On/off-policy のハイブリッド深層強化学習とシミュレーション環境での制御問題への応用

On/off-policy Hybrid Deep Reinforcement Learning and Simulation in Control Tasks

知能機能システム専攻 博士前期課程 1 年次
201820844 王 伯 楠 (Wang Bonan)
(指導教員: 延原肇，古賀弘樹，澁谷長史)

Abstract — ニューラルネットワークを用いた深層強化学習は幅広い、かつ、複雑なタスクに対応でき、様々な分野で成果を出している。特にゲーム AI や制御などのタスクでは素晴らしい性能を示している。しかし従来手法では探索が進まないなどの問題がある。本研究は長期経験と短期経験の両方を活用した on/off-policy エージェントと訓練アルゴリズムを提案する。これによって、従来法の問題を解決し、性能の向上を図る。比較実験の結果、提案手法は従来手法に比べて良い性能を示している。

1 はじめに

近年、ニューラルネットワークを用いた深層強化学習は様々な分野で成果を出している [1]。特にゲーム AI や Robotics コントロール問題では素晴らしい性能を示している [2]。ニューラルネットワークは高い汎用性を持つ近似器として幅広い、かつ、複雑な問題に適用される。

強化学習アルゴリズムは主に on-policy と off-policy の二種類に分けられる。On-policy のアルゴリズムでは探索するエージェントと訓練するエージェントが同一である一方、off-policy のアルゴリズムではそれらは異なっている。On-policy のアルゴリズムは off-policy のアルゴリズムと比べて訓練が速く、安定性も高い。一方で、訓練の時には常に新しいデータが必要である。しかし、探索と訓練を同一のエージェントが行うため、訓練が進むとエージェントの動きが固定され、新しいデータが収集できなくなり、このことはデータから学習する知識の減少を招くため、学習が遅くなるあるいは進まない可能性がある。この問題はタスクが複雑なほど起きやすい。Off-policy のアルゴリズムは過去の経験から学習するため、この問題を緩和できる。しかし off-policy のアルゴリズムは on-policy のアルゴリズムと比べて安定性が悪い。しかも、どの種類のアルゴリズムでも、あらかじめ設計された報酬関数が必要となる。

本稿では、従来の on-policy と off-policy の深層強化学習の問題を解消するため、1) ハイブリッドなエージェントと 2) 訓練アルゴリズムを提案する。具体的には、1) ハイブリッドなエージェントとして、汎用性と効率的な学習のため、DDPG(Deep Deterministic Policy Gradient) アルゴリズム [1] で使われる Actor-Critic 型エージェントを提案するエージェントのベースとする。さらに、2) その訓練アルゴリズムとして、長期経験と短期経験の両方を活用するため、データを複数の短い時系列になるよう分割して、エージェントを学習させる。また、この時系列での訓練アルゴリズムに対応するために、ネットワークとして LSTM(Long Short-Term Memory) を用いる。提案手法では、目的関数に DDPG の価値関数と DPPO(Distribute Proximal Policy Optimization)[3] の価値関数の両方を用いる。

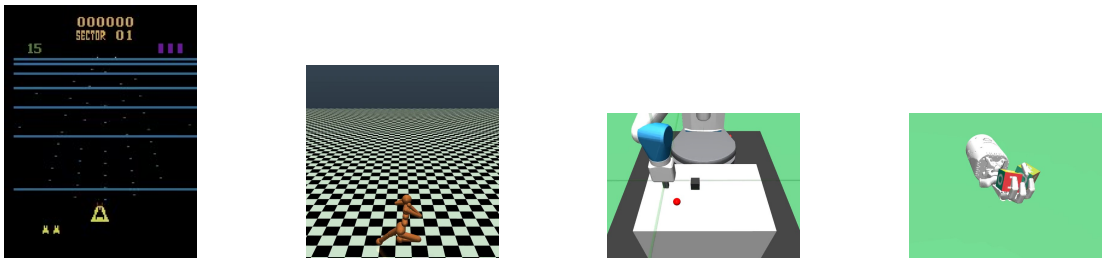
本稿は次のように構成される。第 2 章では関連する従来研究を紹介する。第 3 章では提案手法を説明する。第 4 章では評価実験をその結果を示す。最後に第 5 章で結論を述べる。

2 関連研究

この章では、主に関連研究の DDPG と DPPO アルゴリズムを説明する。それに先立って、いくつかの表記法を説明する。

- $t \in \{1, 2, \ldots, T\}$ — 時刻あるいはステップ数を示す。T は最後のステップとする。
- $s \in \mathbb{R}^m$ — 環境の状態の観測値ベクトルを示す。ただし、m は状態空間の次元数である。また、t ステップ目におけるs をs_t と表す。
- $a \in \mathbb{R}^n$ — エージェントの行動ベクトルを示す。ただし、n は行動空間の次元数である。また、t ステップ目におけるa をs_t と表す。
- $R(s, a) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ — 状態の観測値ベクトルs に対して行動a の報酬を表す報酬値関数を示す。また表記を簡便にするために、t 時刻の報酬値をr_t で示す。すなわち、

$$r_t = R(s_t, a_t), \tag{2.1}$$



(a) Atari (b) Mujoco (c) Robotics (d) Robotics

Figure 1.1 Environments Example

である。

- θ：ネットワークの重みなどのパラメーターを示す。

2.1 DDPG(Deep Deterministic Policy Gradient)

DDPG アルゴリズム [1] は典型的な off-policy の Actor-Critic アルゴリズムである。DDPG エージェントは行動を決めるネットワークπ と評価ネットワークQ で構成されている。π は状態の観測値s を入力とし、行動a あるいは行動の分布を出力する。Q ネットワークは状態s と対応する行動a を評価し、評価値q を出力する。

Q ネットワークの目的関数は Q-learning を参照する。訓練する時、まずQ を更新する。Q を訓練した後、それを教師とし、π を訓練する。π はq を最大化するようにθ^{π} を更新する。

DDPG は Replay Buffer から経験データをランダムでサンプリングして訓練する。よって、過去のすべての経験を使う。Replay Buffer を用いるため、DDPG アルゴリズムは自由に探索できるが、動作の変化が激しかったり、収束が遅かったりするなどの問題がある。

2.2 DPPO(Distribute Proximal Policy Optimization)

DPPO アルゴリズム [3] は典型的な on-policy のアルゴリズムである。DPPO エージェントは行動ネットワークπ と価値ネットワークV で構成されている。π ネットワークは状態の観測値s を入力とし、行動の確率分布N^n を出力する。V ネットワークは状態の観測値ベクトルs を入力とし、価値の予測値v を出力する。

DPPO アルゴリズムは最近の経験データしか使えない。よって、DPPO アルゴリズムは常に新しいデータが必要である。DPPO アルゴリズムは学習がはやく、かつ安定性も高い。しかし、DPPO は訓練が進むと探索が遅くなる。

3 提案手法

3.1 ネットワーク構造

従来法の問題を解消するため、本論文は短期経験と長期経験の両方を活用した on/off-policy のハイブリッドエージェントと強化学習アルゴリズムを提案する。提案手法では Actor-Critic 型エージェントを採用する。短期と長期経験の特徴に対応して、行動策略部分の Actor は LSTM ネットワークを採用する。LSTM ネットワークはK 個のセルで構成され、各 LSTM セルをπ_k とする。π は一連の状態の観測値$s_t, s_{t-1}, \ldots, s_{t-K+1}$、LSTM 初期細胞状態$c_{init}$ と初期出力h_{init} を入力とし、行動の確率分布を出力する。

$$\pi_t(s_t, s_{t-1}, \ldots, s_{t-K+1}, c_{init}, h_{init}) \sim N^n. \tag{3.1}$$

LSTM の各ゲートの活性化関数の前に Layer Normalization を用いる。各セルの出力に全結合層をつけて最後の出力にする。

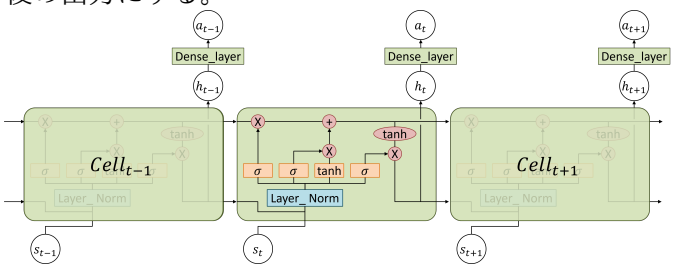


Figure 3.1 π ネットワーク構造

価値関数部分の Critic ネットワークは MLP を採用し、Q とする。DDPG の Critic ネットワークと同じくs とa を入力とし、価値の予測値を出力する。負数の報酬値に対応するため、活性化関数は Leaky Relu を用いる。

安定性のため、ネットワークは main ネットワークと target ネットワークで構成されている。main ネットワークは学習によって更新され、target ネットワークは main ネットワークのパラメーターを用いて更新される。

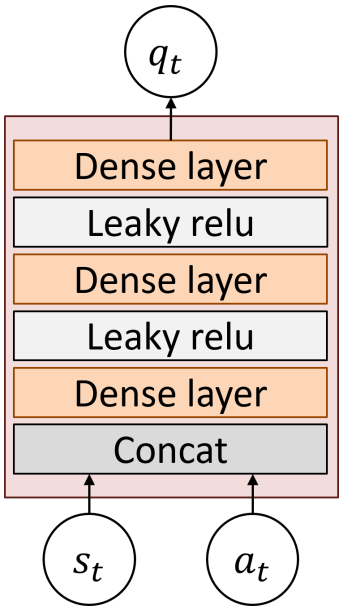


Figure 3.2　Q ネットワーク構造

3.2　目的関数

長期経験と短期経験の両方を活用するため、それぞれQ とπ の目的関数を設計した。短期経験を使う on-policy 的な目的関数は$L^{vf}(\theta^Q)$ と$L^{pg}(\theta^\pi)$ があり、長期経験を使う off-policy な目的関数は$L^Q(\theta^Q)$ と$L^\pi(\theta^\pi)$ がある。アルゴリズムはこの４つの目的関数を最適化する。

on-policy の価値目的関数$L^{vf}(\theta^Q)$：

$$L^{vf}(\theta^Q) = \mathbb{E}(Q(s_t, a_t) - Q_t^{'})^2 \tag{3.2}$$

where (3.3)

$$Q_t^{'} = \sum_{i=t}^{T}(\gamma^{i-t} * r_i) + Q(s_{T+1}, \pi(s_{T+1})). \tag{3.4}$$

on-policy の行動目的関数$L^{pg}(\theta^\pi)$：

$$L^{pg}(\theta^\pi) = \mathbb{E}(\frac{\pi(a_t \mid s_t)}{\pi^{'}(a_t \mid s_t)} * \hat{A}_t), \tag{3.5}$$

$$\hat{A}_t = \delta_t + \gamma\lambda\delta_{t+1} + ... + (\gamma\lambda)^{T-t}\delta_T, \tag{3.6}$$

$$\delta_t = r_t + \gamma * Q_{t+1} - Q_t. \tag{3.7}$$

off-policy の価値目的関数$L^Q(\theta^Q)$：

$$L^Q(\theta^Q) = \mathbb{E}(Q(s_t, a_t) - c_1 * Q_1^{'} - c_2 * Q_2^{'})^2, \tag{3.8}$$

$$Q_1^{'} = r_t + \gamma * Q(s_{t+1}, \pi(s_{t+1})), \tag{3.9}$$

$$Q_2^{'} = r_k + \gamma * r_{k+1} + ... + \gamma^{K-k} * r_K + \tag{3.10}$$

$$\gamma^{K-k+1} * Q(s_{K+1}, \pi(s_{K+1})). \tag{3.11}$$

K は LSTM ネットワークのセル数、k は LSTM のセル順番数、c_1 とc_2 は重み引数である。

off-policy の行動目的関数$L^\pi(\theta^\pi)$：

$$L^\pi(\theta^\pi) = \mathbb{E}(Q(s_t, \pi(s_t))). \tag{3.12}$$

$L^Q(\theta^Q)$ と$L^{vf}(\theta^Q)$ を最小化し、$L^\pi(\theta^\pi)$ と$L^{pg}(\theta^\pi)$ を最大化する。

3.3　訓練アルゴリズム

訓練する時、先ず複数のエージェントを使って探索し、データを収集する。各エージェントは環境の中で一定のステップ数T を実行し、これを一つの iteration とする。収集したデータ$(s_t, a_t, r_t, s_{t+1}, c_t, h_t)$ を Replay Buffer $\mathcal{M}$ に保存する。探索が効率よく進むため、各エージェントは探索の途中、小さい確率ϵ でランダムな行動を取る。ϵ はエージェントによって違う。安定のため、target ネットワークを用いて探索を進め、main ネットワークを更新する。

Algorithm 1 on/off-policy algorithm

initialize Replay Buffer $\mathcal{M}$;
initialize Network $\pi, \pi^{'}, Q, Q^{'}$;
for i **in** $Max\_Iterations$ **do**
  for w **in** $N\_Workers$ **do**
    reset Iteration Buffer $\mathcal{I}$;
    for t **in** T **do**
      $a_t, c_t, h_t \leftarrow \pi_{target}(s_t)$;
      $a_t \leftarrow Random\ Action\ with\ \epsilon$;
      $s_{t+1}, r_t \leftarrow Environment(a_t)$;
      store $(s_t, a_t, r_t, s_{t+1}, c_t, h_t)$ in $\mathcal{I}$;
    end for
    store $\mathcal{I}$ in $\mathcal{M}$;
    update θ^Q_{main} with $L^{vf}(\theta^Q)$;
    update θ^π_{main} with $L^{pg}(\theta^\pi)$;
    $\theta^Q_{target} \leftarrow (1-\tau) * \theta^Q_{target} + \tau * \theta^Q_{main}$;
    $\theta^\pi_{target} \leftarrow (1-\tau) * \theta^\pi_{target} + \tau * \theta^\pi_{main}$;
  end for
  sample random transition $Batch$;
  update θ^Q_{main} with $L^Q(\theta^Q)$;
  update θ^π_{main} with $L^\pi(\theta^\pi)$;
  $\theta^Q_{target} \leftarrow (1-\tau) * \theta^Q_{target} + \tau * \theta^Q_{main}$;
  $\theta^\pi_{target} \leftarrow (1-\tau) * \theta^\pi_{target} + \tau * \theta^\pi_{main}$;
end for

4　評価実験

4.1　実験条件

評価実験は深層強化学習においてベンチマーク的なタスク Humanoid タスクを用いる。Humanoid タスクの目的はシミュレーション環境の中の人間を前進することを学習させることである。エージェントは人間の各関節の動作を決める。

Table 4.1　実験条件

Deep Learning Library	Tensorflow 1.9.0
Simulation Library	openAI gym
Env	Humanoid-v2

訓練する時、Worker ことに on-policy の目的関数を用いてエージェントを更新する、五つの Worker ことに off-policy の目的関数を用いてエージェントを更新する。

Table 4.2　Hyper Parameters

T	1024
N_Worker	5
π learning rate	0.00004-0.000001
Q learning rate	0.00008-0.000002
batch_size	32
epoch	5
sample_size	640
τ	0.2

4.2　実験結果

本研究はベンチマークタスク Humanoid 環境を用いて、従来の DDPG、DPPO と提案手法に対して比較実験を行いました。実験の結果は図 (4.1) に示したように、提案手法は学習が速く、かつ高い得点を獲得している。

Humanoid タスクにおいて、提案エージェントは前進することを学習した。DPPO エージェントはある程度前進ができ、DDPG エージェントはできなかった。

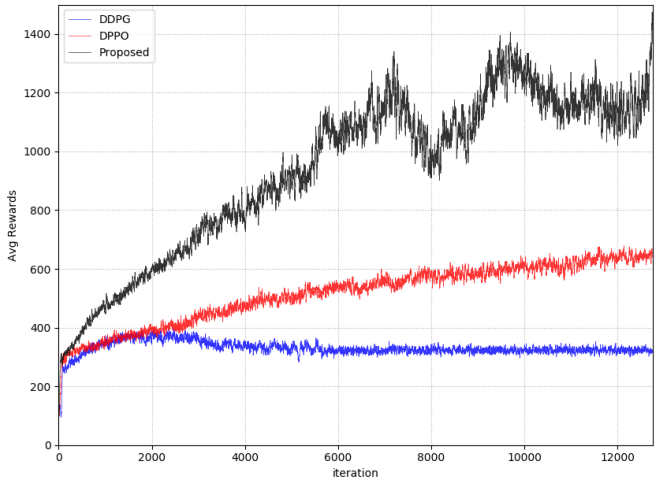


Figure 4.1　**Performance Comparison** of DDPG, DPPO and Proposed Method on benchmark task **Humanoid**

5　おわりに

本研究は従来の深層強化学習アルゴリズムの問題点を解決するため、短期経験と長期経験の両方を活用する on/off-policy のハイブリッド深層強化学習アルゴリズムを提案した。Humanoid タスクにおいて、提案手法は従来手法より優れた性能を示した。On-policy の部分では、安定の学習を実現していて、さらに off-policy の部分は探索を進ませることが原因と思われます。

Humanoid タスクのような、状態空間と行動空間は連続で、報酬値も丁寧に設計されているタスクに対して、提案手法は有効であると思われる。しかし現実の問題おいて、報酬値は二進値のタスクは多数存在している。エージェントの汎用性を向上させるため、これからは HER[4] 手法の投入と Robotics タスク [5] への実装を検討している。

参考文献

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2015. arXiv: 1509.02971. [Online]. Available: http://arxiv.org/abs/1509.02971.

[2] OpenAI, “Learning dexterous in-hand manipulation,” *CoRR*, vol. abs/1808.00177, 2018. arXiv: 1808.00177. [Online]. Available: http://arxiv.org/abs/1808.00177.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, “Hindsight experience replay,” *CoRR*, vol. abs/1707.01495, 2017. arXiv: 1707.01495. [Online]. Available: http://arxiv.org/abs/1707.01495.

[5] OpenAI. (2018). Ingredients for robotics research, [Online]. Available: http://blog.openai.com/ingredients-for-robotics-research/.