回避行動の再利用メカニズムを持つ強化学習手法の提案と多関節ロボットへの応用

○山口明彦 (奈良先端大, ATR-CNS)　杉本徳和 (ATR-CNS)　川人光男 (ATR-CNS, 奈良先端大)

Reinforcement Learning with Reusing Mechanism of Avoidance Actions

—Proposal and its Application to Motion Learning of Multi-Link Robot—

*Akihiko YAMAGUCHI (NAIST, ATR-CNS), Norikazu SUGIMOTO (ATR-CNS),
Mitsuo KAWATO (ATR-CNS, NAIST)

Abstract— Learning complex motions, such as a vaulting horse, requires a large trial number. Reusing the knowledge already learned is an essential mechanism to learn such motions efficiently, like our humans. In this paper, we propose a reusing mechanism for reinforcement learning where the avoidance actions, such as avoiding falling down, are learned separately from primary actions, and they are reused in learning new actions; the simulation results of grid-world demonstrates this method works effective. Furthermore, we apply the suggested method for multi-link robots to learn whole body motions, and evaluate with a tennis-serve task on the simulation.

Key Words: Reusing, reinforcement learning, motion learning

1. 再利用メカニズムの導入による強化学習の性能向上

将来のロボットに不可欠な「目的が与えられただけでロボットが自律的に実現方法を獲得する技術」のひとつである強化学習の課題として，問題が大きくなると，つまり次元が大きくなったり複雑な行動を獲得させようとすると，学習に膨大な時間が掛かってしまうことがあげられる．このためロボットの行動学習に適用されている強化学習の多くは，次元縮約のためにモデルを用いたり [1] 階層構造を導入したり [2, 3, 4] することによってこの問題に対処している．しかし前者のモデルを用いる方法は問題依存で汎用性が無く，後者の手法の多くは状態空間を分割するタイプの階層化であり，センサ情報から作られる状態空間が高次元である問題の解決を目的とした方法である．すなわちこれらの手法は跳び箱運動やジャンピングサーブのように，時間軸方向に複雑な行動を複数獲得することを主目的としたものではない．

そこで本研究では過去に学習した動作知識を「再利用」するメカニズムを強化学習に導入することによって，汎用性を損なうことなく強化学習の性能向上を図ることを提案する．例えば転倒回避や走行と言った行動が跳び箱運動のような動作を学習するときに再利用されると，全く新しい動作として学習する場合に比べて学習速度の向上や学習コスト（転倒によるダメージなど）の軽減が大幅に期待できる．

再利用する動作知識には大きく分けて，複数の目的に共通の動作知識と，目的に固有の動作知識がある．前者は転倒などの回避行動，後者は跳躍や走行などである．跳躍とサーブを組み合わせてジャンピングサーブを実現すると言った目的依存型の動作知識の再利用には，個々の動作知識をどのように組み合わせればよいかを「推論」する必要があり高度である．そこで本稿では，回避行動（転倒を回避，人を傷つけないなど）の再利用メカニズムを強化学習に導入する方法を提案し，多関節ロボットの全身運動学習に応用する方法を示す．

2. 回避行動の分離学習と再利用

通常強化学習では本来のタスクの成功に対して与える報酬 (通常正) とは別に，転倒などの望ましくない行動に対して負の報酬 (罰) を与え，そのような行動を避けつつ学習させる．つまり，ある状態 s (簡単のため離散とする) で行動 a (同様) を取ったときの報酬関数を $R(s,a) \in \mathbb{R}$ と書くと，これは負の報酬と本来のタスクの報酬の和で与えられる．したがって報酬関数 $R(s,a)$ には二つの目的（本来のタスクと回避）が混在していることになるが，これを目的別に分離して学習することができれば，回避行動を別の新たなタスクの学習において再利用できると考えられる (**Fig.**1).

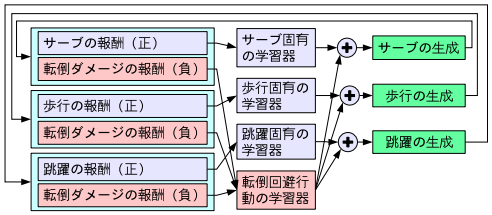

Fig.1: 正と負の報酬分離による再利用

ここで留意しておくべきなのは，再利用の対象となる回避行動は「(転倒などの) 回避を目的とした行動」ではなく「(転倒などにつながるため) 回避すべき行動」である点である．負の報酬に対する行動価値関数は実際に行動を行って得た報酬が負であったときのみ (負の方向に) 更新されるので，後者の実現になっている．逆に転倒を回避できたときに正の報酬を与えるというアプローチだと前者に該当し，これを再利用すると不必要に行動を制限する可能性がある．

2·1 正と負の報酬の分離学習

もっともシンプルな実現方法として，本来のタスクの報酬関数（ゼロ以上とする）を $R_0(s,a)$，回避すべき行動に対する報酬関数（ゼロ以下とする）を $R_1(s,a)$ と表したとき，それぞれの報酬関数に対して強化学習器を用意することが考えられる．本来学習したい行動は報酬関数 $\sum_{i\in\{0,\,1\}} R_i(s,a)$ に対して強化学習を行った場合の行動であるから，この行動が分離した学習器によって獲得した行動から合成することができれば，回避行動を分離して学習したと言うことができる．

2·2　分離学習が可能な条件

個々の報酬関数に対して Q-learning を行うと，獲得される行動価値関数はベルマンの最適方程式

$$Q_i^\star(s,a) = R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \max_{a' \in \mathcal{A}} Q_i^\star(s',a'),$$

$$\text{for} \quad i \in \{0,\, 1\} \tag{1}$$

を満たすように学習される．ただし $\mathcal{S}$ は状態集合，$\mathcal{A}$ は行動集合，$P_{ss'}(a)$ は状態 s で行動 a を選択したとき状態 s' に遷移する確率を表す．これを $i = 0,\, 1$ について足し合わせると

$$\sum_{i \in \{0,\, 1\}} Q_i^\star(s,a) = \sum_{i \in \{0,\, 1\}} R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \sum_{i \in \{0,\, 1\}} \max_{a' \in \mathcal{A}} Q_i^\star(s',a') \tag{2}$$

となるが，もし

$$\sum_{i \in \{0,\, 1\}} \max_{a' \in \mathcal{A}} Q_i^\star(s',a') = \max_{a' \in \mathcal{A}} \sum_{i \in \{0,\, 1\}} Q_i^\star(s',a') \tag{3}$$

が成立するとすれば，式 (2) は $\sum_{i \in \{0,\, 1\}} Q_i^\star(s,a)$ が報酬関数 $\sum_{i \in \{0,\, 1\}} R_i(s,a)$ に対して Q-learning を行ったときの行動価値関数に等しいことを示しているから，分離学習した行動価値関数から，本来学習したい行動価値関数が合成できる．本来のタスクを実現する行動と回避すべき行動が完全に分離できる場合[1]，すなわち回避すべき行動が全体の行動の制約と考えられるような場合には式 (3) が成立するから分離学習可能であり，転倒のような回避行動は（マルコフ決定過程においては）このケースに該当する．

2·3　再利用のための行動選択

エージェントの行動選択は合成した行動価値関数 $Q^\star(s,a) \triangleq \sum_{i \in \{0,\, 1\}} Q_i^\star(s,a)$ に基づいて行われる．再利用時には本来のタスクを学習する行動価値関数 Q_0 は初期化されるが既に回避すべき行動を学習した Q_1 はそのまま再利用されるため，合成した行動価値関数では回避すべき行動は負の値を示す．この情報をもとにエージェントは新たなタスクを実現するための行動を探すわけだが，負の行動価値を示す行動が避けられるような行動選択の方法が必要である．ボルツマン選択は各行動価値に基づいて選択される確率を決定するためこのような要求を満たす行動選択の方法である．すなわち確率

$$P(a|s) = \frac{\exp\left(\frac{1}{\tau} Q^\star(s,a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\tau} Q^\star(s,a')\right)} \tag{4}$$

で行動 a を選択する．

ここで τ は温度定数であり，これが大きいほど各行動間の確率に差が無くなってランダムに行動が選ばれるようになる．したがって回避行動を再利用しながら新しい行動を学習する場合，多様な行動を選択しつつ回避すべき行動は選択しないようにしなければならないから，τ の決定は重要である．

そこで，状態 s における回避行動の平均行動価値 $\frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} Q_1^\star(s,a')$ の選択確率（「平均的な良くない (回避すべき) 行動を選択してしまう確率」と解釈できる）が ϵ を超えないように τ を決定する．すなわち，

$$\max_{s \in \mathcal{S}} \frac{\exp\left(\frac{1}{\tau |\mathcal{A}|} \sum_{a' \in \mathcal{A}} Q_1^\star(s,a')\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\tau} Q_1^\star(s,a')\right)} \leqslant \epsilon \tag{5}$$

を満たす最大の τ を用いる．ϵ としては，例えば 1% 程度に設定する．温度定数 τ の代わりに ϵ を選択する必要性が生じてしまうわけだが，ϵ を使った τ の決定には獲得した回避行動の行動価値関数を使っているためある程度問題への依存度を軽減した選び方ができ，τ の決定よりも直観的である．

[1] 式的には $\prod_{i \in \{0,\, 1\}} Q_i^\star(s',a') = 0$ が成立するとき．

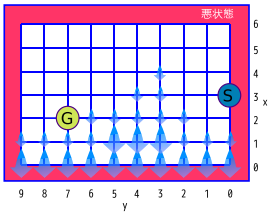

Fig.2: 格子状空間におけるタスク

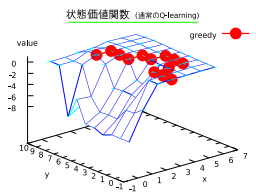

Fig.3: 通常の Q-learning で学習した状態価値関数

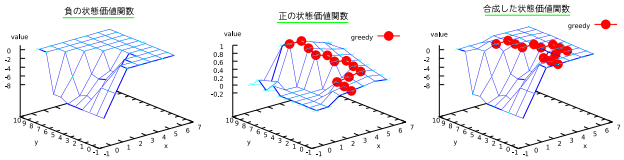

Fig.4: 得られた正と負の状態価値関数及びそれらを合成したもの

2·4　シミュレーション実験

回避行動の分離学習が適切に行われることを確認するために，**Fig.2** に示すような格子状空間における移動タスク（スタート S からゴール G に移動する）を学習させた．図の赤い領域「悪状態」は転倒などに相当する行ってはならない状態，青の矢印はエージェントの行動を変化させる「風」を表しており，どのような行動を取っても必ず「悪状態」に遷移する状態が存在する（転倒しかけの状態を想定）．悪状態に遷移する行動を回避すべき行動として本来のタスクから分離学習した結果 **Fig.4** 左・中央のような負と正の状態価値関数（行動価値関数から計算して表示）が得られ，これらを合成した **Fig.4** 右と **Fig.3** に示す通常の Q-learning で学習した状態価値関数がほぼ等しいことから，分離学習に成功していることがわかる．次にゴールの位置を変えて正の報酬に対する行動価値関数 Q_0 をすべてゼロに初期化し，負の報酬に対する行動価値関数 Q_1 を再利用して学習を行ったところ，試行回数（エピソード回数）に対する報酬の累積が再利用しない場合と比較して **Fig.5** のように変化し，再利用によって「悪状態」に遷移して負の報酬を得る回数が減少していることがわかる．また，その結果学習の速さも向上している．なお **Fig.5** では通常の Q-learning との比較のために，式 (5) を用いた τ の決定法は用いていない．

3.　多関節ロボットへの応用

本節では提案手法を多関節ロボットの運動学習へ応用する方法と，現在得られている結果について述べる．想定しているロボットは地面非固定型のロボットで，対象とする動作は跳躍やテニスサーブなどの全身運動である．ロボット自身が全身運動を獲得したことを明確にするため，また汎用性を損なわないために次元縮約のためのモデルは用いない．このときの問題は状態空間が高次元になるため学習時間が膨大になること，土台が固定されていないため劣駆動系であり，かつ力学的な拘束条件が変化するため制御問題としても難しく，結果としてマルコフ性からのずれが大きくなり強化学習が困難になることである．

これらの問題に対処する方法として，(1) ロボットの非線形ダイナミクスを線形ダイナミクスをひとつひとつのユニット（以下シンボルと呼ぶ）が持つ正規化ガウシ

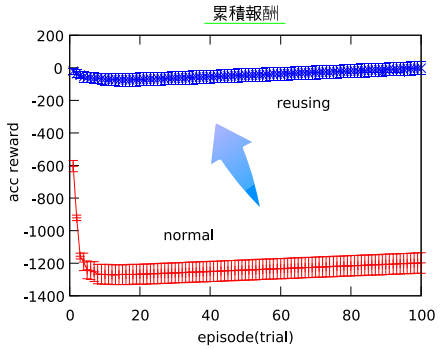

Fig.5: 通常の Q-learning (normal) と負の報酬関数 (ゴール位置が異なるタスクで 500 試行学習させたもの) を再利用した場合 (reusing) の累積報酬．20 回の平均と標準偏差を表示

アンネットワーク [5](Normalized Gaussian Network, 以下 NGnet) で近似させ，このとき学習した基底関数を行動価値関数の近似にも用いる，(2) 行動は学習したシンボル集合からひとつを選び，その中心（ガウシアンの中心）を目標として学習したダイナミクスを用いて制御を行う（すなわち離散行動）という手法を用いる．このような方法を採用した裏には (a) ダイナミクスが線形なら比較的容易に制御が行え，近接するシンボルでは比較的ダイナミクスが近いためシンボル間の制御が比較的容易である，(b) 全身運動は獲得されたシンボルの遷移によって実現できるという 2 つの仮定をおいている．これらの仮定が成立するなら，上述の方法は固定した基底関数のもと，現実的な学習回数で比較的安定に行動価値関数を獲得できると期待できる．以下，その詳細を述べる．

3·1 関数近似器

状態が離散の場合は行動価値関数をテーブルとして学習させることができるが，状態が連続の場合は適当な関数近似器を用いる必要がある．ここでは非線形の関数近似能力が比較的高く，強化学習でも比較的よく用いられる NGnet を使うことにする．NGnet とは入力 x，出力 y の非線形関数を

$$y \approx f(x) \triangleq \sum_i N_i(x)\big(W_i x + b\big) \tag{6}$$

$$N_i(x) \triangleq \frac{G(x;\mu_i,\Sigma_i)}{\sum_j G(x;\mu_j,\Sigma_j)} \tag{7}$$

によって近似する関数近似器である．ここで $G(x;\mu,\Sigma)$ は中心 μ，分散 Σ のガウス関数であり，$N_i(x)$ はある入力 x における合計が 1 になるように正規化された重みである．

3·2 ダイナミクスの学習

ロボットのダイナミクスは，

$$A_{\rm s}(\xi)\ddot{\xi} + b_{\rm s}(\xi,\dot{\xi}) = \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} \tag{8}$$

によって表されるとする．ここで $\xi \in \mathbb{R}^{D_{\rm X}\times 1}$ は土台の位置と姿勢と関節角 ($D_{\rm Q}$ 次元) をまとめたベクトル，$u \in \mathbb{R}^{D_{\rm Q}\times 1}$ は関節トルク，$A_{\rm s}(\xi)$ は慣性行列，$b_{\rm s}(\xi,\dot{\xi})$ は速度項や重力項をまとめたベクトルである．ロボットは直接トルクを指定して駆動できるものとする[2]．このダイナミクスを

$$\frac{\dot{q}(t+\Delta t)-\dot{q}(t)}{\Delta t} = B\begin{bmatrix} \xi \\ \dot{\xi} \end{bmatrix} + d + Au \tag{9}$$

のように線形化・離散化したもの ($A \in \mathbb{R}^{D_{\rm Q}\times D_{\rm Q}}$, $B \in \mathbb{R}^{D_{\rm Q}\times D_{\rm X}}$, $d \in \mathbb{R}^{D_{\rm Q}\times 1}$) を NGnet の基底関数に対応させて，ロボットの全状態空間に対応した非線形ダイナミクスを学習させる[3]．式 (8) は 2 次のダイナミクスだから状態は $(\xi,\dot{\xi})$ であり，この状態から非線形要素が一意に決定される（制御入力 u には依存しない）ため，シンボルは状態空間に配置され，シンボル中心 μ の次元は $2D_{\rm X}$ である．式 (9) でダイナミクスの次元を $\mathbb{R}^{D_{\rm X}\times 1}$ から $\mathbb{R}^{D_{\rm Q}\times 1}$ に落としているのは，劣駆動系での制御に対応させるためである．

3·3 下位レベルの制御器

エージェント（ロボット）の行動は，NGnet のシンボル集合の中からひとつシンボル $s(\in \mathcal{S})$ を選び，その中心 μ_s を目標として制御を行う．このときの制御則は，まず関節角空間 $(q,\dot{q})$ で加速度の時間積分が最小となるように，すなわち

$$\int_0^{T_{\rm f}} \left\| \frac{{\rm d}^2 q}{{\rm d}t^2} \right\|^2 {\rm d}t \to \min \tag{10}$$

によって軌道計画し，次いで学習されたダイナミクスを用いてこれを追従する．

3·4 行動価値関数の学習

行動価値関数は連続状態 $x = (\xi,\dot{\xi}) \in \mathbb{R}^{2D_{\rm X}\times 1}$ に対してのみ近似を行い，離散行動に対してはテーブルルックアップを用いる．すなわち，

$$\begin{aligned} Q_i^\star(x,a) &\approx \tilde{Q}_i(x,a) \\ &\triangleq \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} N_s(x)\delta_{a'}(a)Q_i(s,a') \end{aligned} \tag{11}$$

によって行動価値関数を近似する（$N_s(x)$ は式 (7) で定義されている．パラメタは 3·2 項で学習したものを使う．$\delta_{a'}(a)$ は $a'=a$ で 1，それ以外は 0 を取る関数）．$Q_i(s,a)$ は実数のテーブルであり，これが学習対象のパラメタとなる．Q(λ)-learning のパラメタ更新則を [7, 8] を参考に導出すると，以下のようにまとめられる．

```
for each i ∈ {0, 1} do
  e_it ← R_i(x_t, a_t) + γṼ_it(x_{t+1}) − Ṽ_it(x_t)
  e'_it ← R_i(x_t, a_t) + γṼ_it(x_{t+1}) − Q̃_it(x_t, a_t)
  for each (s, a) ∈ S × A do
    Tr(s, a) ← (γλ) Tr(s, a)
    Q_{it+1}(s, a) ← Q_it(s, a) + α Tr(s, a) e_t
  end for
  for each s ∈ S do
    Q_{it+1}(s, a) ← Q_{it+1}(s, a) + α e'_it N_s(x_t)
    Tr(s, a) ← Tr(s, a) + N_s(x_t)
  end for
end for
```

ここで $Tr(s,a)$ は activity-trace と呼ばれ，行動の履歴を表す．なお，$\tilde{V}_i(x) = \max_{a\in\mathcal{A}} \tilde{Q}_i(x,a)$ である．この行動価値関数の更新は，下位レベルの制御器でシンボルまで到達したときに行う．

---

[2]ただし u を関節に取り付けられたモータの電圧と見ることもできる．状態が $(\xi,\dot{\xi})$ としてセンシングでき，入力が u でかつそれらの間のダイナミクスが式 (8) で表されるシステムなら，以下で述べる手法はそのまま適用可能である．

[3]学習は適当にロボットを動かしてサンプルを取り，それをもとに EM で最適化する．EM だけだとシンボル数が決められない，局所解から脱出できない，と言った問題があるため Ueda らの SMEM[6] をロボットのダイナミクス学習用に変更したものを用いている．

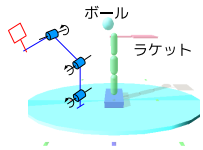

Fig.6: 4 リンク 3 関節の土台非固定型ロボット

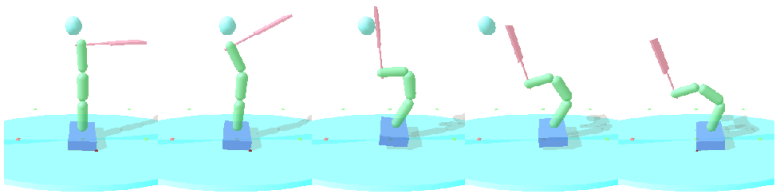

Fig.7: 獲得されたサーブ動作

ここで注意すべきなのは，回避すべき行動の学習 ($i = 1$) では，$\lambda = 0$ としなければならない点である．なぜなら，(基本的には) 取った行動すべてが更新の対象となるので「回避する必要は無いがたまたま負の報酬を得る結果につながった行動」が負の方向に更新されてしまう．正の報酬（本来のタスク）の学習にはこの方が学習を早める効果が期待できる（局所解に陥る危険性も増える）が，回避行動で Q(λ) を行った場合安全な行動も負の方向に更新する危険性がある．

3·5 サーブ課題

以上に述べた手法を，多関節ロボットの全身運動学習に適用する．使用するロボットは 4 リンク 3 関節の土台非固定型で，手先にラケットが取り付けられている (**Fig.**6)．獲得させる動作はボールをできるだけ速く水平に打つというサーブタスクであり，正の報酬は

$$R_0 = \begin{cases} v_x - \frac{\sqrt{v_y^2+v_z^2}}{2} - r_{0\max} & \text{for } \theta_{\text{ball}} < 15^\circ \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

によって与える（$v_{\{x,y,z\}}$ はボールの速度，θ_{ball} はボールの速度の方向が x 軸となす角）である．R_0 が負になった場合は強制的にゼロにする．$r_{0\max}$ は過去 (1 試行内) の最大報酬であり，これを引いておくことで 1 試行の累積報酬がボールの速度に一致する．本来ボールを打つなどのタスクではボールの状態をロボットの状態と合わせて強化学習器の「状態」とするべきだが，ここでは状態空間削減のため，ボールに重力に相当する力を常に上向きに加え，初期状態では静止させている．これにより強化学習器の状態はロボットの状態のみで学習できるようになる．回避すべき行動は転倒であり，負の報酬は各リンクに加わる力のノルムがある閾値 (10000N 程度) を超えたら，力の時間積分に適当な定数を掛けたものを与えている．負の報酬はおおよそ正の報酬の -100 倍程度である．

1 回の試行は直立姿勢・静止，ボールが初期位置 (静止) にある状態からはじまり，シンボルへの遷移が終了したときに

(a) 転倒によるダメージを受けている
(b) ボールが指定された方向以外に飛んで行った
(c) ボールが指定された方向に十分飛び，かつロボットの速度のノルムがある閾値以下

のいずれかを満たした時点で終了する．(c) のみが成功で，「フォロースウィング」も含めて学習させている．

学習を行わせた結果[4]，**Fig.**7 のようなサーブ動作を獲得できた．軽く跳躍しており，ダイナミックなサーブが獲得されていることがわかる．さらにボールの位置を変えて式 (5) で $\epsilon = 0.01$ になる τ を決定し回避行動の再利用下で学習させたところ，転倒によるダメージが約 50 分の 1 程度に軽減することが確認できた．

[4] ダイナミクスのシミュレーションは Open Dynamics Engine[9] を用いて行っている．

4. まとめと今後の課題

本稿では過去の学習で得た転倒などの回避行動を将来の学習に再利用するメカニズムを強化学習に導入し，格子状空間における有効性をシミュレーション実験で示した．さらに提案手法を 4 リンク 3 関節の土台非固定型ロボットのサーブ動作獲得問題に適用する方法を示し，再利用によるダメージの軽減を確認した．しかしながらこの結果だけでは両者が同じサーブタスクであるため得られたものである可能性が否定できず，ロボットの運動学習において回避行動の再利用が有効であることを真に示したとは言いにくい．今後はサーブ以外のタスクで実証していく予定である．

さらに本稿で提案した手法は行動が固定された行動集合 $\mathcal{A}$ の中から選択しているため，同じ行動集合を使う学習にしか再利用できない．ここで用いている行動集合は主に全身運動を想定したものであるため，例えば手先空間における目標位置などを行動とした場合にはその行動が避けられるべきかどうかの判断に使えない．この問題を解決するためには適当な方法で行動を汎化しておく必要があり，より多くの場面で再利用ができるように今後検討していく予定である．

より高度な再利用メカニズムが強化学習に組み込まれれば，「学習すればするほど知識が蓄積されて知的に行動するようになるロボット」が実現される．これは今後，要求や状況が多様に変化する我々の生活の中にロボットが入って行動を共にするような場合，あるいは宇宙空間や地球外惑星などにおいて自律的に行動するような場合に不可欠な技術であり，提案手法はその基礎となるものである．

参考文献

[1] 小林, 藤井, 細江：“一次元空間への低次元化写像を用いた対象物操作の強化学習”, 計測自動制御学会論文集, **42**, 7, pp. 814–821 (2006).
[2] 森本, 銅谷：“階層型強化学習を用いた 3 リンク 2 関節ロボットによる起立運動の獲得”, 日本ロボット学会誌, **19**, 5, pp. 32–37 (2001).
[3] 高橋, 浅田：“階層型学習機構における状態行動空間の構成”, 日本ロボット学会誌, **21**, 2, pp. 38–45 (2003).
[4] 鮫島, 片桐, 銅谷, 川人：“複数の予測モデルを用いた強化学習による非線形制御”, 電子情報通信学会論文誌 D-II, **J84-D2**, 9, pp. 2092–2106 (2001).
[5] M. Sato and S. Ishii: “On-line EM algorithm for the normalized gaussian network”, Neural Computation, **12**, 2, pp. 407–432 (2000).
[6] N. Ueda, R. Nakano, Z. Ghahramani and G. E. Hinton: “SMEM algorithm for mixture models”, Neural Computation, **12**, 9, pp. 2109–2128 (2000).
[7] J. N. Tsitsiklis and B. V. Roy: “An analysis of temporal-difference learning with function approximation”, Technical Report LIDS-P-2322 (1996).
[8] J. Peng and R. J. Williams: “Incremental multi-step q-learning”, International Conference on Machine Learning, pp. 226–232 (1994).
[9] http://www.ode.org/.