計測自動制御学会論文集
Vol.33, No.1, 1/9 (1997)

確率的２分木の行動選択を用いたActor-Criticアルゴリズム†

—多数の行動を扱う強化学習—

木村　元*・小林重信*

An Actor-Critic Algorithm using a Binary Tree Action Selector

—Reinforcement Learning to Cope with Enormous Actions—

Hajime KIMURA* and Shigenobu KOBAYASHI*

In real world applications, learning algorithms often have to handle several dozens of actions, which have some distance metrics. Epsilon-greedy or Boltzmann distribution exploration strategies, which have been applied for Q-learning or SARSA, are very popular, simple and effective in the problems that have a few actions, however, the efficiency would decrease when the number of actions is increased. We propose a policy function representation that consists of a stochastic binary decision tree, and we apply it to an actor-critic algorithm for the problems that have enormous similar actions. Simulation results show the increase of the actions does not affect learning curves of the proposed method at all.

Key Words: reinforcement learning, actor-critic, action selector, binary tree, exploration strategy

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[4]．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．代表的な強化学習法であるQ-learning[11]やSARSA[8]は離散的な状態・行動を対象としている．そのため連続な状態空間や行動空間における状態評価関数 (state value function) や状態-行動評価関数 (state-action value function) を表現するために関数近似としてCMACを用いる方法[8]，ファジィを用いる方法[3]，過去の経験を用いて補間する方法[2]などが提案されている．しかし多数の行動を扱うためには，状態 (-行動) 評価関数の表現を工夫するだけでなく，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．一般によく用いられるε-greedy戦略やボルツマン選択戦略は，仕組みが単純なわりに，行動数が少ない場合は強力である．ところがこれらは行動数の増加に比例して学習時間が増加したり，パラメータのスケジューリングを行動数に応じて適切に設定する必要があるという問題がある．これらは，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間に距離を定義できる，つまり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る．そこで本論文では数十以上の多数の類似した行動を効率良く扱うために階層的行動選択を行う強化学習方式を提案する．提案手法はパラメータ化された確率的政策関数表現として２分木構造を導入し，actor-criticアルゴリズム[1)5)8)]によって政策パラメータを更新する．本手法は行動集合間の半順序関係についての知識を木構造の階層に反映させることにより，学習の効率化を図る．ただし，本論文では階層構造は設計者が与える．実験により，行動空間に位相構造が存在する環境では行動の個数が増加しても提案手法の学習性能が低下しにくいことを示す．

2. 問題設定

2.1 マルコフ決定過程 (MDP)

状態空間を$\mathcal{S}$，行動空間を$\mathcal{A}$，上界と下界を持つ実数の集合を$\mathcal{R}$と表す．各時刻tでエージェントは状態観測$s_t \in \mathcal{S}$に

† 第27回SICE知能システムシンポジウムにて発表 (2000・3)

* 東京工業大学大学院総合理工学研究科
* Interdisciplinary Graduate School of Sci. and Eng. Tokyo Institute of Technology

(Received January 1, 1997)
(Revised January 1, 1997)

基づいて行動 $a_t \in \mathcal{A}$ を実行し，状態遷移に伴う報酬 $r_t \in \mathcal{R}$ を得る．多くの強化学習法が環境のモデルとして仮定するマルコフ決定過程 (MDP) では，一般に次の状態や報酬は確率的で，その分布は s_t と a_t にのみ依存する．MDP では次の状態 s_{t+1} は遷移確率 $T(s_t, a, s_{t+1})$ に従って決まり，報酬 r_t も期待値 $r(s_t, a)$ によって与えられる．エージェントは予め $T(s_t, a, s_{t+1})$ や $r(s_t, a)$ についての知識は持っていない．強化学習の目的はエージェントの挙動を最適化する政策を得ることである．無限期間のタスクにおける自然な評価規範として，以下のような割引報酬の合計がある．

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

割引率 $0 \leq \gamma \leq 1$ は未来に得るであろう報酬の現時点での重要度を表し，V_t は時刻 t の評価値 (value) を表す．MDP では評価関数は以下に定義される．

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

ただし $E\{\cdot\}$ は期待値を表す．MDP における学習の目的は，各状態 s において式 2 で定義される評価値を最大化するような最適政策を見つけることである．

2.2 多数の類似行動を有する環境

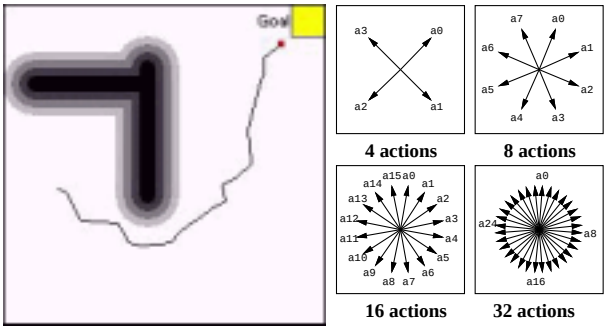


Fig. 1 A modified puddle-world problem. There are 4, 8, 16, 32, or 64 actions, which moves approximately 0.05 in these directions. The state space is continuous and two-dimensional, which is bounded by $[0, 1]$ for each dimension.

多くの実問題では，隣り合う行動の多くは類似した状態遷移となる性質を有している．**Fig. 1** は puddle world と呼ばれる経路探索問題[7] を示す．状態空間は 2 次元の連続値で各軸について $[0, 1]$ の範囲である．エージェントは現在位置の座標を状態として観測する．エージェントが行動を選択すると，上記の範囲を超えない限り選択された方向へおよそ 0.05 の距離を移動する．移動の際，標準偏差 0.01 のガウス分布のノイズが各軸方向に付加される．行動の種類が増加すると隣接する行動が類似してくることは Fig. 1 より明らかである．各時間ステップにおいて -1 の報酬に加えて，水たまり (puddle) に入ることでさらに負の報酬が与えられる．この水たまりによる負の報酬は，水たまり内のエージェント位置と，水たまり外周との最短距離に -400 を掛けた値である．水たまりは半径 0.1 で中心座標は $(0.1, 0.75)$ から $(0.45, 0.75)$ までおよび $(0.45, 0.4)$ から $(0.45, 0.8)$ までの位置に配置される．各試行 (episode または trial と呼ばれる) における初期状態はゴール領域を除いた位置に一様分布で生成される．ゴールは $[0.9, 0.9]$ から $[1.0, 1.0]$ までの矩形領域である．

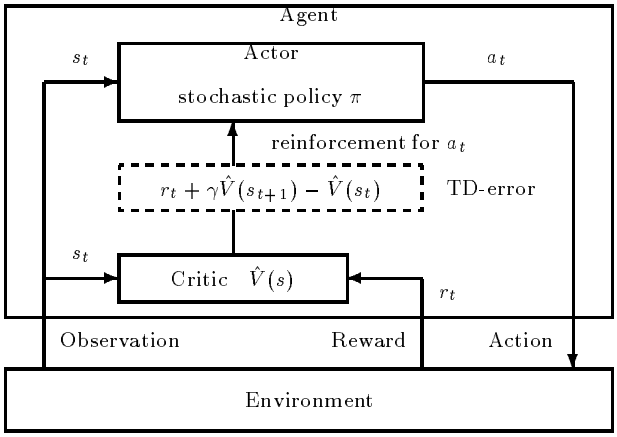


Fig. 2 A standard actor-critic architecture.

3. 2分木構造の確率的政策表現を用いた Actor-Critic 強化学習アルゴリズム

代表的強化学習法である Q-learning や SARSA 等の DP-based または value-based な手法は，行動が離散だが行動間に類似度や距離が存在するような環境においては行動空間の value 関数を汎化することで対処する方法がほとんどだが，環境の非マルコフ性に対して弱く，状態×行動の膨大な空間における value 関数を推定した後でなければ政策を得ることができないなどの問題がある．それに対して actor-critic アルゴリズムは非マルコフ性に対して頑健な実装が簡単であり，政策改善は状態×行動空間の value 関数を推定するよりずっと速く，また critic が機能しなくても学習可能などの優れた特徴を持つアルゴリズムだが，離散行動間に類似度や距離が存在するような環境においてその性質を生かす行動選択方法が無かった．そこで上記の問題に対して有効な actor-critic の実装方法を提案する．

3.1 Actor-Critic アルゴリズム

代表的な強化学習アルゴリズムとしては Q-learning[11] や SARSA[8] が知られているが，本論文では，木構造の特殊な行動決定を行うため，状態–行動の評価値 (Q 値) を用いずに行動選択を行う actor-critic を用いる．この学習アルゴリズムは，Q 値の代わりに行動選択確率を制御する政策パラメータを明示的に保持する．本手法は，行動選択を行う actor と，状態の評価を行う critic の 2 つの部分より構成される (**Fig. 2**)．Actor は状態から行動への写像を確率的政策 (stochastic policy) と呼ばれるパラメータ化された確率分布に従って行動を実行する．Critic は，actor の政策のもとで

(1) Observe state s_t, choose action a_t with probability $\pi(a_t, \theta, s_t)$ in the actor, and perform it.
(2) Observe immediate reward r_t, resulting state s_{t+1}, and calculate the TD-error according to

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

where $0 \leq \gamma \leq 1$ is the discount factor, $\hat{V}(s)$ is an estimated value function by the critic.
(3) Update the estimating value function $\hat{V}(s)$ in the critic according to the TD(λ) method as follows:

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

where e_v denotes the eligibility of the parameter w in the function approximator $\hat{V}(s)$, $\overline{e_v}$ is its trace, and α_v is a learning rate.
(4) Update the actor's stochastic policy according to

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

where e_π is the eligibility of the policy parameter θ, $\overline{e_\pi}$ is its trace, and α_π is a learning rate.
(5) Discount the eligibility traces as follows:

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

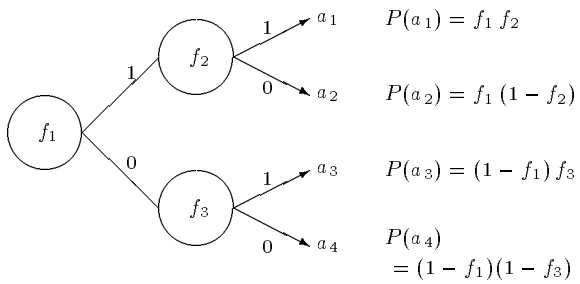
where λ_v and λ_π $(0 \leq \lambda_v, \lambda_\pi \leq 1)$ are discount factors in the critic and the actor respectively.
(6) Let $t \leftarrow t+1$, and go to step 1.

Fig. 3 An actor-critic algorithm making use of eligibility traces in both the actor and the critic.

の状態評価値 $\hat{V}(s)$ を推定する．Actor は，環境から直接受け取る報酬 r_t および critic で推定される評価値 $\hat{V}(s)$ より計算される TD エラーを手がかりに政策を改善する．行動を実行して状態遷移した結果，TD エラーが正ならば，実行した行動（あるいは行動の時系列）によって良い状態へ遷移したと考えられるので，その行動（あるいは行動の時系列）の選択確率を高める．逆に TD エラーが負ならば，悪い状態へ遷移したと考えられるので，その行動（あるいは行動の時系列）の選択確率を低めるよう更新することで政策改善を行うものである．

Fig. 3 は本論文で用いた actor-critic の詳細を示す[5]．Critic における処理手順は Fig. 3 のステップ 3,4,5 に示されている．$\hat{V}(s)$ はパラメータ w による線形関数として表される．実装方法についての詳細は 3.3 節にて説明する．パラメータ λ_v は TD($\lambda = \lambda_v$) の適正度の履歴を表す．政策は actor においてパラメータ化された関数として明示的に表され，式 1 の評価関数 V_t あるいは $\hat{V}(s)$ の勾配を登る方向へ政策パラメータを更新する[5) 9)]．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．パラメータ θ はエージェントの政策パラメータベクトルを表す．エージェントは θ を調節することにより確率的政策 π を変える．$\pi(a, \theta, s)$ の関数形については，エージェントに実装できる計算資源の制限など，一般に個別の問題ごとに制約が存在するが，本論文では 2 分木構造の行動選択を行う政策関数を用いる．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 REINFORCE algorithm[12] で定義されているものと同一で，選択した行動が政策パラメータあるいは行動選択確率に与えるインパクトの大きさを表す．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) の場合とはやや異なる．λ_π が 0 に近い場合，critic が学習している $\hat{V}$ の勾配により政策が改善され，λ_π が 1 に近い場合は式 1 で定義される実際の利得の勾配によって政策が改善される[5]．

3.2　2 分木構造の確率的行動選択



(1) Start at the root of the tree, choose 1 with probability f_1 (or 0 with probability $1 - f_1$), and take the corresponding branch.
(2) At the selected branch node, continue the same process until a leaf is encountered. If the node f_2 is selected, choose 1 with probability f_2 (or 0 with probability $1 - f_2$). If the selected node is f_3, then choose 1 with probability f_3, etc. And take the corresponding branch.
(3) At the resulting leaf node, execute the corresponding action a.
(4) Calculate eligibilities for all nodes by

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

where $y_j \in \{0, 1\}$ is the outcome of the node f_j.

Fig. 4 A binary-tree action selection scheme for four actions, and its policy representation. f_i denotes a probability distribution function.

Fig. 4 は 2 分木構造の行動選択方法について，行動が 4 個の場合の例を示す．葉ノードはそれぞれ行動に対応する．それ以外の各ノードでは確率 f_i に応じて確率的に分岐する．この確率的政策関数は以下のように記述される．

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

このとき f_1 の適正度は以下の式で与えられる：

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

ただし $y_1\in\{0,1\}$ はノード f_1 が選択した値を表す．同様に f_2 や f_3 の適正度は以下のようになる．

$$\frac{\partial}{\partial f_2}\ln\pi(a,\theta,s) = \begin{cases} 1/f_2 & a=a_1\\ -1/(1-f_2) & a=a_2\\ 0 & a=a_3\\ 0 & a=a_4 \end{cases}$$
$$= \begin{cases} \frac{y_2-f_2}{f_2(1-f_2)} & \text{if } f_2 \text{ is active}\\ 0 & \text{otherwise.} \end{cases} \qquad (6)$$

以上より，各ノードの計算処理は全く同じで，しかも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 $\frac{y_j-f_j}{f_j(1-f_j)}$ 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 **Fig. 5** に示すように(行動数) − 1に等しい．

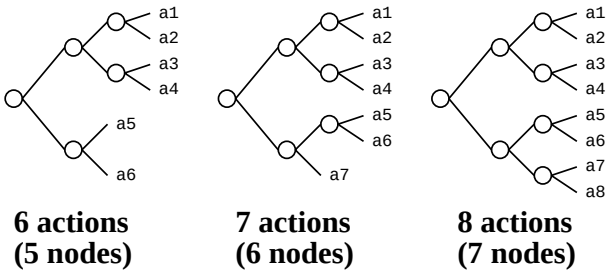


Fig. 5 The binary trees for 6, 7 and 8 actions.

3.3 2分木構造の Actor を用いた Actor-Critic

本節では 3.2 節の2分木構造の行動選択器を 3.1 節の actor-critic の actor へ適用した実装例の詳細について述べる．まず，エージェントは現在の状態 s を特徴ベクトル $(x_1,x_2,\cdots,x_i,\cdots,x_n)$ によって観測するものとする．ただし時刻 t の特徴ベクトルであることを強調するときは $(x_1(t),x_2(t),\cdots,x_i(t),\cdots,x_n(t))$ のように t の関数として記述する．通常，センサ入力などの刺激からどのように状態表現を生成すべきかという問題は学習対象として扱われることが多いが，本論文では対象外とし，状態表現は特徴ベクトルとしてトップダウンに与えられるものとする．Critic では状態評価関数をパラメータ w_i と状態特徴ベクトルの要素 x_i との線形結合によって以下のように近似表現する：

$$\hat{V}(s)=\sum_{i=1}^{n} w_i\,x_i\ . \qquad (7)$$

上記の近似表現は線形アーキテクチャ(linear architecture)と呼ばれ，TD 法により，ある条件のもとで二乗誤差最小の推定値への収束が証明されている[10]．パラメータ w_i の適正度 e_{vi} は $(\partial\hat{V}(s))/(\partial w_i)=x_i$ で与えられる．

Actor における2分木行動選択の各ノードの分岐確率 f_j も状態特徴ベクトルを用いて以下のように表す：

$$f_j=\frac{1}{1+\exp(-\sum_{i=1}^{n}\theta_{ji}x_i)}\ , \qquad (8)$$

ただし θ_{ji} は政策パラメータである．この関数はシグモイド関数で，値域が 0 から 1 までの単調増加関数である．θ_{ji} に関する適正度 $e_{\pi ji}$ は式 5, 6 および式 8 より $\partial f_j/\partial\theta_{ji}=x_i\,f_j(1-f_j)$ であることから以下で与えられる：

$$e_{\pi ji}=\frac{\partial f_j}{\partial\theta_{ji}}\frac{\partial}{\partial f_j}\ln\pi(a,\theta,s)$$
$$=\begin{cases} (y_j-f_j)x_i & \text{if } f_j \text{ is active}\\ 0 & \text{otherwise.} \end{cases} \qquad (9)$$

ただし $y_j\in\{0,1\}$ はノード f_j が選択した値を表す．

以上より，状態特徴ベクトルの要素 n 個，行動数 m 個の場合における提案手法の詳細は以下のようになる．

(1) 初期化：Critic の n 個の変数 w_i (ただし $i=1\cdots n$) およびその適正度の履歴を保持する n 個の変数 $\overline{e_{vi}}$，actor の $n(m-1)$ 個の政策パラメータ変数 θ_{ji}(ただし $j=1\cdots m-1$) およびその適正度の履歴を保持する $n(m-1)$ 個の変数 $\overline{e_{\pi ji}}$ を用意して値を初期化する．

(2) 状態 s_t の特徴ベクトル $(x_1(t),x_2(t)\cdots x_n(t))$ を観測

(3) Actor の2分木で行動選択

(a) ルートノード $j=1$ より以下の計算を始める：

(b) ノード f_j において以下を計算する：

$$f_j=\frac{1}{1+\exp(-\sum_{i=1}^{n}\theta_{ji}x_i(t))}$$

f_j の確率で 1 を選択 (または $1-f_j$ の確率で 0 を選択)．選択した数字のリンクの方向へ進む．

(c) 進んだ先のノードで同様の処理を繰り返す．

(d) 最終的にたどり付いた葉ノードの行動 a を実行

(e) 全ノード j について以下の適正度を計算：

$$e_{\pi ji}(t)=\begin{cases} (y_j-f_j)x_i(t) & \text{if } f_j \text{ is active}\\ 0 & \text{otherwise,} \end{cases}$$

ただし $y_j\in\{0,1\}$ はノード f_j の出力を表す．

(4) 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．

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

$$= r_t + \gamma \left(\sum_{i=1}^{n} w_i\, x_i(t+1) \right) - \sum_{i=1}^{n} w_i\, x_i(t)$$

（5） Critic の $\hat{V}(s)$ を以下のように更新する．

$$\begin{aligned} e_{vi}(t) &= x_i(t)\ , \\ \overline{e_{vi}}(t) &\leftarrow e_{vi}(t) + \overline{e_{vi}}(t)\ , \\ \Delta w_i(t) &= (\text{TD-error})\, \overline{e_{vi}}(t)\ , \\ w_i &\leftarrow w_i + \alpha_v\, \Delta w_i(t)\ , \end{aligned}$$

（6） 適正度 $e_{\pi_{ji}}(t)$ で actor の行動選択確率を更新する

$$\begin{aligned} \overline{e_{\pi ji}}(t) &\leftarrow e_{\pi_{ji}}(t) + \overline{e_{\pi ji}}(t)\ , \\ \Delta\theta_{ji}(t) &= (\text{TD-error})\, \overline{e_{\pi ji}}(t)\ , \\ \theta_{ji} &\leftarrow \theta_{ji} + \alpha_\pi\, \Delta\theta_{ji}(t)\ , \end{aligned}$$

（7） 適正度の履歴を以下のように割り引く．

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

（8） 時刻を $t \leftarrow t+1$ に進めてステップ 2 へ戻る．

3.4　提案手法の特徴

行動の選択は行動数 m に対して $\log_2 m$ 回の二者択一によって行われる．選択のための計算や適正度の計算は，木構造上で選択されたパス上だけで，個々のノードのローカルな値を用いた単純計算で実行される．よって大規模問題への適用に有効である．Actor-critic アルゴリズムを用いているため，ある程度の非マルコフ性を有する環境でも Q-learning ほどの影響を受けずに学習が可能である [5]．

3.5　既存手法および関連研究

Actor-critic では離散的な行動選択として Boltzmann 選択が提案されている [8]．3.3 節の設定と同様の n 次元状態特徴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：

$$\Pr(a_k|s) = \frac{\exp\left(\sum_{i=1}^{n} x_i \theta_{ki}\right)}{\sum_{j=1}^{m} \exp\left(\sum_{i=1}^{n} x_i \theta_{ji}\right)} \qquad (10)$$

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 [6]：

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

これらの処理を 3.3 節中の 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．

Q-learning と SARSA(λ) では，状態-行動評価値を $Q(s,a) = \sum_{i=1}^{n} x_i w_{ai}$ と表す．行動選択戦略としては Boltzmann 選択の他，ϵ-greedy 選択がよく用いられる．これは確率 ϵ でランダムに行動を選び，残りの確率で最も大きな Q 値を持つ行動を選ぶ方法である．時刻 t の状態を s_t，その特徴量を $x_i(t)$，そのとき選択した行動を a_t，状態遷移後の状態と特徴量を s_{t+1}, $x_i(t+1)$，時刻 $t+1$ で選んだ行動を a_{t+1} とすると，Q-learning の学習則は

$$\begin{aligned} \Delta w_{a_t i} &= x_i(t) \left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \\ w_{a_t i} &\leftarrow w_{a_t i} + \alpha \Delta w_{a_t i}\ , \end{aligned}$$

これを全ての $i \in \{1, 2, \cdots n\}$ について行う．α は学習率．SARSA(λ) の学習則は以下のとおり

$$\begin{aligned} (\text{TD-error}) &= r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \\ \overline{e_{a_t i}}(t) &= x_i(t) + \overline{e_{a_t i}}(t) \\ \Delta w_{ai} &= \overline{e_{ai}}(t)\, (\text{TD-error}) \quad \text{for all } a \\ w_{ai} &\leftarrow w_{ai} + \alpha \Delta w_{ai} \quad \text{for all } a \\ \overline{e_{ai}}(t+1) &= \gamma\, \lambda\, \overline{e_{ai}}(t) \quad \text{for all } a, \end{aligned}$$

これを全ての $i \in \{1, 2, \cdots n\}$ について行う．$\overline{e_{ai}}$ は適正度の履歴である．Q-learning では $Q(s,a)$ を近似するため変数 w_i を $n \times m$ 個要する．SARSA(λ) では変数 w_i とその履歴 $\overline{e_{ai}}$ のために $2 \times n \times m$ 個の変数を要する．SARSA が消費するメモリーは提案手法と全く同じである．Q-learning や SARSA では，本論文のように大量の行動を扱う場合には，実行した行動 a_t について $Q(s_t, a_t)$ を更新するだけでなく，a_t の近傍の行動についても同様の更新を行う方法がある [8]．

4.　計算機シミュレーション実験

行動数の増加がアルゴリズムに与える影響を調べる．Fig. 1 の水たまり問題において提案手法，フラットな行動選択を行う actor-critic，ϵ-greedy 戦略を用いる Q-learning および SARSA(λ) を比較する．状態入力から特徴ベクトル $(x_1, x_2, \cdots x_{140})$ を生成するためタイルコーディング [8] を用いる．2 次元状態入力から広域および局所的特徴量を含む特徴ベクトルを生成するため **Fig. 6** のように 140 コの特徴量を生成するタイルコーディングを用いた．状態空間全部をカバーする 1×1 から 7×7 まで分割したタイルを用いるため，合計 $1^2 + 2^2 + 3^2 + \cdots + 7^2 = 140$ コの特徴量となる．個々のタイルは特徴ベクトル $(x_1, x_2, \cdots x_{140})$ の各要素に対応する．状態入力がある $m \times m$ 分割のタイル内にある場合，そのタイルは発火し，対応する特徴量ベクトルの要素は $m^2/140$ の値をとり，それ以外の要素はゼロである．

割引率は $\gamma = 1$，提案手法の $\alpha_v = 0.1$，$\alpha_\pi = 0.01$，$\lambda_v = \lambda_\pi = 0.9$ とした．Q-learning と SARSA(λ) の探査パラメータ $\epsilon = 0.1$，Q-learning の学習率 $\alpha = 0.5$，SARSA(λ) の学習率 $\alpha = 0.1$，$\lambda = 0.9$ とした．

Figs. 7，**8**，**11**，**12** は各手法の学習結果を示す (割引率 $\gamma = 1$)．各グラフの縦軸 (cost) はスタートからゴール領域までのコスト (負の報酬) を表し，横軸 (trial) はスタートからゴール領域に到達するまでを 1 trial として表している．全ての手法において学習の進行すなわち trial の増加にともなってコストが減少している．特に提案手法では，行動の個数が増加しても，性能がほとんど変化しない．一方フラットな行動選択を用いた actor-critic，Q-learning および

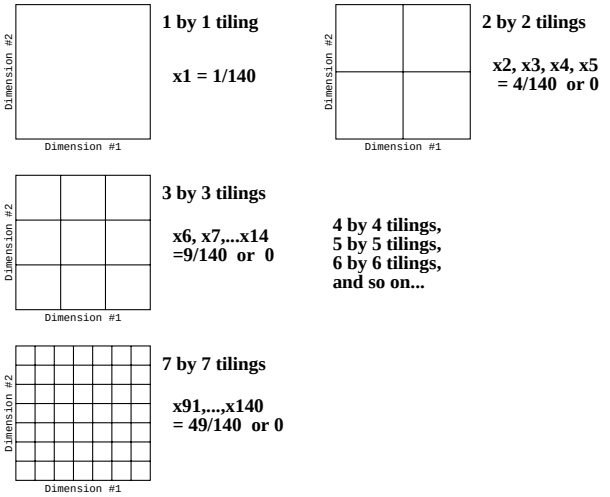


Fig. 6 A tile coding method to generate a feature vector, which contains all features from broad to narrow over a continuous two-dimensional state space. Any state is in exactly one tile of each tiling. Each tile is associated with one element of the feature vector $(x_1, x_2, \cdots x_{140})$. The number 140 is come from $1^2 + 2^2 + 3^2 + \cdots + 7^2$. When a state input is in some certain tiles, the corresponding tiles are activated and their elements are set to some positive value (e.g., 1×1 tiling is 1/140, etc.). The other elements are set to zero.

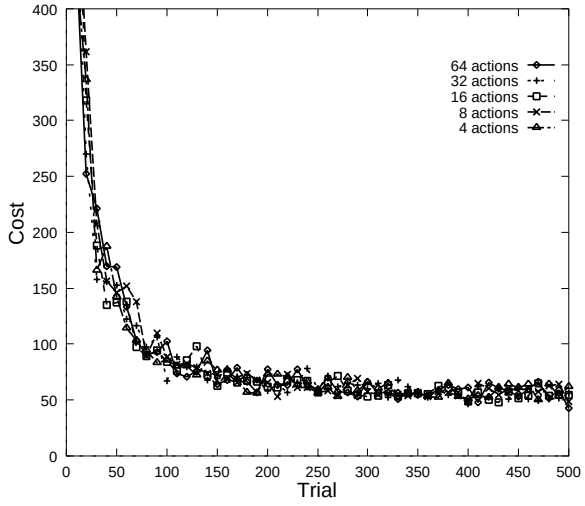


Fig. 7 Performance on the actor-critic method using the binary-tree action selector, averaged over 100 trials. $\alpha_v = 0.1$, $\alpha_\pi = 0.01$, $\lambda_v = 0.9$, $\lambda_\pi = 0.9$.

SARSA では，行動の個数によって性能が大きく影響を受ける．**Fig. 10** は行動が **Fig. 9** に示すようにランダムに配置されている場合の 2 分木行動選択の学習のようすを示す．行動がランダムに配置されていても，この問題では 2 分木行動選択は少なくともフラット選択と同等の性能を示した．

5. ロボットへの適用

Fig. 13 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

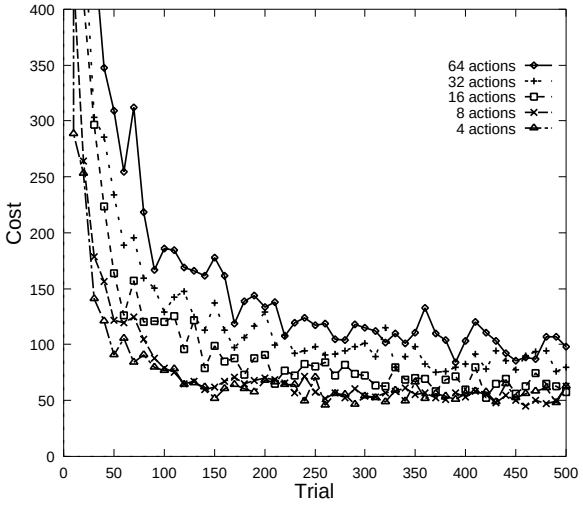


Fig. 8 Performance on the actor-critic method using the flat action selector, averaged over 100 trials. $\alpha_v = 0.1$, $\alpha_\pi = 0.01$, $\lambda_v = 0.9$, $\lambda_\pi = 0.9$.

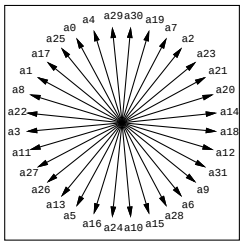


Fig. 9 An example which are randomly labeled actions in the puddle world.

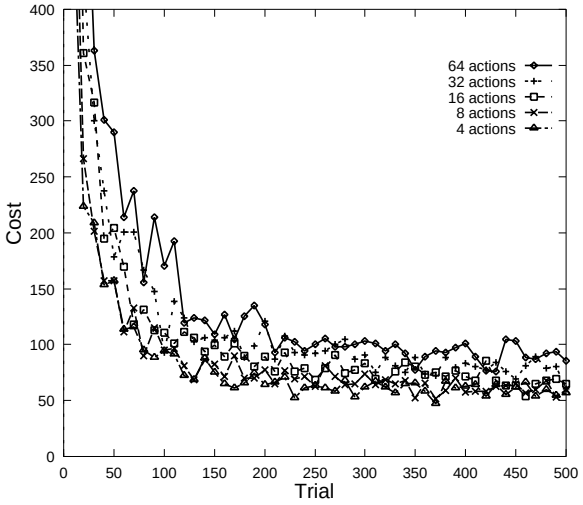


Fig. 10 Performance on the actor-critic method using the binary-tree action selector in the puddle world with randomly labeled actions, averaged over 100 trials. $\alpha_v = 0.1$, $\alpha_\pi = 0.01$, $\lambda_v = 0.9$, $\lambda_\pi = 0.9$.

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

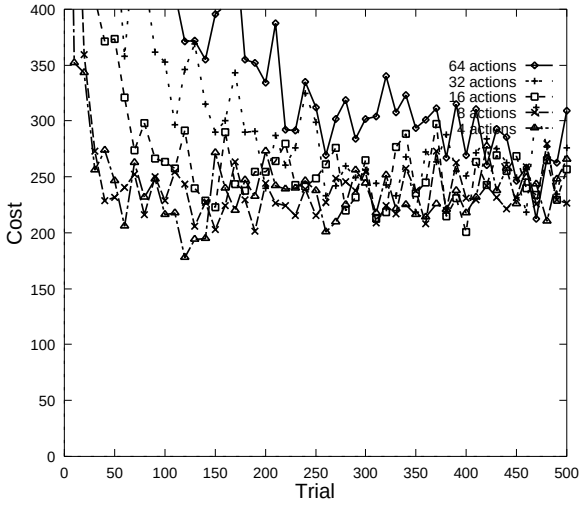


Fig. 11 Performance on Q-learning using ϵ-greedy policy averaged over 100 trials. $\epsilon = 0.1$, learning rate $= 0.5$.

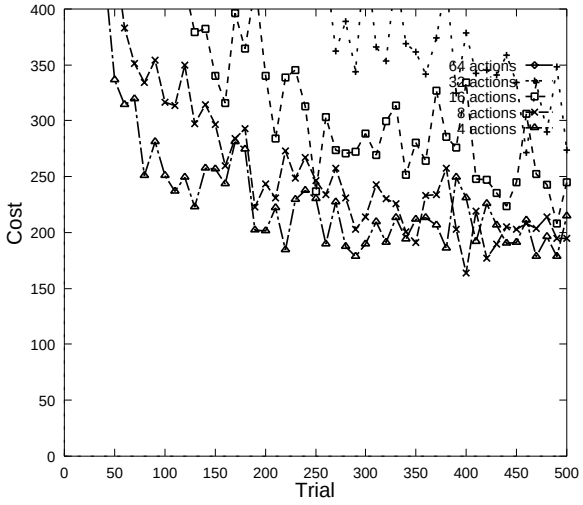


Fig. 12 Performance on SARSA(λ) using ϵ-greedy policy averaged over 100 trials. $\epsilon = 0.1$, learning rate $= 0.1$, $\lambda = 0.9$. Note that this algorithm costs the memory exactly the same size as our method's.

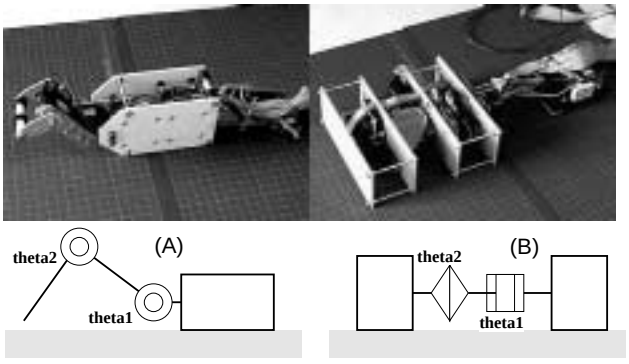


Fig. 13 Real two robots and their models. The Robot-A has a two-link arm. The Robot-B consists two boxes with an actuated joint which bends and twists the body.

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） エージェントは状態観測としてロボットの関節の角度 θ_1, θ_2 を受け取る．

（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

（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，その値を報酬としてエージェントに与える．

（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 までの整数値をとる．報酬の値は -128〜127 の範囲の整数値をとり，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．

エージェントは各関節毎に 8 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．もっと良い実装方法があるかもしれないが，この実装は，２次元の行動空間を木構造に反映させる最も単純な方法の一例である．状態入力は水たまり問題と同じ 2 次元の連続値なので，本実験でも同様に Fig. 6 に示すタイルコーディングで特徴ベクトルを生成した．

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 3cm 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 50cm 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．本実験は水たまり問題とは異なり，ゴールのような終端状態が存在しないので割引率 $\gamma = 0.9$ に設定した．また，実時間で学習させるため，$\alpha_\pi = 0.1$，と大きめに設定した．それ以外の学習パラメータは水たまり問題と同じ学習率 $\alpha_v = 0.1$，$\lambda_v = 0.9$, $\lambda_\pi = 0.9$ とした．

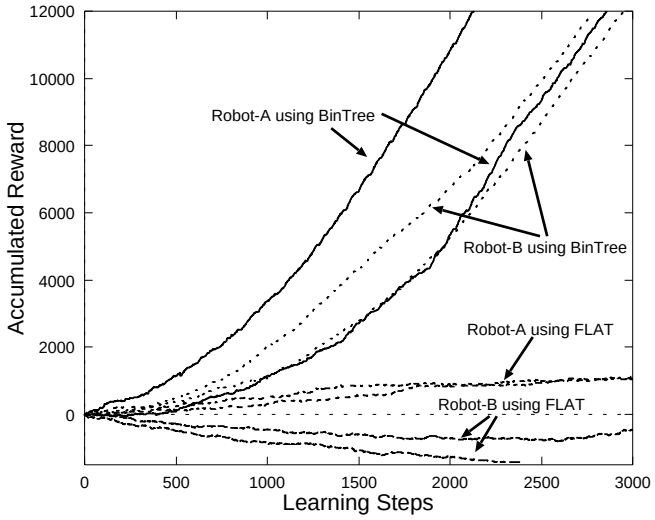


Fig. 14 Performances of actor-critic algorithms on the real robots.

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. 14** に示す．A と B は全く異なるメカニズムであるにもかかわらず，提案手法を用いた場合にはどちらのロボットも順調に学習を行っている．特に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，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．それに対してフラットな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らず，ほとんど学習が進んでいない．行動選択方法のちがいによる性能の差は水たまり問題よりも顕著に現れた．

6. 考察

単純な水たまり問題でも，単に類似した行動を増やすだけで，フラットな行動選択を用いた場合には学習アルゴリズムに関係なく学習性能が行動数の影響を受ける．ところが 2 分木行動選択を用いると，類似した行動が順番に並んでいる場合には行動数の影響をほとんど受けずに学習できる．興味深いのは，行動がランダムに配置された場合でもフラットな行動選択と同等かそれ以上の性能だった点である．これは，類似した行動が多数存在する問題においては，行動をランダムにグループ化しても，階層的な行動選択により効率良く学習できるためと考えられる．階層的な行動選択が苦手とする環境として考えられるのは，あるグループ化された行動集合中において，低い評価値を持つ行動がほとんどの割合を占める中に最適な行動がぽつんと存在するような場合である．行動グループの中に最適な行動が含まれていたとしても，その行動グループの平均評価値が低ければ，上位階層ではその行動グループを選択しなくなるので最適行動を学習できないと考えられる．しかし実問題ではこのように意地の悪い問題はほとんど無いだろう．フラットな行動選択を 2 分木行動選択に置きかえることにより，学習の高速化だけでなく，行動の個数に応じてパラメータをチューニングする手間も省けることが期待できる．

ロボットの実験ではフラットな行動選択と階層的な行動選択との性能の差がさらに顕著に現れた．水たまり問題と大きく異なるのは，行動が 2 次元である点なので，これが影響していると考えられるが，行動次元の増加と学習性能との因果関係についての詳しい解析は今後の課題である．本論文では 1 次元および 2 次元空間に行動が整然と配置されている環境において，提案手法が効率良く機能することを示した．行動がさらに多次元になった場合，階層的行動選択ではどの次元から選ぶかによって性能が大きな影響を受けると予想される．各軸で独立の出力を行うような実装も可能であるし，木構造の決定自体も学習対象となりうると考えられる．高次元の行動空間の場合にどのような木構造にすべきかについては今後の課題である．

7. おわりに

本稿では 2 分木による確率的政策表現方法を提案した．本手法は類似の状態遷移をもたらす行動が多数ある環境にて効果を発揮する．行動選択や適正度の計算は単純かつローカルな情報だけで可能であり，大規模問題への拡張が容易である．従来のフラットな行動選択と置き換えることにより，学習性能の向上だけでなく，行動の個数に応じたパラメータチューニングも不要となる．提案手法は単に確率的政策表現方法の一つであり，理論的にも単純なため，actor-critic 以外のアルゴリズムとの組合わせも可能である．

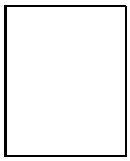
参 考 文 献

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) 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，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) 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).
7) Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coarse coding, *Advances in Neural Information Processing Systems 8 (NIPS8)*, pp.1038–1044 (1996).
8) Sutton, R. S. & Barto, A.: Reinforcement Learning: An Introduction, *A Bradford Book*, The MIT Press (1998).
9) Sutton, R. S., McAllester, D., Singh, S. & Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with

Function Approximation, *Submitted to Advances in Neural Information Processing Systems 12 (NIPS12)*, (2000).
10) Tsitsiklis, J. N., & Roy, B. V.: An Analysis of Temporal-Difference Learning with Function Approximation, IEEE Transactions on Automatic Control, Vol.42, No.5, pp.674–690 (1997).
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).

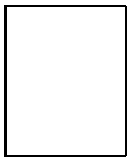
［著　者　紹　介］

木　村　　元（正会員）



1997 年東京工業大学大学院知能科学専攻博士課程修了，同年 4 月日本学術振興会 PD 研究員，1998 年 4 月，東京工業大学大学院総合理工学研究科助手，現在に至る．人工知能，特に強化学習に関する研究に従事．

小　林　重　信（正会員）



1974 年東京工業大学大学院博士課程経営工学専攻修了．同年 4 月，同大学工学部制御工学科助手．1981 年 8 月，同大学大学院総合理工学研究科助教授．1990 年 8 月，教授．現在に至る．問題解決と推論制御，知識獲得と学習などの研究に従事．