2003 年情報論的学習理論ワークショップ
2003 Workshop on Information-Based Induction Sciences (IBIS2003)
Kyoto, Japan, Nov 11 - 12, 2003.

強化学習による８自由度ロボットの制御規則獲得

Reinforcement learning of control rules for a 8 d.o.f robot

木村 元　　小林 重信*
Hajime Kimura　　Shigenobu Kobayashi

Abstract: We investigate a reinforcement learning of walking behavior for a four-legged robot. The robot has two servo motors per leg, so this problem has eight-dimensional continuous state-action space. Applying actor-critic algorithms with some generalization techniques will the best way to deal with the learning control in such continuous space. In real robots, the learning controllers are often faced with incomplete state observation that makes the environment to partially observable Markov decision processes (POMDPs). We present an actor-critic algorithm that can deal with a certain class of POMDPs and also present action selection scheme for the actor-critic, in which the actor selects continuous action from its bounded action space by using normal distribution. The experimental results show that the robot successfully learns to walk in practical learning steps.

1　はじめに

強化学習は，報酬という単純な指示から，それよりずっと複雑な制御規則を自動的に獲得するための手法として有望である [4]．強化学習で最も一般的なのは Q-learning[11] のように状態-行動の評価値 (value) を推定した後に最適な政策を得る方法である．強化学習を実問題へ適用するために，連続な状態空間を扱うための汎化テクニックと組み合わせる方法 [9] が知られている．しかし実環境におけるロボットの学習制御においては連続で膨大な状態だけでなく，連続な行動空間の扱いも求められるため，Q-learning によって全ての状態-行動空間における評価値を推定することは，メモリ容量の点からも探索のための試行回数の点でも事実上実行不能である．

一方，actor-critic アルゴリズムは，政策勾配法に基づく強化学習法で，連続な状態-行動空間への適用が比較的容易である．この手法は，ある行動政策（状態から行動への分布関数）のもとでの各状態の評価値を推定し，これを手がかりにしてその行動政策を改善していくことを基本としている．Q-learning のように全状態-行動の評価値を推定してから最適な状態-行動対を見出すわけではないため，少ないメモリーでの実装が可能であり，学習も高速である．さらに，行動選択を特徴付ける政策関数には任意の確率分布関数または確率密度関数が使用できるため，ガウス分布など計算が簡単な分布を用いれば連続な行動空間への適用が容易である．このアプローチは局所解に陥る可能性を伴うものの，これ以外にも適正度の履歴を用いることにより，ある POMDP の環境下でも学習可能であるなど多くの利点を有するため，高次元の空間を持つ強化学習の実問題においてたびたび用いられている [8]．

本稿では actor-critic アルゴリズムを４足ロボットの歩行動作獲得問題へ適用するための実装例を紹介する．扱うロボットは１脚あたりモータ２個を持ち，全体で８次元の連続な状態空間と同時に８次元連続値の行動空間を持つ強化学習問題である．このような問題において actor の行動選択に正規分布を用いる方法と，選択した行動が行動空間の上界または下界の値からはみ出す場合の計算上の対処方法などについて紹介する．実機を用いた実験では，ロボットは actor-critic 法により現実的な時間内で歩行動作を獲得できることを示す．

2　問題設定

2.1　４足ロボット

本稿では図１に示す２種類の４足ロボット（クモ型の OCT1 とイヌ型の OCT2）を扱う．ロボットの各脚は図１に示すように２個の位置制御サーボモータで駆動され，全体で８自由度である．クモ型ロボットの OCT1 では，

*東京工業大学 大学院総合理工学研究科, 226-8502 横浜市緑区長津田 4259, tel. 045-924-5648, e-mail kobayasi@dis.titech.ac.jp, Tokyo Institute of Technology, Interdisciplinary Graduate School of Sci. and Eng., 226-8502 Nagatsuta Midor-ku Yokohama JAPAN

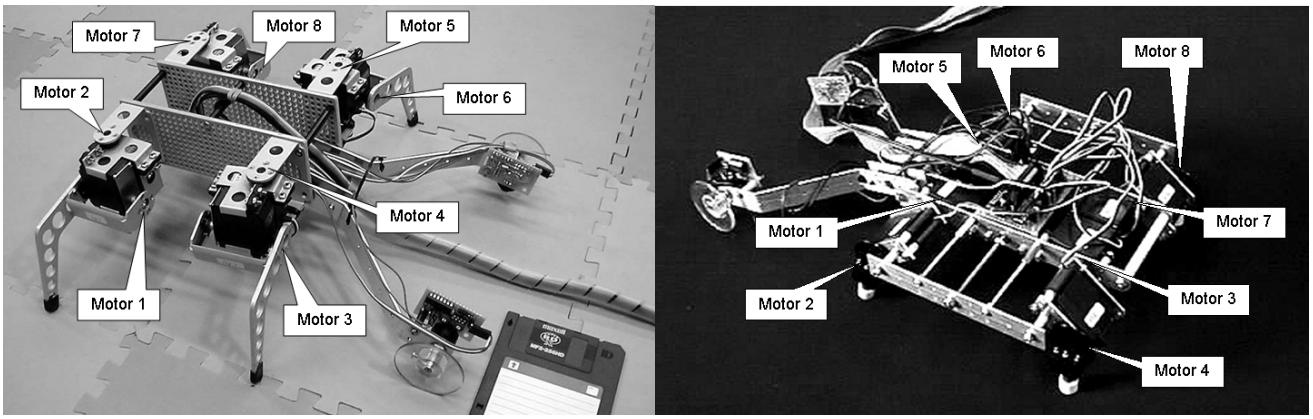

図 1: 学習対象のロボット（左 OCT1, 右 OCT2）各足は角度指令で動作するサーボモータ 2 個を持つ.

各脚の 2 つのモータはそれぞれリフト脚，スイング脚を受け持つ．リフト脚は上下方向，スイング脚は前後方向へ足先を運ぶ役割を持つ．これに対しイヌ型ロボットの OCT2 では，各脚の 2 つのモータが駆動する関節軸は平行であり，足先を運ぶ方向について役割分担がなく，互いの動作が干渉するため，制御規則の構築は OCT1 よりも困難である．各モータと関節は図中で示すとおり，OCT1 では反時計回りに対応付けられているのに対し，OCT2 では左側前後，右側前後の順番で割り当てられている点でも構造的に異なる．

学習目標は，なるべく速くまっすぐに前進する制御規則を獲得することである．このロボットを制御するコントローラが学習主体の「エージェント」になり，ロボット本体を含めた外界がエージェントにとっての「環境」になる．エージェントは事前に環境やロボットのダイナミクスを知らされていないため，試行錯誤を通じて制御規則を形成していくことが求められる．エージェントの保持する制御規則は，状態入力から行動への確率分布という形式の政策関数で表現される．エージェントは試行錯誤を通じて政策関数のパラメータを改善する．

各時間ステップにおいてエージェントは 8 個のサーボモータの角度を観測して現在の状態とし，政策関数に従って行動を選択する．行動は 8 個のサーボモータの角度の目標値である．よって現在の状態は 1 ステップ前に出力した行動に等しい．行動を実行後，約 0.3 秒後に状態遷移結果として報酬が与えられ，次の時間ステップへと進む．図 1 のロボット後方に示される 2 個の車輪は，ボディの移動を検出して報酬信号を生成する．2 つの車輪の移動速度平均はロボットの前進速度を示し，2 つの車輪の移動速度の差分はロボットが旋回していることを示す．ロボットの学習目標はまっすぐに前進することなので，各時間ステップでの報酬は，2 つの車輪の移動速度平均から差分の絶対値を引いた値とした．

2.2 マルコフ決定過程による環境モデル化

本稿ではロボットの学習問題をマルコフ決定過程 (MDP) における強化学習問題として以下のように定式化する．状態空間を $\mathcal{S}$，行動空間を $\mathcal{A}$，実数の集合を $\mathcal{R}$ と表す．各離散時間ステップ t において，エージェントは状態 $s_t \in \mathcal{S}$ を観測して行動 $a_t \in \mathcal{A}$ を実行し，状態遷移の結果，報酬 $r_t \in \mathcal{R}$ を受け取る．一般に報酬と遷移先の状態は確率変数だが，MDP ではその分布は s_t と a_t だけに依存する．遷移先の状態 s_{t+1} は遷移確率 $T(s_t, a, s_{t+1})$ に従い，報酬 r_t は期待値 $r(s_t, a)$ に従って与えられる．

学習エージェントは $T(s_t, a, s_{t+1})$ や $r(s_t, a)$ についての知識は事前に持たない．強化学習の目的はエージェントのパフォーマンスを最大化する政策 (policy) を生成することである．無限期間のタスクにおいて自然な評価規範として，以下の割引報酬合計による評価がある．

$$V_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \ , \tag{1}$$

ただし割引率 $0 \leq \gamma \leq 1$ は未来の報酬の重要度を示し，V_t は時刻 t の評価値を表す．MDP では，評価値は以下のように定義できる：

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \, r_t \middle| \, s_0 = s, \pi \right] \ , \tag{2}$$

ただし $E\{\cdot\}$ は期待値演算を表す．MDP では式 2 で定義される各状態 s における評価値を最大化する政策を探すことが目標である．本稿のロボット学習問題では，状態空間が上下界の存在する 8 次元の連続な空間であり，行動空間も同じである．また，ここでは，ロボット学習問題を MDP でモデル化したが，実際にはセンサが不完全なため，観測入力と真の状態が一部で食い違う部分観測 MDP (partially observable MDP: POMDP) となる．

3 強化学習アルゴリズム

3.1 Actor-Critic アルゴリズム

Actor-critic アーキテクチャ [1](図 2) は基本となる仕組みが単純な上，連続空間への適用が容易で，MDP だけでなく非マルコフ問題の一種である POMDP におけるあるクラスにおいても有望な手法である．Actor は，状態から行動への確率分布である確率的政策 π に従って行動を実行する．Critic は，actor の政策のもとでのそれぞれの状態の評価値を推定する．ある状態 s_t で行動 a_t を選択後，報酬 r_t を受け取って状態 s_{t+1} へ遷移した時，critic は以下の TD 法に従って状態評価の推定値 $\hat{V}(s_t)$ を更新する：

$$\text{TD\_error} = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \quad (3)$$

$$V(s_t) \leftarrow V(s_t) + \alpha \text{TD\_error} \quad (4)$$

ただし $0 < \alpha \leq 1$ は学習率，$0 \leq \gamma < 1$ は割引率を表す．Actor は，critic で計算される TD_error を手がかりにして政策を改善する．TD_error が正のとき，エージェントは良い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を少し高く修正する．逆に TD_error が負のとき，エージェントは悪い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を少し低く修正する．この原理は行動空間が連続であってもそのまま拡張できる．

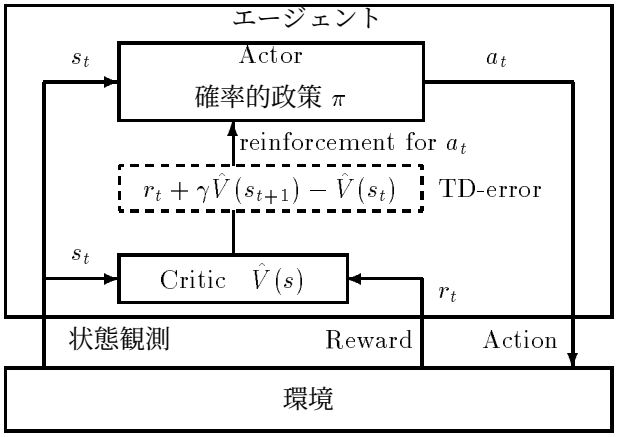

図 2: 一般的な actor-critic アルゴリズムの枠組．Critic は状態評価値を推定し actor に TD-error を与える．Actor はそれに基づいて政策を改善する．TD-error > 0 の場合，行動 a_t は良い行動と考えられるので a_t をとる確率を上げる．さもなければ a_t の確率を下げる．

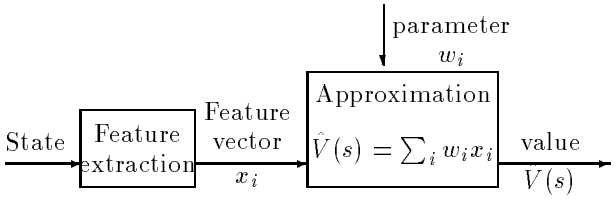

図 3: 特徴量ベクトルへの変換を介した線形アーキテクチャによる汎化.

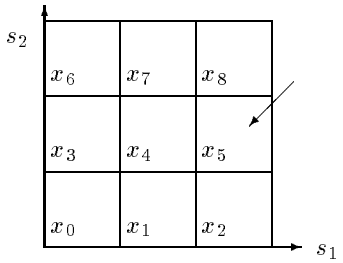

図 4: タイルコーディングによる特徴量ベクトル生成例. 状態入力 (s_1, s_2) が矢印の位置のとき，特徴量ベクトル $(x_0, x_1, \cdots x_8)$ は，該当するタイル要素 $x_5 = 1$，それ以外の要素は全てゼロ.

3.2 観測入力からの状態表現生成

強化学習の基本は，センサ入力（観測入力）から生成された状態表現から，状態評価値や行動確率分布への写像関数を表現することである．この観測入力からの状態表現生成は特徴量抽出 (feature extraction) に関連するが，強化学習ではこの部分については扱わず，予め与えられるものとするのが一般的である．観測入力が連続空間の場合，このテクニックは汎化 (generalization) や関数近似と呼ばれ，強化学習に限らずニューラルネットワークなど幅広い分野で研究されており，強化学習または DP と関数近似法を組み合わせた計算法は neuro-dynamic programming[2] と呼ばれる．Q-learning の最適解への収束が保障される簡便な汎化方法として線形アーキテクチャ (linear architecture) がある．これは固定された基底関数の集合を用いて関数近似を行うもので，ラジアル基底やウエーブレット関数基底を用いたものなど様々だが，状態空間中の座標を基底関数によって特徴量ベクトルに変換し，このベクトルの線形重み付け和で関数を表現する点は同じである（図 3）．特徴量ベクトルの生成方法は問題に応じて設計者が与える必要があり，ベクトルのノルムが常に 1 となることが望ましい．図 4 は最も単純な特徴ベクトル生成方法の例を示す。2 次元空間を 1 枚のタイル（グリッド）によって分割し，各タイル要素に特徴量ベクトルの要素を割り当てる．状態入力があるタイル要素内にあるとき，該当する特徴量ベクトル要素の値は 1 に，それ以外の要素はゼロとする．V 値を

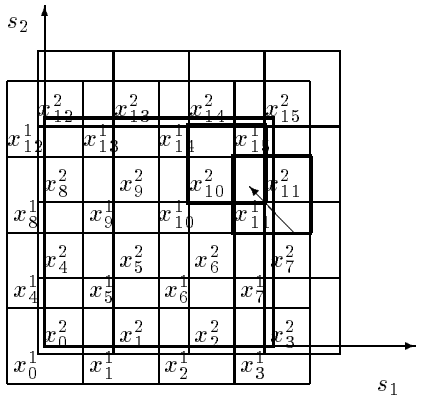

図 5: 2枚のタイルをずらせて重ねる CMAC による特徴量ベクトル生成．大きな太線の枠は状態の定義域，2つの小さい太線枠は状態入力に対応するタイルを表す．状態入力が矢印の位置のとき，特徴量ベクトルは，該当するタイル要素 $x^1_{11}=0.5$，$x^2_{10}=0.5$，それ以外の全要素はゼロ．

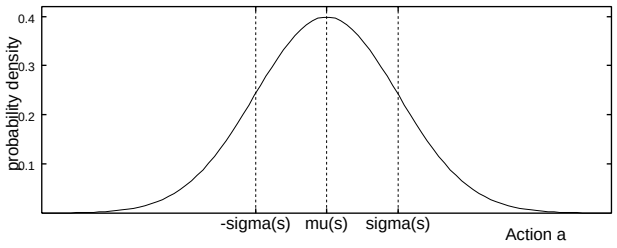

図 6: Actor において 1 次元連続値行動を扱う例．中心値 $\mathrm{mu}(s)$，標準偏差 $\mathrm{sigma}(s)$ の正規分布に従ったランダムサンプルによって行動 a を選択．状態遷移の結果，TD_error が正なら，中心値 $\mathrm{mu}(s)$ を a の方向へ修正．行動 a が $\pm\mathrm{sigma}(s)$ の内側だったなら，$\mathrm{sigma}(s)$ を小さくする方向へ，外側なら大きくする方向へ修正する．TD_error が負なら逆の操作を行う．

表現するときは，特徴ベクトル要素と同数のパラメータ w^a_i を用いて線形関数表現する：

$$\hat{V}(s)=\sum_{i=1}^{n} w_i x_i \tag{5}$$

ただし n は特徴ベクトル要素の個数．線形アーキテクチャによる V 値更新は，単純な勾配法に基づきパラメータ w_i を更新することで達成される：

$$w_i \leftarrow w_i+\alpha\frac{\partial\hat{V}(s)}{\partial w_i}\mathrm{TD\_error} \tag{6}$$

式 (5) より $\partial\hat{V}(s)/\partial w_i=x_i$ なので，

$$w_i \leftarrow w_i+\alpha\, x_i\,\mathrm{TD\_error} \tag{7}$$

つまり線形アーキテクチャを用いた更新では，更新するパラメータ w_i に対応する特徴の要素の値 x_i だけを使って簡単に更新できる．図 4 のような単純なグリッド分割による線形アーキテクチャの更新式は離散的 TD 法の更新式（式 (4)）と等価になる．

3.3 連続な行動空間を扱う Actor-Critic

前述のように，actor の確率的政策 π を確率分布関数とすることで，actor-critic を連続行動空間へ拡張できる．Actor では，一般に確率的政策をパラメータ関数表現し，そのパラメータを勾配法によって更新していく．政策関数の場合，価値関数とは異なりこの部分は必ずしも線形関数である必要はなく，非線形関数を用いることができる．図 6 は確率的政策として正規分布を用いた例を示す．この actor を用いたエージェントは，中心値 $\mathrm{mu}(s)$，標準偏差 $\mathrm{sigma}(s)$ の正規分布に従ったランダムサンプリングによって行動 a を選択する．状態遷移後，式 3 に従って TD_error を計算し，この値に基づいて actor の政策パラメータを更新する．TD_error が正のとき，エージェントは良い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を高める方向へ修正するが，正規分布の場合は中心値 $\mathrm{mu}(s)$ を a の方向へ修正し，行動 a が $\pm\mathrm{sigma}(s)$ の内側だったなら，$\mathrm{sigma}(s)$ を小さくする方向へ，外側なら大きくする方向へ修正することで実現できる．TD_error が負の場合は逆の操作を行う．本ロボットの問題においては，行動空間に上下界が存在するため，この正規分布による行動選択に修正を加えている．

3.4 適正度の履歴を用いた Actor-Critic

図 7 は本ロボットに適用した actor-critic の詳細である．Actor と critic の両方に適正度の履歴 (eligibility trace)[9] を用いているのが特徴である．Critic では TD(λ) でそれを用いており，actor では政策パラメータについて用いている [5]．

Critic における TD(λ) の処理は図 7 のステップ 3 と 5 の部分である．パラメータ λ_v は適正度の履歴を特徴付け，TD($\lambda=\lambda_v$) となる．Actor では，確率的政策はパラメータ関数表現され，そのパラメータを価値関数の勾配を用いて更新していく [5][10]．エージェントが政策 π のもとで観測 s において行動 a を選択する確率を関数 $\pi(a|\theta,s)$ で表す．政策 $\pi(a|\theta,s)$ は行動 a の集合が連続値の場合は確率密度関数である．エージェントは内部変数 θ を調節することにより確率的政策 π を変える．Actor の更新処理は図 7 のステップ 4 と 5 である．パラメータ λ_π は actor の適正度の履歴を特徴付けるが，TD(λ) の場合とは性質がやや異なる．λ_π が 0 に近い場合，critic で推定された価値関数 $\hat{V}$ を用いて政策が更新される．λ_π

1. 状態 s_t を観測して行動 a_t を確率 $\pi(a_t|\theta, s_t)$ に従って選択
2. 報酬 r_t を受取り，次の状態 s_{t+1} を観測したら以下に従って TD-error を計算：

$$(\text{TD-error}) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \ , \quad (8)$$

ただし γ $(0 \leq \gamma \leq 1)$ は割引率，
$\hat{V}(s)$ は Critic が出力した割引報酬の期待値

3. TD 法を用いて critic の推定値 $\hat{V}(s)$ を更新

$$\begin{aligned} e_v(t) &= \frac{\partial}{\partial w}\hat{V}(s_t) \ , \quad (9) \\ \overline{e_v}(t) &\leftarrow e_v(t) + \overline{e_v}(t) \ , \\ \Delta w(t) &= (\text{TD-error})\, \overline{e_v}(t) \ , \\ w &\leftarrow w + \alpha_v \, \Delta w(t) \ , \end{aligned}$$

ただし e_v は関数 $\hat{V}(s)$ におけるパラメータ w に関する適正度，$\overline{e_v}$ はその履歴，α_v は学習率.

4. TD-error を用いて actor の行動選択確率更新

$$\begin{aligned} e_\pi(t) &= \frac{\partial}{\partial \theta} \ln\Big(\pi(a_t|\theta, s_t)\Big) \ , \quad (10) \\ \overline{e_\pi}(t) &\leftarrow e_\pi(t) + \overline{e_\pi}(t) \ , \\ \Delta\theta(t) &= (\text{TD-error})\, \overline{e_\pi}(t) \ , \\ \theta &\leftarrow \theta + \alpha_\pi \, \Delta\theta(t) \ , \end{aligned}$$

ただし e_π は政策パラメータ θ に関する適正度，$\overline{e_\pi}$ はその履歴，α_π は学習率.

5. 適正度の履歴を以下のように割り引く：

$$\begin{aligned} \overline{e_v}(t+1) &\leftarrow \gamma \lambda_v \, \overline{e_v}(t) \ , \\ \overline{e_\pi}(t+1) &\leftarrow \gamma \lambda_\pi \, \overline{e_\pi}(t) \ , \end{aligned}$$

ただし λ_v と λ_π $(0 \leq \lambda_v, \lambda_\pi \leq 1)$ はそれぞれ critic と actor の適正度の減衰率である.

6. 時刻を $t \leftarrow t+1$ に進め step1 から繰返す

図 7: 適正度の履歴を用いた actor-critic アルゴリズム

が 1 に近い場合，式 1 で定義される実際の報酬のサンプル合計による評価値を用いて政策が更新される．すると，更新が環境のマルコフ性に依存せず実行可能となり，ある程度の POMDP 環境下での政策改善が可能になる．

3.5 4足ロボット学習問題のための実装

ロボットの 8 個の関節モータ角度をベクトル $(s_1, \cdots, s_8)$ によって表現し，これを状態とする．ただし各要素は $-1 \leq s_i \leq 1$，$i = 1, 2, \cdots, 8$ のように正規化されているものとする．Critic では 8 次元の連続な状態空間は各座標軸毎に 2 分割，合計 2^8 個の超矩形領域によって離散化され，状態は 2^8 の長さを持つ単位ベクトル $(x_1, x_2, \cdots, x_{256})$ を用いて，該当する領域の要素のみ 1，それ以外の要素は 0 という具合に表現される．図 7 中で表される推定値 $\hat{V}(s_t)$ は以下のように計算される：

$$\hat{V}(s_t) = \sum_{b=1}^{256} x_b w_b \ , \quad (11)$$

ただし w_b は関数近似パラメータである．式 9 の $e_v(t)_b$ は b^{th} 番目のパラメータ w_b の適正度 (eligibility) である．すなわち $e_v(t) = (e_v(t)_1, e_v(t)_2, \cdots e_v(t)_b \cdots, e_v(t)_{256})$，このとき式 11 より

$$e_v(t)_b = \begin{cases} 1 & , \text{where } x_b = 1 \ , \\ 0 & , \text{where } x_b = 0 \ . \end{cases} \quad (12)$$

ロボットの 8 個の関節モータ角度の目標値をベクトル $(a_1, a_2, \cdots, a_8)$ によって表現し，これを行動とする．ただし各要素は状態ベクトルと同様に $-1 \leq a_i \leq 1$，$i = 1, 2, \cdots, 8$ のように正規化されているものとする．Actor は以下のような行動選択を行う：各モータ (i) において，actor は正規分布 $N(\mu_{(i)}, \sigma^2_{(i)})$ に従い，$[-1, 1]$ の範囲の値を得るまでランダムサンプルをとり続け，その値が出たらそれを選択する．正規分布のパラメータ $\mu_{(i)}$ と $\sigma_{(i)}$ は以下のシグモイド関数によって与える：

$$\begin{aligned} \mu_{(i)} &= \frac{2}{1 + \exp\left(-\sum_{k=1}^{8} s_k \theta_{k,(i)}\right)} - 1 \ , \\ \sigma_{(i)} &= \frac{1}{1 + \exp\left(-\theta_{9,(i)}\right)} \ , \end{aligned}$$

ただし $\theta_{k,(i)}$ $(k = 1, 2, \cdots 9)$ は政策パラメータである．正規分布 $N(\mu_{(i)}, \sigma^2_{(i)})$ が 1 回の試行で $[-1, 1]$ の範囲のサンプルを生成する確率を P_{in} とすると，

$$P_{in} = \int_{-1}^{1} \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_{(i)})^2}{2\sigma^2_{(i)}}\right) dx \quad (13)$$

このとき政策関数は以下で表される：

$$\begin{aligned} &\pi(a_{(i)}|\theta, s_t \ ) \\ &= \big(1 + (1 - P_{in}) + (1 - P_{in})^2 + \cdots\big) \\ &\quad \times \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(a_{(i)} - \mu_{(i)})^2}{2\sigma^2_{(i)}}\right) \\ &= \frac{1}{P_{in}} \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(a_{(i)} - \mu_{(i)})^2}{2\sigma^2_{(i)}}\right) \ , \quad (14) \end{aligned}$$

ただし $a_{(i)}$ は $[-1, 1]$ の範囲に制限される．i^{th} 番目のモータにおける k^{th} 番目の政策パラメータ $\theta_{k,(i)}$ の適正度を $e_\pi(t)_{k,(i)}$ と表すと，式 10 中の θ と $e_\pi(t)$ は 9×8 のマトリクスによって表現される：

$$\theta = \begin{Bmatrix} \theta_{1,(1)} & \theta_{2,(1)} & \cdots & \theta_{9,(1)} \\ \theta_{1,(2)} & \theta_{2,(2)} & \cdots & \theta_{9,(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{1,(8)} & \theta_{2,(8)} & \cdots & \theta_{9,(8)} \end{Bmatrix}$$

$$e_\pi(t) = \begin{Bmatrix} e_\pi(t)_{1,(1)} & e_\pi(t)_{2,(1)} & \cdots & e_\pi(t)_{9,(1)} \\ e_\pi(t)_{1,(2)} & e_\pi(t)_{2,(2)} & \cdots & e_\pi(t)_{9,(2)} \\ \vdots & \vdots & \ddots & \vdots \\ e_\pi(t)_{1,(8)} & e_\pi(t)_{2,(8)} & \cdots & e_\pi(t)_{9,(8)} \end{Bmatrix}$$

式 10，14 より各 $e_\pi(t)_{k,(i)}$ は以下で与えられる．

$$\begin{aligned} e_\pi(t)_{k,(i)} &= \frac{\partial}{\partial\theta_{k,(i)}} \ln \pi(a_{(i)}|\theta, s_t) \\ &= \frac{\partial\mu_{(i)}}{\partial\theta_{k,(i)}} \frac{\partial}{\partial\mu_{(i)}} \ln \left(\frac{1}{P_{in}\sigma_{(i)}\sqrt{2\pi}} \exp \frac{(a_{(i)}-\mu_{(i)})^2}{-2\sigma_{(i)}^2} \right) \\ &= \frac{s_i(1-\mu_{(i)})(1+\mu_{(i)})}{2} \\ &\quad \times \left(\frac{a_{(i)}-\mu_{(i)}}{\sigma_{(i)}^2} + P_{in}\frac{\partial}{\partial\mu_{(i)}}\frac{1}{P_{in}} \right) \end{aligned} \quad (15)$$

ただし $k = 1, 2, \cdots 8$ である．$k = 9$ の場合の適正度は

$$\begin{aligned} e_\pi(t)_{k,(i)} &= \frac{\partial}{\partial\theta_{k,(i)}} \ln \pi(a_{(i)}|\theta, s_t) \\ &= \frac{\partial\sigma_{(i)}}{\partial\theta_{k,(i)}} \frac{\partial}{\partial\sigma_{(i)}} \ln \left(\frac{1}{P_{in}\sigma_{(i)}\sqrt{2\pi}} \exp \frac{(a_{(i)}-\mu_{(i)})^2}{-2\sigma_{(i)}^2} \right) \\ &= \sigma_{(i)}(1-\sigma_{(i)}) \\ &\quad \times \left(\frac{(a_{(i)}-\mu_{(i)})^2 - \sigma_{(i)}^2}{\sigma_{(i)}^3} + P_{in}\frac{\partial}{\partial\sigma_{(i)}}\frac{1}{P_{in}} \right), \end{aligned} \quad (16)$$

ただし $k = 9$ である．式 15 と式 16 の第 2 項は定積分を $\mu_{(i)}$ または $\sigma_{(i)}$ で微分する操作が求められる．本稿ではこれを近似数値計算により与えたが，これらの関数は上下界を持つ $\mu_{(i)}$ や $\sigma_{(i)}$ にのみ依存するので，テーブル等を用いることで計算コストを減らせる．

ここで $\sigma_{(i)}$ が 0 に近付くと適正度が発散してしまうことに注意が必要である．これは式 15，16 において $\sigma_{(i)}$ が分母になっているためである．適正度が発散すると政策更新のステップ幅が不当に大きくなりすぎて学習が失敗してしまう．そのため本稿では更新のステップサイズを $\sigma_{(i)}^2$ に比例させるというヒューリスティクスを用いた．このとき適正度は以下のように与えられる．

$$\begin{aligned} e_\pi(t)_{k,(i)} &= \frac{s_i\,(1-\mu_{(i)})(1+\mu_{(i)})}{2}\,(a_{(i)}-\mu_{(i)}) \\ &\quad + \frac{s_i\,(1+\mu_{(i)})(1-\mu_{(i)})}{\sigma_{(i)}^2}\left(P_{in}\frac{\partial}{\partial\mu_{(i)}}\frac{1}{P_{in}}\right), \quad (17) \\ &\quad \text{where } k = 1, 2, \cdots 8, \\ &= (1-\sigma_{(i)})\left((a_{(i)}-\mu_{(i)})^2 - \sigma_{(i)}^2\right) \\ &\quad + (1-\sigma_{(i)})\sigma_{(i)}^3\left(P_{in}\frac{\partial}{\partial\sigma_{(i)}}\frac{1}{P_{in}}\right), \text{where } k = 9. \end{aligned}$$

4 実験結果

OCT1 は比較的よく滑べるカーペット上および摩擦の大きなゴムマット上で実験を行った．OCT2 はカーペット上のみで実験を行った．パラメータ設定は $\gamma = 0.9$，$\alpha_v = 0.1$，$\alpha_\pi = 0.002$，$\lambda_v = 1.0$，$\lambda_\pi = 1.0$ である．

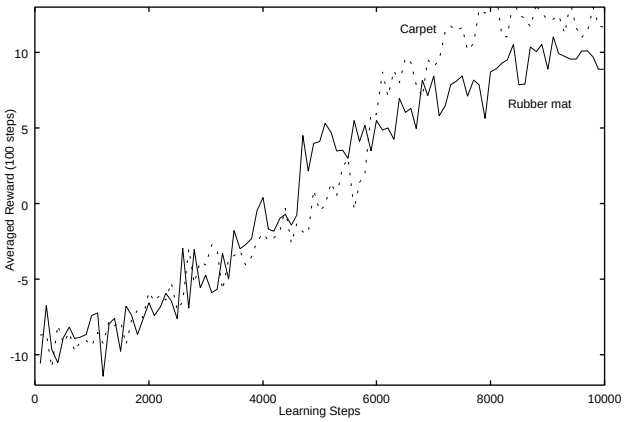

図 8: ＯＣＴ１の学習の様子．各試行は 10000 ステップ（約 80 分）．

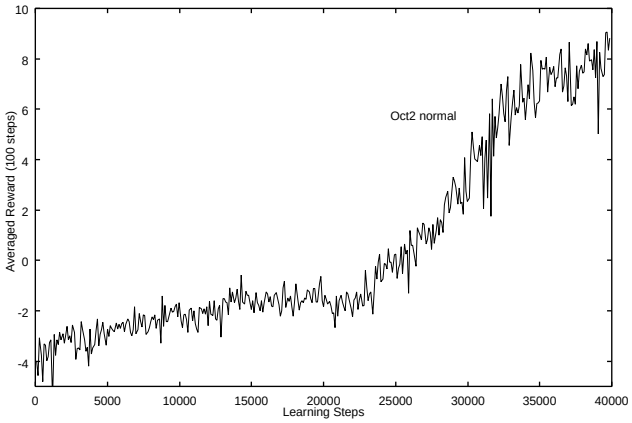

図 9: ＯＣＴ２の学習の様子．各試行は 40000 ステップ（約 160 分）．

図 8 は OCT1 における学習曲線を表す．縦軸は 100 ステップ毎の報酬の平均値を表すが，報酬は (１ステップで２つの車輪が前進した平均値) − (１ステップでの２つの車輪の差分の絶対値) という形式で与えられるため，学習初期において値が負の値を示しているのはロボットが後進しているせいではなく，左右に旋回しているためである．OCT1 ロボットはカーペットおよびゴムマット上の両方の設定において 8000 ステップ (約１時間) 以下という実時間で良好な挙動を獲得している．10000 ステップ後の実際の移動速度は約 5cm/sec である．

図 9 は OCT2 における学習曲線を表す．本ロボットの学習問題では，１ステップ前に出力した行動が現在の状態として観測されるが，場合によってはモータが応答し切れずに実際の状態と異なってしまう隠れ状態問題が存在する．学習初期から中期にかけてこの影響が強く，特に OCT2 で顕著である．Actor に適正度の履歴を用いる

と隠れ状態問題に対処できるため，従来の actor-critic より約 40 分早く行動を獲得できた．OCT1 でも同様の効果が確認された．また OCT2 では各脚の２つのモータは足先を運ぶ方向について役割分担がなく，互いの動作が干渉するという困難さから OCT1 に比べ動作獲得までに３〜４倍の学習ステップ数を要している．それでも約２時間という実時間で動作獲得ができた．

OCT1 では図 10 左側に示すようにカメのような歩行動作を獲得した．OCT1 では右側の前足 (leg4) と左側の後ろ足 (leg2) はほぼ同位相で，また左側の前足 (leg1) と右側の後ろ足 (leg3) がほぼ同位相で同期して動いており，歩容はトロット (trot) に近い．

カーペット上とゴムマット上では OCT1 で獲得された動作にはやや違いが見られた．カーペット上では，ボディは容易に床をすべることが可能なのでしばしばボディを引きずる動作が見られた．これは，支持脚で体重を支える動作よりも前進させる動作を優先して学習した結果と考えられる．一方，摩擦の大きなゴムマット上では，ロボットのボディが床から離れた状態を保持し続ける傾向が見られた．これらの結果は，ロボットがそれぞれ周囲の状況に応じて適応的に動作を獲得できたことを示している．

OCT2 でも図 10 右側に示すように足先以外の脚部やボディをひきずる動作となったが，左右に大きく傾きながらもきちんと直進する動作が得られた．OCT1 とはモータの配置が異なるにもかかわら歩容は OCT1 と類似している．

5　考察

【連続行動空間での行動選択】

Actor での行動選択に正規分布を用いる方法は行動選択時に必要な計算と学習時の計算が単純で簡単だという優れた特徴を有している [12][3][5] が，本ロボットにそのまま実装すると学習に失敗する．失敗の主な原因は，actor が行動空間の定義域についての情報を利用していないことに起因する．従来の正規分布による行動選択では，actor が定義域外の行動を選んだ場合，ロボットはそれを実行したふりをするが，実際には定義域境界上の行動を実行する．もし分布の平均を示すパラメータが定義域のはるか外側の値になってしまったら，ロボットはランダムな行動をとることがほとんどできなくなり，学習できなくなる．本稿で示した手法ではこの問題を単に定義域外の行動選択を棄却することで回避している．しかしその代償として適正度の計算に数値積分の微分演算という余計なコストが要求される．

【適正度の履歴による非マルコフ性への対処】

本実験の critic で用いた粗い分割による量子化は高次元の状態空間を扱う方法として有望だが，非マルコフ性が生じ，またそのように粗く近似された価値関数から良い政策を導くことは困難という問題点がある．本実験の結果は，actor の適正度の履歴が上記の問題点の解決に有効であることを示している．

【状態表現のための特徴ベクトル生成】

文献 [7] では状態-行動価値関数の近似における状態特徴ベクトルとして actor の適正度を使うべきであると主張している．彼らの手法は理論的な根拠に基づいており，高次元空間に対しても有望なため，この手法との組み合わせることでさらに性能向上が期待できる．

特徴ベクトルの与え方は，行動選択方法の与え方と並んで学習性能に大きく影響を与える．特徴ベクトルについては，そのノルムが状態にかかわらず一定であることが望ましいと述べたが，実験で用いた実装ではそのようになっていない．後日，ノルムが一定になるよう特徴ベクトルを工夫して実験を行ったところ，学習に２時間近くを要した OCT2 が１時間足らずで同等の制御規則を得た．

本稿で紹介した特徴ベクトル生成方法は，予め設計者が全状態空間について与えたが，訪問することがない状態に対するメモリ割当は無駄になる．そこで，経験した状態から適応的に生成する方法もある．また，この部分はいわゆる特徴抽出や符号化に関係する．

【得られた制御規則の妥当性】

獲得された動作では，ボディを引きずるなど好ましくないように思える動作が見られた．ボディが床についているかどうかや，エネルギーなどを報酬に反映させれば，よりロボティクス的に好ましい動作が獲得できるだろう．

参考文献

[1] Barto, A.G., Sutton, R.S. & Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC–13, no.5, pp. 834–846 (1983).

[2] Bertsekas, D.P. & Tsitsiklis, J.N.: *Neuro-Dynamic Programming*, Athena Scientific (1996).

[3] Doya, K.: Efficient Nonlinear Control with Actor-Tutor Architecture, *Advances in Neural Information Processing Systems 9*, pp. 1012–1018 (1997).

[4] 木村 元, 宮崎 和光, 小林 重信:強化学習システムの設計指針，計測と制御, Vol.38, No.10, pp.618–623 (1999).

[5] 木村 元，小林 重信：Actor に適正度の履歴を用いた Actor-Critic アルゴリズム– 不完全な Value-Function のもとでの強化学習，人工知能学会誌, Vol.15, No.2, pp.267–275 (2000).

[6] 木村 元，山下 透，小林 重信：強化学習による４足ロボットの歩行動作獲得，電気学会 電子情報システム部門誌, Vol.122-C, No.3, pp.330–337 (2002).

[7] Konda, V.R. & Tsitsiklis, J.N.: Actor-Critic Algorithms, *Advances in Neural Information Processing Systems 12*, pp. 1008–1014 (2000).

[8] Morimoto, J. & Doya, K.: Acquisition of Stand-up Behavior by a Real Robot using Hierarchical Reinforcement Learning, *Proceedings of the 17th International Conference on Machine Learning*, pp.623–630 (2000).

[9] Sutton, R.S. & Barto, A.: Reinforcement Learning: An Introduction, *A Bradford Book*, The MIT Press (1998).

[10] Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation, *Advances in Neural Information Processing Systems 12 (NIPS12)*, pp. 1057–1063 (2000).

[11] Watkins, C.J.C.H. & Dayan, P.: Technical Note: *Q*-Learning, *Machine Learning 8*, pp.279–292 (1992).

[12] Williams, R.J.: Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning 8*, pp. 229–256 (1992).

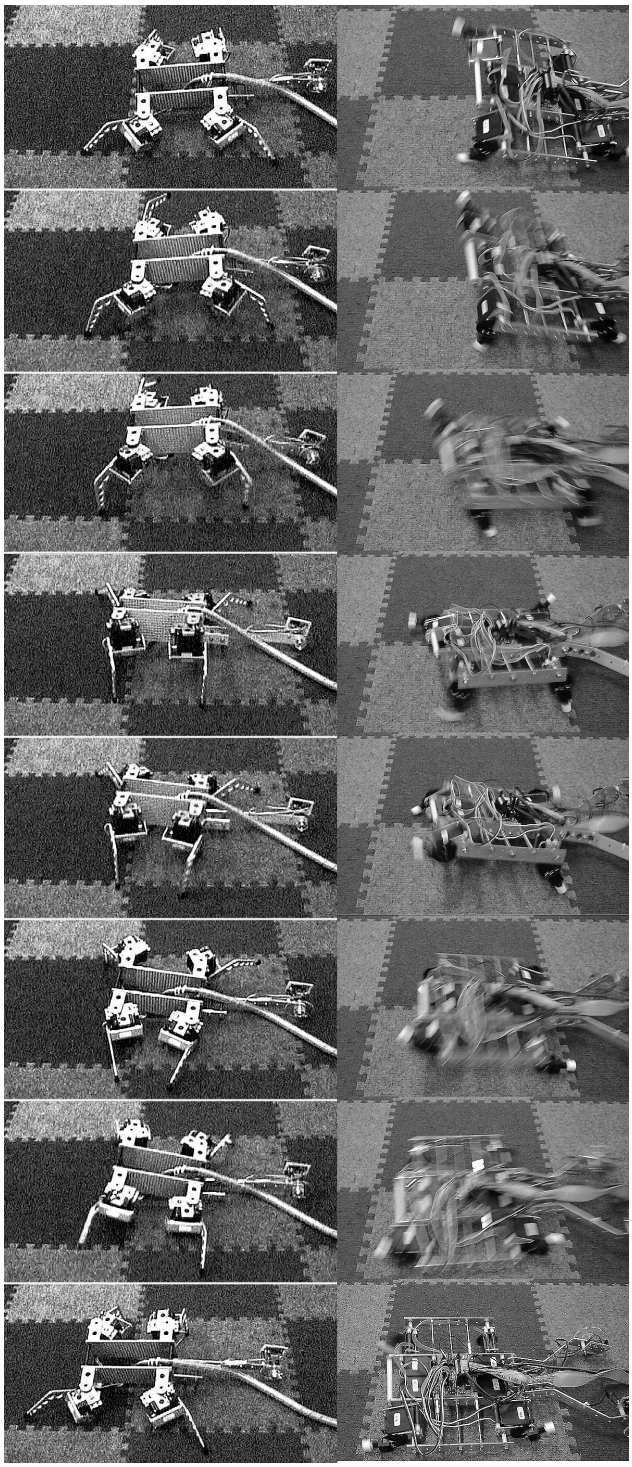

図 10: カーペット上にて 10000 ステップ（約 80 分）学習後に得た動作の一例．左側：約 10000 ステップ後（80 分），右側：約 30000 ステップ後（120 分）．