計測自動制御学会東北支部第337回研究集会(2022.6.27)
資料番号337-5

多脚歩行ロボット制御のための深層強化学習におけるパラメータ調整

Parameter Tuning in Deep Reinforcement Learning for Multilegged Robot Control

○大澤士竜*，釜谷博行*，工藤憲昌*，原元司**
Shiryu Osawa*, Hiroyuki Kamaya*, Norimasa Kudoh*, Motoshi Hara**
*八戸工業高等専門学校，**松江工業高等専門学校
*National Institute of Technology, Hachinohe College,
**National Institute of Technology, Matsue College

キーワード: 深層強化学習(Deep Reinforcement Learning)，多脚ロボット(Multilegged Robot)
連絡先: 〒039-1192 青森県八戸市田面木字上野平16-1 八戸工業高等専門学校　産業システム工学専攻
Tel.: 0178-27-7283，E-mail: kamaya-e@hachinohe-ct.ac.jp

1. はじめに

近年、地震等の災害発生時の救助活動や惑星探査などの分野で多脚ロボットが登場し、注目されている。多脚ロボットは支持脚が多いため安定性が高く不整地での活躍が期待されている[1]。新しい機構や移動方法の開発など、多脚ロボットの不整地での走破性の向上が研究されているが、歩行ロボットの歩行制御は運動学、動力学、状態遷移軌道などの難解な要因を考慮しなければならなく、開発には人の手がかかり、関節が増えるほど自由度が増え、歩行モーションの作成が困難となる。このため、強化学習による歩行動作の獲得が提案されている。

本研究では、ロボット制御に適している連続値を扱える深層強化学習の一つである、Normalized Advantage Function(NAF)に注目し、歩行動作の学習を行う。

2. 深層強化学習

2.1 強化学習

強化学習とは、学習を行う主体であるエージェントが環境との相互作用を通し、目標状態へ到達するための行動を獲得するための手法である。その際、環境等に関する先験的な知識は必要なく、各状態、および行動に対して評価を与えることにより、エージェントは学習を進めることが可能である。

2.2 Q-learning

Q-learning は、方策に依存しない学習手法である。この手法では、ある状態において、行動を選択し実行する。これによって得られた報酬から行動の価値 Q 値を更新することで、より価値の高い行動をとるよう学習を進めていく。Q-learning は状態、行動ともに離散値のみを扱う。

2.3 Normalized Advantage Function (NAF)

NAF は Q-learning のアプローチに基づいた手法であり、行動、状態ともに連続値を扱う。NAF は DDPG(Deep deterministic policy Gradient)よりも多くのタスクにおいて、収束までにかかるエピソード数が少ないことが指摘されている[2]。

NAF の処理の流れを Algorithm1 に示す。Algorithm1 において、Q は Q 関数全体を、Q′はそのターゲットネットワークを、μ は方策関数を、R は過去ログデータを蓄積するリプレイ

バッファを指す。また、Iはネットワーク更新頻度であり、本研究では$I=1$とする。γは割引率、τは学習率である。14行目において、$Q(\boldsymbol{x}_i,\boldsymbol{u}_i|\theta^Q)$は式(1)のように、アドバンテージ関数と状態価値$V(\boldsymbol{x}|\theta^V)$から求められる。

$$Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q)=A(\boldsymbol{x},\boldsymbol{u}|\theta^A)+V(\boldsymbol{x}|\theta^V) \quad (1)$$

$$A(\boldsymbol{x},\boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u}-\mu(\boldsymbol{x}|\theta^\mu))^T\boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u}-\mu(\boldsymbol{x}|\theta^\mu)) \quad (2)$$

$$\boldsymbol{P}(\boldsymbol{x}|\theta^P)=\boldsymbol{L}(\boldsymbol{x}|\theta^P)\boldsymbol{L}(\boldsymbol{x}|\theta^P)^T \quad (3)$$

アドバンテージ関数は式(2)で近似され、$\boldsymbol{x}$において行動$\boldsymbol{u}$をとった価値と、行動$\mu(\boldsymbol{x}|\theta^\mu)$をとった価値の差を表す。ここで、$\boldsymbol{P}$は正定行列であり、一般に(3)式を満たす下三角行列$\boldsymbol{L}(\boldsymbol{x}|\theta^P)$を持つ。従って、NAF ではこの$\boldsymbol{L}(\boldsymbol{x}|\theta^P)$を学習する。

Algorithm1
Normalized Advantage Function

1 Q 関数のランダム初期化　$Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q)$
2 ターゲットネットワークの初期化
$\theta^{Q'}\leftarrow\theta^Q$
3 リプレイバッファの初期化　$R\leftarrow\emptyset$
4 for episode = 1, max episode do
5 ノイズプロセスNの初期化
6 初期状態を観測 $\boldsymbol{x}_1\sim p(\boldsymbol{x}_1)$
7 for t = 1, max timestep do
8 行動$\boldsymbol{u}_t$の選択　$\boldsymbol{u}_t=\mu(\boldsymbol{x}_t|\theta^\mu)+N_t$
9 行動$\boldsymbol{u}_t$の実行
10 遷移$(\boldsymbol{x}_t,\boldsymbol{u}_t,r_t,\boldsymbol{x}_{t+1})$を$R$に追加
11 for iteration=1, I do
12 Rより、バッチサイズ分の遷移データをランダムサンプリング
13 $y_i=r_i+\gamma V'(\boldsymbol{x}_{i+1}|\theta^{Q\prime})$を代入
14 損失関数Lが最小となるようにθ^Qを更新
$$L=\frac{1}{n}\sum_i(y_i-Q(\boldsymbol{x}_i,\boldsymbol{u}_i|\theta^Q))^2$$
15 ターゲットネットワークの更新
$\theta^{Q'}\leftarrow\tau\theta^Q+(1-\tau)\theta^{Q'}$
16 End for
17 End for
18 End for

行動の探索のため、ネットワークから出力された行動$\boldsymbol{x}_t$にはノイズを付与する。本研究では、$\boldsymbol{x}_t$を平均値、$\alpha\boldsymbol{P}^{-1}$を共分散行列とした多変量正規分布を生成し、この分布からのランダムサンプリングによるノイズシステムを導入する。α はノイズの範囲を調整するハイパーパラメータであり、目的地に到達できるよう調整する。

3. シミュレーション環境

本研究では、Unity によって構築したシミュレーション環境内で学習を行う。Unity は、レンダリング、物理エンジン、Unity Editor と呼ばれるグラフィカルユーザーインターフェイスで構成されるリアルタイム 3D 開発プラットフォームである。Unity は、さまざまなプラットフォーム、開発者の技術レベル、およびゲームタイプをサポートする汎用エンジンとして開発された。したがって、Unity エンジンは AI 研究の理想的なシミュレーションプラットフォームである[3]。

3.1 Unity ML-Agents Toolkit

Unity ML-Agents Toolkit は、研究者や開発者が Unity Editor を使用してシミュレートされた環境を作成し、PythonAPI を介してそれらと通信できるようにするオープンソースプロジェクトである。ツールキットは、学習パイプラインを構築するためのコア C＃スクリプトとともに、Unity エディタ内の環境を定義するために必要なすべての機能を含む ML-Agents

SDK を提供する。また、ML-Agents Toolkit は、多くの研究者が使用する標準の gym インターフェースの使用を可能とするラッパーAPI のセットを提供している。これにより、gym ラッパーを介して独自の学習アルゴリズムの実装が可能となる。

3.2 ロボットモデル

本研究では，4 脚ロボットについて学習を行う。ロボットモデルを Fig.1 に示す。モデルは、ML-Agents Toolkit のサンプルである 4 脚ロボットモデル（Crawler）を使用した。Fig.2 に脚の正面図、Fig.3 に脚の平面図を示す。関節は、各パーツの接合部であり、各脚に二か所存在する。図中の矢印が関節の回転角を示している。関節の可動域を Table 1 に示す。各脚の自由度は 3 となる。サイズについて、Fig.1 の矢印の幅を 2[m]と定義する。

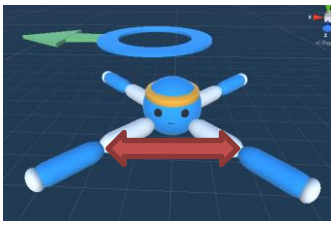

Fig.1 Crawler

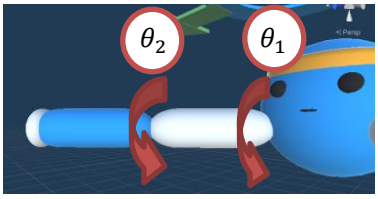

Fig.2 脚の正面図

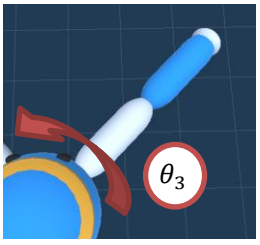

Fig.3 脚の平面図

Table 1 各関節の可動域

関節角	下限［°］	上限［°］
θ_1	−30	30
θ_2	60	120
θ_3	−30	30

3.3 シミュレーション環境

ML-Agents Toolkit のサンプル Crawler をもとにシミュレーション環境を構築する。Fig.4 に構築した環境を示す。地形は平地であり、乗り越えることのできない壁に囲まれている。学習を行うモデルは 3.2 節で述べた 4 脚ロボットモデルである。学習の目的を「ターゲット地点への到達」とし、かつ移動の際は「ターゲットの方向を向く」ことを目指す。Fig.4 における緑色のキューブがターゲット地点となり、ロボットから 30[m]離れた地点とする。

Fig.4　構築した環境

4. 学習パラメータ

4.1 実験方法

各ハイパーパラメータを Table 2 に示す。1 試行あたり 5,000 ステップを最大とし、1,000,000 ステップで学習を行う。ロボットが転倒した場合や、ボディが地面に接触した場合にはロボットの位置がリセットされる。初期位置のボディの向きは、ターゲットの方向を向くよう設定した。

Table 2 ハイパーパラメータ

ハイパーパラメータ	値

最適化手法	Adam
割引率	0.99
ターゲット更新係数	0.005
学習率	0.001
損失関数	MSE
活性化関数	tanh
リプレイバッファサイズ	100,000
バッチサイズ	2048
Qネットワーク、ターゲットネットワークの隠れ層の数	4
Qネットワーク、ターゲットネットワークの隠れ層のユニット数	512

4.2 学習要素

学習要素である状態、行動、報酬は Unity 内で定義される。これを gym ラッパーにより Pytorch が受け取り、行動パラメータを送信することでロボットの制御を行う。

4.2.1 行動

Table 3 に、行動として Pytorch から送信されるデータを示す。行動空間の合計の次元数は 20 となる。

Table 3 行動データ

データ	次元数
上脚の回転角	2×4
前脚の回転角	1×4
上脚の力	1×4
前脚の力	1×4
合計	20

4.2.2 状態

Table 4、Table 5 に、状態として Pytorch で受信するデータを示す。状態空間の合計の次元数は 158 となる。また、回転角はクオータニオンで表わされる。

Table 4 状態データ

データ	次元数
速度ベクトルと目標速度ベクトルの差	1
速度ベクトル	3
目標速度ベクトル	3
ボディ方向からキューブ方向への回転角	4
ターゲットまでの距離	3
ボディ下部の地面までの距離	1
ボディのパーツ情報	15
上脚のパーツ情報	16×4
前脚のパーツ情報	16×4
合計	158

Table 5 パーツ情報

データ	次元数
ボディのパーツ情報	15
↳地面の衝突判定	1
↳速度	3
↳回転角	4
↳ボディからの相対座標	3
↳ボディからの相対回転角	4
上脚、前脚のパーツ情報	16
↳地面の衝突判定	1
↳速度	3
↳回転角	4
↳ボディからの相対座標	3
↳ボディからの相対回転角	4
↳力	1

4.2.3 報酬

報酬は大きく分けて 2 種類定義する。まず、ターゲット地点到達時のゴール報酬として$r = 1$を与える。次に、ステップ毎のステップ報酬である。学習の目的から、Table 6 のようにステップ報酬を設計した。vは目標速度ベクトルとロボットの速度ベクトルの内積、θ[rad]はロボ

ットの向きとターゲット方向の成す角を示す。

Table 6　ステップ報酬

	$v > 0.2$	$0.2 \geq v > 0$	$v \leq 0$
$\frac{\pi}{2} < \lvert\theta\rvert \leq \pi$	-1	-1	-1
$\frac{\pi}{4} < \lvert\theta\rvert \leq \frac{\pi}{2}$	0	-1	-1
$0 \leq \lvert\theta\rvert < \frac{\pi}{4}$	1	0	-1

5. 実験

行動空間ノイズのパラメータαについて、α = 0.1、α = 0.3、α = 0.5、α = 1、α = 2それぞれの場合について比較する。Table 7 にゴール回数を、Fig.5 にターゲットとロボット間の最小距離を示す。Table 7 より、α = 0.3においてもっともゴール回数が多くなり、そこからαの値が離れるほどゴール回数が減っていることが分かる。

Table 7　各αにおけるゴール回数

α	0.1	0.3	0.5	1	2
ゴール回数	7	94	11	1	0

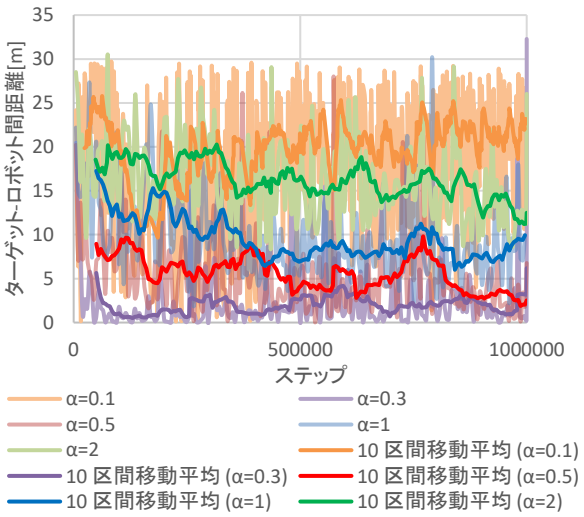

Fig.5 ターゲット-ロボット間距離

動作について、ノイズパラメータに関わらず、脚を小刻みに動かして移動する行動が得られた。特にα = 0.1については、全脚が内側に折りたたまれた際に、元に戻すことが出来ず転倒してしまう挙動が多く見受けられた。

6. おわりに

今回は行動空間ノイズの大きさによる学習結果を比較した。結果として、ほぼすべてのαにおいて歩行動作が得られた。特に、α = 0.3で優れた結果が得られた。これについて、学習条件の多くを単純化しているため、探索の必要性が少なく、小さいほど良い結果が得られたと考えられる。今後の課題として、ロボットがランダムな方向を向いて生成され方向転換が必要な場合等の環境での学習を行っていきたい。

参考文献

[1]. 渡邉宗一郎,西山雄輝,程島竜一,琴坂信哉,“ザトウグモ型 6 脚歩行ロボット ASURA I の開発-第 4 報: 脚機構の解析と基本動作実験”,ロボティクスメカニクス講演会 1A1-P10(2013).

[2]. Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, Sergey Levine,” Continuous Deep Q-Learning with Model-based Acceleration”,arXiv:1603.00748v1 (2016)

[3]. Arthur Juliani, Vincent-Pierre Berges, Ervin Teng , Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, Danny Lange, ”Unity: A General Platform for Intelligent Agents”, arXiv:1809.02627 (2020).