リセット機能を活用したシミュレータにおける効率的な方策学習

橋本 大世[1,a)]　鶴岡 慶雅[1]

概要：強化学習ではシミュレータを使った方策学習が一般的である．これは, シミュレータでは実環境よりも速くかつ安全にデータを収集できるためである．強化学習は試行錯誤を繰り返しながら学習するため一般的に大量のデータが必要であり, シミュレータを使っても学習に長時間かかることが多い．そのため強化学習の実応用に向けて, シミュレータにおける方策学習のサンプル効率を高めることが重要である．サンプル効率の向上を目的とする研究は数多く存在するが, シミュレータでの学習の特性を利用する研究は不十分であり改善の余地がある．そこで本研究では, シミュレータが備えるリセット機能を活用して方策学習を効率化する手法を検討する．具体的には, 累積報酬の高い軌跡を素早く見つけることで学習効率を高める．そのために, リセットする状態を選ぶ基準や, 不必要なデータ収集を避ける方法を提案する．実験では, CartPole という古典的なタスクと, Pong, Boxing というビデオゲームのタスクにおいて提案手法の有効性を定量的に検証した．加えて, 提案手法の動作に関する定性的な分析も行った．

Exploiting Reset Functions for Efficient Policy Learning on Simulators

Taisei Hashimoto[1,a)]　Yoshimasa Tsuruoka[1]

Abstract: In reinforcement learning, it is common to use a simulator for policy learning. This is because an agent can collect data faster and more safely on a simulator than in a real environment. Reinforcement learning generally requires a large amount of data because its learning process is a trial-and-error, and training often takes a long time to learn even using a simulator. Therefore, it is crucial to improve the sample efficiency of policy learning on a simulator for practical applications of reinforcement learning. Although there is a large body of work aiming to improve the sample efficiency, there are not many studies that exploit the characteristics of learning on a simulator. Therefore, in this study, we investigate an approach to improve the efficiency of policy learning by utilizing the reset function of a simulator. Specifically, we improve the learning efficiency by quickly finding trajectories with high cumulative rewards. For this purpose, we propose a criterion to select reset states and a method to avoid unnecessary data collection. In the experiments, we quantitatively verified the effectiveness of the proposed method in a classical task called CartPole and video game tasks of Pong and Boxing. In addition, we conducted a qualitative analysis of the behavior of the proposed method.

1. はじめに

強化学習は意思決定問題を扱う機械学習の一種である．これまでゲーム AI [1], ロボット制御 [2], 医療 [3], 金融 [4] など様々な分野での応用が研究されてきた．近年は深層学習を取り入れた深層強化学習が活発に研究されており, 非常に複雑な制御も学習できるようになりつつある．

強化学習ではシミュレータが広く利用されている．その理由として, 学習に多くのデータが必要, すなわちサンプル効率が低いことが挙げられる．また, 本研究とは直接の関連はないが, 実環境での学習は安全性に問題があることも大きな理由である．ロボット制御などの分野ではシミュ

[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)] hashi@logos.t.u-tokyo.ac.jp

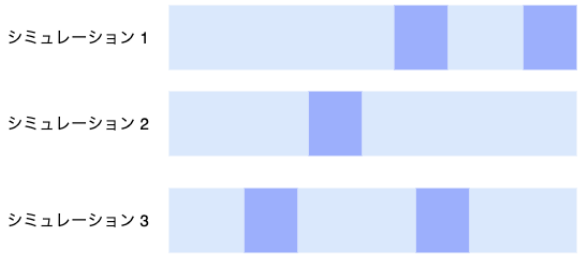
レータ上で学習した方策を実機で利用することが一般的である．一方ゲーム AI のように，目的の環境そのものがシミュレータである場合もある．

シミュレータ上のデータ収集は実環境で行うよりも一般的に高速であるが，それでもサンプル効率の低さゆえ学習に長時間かかることが多い．学習時間を短縮することはそれ自体も有益であるが，強化学習の実応用を考えると学習の高速化はさらに重要になりうる．なぜなら，方策学習を成功させるためには多くのハイパーパラメータを適切に設定する必要があり，学習と得られた方策の評価を繰り返すことが求められるからである．特に近年盛んに研究されている深層強化学習では，強化学習アルゴリズム関連のハイパーパラメータだけでなく，ニューラルネットワークの構造や最適化方法など，選択しなければならない設定が非常に多くなっている．学習を素早く行うことができると学習と評価のサイクルを多く繰り返すことができ，結果的により良い方策を得られることにつながる．以上のことから，シミュレータにおける方策学習を効率化することは重要である．

シミュレータの特性を活かして学習を効率化するアプローチとしては，多数のシミュレーションを同時に実行して大量のデータを素早く集めるという方法がある [5], [6]．この方法は必要な環境が準備できる場合は有効性が高いが，多数のシミュレータやそれを動作させる計算資源を利用できない場合もある．また，学習中に似たようなデータが大量に集められることも考えられ，学習の時間効率は高くてもエネルギー効率は低くなってしまうという懸念がある．

本研究では別のアプローチとして，シミュレータのリセット機能を活用して効率化を図る．ここでのリセット機能とは，学習中に訪れた状態を記録しておけば，リセット機能を使うことでいつでもその状態に戻ることができるというものである．通常の強化学習はこのような機能を前提としないが，本研究ではリセット機能を活用し，学習に有用なデータを重点的に集めることで方策学習を効率化する．上述した多数のシミュレーションを同時実行するアプローチと比較したイメージ図を図 1 に示す．図 1a では複数のシミュレーションを同時に実行することでデータの量を増やしている．一方，図 1b では単一のシミュレーション内で適切な状態にリセットすることで，方策学習という観点におけるデータの質を高めている．当然これらのアプローチを組み合わせてさらに効率的な学習を行うことも可能である．

リセット機能を活用する既存手法はいくつか存在する [7], [8], [9] が，ドメインに特化しているため汎用性に乏しかったり，リセットする状態の選び方がヒューリスティックで効果が小さかったりと改善の余地がある．本研究ではリセット機能を用いて累積報酬の高い軌跡を素早く見つけることで，サンプル効率の高い学習を行う．累積報酬という一般的な指標を用いることで，環境ごとに独自の指標をつくる必要がなく，汎用的な手法となりうる．



(a) 多数のシミュレーションを並列実行するアプローチ

(b) 本研究のアプローチ

図 1: シミュレータの特性を利用する 2 つのアプローチのイメージ図．各ボックスが 1 シミュレーションから得られるデータを表している．色の濃い部分はそのデータが方策学習に特に重要であることを示している．

2. 背景

2.1 強化学習

強化学習では，マルコフ決定過程 (Markov Decision Process, MDP) [10] で表される環境の中でエージェントが方策を学習する．MDP は $(\mathcal{S}, \mathcal{A}, \rho_0, p, r, \gamma)$ で定義される．$\mathcal{S}$ は状態空間，$\mathcal{A}$ は行動空間である．本稿では，状態空間は連続，行動空間は離散の場合を扱う．$\rho_0 : \mathcal{S} \to [0, \infty)$ は初期状態分布であり，各エピソードは ρ_0 からサンプルされた初期状態から開始する．p: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ は遷移確率であり，$p(s_{t+1}|s_t, a_t)$ は状態 $s_t \in \mathcal{S}$ において行動 $a_t \in \mathcal{A}$ を取った際に次状態が s_{t+1} になる確率を表す．r: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ は報酬関数であり，状態 s_t において行動 $a_t \in \mathcal{A}$ を取った際に得られる報酬を表す．p, r はエージェントには与えられず，必要であればデータから推定する必要がある．$\gamma \in [0, 1)$ は割引率である．強化学習の目的は，割引された累積報酬 $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)\right]$ を最大化する方策 $\pi(a_t|s_t)$ を学習することである．

本研究では，エージェントが観測する状態 $s_t \in \mathcal{S}$ に加えてシミュレータの内部状態 $h_t \in \mathcal{H}$ を考える．この内部状態 h_t をシミュレータにセットすることで，シミュレータは時刻 t と完全に同じ状態になる．すなわち h_t をセットしてから行動系列 $a_t, a_{t+1}, \cdots, a_{t+k-1}$ を取ると，状態系列 $s_{t+1}, s_{t+2}, \cdots, s_{t+k}$ が必ず得られる．[1]

[1] 環境が確率的な遷移を持つ場合は一般にこのような再現性を保証することはできない．しかし，シミュレータ内でのランダム性の実現には擬似乱数が用いられており，乱数生成器の状態も内部状態に含めて考えると，多くのシミュレータにおいてこの仮定は成立する．

2.2 Deep Q-Network

Deep Q-Network [11] は離散制御タスクにおける深層強化学習の代表的なアルゴリズムである．状態 s において行動 a を取った後に得られる累積報酬の期待値は Q 値と呼ばれ, (s,a) から Q 値への写像 $Q(s,a)$ は Q 関数と呼ばれる．Deep Q-Network では, Q 関数をニューラルネットワーク $Q_\theta(s,a)$ で推定する．ニューラルネットワークのパラメータ θ は, 次式で表される TD 誤差を繰り返し最小化することで学習される．

$$L_\theta(\mathcal{D}) = \mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[(y - Q_\theta(s,a))^2\right]$$
$$y = r + \gamma \max_{a'} Q_{\theta'}(s',a')$$

ここで, $\mathcal{D}$ は状態遷移を保存したリプレイバッファ, s' は状態 s の次の状態である．θ' は固定された古いパラメータであり, 一定のステップ間隔で θ と同期される．

学習された Q ネットワーク $Q_\theta(s,a)$ に対し, 方策は以下のように決定される．

$$\pi(a|s) = \begin{cases} 1 & (a = \arg\max_a Q_\theta(s,a)) \\ 0 & (\text{otherwise}) \end{cases}$$

3. 関連研究

3.1 リセット機能による多様な状態の収集

本研究と直接関連するのは, Tavakoli らの研究 [7] である．彼らはシミュレータのリセット機能を活用して多様な状態を収集する手法を提案した．この研究は on-policy 学習を前提としており, 過去のデータを直接は利用できない on-policy 学習において, 過去に訪れた状態にリセットするという形で過去のデータを活用することを目的としている．

リセットする状態を選択する方法はヒューリスティックに決められており, タスクの性質に応じて以下の 3 つが使用される．

一様サンプリング: これまでに得られた状態から一様にサンプリングする．常に初期状態からエピソードを開始するのではなく様々な状態から始めることで, より多様なデータが集められることを期待する．

状態価値推定の誤差に応じたサンプリング: 状態価値 $V^\pi(s)$ は状態 s から方策 π にしたがって行動したときに得られる累積報酬の期待値であり, Q 値と以下の関係式が成り立つ．

$$V^\pi(s) = \mathbb{E}_\pi\left[Q^\pi(s,a)\right]$$

状態価値関数も Q 関数と同様に TD 誤差の最小化により学習できる．損失関数の値が大きい状態は学習が進んでいないと考えられるため, そのような状態を重点的に収集する．

エピソード内で得られた報酬に応じたサンプリング: 報酬を得られる状態が少ない環境は報酬が疎な環境と呼ばれる．例えばゲームを完全にクリアしたときに報酬 1 が与えられ, それ以外は常に報酬 0 が与えられる場合は報酬が疎であると言える．このような場合, 報酬が得られなかったエピソードからは得られる学習信号が少なく, 成功した経験から優先的に学習することが有効である [12]．そのため, 累積報酬が高いエピソードを重点的にサンプリングし, さらにその中から 1 つの状態を一様にサンプリングする．

タスクの性質に応じてこれらの方法を使い分けることで, 実験では一定の効果が示された．しかし, on-policy 学習は off-policy アルゴリズムに比べて一般的にサンプル効率が低いことが知られている．実際に, 彼らの実験では Proximal Policy Optimisation (PPO) [13] という on-policy アルゴリズムが使用されているが, MuJoCo [14] という広く使用されているタスクでの学習効率は, 後に提案された Soft Actor-Critic (SAC) [15] などの off-policy アルゴリズムに比べて大きく劣っている．本研究はサンプル効率の向上を目的とするため, off-policy アルゴリズムを前提とする．

3.2 リセット機能によるカリキュラム生成

Salimans らは Montezuma's Revenge という探索の難しいゲームを, デモンストレーションを 1 つだけ用いて解く手法を提案した [9]．探索の難しい環境では, ランダムな行動による探索では一向にゴールまで到達できず, 全く学習が進まないということが往々にしてある．そこで彼らはエージェントをゴールに近い位置にセットし, そこからゴールまでの短い距離を学習させることで探索を容易にした．デモンストレーションを使えば, ゴールに近い位置をサンプルすることは容易である．ゴールに近い位置からタスクを解けるようになったら, エージェントをセットする位置を次第にゴールから遠ざけていく．このようにカリキュラムを生成することで, 難しい探索を回避して方策を学習することができる．

この手法にはデモンストレーションが必要であり, また探索の難しい環境に特化したものとなっている．本研究では, 外部情報を極力用いずにより汎用的なアルゴリズムの考案を目指す．

3.3 リセット機能を補助的に用いる研究

Ecoffet らが提案した Go-Explore [8] は探索が困難な問題を解くための新しい枠組みである．Go-Explore では 2 つのフェーズに分けて問題を解く．第 1 フェーズでは環境の遷移が決定的であるとして解法を見つける．第 2 フェーズでは見つけた解法をもとに, 遷移にランダム性がある場合にも有効な方策を学習する．どちらにも強化学習は必須ではなく, 行動決定問題への新しい枠組みと言える．

ただし, 彼らが提案した具体的なアルゴリズムは Montezuma's Revenge のようなビデオゲームを解くために特化したものであり, いくつかの点でドメイン知識が用いら

れている．特に重要なのは，第 1 フェーズで用いられるセル表現の抽出とリセットするセルの選択である．

Go-Explore では意味のある形で異なる状態を識別するためにセル表現を用いる．セル表現とは，ゲーム画面を白黒化，縮小などすることで粗い表現にしたものである．例えば画面のうち 1 ピクセルだけが異なっていたとしても実質的に同じものとして扱うべきであるが，状態の代わりにセル表現を使うことでそのような扱いが可能になる．Montezuma's Revenge の実験ではさらに，エージェントの位置や部屋番号などの外部情報によりセル表現を構築することも試されている．

上記のように得られたセルのアーカイブから，リセットするセルを選択し，その状態からランダムな行動を取って探索を行う．セルの選択のため，各セルにはヒューリスティックに決められたスコアがつけられ，スコアに応じて確率的にセルが選ばれる．このスコアはいくつかのサブスコアから構成され，各サブスコアはそのセルが何度リセットに選ばれたか，何度訪れたか，隣接するセルがいくつあるかなどといった情報から計算される．

セル表現の抽出やリセットするセルの選択は Montezuma's Revenge のようなゲームに特化したものであり，ユーザが決めなければならないハイパーパラメータも数多く存在する．そのためより汎用的な手法が求められる．

4. 提案手法

リセット機能を活用し，なるべく少ない試行回数で累積報酬の高い軌跡を発見する手法を考案する．基本的なアイディアは，これまでに見つけた最も累積報酬の高い軌跡の途中の状態にリセットし，そこから再度探索することで，さらに累積報酬の高い軌跡を見つけるというものである．エピソード途中までのデータを再利用することにより，必要なデータを少なくすることができる．

4.1 全体の流れ

初期状態分布 $\rho_0(s)$ からサンプルされた初期状態にリセットすることを通常リセット，提案手法により選ばれた状態にリセットすることを選択的リセットと呼ぶこととする．通常リセットは通常の学習で行うリセットのことである．選択的リセットでは以下で説明する方法でリセットする状態を選び，その状態からエピソードを開始する．エージェントは，通常リセットで集められるデータと選択的リセットで集められるデータの比が一定になるようデータを収集する．

4.2 リセットする状態の選択

リセットする状態には，その状態から再開することでより高い累積報酬が得られる見込みの高いものを選ぶ必要がある．そこで，Q 値を用いた次のスコアを考える．

$$StateScore(s) = \max_a Q_\theta(s,a) - G$$

ここで，s は軌跡に含まれる状態，G は s から先で得られた最大の割引累積報酬である．このスコアは状態 s から現在の方策に従って行動した場合に期待できる，割引累積報酬の改善幅を表している．ただし，状態 s における Q 値が過大に推定されている場合，このスコアも実際よりも高く見積もられてしまう．それを防ぐため，以下のように計算方法を改良する．

$$StateScore(s) = \max_a Q_\theta(s,a) - \max(G, \max_{a \in A(s)} Q_\theta(s,a)) \quad (1)$$

ここで，$A(s)$ はこれまでに s で取られた行動の集合である．追加した項により，Q 値が過大に推定されても不当にスコアが高くならず，悪影響を軽減できる．スコア計算の具体例を図 2 に示す．

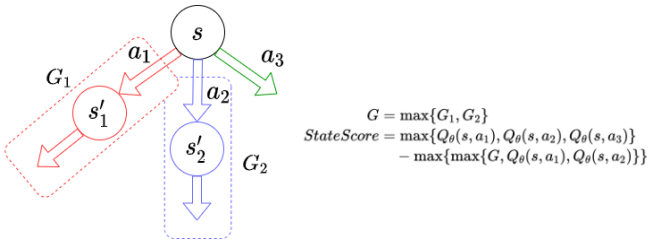


図 2: スコア計算の具体例．これまでに状態 s からは行動 a_1, a_2 を取ったことがあり，a_3 は取ったことがない．すなわち $A(s) = \{a_1, a_2\}$.

$StateScore$ が高い状態を選択してリセットすることで，累積報酬が高い軌跡を見つけることが期待できる．ただし，単純にスコアが最大の状態を選んだ結果，エピソードの序盤の状態が選ばれると，初期状態からエピソードを開始する通常の学習とあまり違いがなくなってしまう．そこで，エピソードの終端に近い状態を優先的に選び，積極的にデータの再利用を図る．具体的には，軌跡に含まれる全状態の $StateScore$ を計算し，スコアが全体の c パーセンタイル以上となる状態のうち，最もエピソードの終端に近い状態を選ぶ．こうすることで，大きな累積報酬の改善が期待でき，かつ多くのデータが再利用されるような状態をリセットに選ぶことができる．c はハイパーパラメータであり，$c = 100$ の場合は $StateScore$ が最大の状態を選ぶことになり，$c = 0$ の場合はエピソード終端の状態を選ぶことになる．

4.3 その他の工夫

累積報酬の更新が難しくなった場合，そのエピソードを中断して新しい状態にリセットすることもサンプル効率向上のために重要である．そのため 1 ステップごとに次式で

表される N ステップリターンの上限値を計算し, これが現時点での割引累積報酬の最大値 G を P 回連続で下回った場合にエピソードを中断する.

$$\sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n (\max_a Q_\theta(s_{t+n}, a) + Q_u)$$

ここで, t はリセットに選ばれた状態のタイムステップ, n はリセットから現時点までの経過ステップ, Q_u は Q 関数の不確かさである. Q_u の真の値を得ることはできないため何らかの指標で代替する必要があるが, 今回は簡単のため TD 誤差の指数移動平均 $\overline{\delta^2}$ を使い,

$$Q_u = w\sqrt{\overline{\delta^2}}$$

とする. この値は Q 値の予測誤差を大まかに測ったものになっており, w はハイパーパラメータである.

5. 実験

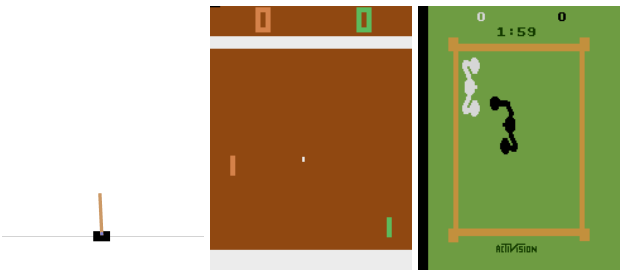
5.1 実験設定

OpenAI Gym に含まれる以下の 3 つの環境において, 提案手法を通常の学習と比較する.

CartPole ポールが取り付けられた台車を左右に動かし, ポールが倒れないようにする. 状態は台車の位置, 速度などを表す 4 次元の実数値で与えられる. 取りうる行動の数は 2.

Pong 卓球を模したビデオゲーム. パドルを上下に操作し, ボールを打ち返す. 状態は画像で与えられる. 取りうる行動の数は 6.

Boxing ボクシングを模したビデオゲーム. 左右の腕を動かし, 相手にパンチを当てる. 状態は画像で与えられる. 取りうる行動の数は 18.



(a) CartPole (b) Pong (c) Boxing

図 3: 各環境のサンプル画像

通常リセットで集められるデータと選択的リセットで集められるデータの比は 1:9 で固定し, 最適化していない. 強化学習アルゴリズムには Deep Q-Network を用いた. モデルの構成やその他のハイパーパラメータについては Appendix A.1, A.2 で述べる. すべての実験は 5 個のランダムシードで行った.

5.2 実験結果

結果を図 4 に示す.

CartPole では提案手法により大幅に学習効率が上昇している. また, ハイパーパラメータ c の影響がほとんどないこともわかる. $c = 100$ の場合は単純に $StateScore$ が最大の状態を選ぶことに相当するが, その場合も高い学習効率を達成している. これは, CartPole のタスクの性質によるものであると考えられる. CartPole はポールが倒れるまで常に報酬 1 が与えられ, 倒れるとエピソードが終了する. そのため $StateScore$ が大きくなるのはポールが倒れる直前, すなわちエピソードの終端付近になる可能性が高い.

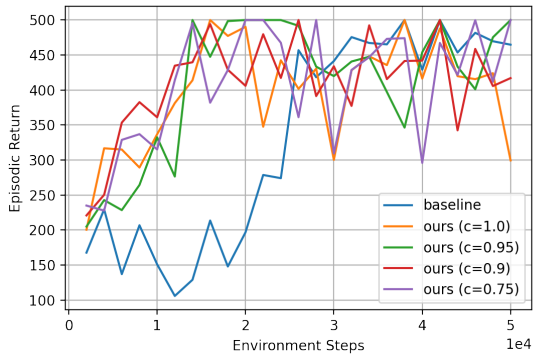
Pong においても提案手法による学習効率の上昇が見られる. 最もサンプル効率が良いのは $c = 95$ の場合である. $c = 100$ の場合はデータの再利用が十分でなく, また $c = 90, 75$ の場合は $StateScoer$ があまり高くない状態を選んでしまうために効率が低下すると考えられる. 実際に Pong でリセットに選択された状態のタイムステップの平均値を比較すると, 図 5 のようになる. c の値が大きいほど, タイムステップの小さい, すなわちエピソードの終端から遠い状態が選ばれていることがわかる.

一方, Boxing では提案手法による改善は見られなかった. 提案手法で得られた累積報酬の最高値は, 図 6 で示すように c の値によらずテスト時の累積報酬よりも大幅に高い. つまり累積報酬の高い軌跡は見つけられているが, モデルが汎化できていないということがわかる. この原因は定かではないが, Boxing は他のタスクに比べて入力が複雑であり (高い累積報酬を得るには自分と相手の位置関係, 腕の状態などを正確に把握する必要がある), 汎化しやすい表現が抽出できなかった可能性が考えられる. 他にも Boxing は今回扱ったタスクの中で行動の数が最も多く, 実験したステップ数では学習が難しかった可能性がある.

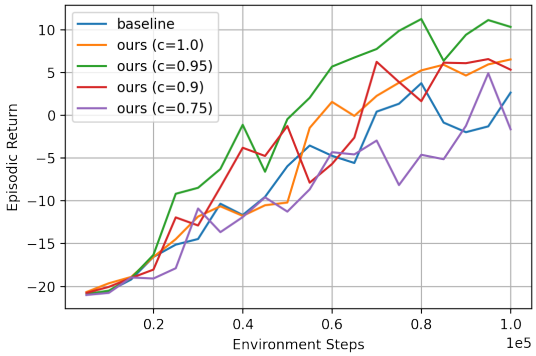
提案手法の動作に関して理解を深めるため, リセットに選択された状態の例を図 7 に示す. CartPole では, 上段に示すもとの軌跡ではすぐにポールが倒れてエピソードが終了しているが, 下段では操作が改善されて持ちこたえている. なお, 紙面の都合上 4 ステップ分しか表示していないが, 下段のエピソードはその先も続いている. Pong では, 上段では直後に失点しているが, 下段ではボールを打ち返して失点を防いでいる. Boxing では, 上段で得点できなかった所をリトライしているようにも見えるが, 実際には改善できていない.

6. おわりに

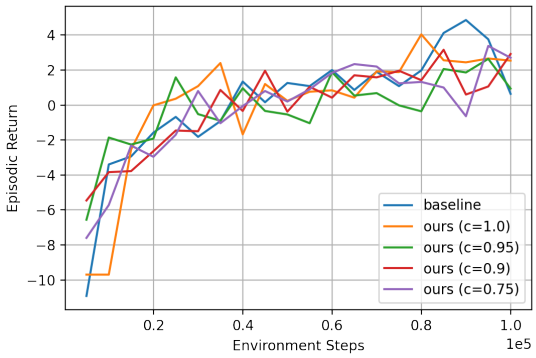
本研究ではシミュレータを用いた方策学習を効率化するため, シミュレータのリセット機能を活用する手法を考案し, その効果を検証するための実験を行った. その結果, 一部の環境ではサンプル効率を改善し, また意図した形でリ



(a) CartPole



(b) Pong



(c) Boxing

図 4: 各タスクの学習曲線．横軸は学習に使ったステップ数, 縦軸はエピソードあたりの累積報酬を表す．5 回の実験の平均値を示している．

セットが行われていることが確認できた．その一方で, 提案手法の効果がはっきりと現れない環境があることもわかった．

今後の課題は以下の 3 つである．

手法の改善　現状ではリセットする状態の選び方に恣意性が残っており, 最適でない可能性がある．よりシステマチックな方法を模索することで, さらにサンプル効率を高められる可能性がある．

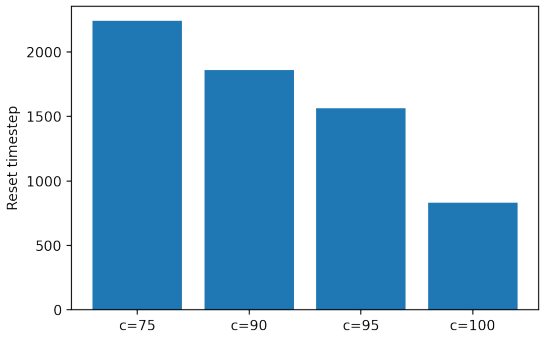


図 5: リセットに選択された状態のタイムステップの平均値．

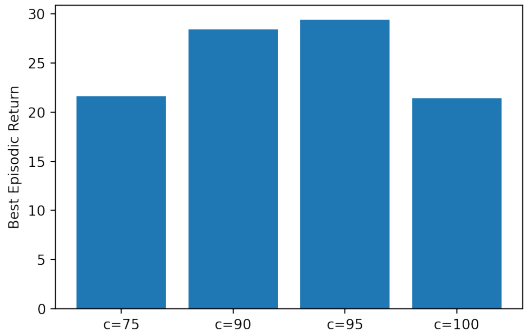


図 6: 各初期状態に対する累積報酬の最高値．

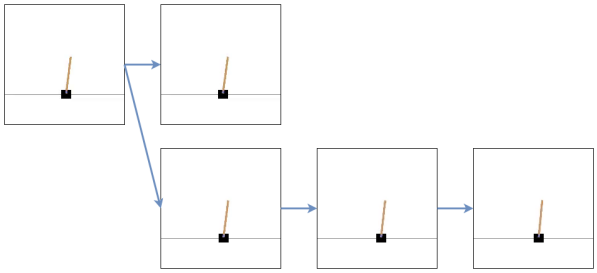
効果が見られなかった原因の特定　Boxing では提案手法の効果が見られなかったが, この原因を特定することで手法を改善したり, 手法がどのようなタスクに効果的かを理解したりすることができる．

実験の拡充　今回は 3 つの環境で実験を行ったが, より多くの環境で実験することで提案手法の有効性を検証したい．また, 提案手法は離散行動のタスクだけでなく連続行動のタスクにも適用できるため, ロボット制御などの連続制御タスクでも実験をしていきたい．

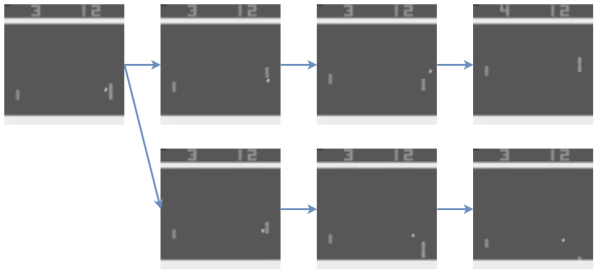
昨今は事前に集められたデータのみで学習するオフライン強化学習が盛んに研究されている [16], [17]．オフライン強化学習ではシミュレータが必須ではないこともあり, 全体的な流れとして学習のシミュレータに対する依存度を下げる方向に研究が進んでいる [18]．しかし学習のために十分な量と質のデータを事前に集めることは容易ではなく, 方策学習におけるシミュレータの利用は今後も続くと考えられる．そのため, シミュレータの特性を活用するという方向性の研究にも大きな意義があると考えている．

参考文献

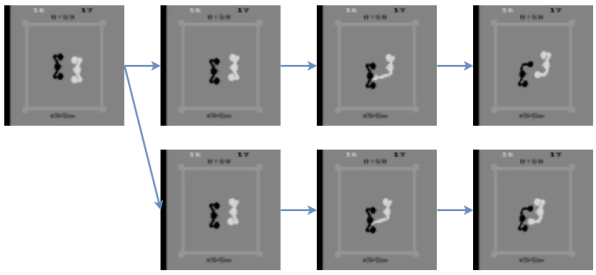
[1] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q.,



(a) CartPole



(b) Pong



(c) Boxing

図 7: リセットに選択された状態の例．上段が更新前の軌跡の一部, 下段が更新後の軌跡の一部を示す．ここではわかりやすさのため CartPole についても画像を表示しているが, 実際の状態は 6 次元の実数値であり, 画像ではない．

Hashme, S., Hesse, C. et al.: Dota 2 with Large Scale Deep Reinforcement Learning, *arXiv preprint arXiv:1912.06680* (2019).

[2] Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R. et al.: Solving Rubik's Cube with a Robot Hand, *arXiv preprint arXiv:1910.07113* (2019).

[3] Yu, C., Liu, J. and Nemati, S.: Reinforcement Learning in Healthcare: A Survey, *arXiv preprint arXiv:1908.08796* (2019).

[4] Fischer, T. G.: Reinforcement Learning in Financial Markets - a Survey, Technical report, FAU Discussion Papers in Economics (2018).

[5] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning, *International conference on machine learning* (2016).

[6] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S. and Kavukcuoglu, K.: IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, *International Conference on Machine Learning* (2018).

[7] Tavakoli, A., Levdik, V., Islam, R., Smith, C. M. and Kormushev, P.: Exploring Restart Distributions, *The Multi-disciplinary Conference on Reinforcement Learning and Decision Making* (2019).

[8] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O. and Clune, J.: Go-explore: a new approach for hard-exploration problems, *arXiv preprint arXiv:1901.10995* (2019).

[9] Salimans, T. and Chen, R.: Learning Montezuma's Revenge from a Single Demonstration, *NeurIPS 2018 Deep RL Workshop* (2018).

[10] Bellman, R.: A Markovian Decision Process, *Indiana University Mathematics Journal*, pp. 679–684 (1957).

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., 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* (2015).

[12] Oh, J., Guo, Y., Singh, S. and Lee, H.: Self-Imitation Learning, *International Conference on Machine Learning* (2018).

[13] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).

[14] Todorov, E., Erez, T. and Tassa, Y.: MuJoCo: A physics engine for model-based control, *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012).

[15] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, *International Conference on Machine Learning* (2018).

[16] Agarwal, R., Schuurmans, D. and Norouzi, M.: An optimistic perspective on offline reinforcement learning, *International Conference on Machine Learning* (2020).

[17] Fujimoto, S., Meger, D. and Precup, D.: Off-Policy Deep Reinforcement Learning without Exploration, *International Conference on Machine Learning* (2019).

[18] Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V. and Levine, S.: The Ingredients of Real World Robotic Reinforcement Learning, *International Conference on Learning Representations* (2020).

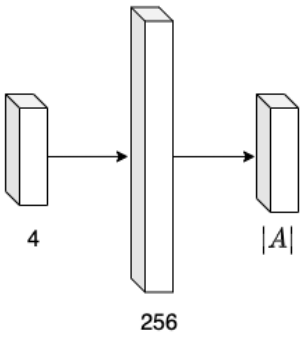
[19] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M. and Freitas, N.: Dueling Network Architectures for Deep Reinforcement Learning, *International Conference on Machine Learning* (2016).

[20] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *International Conference on Learning Representations* (2015).

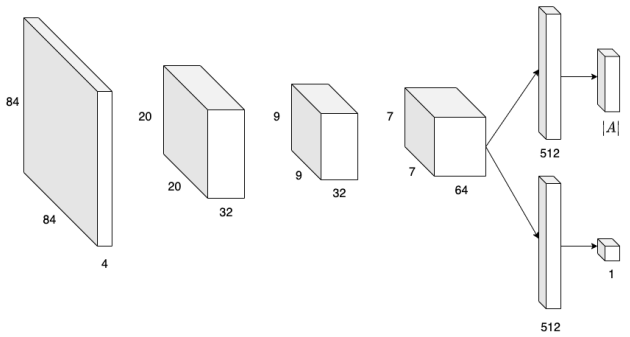
付　　録

A.1　ニューラルネットワークの構成

第 5 節で述べたとおり, 実験では入力が低次元のタスク (CartPole) と画像のタスク (Pong, Boxing) を扱った. それぞれにおける Q ネットワークの構成を図 A·1 に示す. 入力が画像のタスクでは Dueling Network [19] を用いている. 活性化関数には ReLU を使用した.



(a) Cartpole



(b) Pong, Boxing

図 A·1: Q ネットワークの構成図. 図中の $|A|$ は取りうる行動の数であり, CartPole では 2, Pong では 6, Boxing では 18 である.

A.2　その他のハイパーパラメータ

すべての環境に共通の設定を表 A·1 に, 環境固有の設定を表 A·2, A·3 に示す.

表 A·1: 共通のハイパーパラメータ

カテゴリ	種類	値
ニューラルネットワークの学習	オプティマイザ	Adam [20]
	学習率	1.0×10^{-3}
	ステップ当たりのパラメータ更新回数	1
強化学習	割引率 γ	0.99
	リプレイバッファのサイズ	制限なし
提案手法	通常リセットと選択的リセットのステップ比	1:9
	エピソード中断までの回数 P	10
	Q 値の不確かさのスケール w	2.0

表 A·2: CartPole 環境固有のハイパーパラメータ

カテゴリ	種類	値
ニューラルネットワークの学習	バッチサイズ	128
強化学習	N ステップリターン	1
	学習ステップ	50k
	アクションリピート	1
	報酬のクリッピング	なし

表 A·3: Pong, Boxing 環境固有のハイパーパラメータ

カテゴリ	種類	値
ニューラルネットワークの学習	バッチサイズ	32
強化学習	N ステップリターン	5
	学習ステップ	100k
	アクションリピート	4
	報酬のクリッピング	[-1, 1]
画像処理	フレームスタック	4
	画像サイズ	84 x 84