2 分木構造の Actor-Critic による 2 自由度ロボットの強化学習

○ 木村 元， 小林 重信
東京工業大学 大学院総合理工学研究科

Reinforcement learning in 2 degree of freedom robots using binary-tree actor-critic algorithms

○ Hajime Kimura, Shigenobu Kobayashi
Interdisciplinary Graduate School of Science and Eng., Tokyo Institute of Technology

Abstract: Learning control in real world applications requires dealing with both continuous state and several dozens or more of actions. For handling several dozens of actions, not only the techniques of action value approximation but also exploration strategies are essential. A binary tree action selector using an actor-critic algorithm is promising for such problems. This paper investigates properties of the binary tree action selector using the actor-critic through learning tasks in real robots that have 2 degree of freedom.

1 はじめに

強化学習は，ロボットの挙動を自ら改善したり知識の欠落を自ら補うための接近法として有望である．実環境のロボット学習制御において，繊細な制御を行うためには連続で膨大な状態空間の扱いが求められると同時に，多数の行動選択の学習も求められる．連続または膨大な状態空間を扱うための方法として CMAC[8] などの関数近似や instance-based な方法 [2] を用いて状態空間を汎化する接近法が確立されている．しかし多数の行動を扱うためには，action value を汎化する技術だけでなく，行動選択や探査戦略についても特別の工夫が必要である．

Q-learning[9] や SARSA アルゴリズムと共によく用いられるε -greedy 戦略やボルツマン選択戦略は仕組みが単純なのでよく用いられ，行動数が少ない場合は強力である．しかしこれらは行動の数が増加すると効率が悪くなる．これは，全ての行動を区別なくフラットに行動選択を行っていることに起因する．

われわれは日常生活において，たとえばハンドルを操作する場合「右に回す」あるいは「左に回す」のどちらかを選択した後「少しだけ回す」あるいは「普通に回す」を選択するといった具合に，しばしば階層的に行動を選択することで効率良く意思決定を行っている．このとき無意識に「右に少し回す」と「右に普通に回す」を「右に回す」という上位の行動によってグループ化していると考えられる．実環境における学習問題の多くはこのように行動空間に位相構造が存在するため，物理量的に近接・類似する行動をグループ化することで容易に階層化でき，効率的な意志決定が可能である．このように多数の類似した行動を階層的に効率よく扱う強化学習方式として，

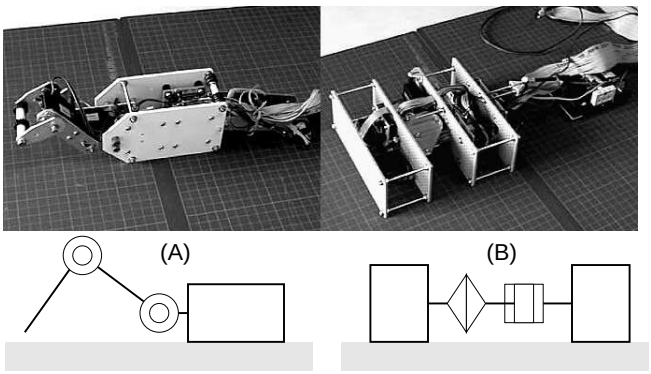


Figure 1: 学習対象としたロボット機構とその模式図．A はボディから 2 節リンクアームが張り出す構造を持ち，B はボディにねじりと曲げを行う構造を持つ．各ロボット後方 (写真右側) に見えるのは移動距離計測器である．

2 分木構造の行動選択器と actor-critic アルゴリズム [1] [8] を組み合わせた方法 [5] がある．本論文では上記の手法を 2 自由度ロボットの学習へ適用し，フラットな行動選択方法との比較を通じて有効性を示す．

2 2 自由度ロボットの学習問題

Fig.1に示すように，モータを 2 個搭載した 2 自由度の機構を持つロボット A および B に対し，完全に同一のアルゴリズムを適用し，効率よく前進する動作の獲得を試みる．エージェント (コントローラ) が獲得すべき制御規則は，現在の関節の角度を状態入力として与えられたとき，前進するような動きとなるようにモータの目標値とすべき関節の角度を出力す

ることである．ロボットの学習目標は，効率よく前進することなので，各時刻におけるボディの前進速度をエージェントが報酬として受け取るよう設定する．エージェントとロボット (環境) は以下のやりとりを行う．

1. エージェントはロボットの関節の角度θ_1, θ_2を環境の状態として観測する．
2. エージェントは行動出力として関節モータの角度の目標値 a_1, a_2を出力する．
3. ロボットは目標角度の方向へ各モータを動かす．
4. 約 0.2 秒後，ロボットはボディが移動した距離を計測し，その値を報酬とする．
5. ステップ 1 に戻って繰り返す．

状態観測である関節の角度θ_1, θ_2および行動出力である関節モータの角度の目標値 a_1, a_2は，それぞれ 0 から 7 (または 0 から 15) までの整数値をとる．報酬の値は$-128 \sim 127$ の整数値をとり，ボディが移動しない場合は 0 である．

関節の駆動モータとして模型用サーボモータを用いたため，本実験における状態観測θ_1, θ_2は，直前のステップにおいてエージェントが出力した行動 a_1, a_2 に等しい値とした．よって，エージェントが直前のステップで出力した値とは大きく異なる値 (値の差がおおむね 3 以上) を出力した場合，モータの応答が追い付かないため，エージェントが観測する角度状態とロボットの真の角度とが食い違う隠れ状態問題を生じやすい．

3 強化学習アルゴリズム

3.1 Actor-Critic アルゴリズム

Actor-critic 学習システムは，政策反復法 (policy iteration) の一種であると言われている．政策反復法は状態評価と政策改善を交互に繰り返す．Actor は，状態から行動への確率分布である確率的政策 (**stochastic policy**) 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 (value) を推定する．Actor は，環境から直接受け取る報酬と，critic で推定する評価値から計算される TD エラーを手がかりに政策を改善する (Fig.2)．TD エラーが正の場合，行動を実行した結果，より評価値の高い状態へ遷移したと考えられるので，その行動を選択する確率を増やす．逆に TD エラーが負の場合，行動を実行した結果，より悪い状態へ遷移したと考えられるので，その行動を選択する確率を減らすという処理を基本とする．Fig.3は本論文で用いたアルゴリズム [4] を示す．

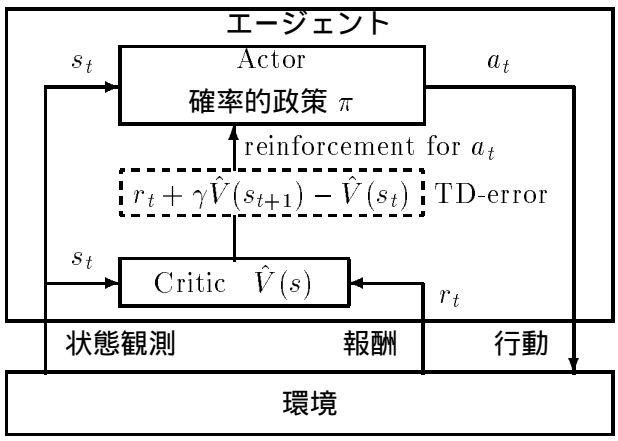


Figure 2: 一般的な actor-critic アルゴリズムの枠組．

Critic における処理手順は Fig.3のステップ 3 から 5 までに示されている．パラメータλ_vは TD($\lambda = \lambda_v$) の適正度の履歴を特徴づける．λ_vが 1 に近い場合，環境の非マルコフ性に対して頑健になる．λ_vを適切な値にすると学習を早める効果もある．

政策は actor において関数として明示的に表現され，政策パラメータθを value 関数の勾配によって更新する．観測 s においてエージェントが行動 a を選択する確率を政策πと呼び，関数$\pi(a, \theta, s)$ で表す．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．

Fig.3のステップ 4,5 は actor の処理手続きである．適正度 e_πは政策パラメータθが実行した行動 a_tに関与したインパクトを表している．パラメータλ_πは actor の適正度の履歴の性質を定めるが，その性質は TD(λ) の場合とほぼ同等である (詳しくは文献 [4] 参照)．

3.2 2 分木構造の確率的行動選択

まず 2 行動の場合に用いられる Bernoulli semilinear unit について紹介する．Fig.4の確率変数 $y \in \{0, \text{or } 1\}$ は分布 fに従い，$\Pr(y = 1) = f$または $\Pr(y = 0) = 1 - f$のように振舞う．エージェントは現在の状態を特徴ベクトル $(x_1, x_2, \cdots, x_i, \cdots, x_n)$ によって観測する．分布関数 fは以下に与えられる：

$$f = \frac{1}{1 + \exp\left(-\sum_i \theta_i x_i\right)} , \qquad (2)$$

ただし x_iは状態入力の特徴ベクトルの要素，θ_iは政策パラメータの要素を表す．エージェントは政策パラメータθの調節により挙動を改善する．エージェントは行動 1 または 0 を選択し，報酬 rを受け取る．そ

1. 状態 s_tを観測し，確率分布$\pi(a_t, \theta, s_t)$ に従って行動 a_tを選択／実行する．

2. 報酬 r_tを受け取り，遷移後の状態 s_{t+1}を観測して以下の TD-error を計算する．

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

γ $(0 \leq \gamma \leq 1)$ は割引率，$\hat{V}(s)$ は critic が推定した割引報酬の期待値を表す．

3. TD 法を用いて critic の$\hat{V}(s)$ を更新する．

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

ただし e_vは$\hat{V}(s)$ のパラメータ wの適正度，$\overline{e_v}$はその適正度の履歴，α_vは学習率を表す．

4. TD-error で actor の行動選択確率を更新

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

ただし e_πは政策パラメータπの適正度，$\overline{e_\pi}$はその適正度の履歴，α_πは学習率を表す．

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およびλ_π $(0 \leq \lambda_v, \lambda_\pi \leq 1)$ はそれぞれ critic と actor の割引率である．

6. 時刻を $t \leftarrow t+1$ に進めてステップ 1 へ

Figure 3: Actor と critic の両方に適正度の履歴を用いる actor-critic アルゴリズム．

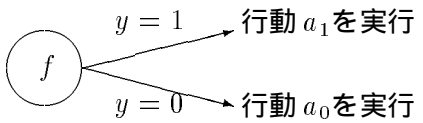


Figure 4: Bernoulli semilinear ユニット．確率変数 $y \in \{0, 1\}$ は分布 $\Pr(y = 1) = f$に従う．

のとき政策パラメータθの適正度 (eligibility) は以下の式で計算される．

$$\begin{aligned} e &= \frac{\partial}{\partial \theta_i} \ln \Pr(\text{実行された行動}) \\ &= \begin{cases} \frac{\partial}{\partial \theta_i} \ln f & \text{if } y = 1 \\ \frac{\partial}{\partial \theta_i} \ln(1-f) & \text{if } y = 0 \end{cases} \\ &= (y - f)\, x_i \end{aligned} \quad (3)$$

Fig.5は Fig.4を多段に拡張し，4 行動の場合の 2 分木行動選択方法の例を示す．葉ノードはそれぞれ行動に対応する．それ以外のノードは Bernoulli semilinear unit (Fig.4) によって構成される．理論上，Fig.5の確率的政策関数は以下のように記述される．

$$\pi(a, \theta, s) = \begin{cases} f_1 \, f_2 & a = a_1 \\ f_1 \, (1 - f_2) & a = a_2 \\ (1 - f_1) \, f_3 & a = a_3 \\ (1 - f_1) \, (1 - f_3) & a = a_4 \end{cases} \quad (4)$$

このとき f_1の適正度は以下の式で与えられる：

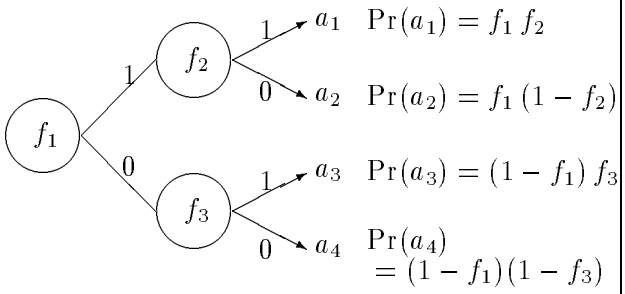
$$\begin{aligned} \frac{\partial}{\partial f_1} \ln \pi(a, \theta, s) &= \begin{cases} 1/f_1 & a = a_1 \\ 1/f_1 & a = a_2 \\ -1/(1 - f_1) & a = a_3 \\ -1/(1 - f_1) & a = a_4 \end{cases} \\ &= \frac{y_1 - f_1}{f_1(1 - f_1)} \ , \end{aligned} \quad (5)$$

ただし $y_1 \in \{0, 1\}$ はユニット f_1が選択した値を表す．ここで分布関数 f_j $(j = 1, 2 \text{ or } 3)$ を式 2のシグモイド関数で与えるもととする．式 3と式 5より，f_1の政策パラメータの適正度は以下のようになる．

$$\frac{\partial}{\partial \theta_i} \pi(a, \theta, s) = \frac{\partial f_1}{\partial \theta_i} \frac{\partial}{\partial f_1} \ln \pi(a, \theta, s) = (y_1 - f_1) x_i \ .$$

同様に f_2や f_3の適正度は以下のようになる．

$$\begin{aligned} \frac{\partial}{\partial f_2} \ln \pi(a, \theta, s) &= \begin{cases} 1/f_2 & a = a_1 \\ -1/(1 - f_2) & a = a_2 \\ 0 & a = a_3 \\ 0 & a = a_4 \end{cases} \\ &= \begin{cases} \frac{y_2 - f_2}{f_2(1 - f_2)} & \text{if } f_2 \text{ is active} \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$



1. 最上位ノードにおいて f_1の確率で 1 を選択 (または $1 - f_1$の確率で 0 を選択)．選択した数字のリンクの方向へ進む．
2. 次のノードにおいても同様に f_2または f_3 の確率で 1 を選択，さもなければ 0 を選択し，選択した数字のリンクの方向へ進む処理を葉ノードにたどり着くまで繰り返す．
3. たどり付いた葉ノードの行動 a を実行
4. 全てのノードについて以下の適正度を計算:

$$\frac{\partial}{\partial f_j} \ln \pi(a, \theta, s) = \begin{cases} \frac{y_j - f_j}{f_j(1-f_j)} & \text{if } f_j \text{ is active} \\ 0 & \text{otherwise,} \end{cases}$$

$y_j \in \{0, 1\}$ はノード f_jの出力を表す．

Figure 5: 行動が 4 個の 2 分木行動選択の処理例と確率的政策表現．f_iは確率分布関数を表す．この処理の項目 $1, 2, 3$ を Fig.3 の項目 1 へ適用し，項目 4 を Fig.3 の項目 4 における適正度の計算に適用する．

よって f_2の政策パラメータについての適正度は

$$\frac{\partial f_2}{\partial \theta_i} \frac{\partial}{\partial f_2} \ln \pi(a, \theta, s) = \begin{cases} (y_2 - f_2) x_i & \text{if } f_2 \text{ is active} \\ 0 & \text{otherwise.} \end{cases}$$

以上より，各ノードの計算処理はローカルな情報だけで実行できる．2 分木のルートノードより葉ノードまでの選択されたパス上の適正度は単純に $(y_j - f_j)x_i$ で与えられ，それ以外のノードでは 0 になる．この性質は木のサイズが変わっても全く同じである．必要なメモリ容量 (ノード数) は 行動数-1 に等しい．

4 実験

4.1 実験設定

エージェントは各関節毎に 8 種類 (または 16 種類) の目標値のうち 1 つを選択するため，2 関節で $8 \times 8 = 64$ コ (または $16 \times 16 = 256$) の行動を扱うことになる．フラットな行動選択では，これら 64 コ (または 256) の行動を区別なく選択する．2 分木の行動選択を行うアルゴリズムでは，ルートノードにおいて関節 1 について上半分または下半分の行動集合どちらかを選択し，第 2 層では関節 2 について上半分または下半分の行動集合を選択する．第 3 層では再び関節 1 について第 1 層で選択した集合のうち上半分または下半分を選択し，第 4 層では関節 2 について同様に選択するといった具合に 2 分木の深さに対して行動の次元を交互に対応させる．状態入力は 2 次元の連続値として $[0, 1]$ に正規化し，CMAC[8] によって 5×5 のタイルを 5 枚重ねた 125 個の基底によって特徴ベクトルを生成した．

提案手法を既存のフラットな行動選択方法と比較するため，Actor-critic における代表的な離散的行動選択方法である Boltzmann 選択 [8] を用いた，提案手法で用いるのと同じ n 次元状態観測ベクトルが与えられ，m 個の行動がある場合，actor では $n \times m$ 個の政策パラメータ変数θ_{ji}(ただし $j = 1 \cdots m$) およびその適正度の履歴を保持する $n \times m$ 個の変数$\overline{e_{\pi}}_{ji}$ を用意する．ある行動 a_kを以下の確率で選択する：

$$\Pr(a_k|s) = \frac{\exp\left(\sum_{i=1}^{n} x_i \theta_{ki}\right)}{\sum_{j=1}^{m} \exp\left(\sum_{i=1}^{n} x_i \theta_{ji}\right)} \qquad (6)$$

行動 a_kを実行した場合の政策パラメータの適正度は以下のように計算する [7]：

$$e_{\pi_{ji}}(t) = \begin{cases} -\Pr(a_j|s)\, x_i & \text{if } a_j \neq a_k, \\ (1 - \Pr(a_j|s))\, x_i & \text{if } a_j = a_k. \end{cases}$$

これらの処理を Fig.3中の actor の部分と置き換えるだけでフラットな行動選択の actor-critic になる．

4.2 実験結果

移動距離を計測するため 1 回転 200 パルスのロータリーエンコーダに直径 5cm の車輪を付け，パルス個数を報酬の絶対値，回転方向を報酬の符号として計測した．報酬 1000 パルスは約 80cm の距離になる．実験は 3000 ステップの学習を行った．実時間で約 8.5 分である．本実験では割引率$\gamma = 0.9$ に設定した．

学習によってエージェントが得た報酬の累積，すなわち初期位置からロボットが移動するようすを Fig.6, 7, 8, 9に示す．ここで注目すべき点は，状態空間が $8 \times 8 = 64$，行動空間が $8 \times 8 = 64$ より全状態-行動ペアが 4092 組あるにもかかわらず，2 分木行動選択ではその半分以下のステップ数でほぼ行動を獲得している点である．フラット行動選択の場合には，学習アルゴリズムやパラメータが同じであるにもかかわ

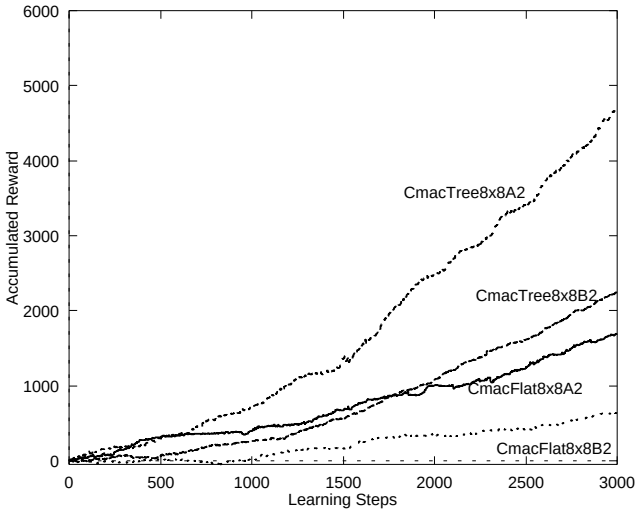


Figure 6: 学習によるロボット A，B の移動距離の変化．行動数 8×8, 学習率$\alpha_v = 0.1$，$\alpha_p = 0.1$, 割引率 $\gamma = 0.9$, $\lambda_v = 0.9$, $\lambda_p = 0.9$ ．CmacFlat8x8A2 はフラット行動選択をロボット A に適用，CmacFlat8x8B2 はフラット行動選択をロボット B に適用，CmacTree8x8A2 は 2 分木行動選択をロボット A に適用，CmacTree8x8B2 は 2 分木行動選択をロボット B に適用した結果である．

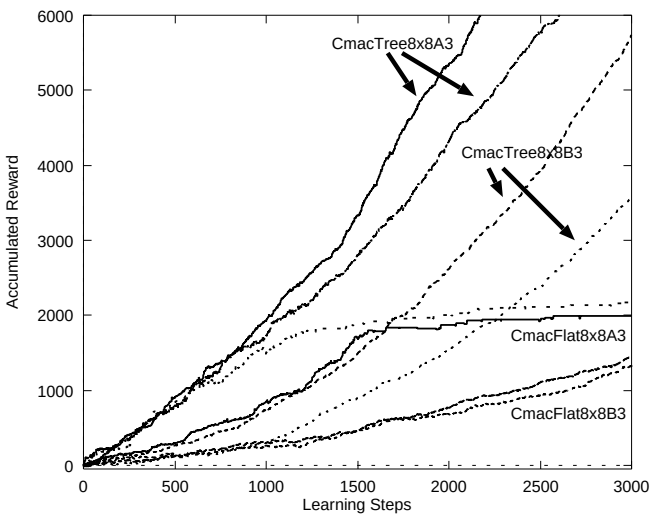


Figure 7: 学習によるロボット A，B の移動距離の変化．行動数 8×8, 学習率$\alpha_v = 0.1$，$\alpha_p = 0.2$, 割引率 $\gamma = 0.9$, $\lambda_v = 0.9$, $\lambda_p = 0.9$ ．CmacFlat8x8A3 はフラット行動選択をロボット A に適用，CmacFlat8x8B3 はフラット行動選択をロボット B に適用，CmacTree8x8A3 は 2 分木行動選択をロボット A に適用，CmacTree8x8B3 は 2 分木行動選択をロボット B に適用した結果である．

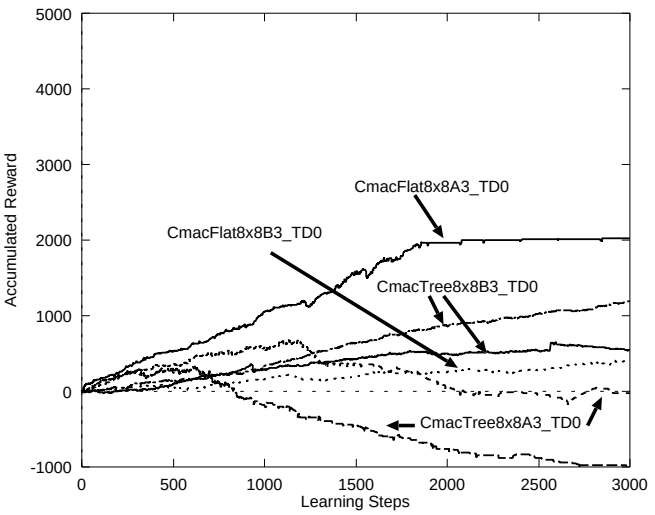


Figure 8: 全手法において actor と critic 共に適正度の履歴を使わない場合の学習によるロボット A，B の移動距離の変化．行動数 8×8, 学習率$\alpha_v = 0.1$，$\alpha_p = 0.2$, 割引率$\gamma = 0.9$, $\lambda_v = 0$, $\lambda_p = 0$

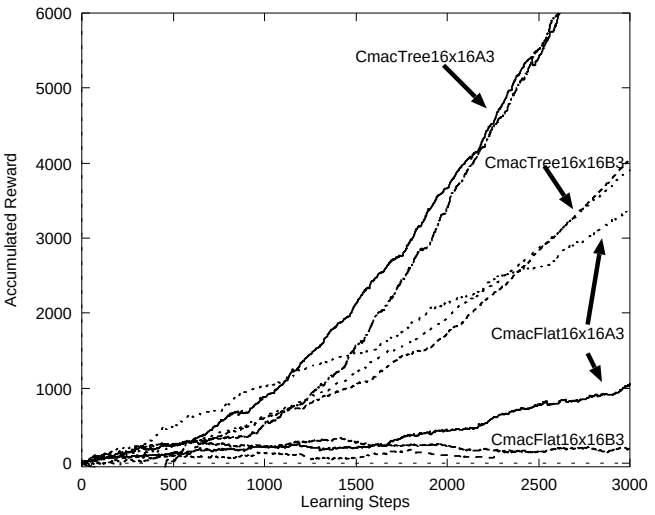


Figure 9: 行動数 16×16 に増加した場合における学習によるロボット A，B の移動距離の変化．学習率 $\alpha_v = 0.1$，$\alpha_p = 0.2$, 割引率$\gamma = 0.9$, $\lambda_v = 0.9$, $\lambda_p = 0.9$ ．CmacFlat16x16A3 はフラット行動選択をロボット A に適用，CmacFlat16x16B3 はフラット行動選択をロボット B に適用，CmacTree16x16A3 は 2 分木行動選択をロボット A に適用，CmacTree16x16B3 は 2 分木行動選択をロボット B に適用した結果である．

らず，学習が半分程度しか進まないか，途中で局所解に捕まってしまい，前進しなくなるかのどちらかである．Fig.6と 7では学習率が異なるだけだが，2 分木行動選択ではどちらでも安定して学習しているのに対し，フラット行動選択の場合には，Fig.7のように学習に失敗するなどパラメータに敏感である．

Fig.8は，2 分木行動選択だが，actor と critic 共に適正度の履歴を用いない場合である．グラフより，ほとんど前進できない局所解に陥っていることが分かる．Fig.7との設定の差異は適正度の履歴の有無だけであることより，環境の非マルコフ性が影響していると考えられる．特に全手法がロボット A において著しく性能劣化している．よって Q-learning 等の非マルコフ性に弱いアルゴリズムでは学習できないと考えられる．

行動が Fig.7の 8×8 から Fig.9の 16×16 へ増加した場合，木構造の actor-critic では性能があまり変化しないのに対し，フラットな actor-critic では，ロボットのタイプや試行によって性能に大きなばらつきが生じ，場合によってはほとんど学習が進まなくなる様子が観察できる．

4.3 実験結果のまとめ

- Actor-Critic アルゴリズムにおいて，2 分木構造の行動選択は，フラットな行動選択よりも学習率や行動の個数などのパラメータ変化に対して頑健である．
- 適正度の履歴を用いない場合は，2 分木とフラットのどちらの行動選択方法でも学習が進まない，または局所解へ陥った．問題の非マルコフ性が影響していると考えられる．

5 おわりに

本論文では数十種類の多数の行動の扱いが必要な 2 自由度ロボットの強化学習問題をとり上げた．2 分木構造の行動選択を行う actor-critic アルゴリズムを適用し，フラットな行動選択を行った場合との比較を行った．2 分木構造の行動選択では，eligibility の計算は各ノードでローカルに極めて簡単に行ことができ，BP のような面倒な情報の伝搬が不要なので，簡単に規模の拡大を行える．また，フラットな行動選択よりも，学習率や行動の個数などのパラメータ変化に対して頑健であることを示した．

関連研究として連続値の行動を扱う actor-critic を用いる階層的な強化学習として文献 [6] がある．今後はさらに高次元の行動出力を行う場合の実装方法についてこれらとの比較検討を行う．

References

[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] 深尾 隆則，稲山 典克，足立 紀彦: 正則化理論を用いた連続的状態と行動を扱う強化学習，システム制御情報学会論文誌，Vol.11, No.11, pp.593–599 (1998).

[3] 堀内 匡，藤野 昭典，片井 修，椹木 哲夫：連続値入出力を扱うファジィ内挿型 Q-learning の提案，計測自動制御学会論文集, Vol.35, No.2, pp.271–279 (1999).

[4] 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).

[5] 木村 元，小林 重信: 確率的 2 分木の行動選択を用いた強化学習による多数の行動の扱いについて，計測自動制御学会 第 27 回知能システムシンポジウム，pp.111–116 (2000).

[6] 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).

[7] Peshkin, L. & Meuleau, N. & Kaelbling, L. P.: Learning Policies with External Memory, *Proceedings of the 16th International Conference on Machine Learning*, pp.307–314 (1999).

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

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