $\mathcal{L}_{UNREAL}$ を式 (8) に示す。

$$\mathcal{L}_{UNREAL}(\theta) = \mathcal{L}_{A3C} + \lambda_{VR}\mathcal{L}_{VR} + \lambda_{PC}\sum_{c}\mathcal{L}_{Q}^{(c)} + \lambda_{RP}\mathcal{L}_{RP}, \tag{8}$$

ここで、$\lambda_{VR}, \lambda_{PC}$ 及び λ_{RP} は各補助タスクの損失関数の重みである。$\mathcal{L}_{A3C}, \mathcal{L}_{VR}$ 及び $\mathcal{L}_{Q}^{(c)}, \mathcal{L}_{RP}$ はそれぞれ損失関数を表し、式 (9) から式 (14) に示す。

A3C の損失関数 $\mathcal{L}_{A3C}$ は、式 (9) のように価値関数の損失関数 $\mathcal{L}_{VR}$ と方策関数の損失関数 $\mathcal{L}_{\pi}$ の合計で求められる。

$$\mathcal{L}_{A3C} = \mathcal{L}_{VR} + \mathcal{L}_{\pi}. \tag{9}$$

© 2019 Information Processing Society of Japan

方策関数の損失関数 $\mathcal{L}_\pi$ は、式 (10) に従って求められる。

$$\mathcal{L}_\pi = -\sum(-\log\pi(s_t)\times A(s_t) - \eta H(\pi(s_t))), \qquad (10)$$

ここで、η は学習率を表し、$0 \le \eta \le 1$ を満たす学習パラメータである。

式 (10) 中のエントロピー $H(\pi(s_t))$ は、式 (11) に従って求められる。

$$H(\pi(s_t)) = -\pi(s_t)\times\log\pi(s_t). \qquad (11)$$

VR の損失関数 $\mathcal{L}_{VR}$ を、式 (12) に示す。式 (12) は n-timestep 先の報酬 R や状態価値 V も考慮した式になるが、UNREAL においては即時報酬のみを考慮するので $\gamma = 0$ となる。よって、式 (12) 中の 2 項目は考慮しない。

$$\mathcal{L}_{VR} = \mathbb{E}[(R_{t:t+n} + \gamma^n V(s_{t+n+1}, \boldsymbol{\theta}) - V(s_t, \boldsymbol{\theta}))^2]. \qquad (12)$$

PC の損失関数 $\mathcal{L}_Q^{(c)}$ を式 (13) に示す。$\mathcal{L}_Q^{(c)}$ も $\mathcal{L}_{VR}$ と同様に $\gamma = 0$ となるので、式 (12) 中の二項目は考慮しない。

$$\mathcal{L}_Q^{(c)} = \mathbb{E}[(R_{t:t+n} + \gamma^n \max_{a'} Q^{(c)}(s', a', \boldsymbol{\theta}) - Q^{(c)}(s, a, \boldsymbol{\theta}))^2], \qquad (13)$$

ここで、c は入力の座標を表す。

RP の損失関数 $\mathcal{L}_{RP}$ を式 (14) に示す。

$$\mathcal{L}_{RP} = -\sum^{t}(pr_t \times \log r_t^{RP}), \qquad (14)$$

ここで、pr_t は獲得した報酬を式 (15) のもと処理した報酬を表す。r_t^{RP} は RP が予測した報酬を表す。

$$pr_t = \begin{cases} 1, & (R_t > 0), \\ -1, & (R_t < 0), \\ 0, & (R_t = 0). \end{cases} \qquad (15)$$

3. 提案手法

3.1 提案手法の概要

本研究では、PPO と補助タスクを組み合わせた階層化方策勾配法を提案する。提案手法でメインの学習器として、A3C ではなく PPO を選択した理由は、Stooke ら [18] の実験にて A3C と PPO を比較した際に PPO の方がアルゴリズムとして優れていることが示されているからである。図 6 に提案手法のフレームワークを示す。UNREAL のフレームワークを踏襲する形でフレームワークを形成しているが、メインの学習器を A3C agent から PPO agent に置き替えている。

3.2 提案手法の方策の更新

提案手法で用いる損失関数 $\mathcal{L}_{PM}(\boldsymbol{\theta})$ を式 (16) に示す。式 (8) 中の損失関数 $\mathcal{L}_{UNREAL}(\boldsymbol{\theta})$ の A3C の損失関数 $\mathcal{L}_{A3C}$ 部分を PPO の損失関数 $\mathcal{L}_{PPO}$ に置き替えた形となる。また、$\mathcal{L}_{PPO}$ は式 (7) の $\mathcal{L}_{CLIP}(\boldsymbol{\theta})$ と同一である。

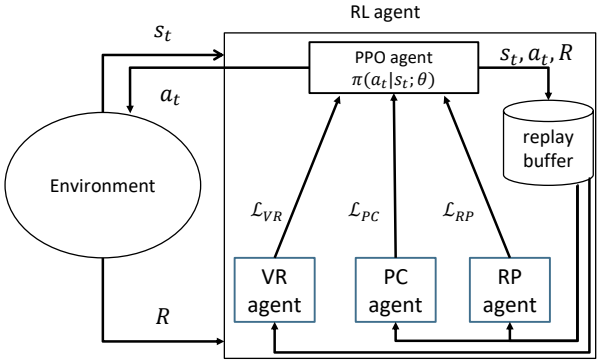

図 6 提案手法の概要.

$$\mathcal{L}_{PM}(\boldsymbol{\theta}) = \mathcal{L}_{PPO} + \lambda_{VR}\mathcal{L}_{VR} + \lambda_{PC}\sum_c \mathcal{L}_Q^{(c)} + \lambda_{RP}\mathcal{L}_{RP}, \qquad (16)$$

ここで、$\lambda_{VR}, \lambda_{PC}$ 及び λ_{RP} は各補助タスクの損失関数の重みである。

3.3 提案手法の学習の流れ

提案手法の学習の流れを図 7 に示す。学習の基本的な流れは一般的な RL と同じであり、状態取得、選択、報酬付与、方策更新の流れを繰り返すが、提案手法では一定 timestep ごとに学習器の更新を行うミニバッチ学習を採用し、一定 timestep 経過後に方策の更新を行い、RL エージェントの学習を進める。提案手法の 1timestep の流れを次に示す。

(1) RL エージェントは現在の環境の状態 s_t を取得する。

(2) PPO エージェントは、s_t における行動 a_t を、方策 $\pi(a_t|s_t)$ を用いて 1 つ選択する。

(3) 環境は、行動 a_t を実施し、状態を s_t から s_{t+1} へ遷移する。

(4) 環境は、報酬 R を RL エージェントに与える。

(5) 一定 timestep 経過後に以下の処理を行う

(a) PPO エージェントは、報酬 R を用いて方策を更新する。

(b) 各補助タスクは、リプレイバッファーから獲得した経験をもとに方策を更新する。

4. 実験と評価

4.1 問題設定

実験には、OpenAI が強化学習用のベンチマークとして提供している Gym [27] を用いる。その中でも物理エンジン MuJoCo [28] の 3D モデルを扱う問題 10 問を対象とする。実験時のシミュレーション環境のパラメータを表 1 に示す。

© 2019 Information Processing Society of Japan

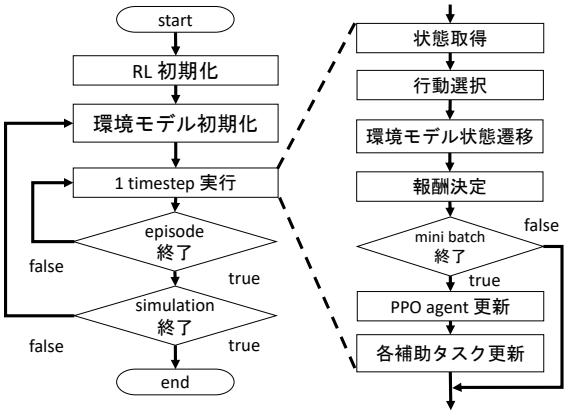

図 7　提案手法の学習の流れ.

表 1　シュミレーション環境のパラメータ

パラメータ名	値
最大 timestep 数	1,000,000
試行回数	10

表 2　実験時の学習パラメータ

パラメータ名	値
α	3×10^{-4}
β	3×10^{-4}
γ	0.99
ϵ	0.2
λ_{VR}	1
λ_{PC}	0.01
λ_{RP}	1

最大 timestep 数は、Schulman ら [17] が 100 万 timestep で実験を行っているため、それに合わせた。また、10 試行回数分の実験結果を集計し、その平均値を RL エージェントの学習結果として扱う。

実験時の RL エージェントの学習パラメータを表 2 に示す。ここで、学習パラメータの値は PPO で用いられていた数値 [17] を採用している。補助タスクのパラメータについては UNREAL で用いられていた数値 [15] とする。　RL エージェントのパラメータ $\boldsymbol{\theta}, \boldsymbol{w}$ はそれぞれ初期値を 0 とする。

4.2　実験結果

実験結果を図 8、図 9 に示す。縦軸が報酬量、横軸が timestep 数を表す。提案手法が最終的に獲得した報酬と UNREAL が最終的に獲得した報酬を比較した際の各問題の増加率をまとめて図 10 に示す。報酬の増加率の計算式を式 (17) に示す。

$$Increase\ \ rate = \frac{PM\ \ reward - UNREAL\ \ reward}{UNREAL\ \ reward} \times 100. \qquad (17)$$

図 8 から、HalfCheetah-v2 では 0 から 50 万 timestep にかけて提案手法と UNREAL の差が大きくなっている。最

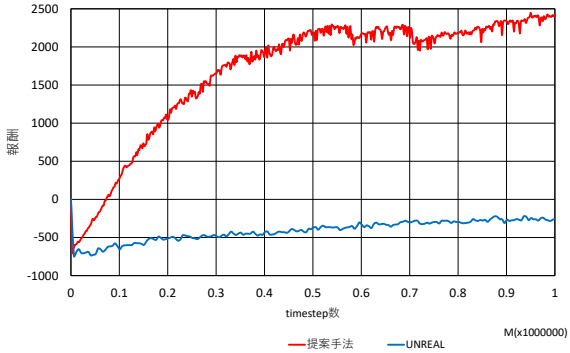

図 8　Halfcheetah-v2 での報酬の推移

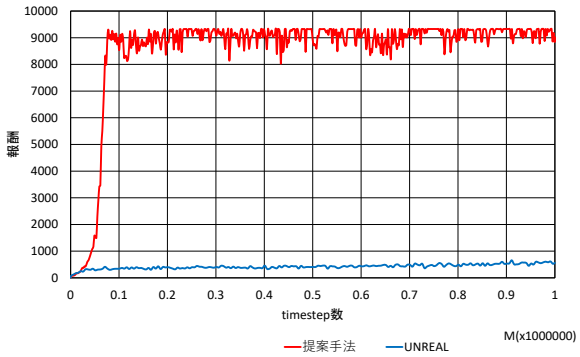

図 9　InvertedDoublePendulum-v2 での報酬の推移

終的には提案手法が UNREAL よりも約 2,700、より多くの報酬を獲得できていることが分かる。これは率にして、1,118% の増加である。

図 9 から、InvertedDoublePendulum-v2 では 0 から 10 万 timestep にかけて提案手法と UNREAL の差が大きくなっている。最終的には提案手法が UNREAL よりも約 8,500、より多くの報酬を獲得できていることが分かる。これは率にして、1,512% の増加である。

図 10 から、全ての実験において、提案手法の方が既存手法よりも多くの報酬を獲得することが出来ていることが分かる。具体的には、10 個の問題を平均して報酬を 412% 増加させることに成功した。この結果から、3D モデルの制御において提案手法は、強化学習の学習効率改善に成功したといえる。

5.　おわりに

2018 年の DeepMind 社による AlphaGo Zero の発表により、深層強化学習が行動選択の学習法として注目されるようになっている。また、自動車の制御技術の獲得や人工知能の自動設計といった、実社会における問題を扱うシステムの学習方法としても強化学習は用いられている。

強化学習の手法は続々と開発されており、その中でも Schulman らが提案した PPO が優れていることが Stooke らにより示されている。

本研究では、強化学習の学習効率を改善するために、PPO

© 2019 Information Processing Society of Japan

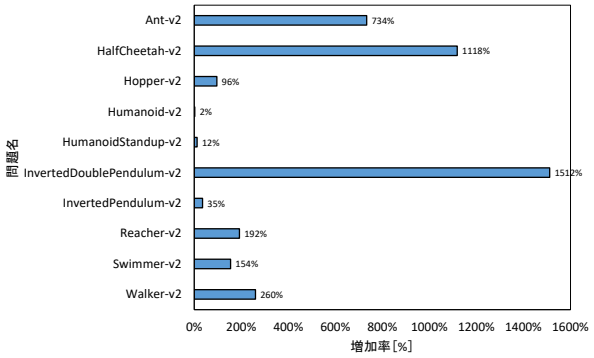

図 10 提案手法と UNREAL を比較した際の最終報酬の増加率.

に補助タスクを加えた階層化方策勾配法を提案した。既存手法である UNREAL と提案手法とで、3D モデル制御における報酬の多寡の比較を行った。

実験により、10 個の問題全てにおいて既存手法よりも多くの報酬を獲得出来ることを示した。また既存手法と比べて平均して 412% 以上の報酬を提案手法が獲得出来ることを示した。

今後の課題として、3D モデルの制御の問題でのみ実験・評価を行っているので、他のベンチマークでの実験・評価を行うことが挙げられる。

参考文献

[1] Sutton, R. S. and Barto, A. G.: *Reinforcement learning: An introduction*, MIT press ((1998)).
[2] 牧野貴樹，澁谷長史，白川真一， 他：これからの強化学習，森北出版 ((2016)).
[3] Szpesari, C.: 速習 強化学習 -基礎理論とアルゴリズム，共立出版 ((2017)).
[4] Crites, R. H. and Barto, A. G.: Elevator Group Control Using Multiple Reinforcement Learning Agents, *Mach. Learn.*, Vol. 33, No. 2-3, pp. 235–262 (online), DOI: 10.1023/A:1007518724497 (1998).
[5] Abbeel, P., Coates, A. and Ng, A. Y.: Autonomous helicopter aerobatics through apprenticeship learning, *The International Journal of Robotics Research* (2010).
[6] Tesauro, G.: TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play, *Neural Computation*, Vol. 6, No. 2, pp. 215–219 (online), DOI: 10.1162/neco.1994.6.2.215 (1994).
[7] Mnih, V., Kavukcuoglu, K. and Riedmiller, M. A.: Playing Atari with Deep Reinforcement Learning, *Computer Research Repository (CoRR)*, Vol. abs/1312.5602 (online), available from ⟨http://arxiv.org/abs/1312.5602⟩ (2013).
[8] : OpenAI Five, https://blog.openai.com/openai-five/ (2019).
[9] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G. v. d., Graepel, T. and Hassabis, D.: Mastering the game of Go without human knowledge, *Nature*, Vol. 550, No. 7676, p. 354 (online), DOI: 10.1038/nature24270 (2017).
[10] Silver, D., Huang, A. and Hassabis, D.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, pp. 484–503 (online), available from ⟨http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html⟩ (2016).
[11] : 分散深層強化学習でロボット制御，https://research.preferred.jp/2015/06/distributed-deep-reinforcement-learning/.
[12] : AutoML — Google Research Blog, https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html (2019).
[13] : グーグルがデータセンターの空調を AI で節約、アルファ碁技術応用, http://ascii.jp/elem/000/001/729/1729344/s/ (2019).
[14] Arulkumaran, K., Deisenroth, M. P., Brundage, M. and Bharath, A. A.: A Brief Survey of Deep Reinforcement Learning, *CoRR*, Vol. abs/1708.05866 (online), available from ⟨http://arxiv.org/abs/1708.05866⟩ (2017).
[15] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D. and Kavukcuoglu, K.: Reinforcement Learning with Unsupervised Auxiliary Tasks, *Computing Research Repository(CoRR)*, Vol. abs/1611.05397 (online), available from ⟨http://arxiv.org/abs/1611.05397⟩ (2016).
[16] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning, Vol. 48, pp. 1928–1937 (online), available from ⟨http://proceedings.mlr.press/v48/mniha16.html⟩ (2016).
[17] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal Policy Optimization Algorithms, *Computing Research Repository(CoRR)*, Vol. abs/1707.06347 (online), available from ⟨http://arxiv.org/abs/1707.06347⟩ (2017).
[18] Stooke, A. and Abbeel, P.: Accelerated Methods for Deep Reinforcement Learning, *Computing Research Repository(CoRR)*, Vol. abs/1803.02811 (online), available from ⟨http://arxiv.org/abs/1803.02811⟩ (2018).
[19] Watkins, C. J. and Dayan, P.: Q-learning, *Machine learning*, Vol. 8, No. 3-4, pp. 279–292 (1992).
[20] A. Rummery, G. and Niranjan, M.: On-Line Q-Learning Using Connectionist Systems, *Technical Report CUED/F-INFENG/TR 166* (1994).
[21] 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（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110002808079/⟩ (1998).
[22] 黒江康明， 飯間， 他：群強化学習法-エージェント間の情報交換に基づく方法-，計測自動制御学会論文集，Vol. 42, No. 11, pp. 1244–1251 (2006).
[23] Mnih, V., Kavukcuoglu, K. and Silver, D.: Human-level control through deep reinforcement learning, *Nature*, Vol. 518, pp. 529–533 (online), DOI: 10.1038/nature14236 (2015).
[24] Nair, A.，Srinivasan, P.，Blackwell, S.，et. al. : Massively Parallel Methods for Deep Reinforcement Learning (2015).
[25] Konda, V. R. and Tsitsiklis, J. N.: Actor-Critic Algorithms, *Advances in Neural Information Processing Systems 12* (Solla, S. A., Leen, T. K. and Müller, K., eds.), MIT Press, pp. 1008–1014 (online), available from ⟨http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf⟩ (2000).
[26] Ghavamzadeh, M. and Mahadevan, S.: Hierarchical Policy Gradient Algorithms, *ICML* (2003).
[27] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym (2016).
[28] Todorov, E., Erez, T. and Tassa, Y.: MuJoCo: A physics engine for model-based control, pp. 5026–5033 (online), DOI: 10.1109/IROS.2012.6386109 (2012).

© 2019 Information Processing Society of Japan