

回避行動の再利用メカニズムを備えた強化学習手法と多関節ロボットの全身運動学習への応用

山口 明彦^{*1*2} 杉本 徳和^{*2} 川人 光男^{*1*2}

Reinforcement Learning with Reusing Mechanism of Avoidance Actions and its Application to Learning Whole-Body Motions of Multi-Link Robot

Akihiko Yamaguchi^{*1*2}, Norikazu Sugimoto^{*2} and Mitsuo Kawato^{*1*2}

In acquiring a motion only from its objective by learning, large cost, such as damage from falling over, and a large number of trials are required if the motion is a complex one, such as jumping serve. Reusing the knowledge already learnt is an essential mechanism to learn such motions efficiently, like humans do. In this paper, we propose a learning method to decompose action-value functions for reusing in the framework of reinforcement learning. *Avoidance actions* that are assumed invariant across different tasks (e.g. avoiding to fall over) are learnt separately from primary actions that are assumed task specific, then the action-value function for the avoidance actions is reused in learning new tasks. Furthermore, we extend the method for multi-link robots to learn whole body motions. The proposed method is applied for moving tasks both in discrete and continuous planes, and is also applied for a tennis-serve and a jump tasks of a 4-link robot. We also demonstrate a issue in reusing of the similar method, *Q-decomposition* [1]. The simulation results show an performance advantage of the proposed method over Q-decomposition in reusing avoidance actions.

Key Words: Motion Learning, Reinforcement Learning, Reusing, Avoidance Actions, Jump, Tennis Serve

1. はじめに

ヒューマノイドに代表される高機能な多関節ロボットは設計者によって“作り込まれた”歩行や跳躍などの様々な動作を身に付けているが、設計者の想定を超える例外的な事態に対応したり組み込まれた機能以上の行動をするために「目的から実現方法を自律的に獲得する技術」が不可欠である。そのような技術の一つとして強化学習が研究されており、ロボットの運動学習にも応用されている [2]~[6].

強化学習の最大の課題の一つとして複雑な行動を獲得させようとした高次元の状態空間を扱ったときに学習に膨大なコスト（転倒によるダメージなど）や時間が掛かることがあげられ、ロボットの運動学習に適用されている強化学習の多くはロボットやタスクに特化したモデルを用いたり [4]~[6], 階層構造を導入したり [2] [7] [8] することでこの問題に対処している。

これらの手法は状態空間が高次元であることへの対処であるが、複雑な行動の獲得にはすでに学習した動作知識を新たなタ

スクの学習で「再利用」することによってコスト軽減や学習時間の短縮を図るアプローチも有効である [4] [9]. 例えば人間がテニスのジャンピング・サーブを学習するときには転倒の回避や跳躍と言った動作知識はすでに獲得されており、それらを再利用しながら効率的に学習を行っていると考えられる。仮にこのように複雑な動作をゼロから学習すれば膨大なコストや学習時間を必要とすると考えられ、再利用のメカニズムは不可欠であると言える。しかしながら強化学習によるロボットの運動学習において、再利用に焦点を絞った研究はそれほど多くない。この理由としてはどの動作知識をいつどのように再利用するか推論する必要があり、多くの場合手法がタスク依存になるからなどが考えられる。汎用性を保ったまま再利用するためには、再利用対象となる動作知識を細分化して議論する必要がある。

そこで再利用対象となる動作知識（タスク）を

A. 時間的に並列に行われるタスク.

A.1 状態行動空間の制約となるようなタスク. 言い替えると、転倒回避や人を傷つけないなどの「回避行動」.

A.2 ほかのタスクとの同時実現が要求されるタスク. 走行中に投球するなど.

B. 時間的に見て順次的に行われるタスク. 歩行や跳躍など. のように分類したとき、A.1 の「回避行動」はほぼすべてのタ

原稿受付 2007年12月14日

*¹奈良先端科学技術大学院大学

*²ATR 脳情報研究所

*¹Nara Institute of Science and Technology

*²ATR Computational Neuroscience Laboratories

■ 本論文は提案性で評価されました.

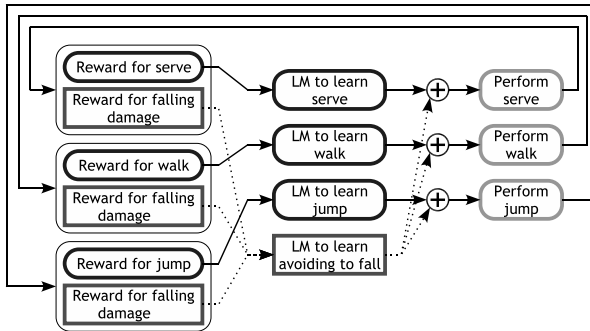


Fig.1 Reusing by separating the primary-task and the avoidance-task rewards. LM stands for a Learning Module

スクにおいて共通に達成されるべき動作知識であり、前述したような推論が不要であるため比較的再利用が容易であると考えられる。そこで本研究では汎用性を損なわずに実現できてかつ再利用されることが重要である「回避行動」の再利用メカニズムを強化学習に導入する。

具体的には報酬を本来のタスクの報酬と回避行動の報酬の二つに分けて別々に行動価値関数を学習し、新たなタスクの学習で回避行動の行動価値関数を再利用するというアプローチを取る (Fig. 1)。したがって回避行動を適切に分離することが重要であり、特にロボットの運動学習のように連続状態空間では関数近似器の構成が回避行動の分離にかかわる最大の要素の一つとなる。そこで本稿では (1) 回避行動を分離し再利用できる強化学習手法を提案、(2) 提案手法を応用した分離学習のための関数近似器修正法を示す、これらを (3) ロボットの運動学習に適用する方法を述べ、(4) 手法の有効性を検証するために格子状空間・二次元連続平面における移動タスク、多関節ロボットのダイナミックな運動学習に適用する。また複数の報酬に対する強化学習手法の一つである Russell らの Q-decomposition [1] の問題点を (1) で指摘し、(4) において比較実験を行う。

Fernández らは過去の学習で獲得した方策を新たなタスクの学習で確率的に再利用する方法を提案しており [9]、これは“確率的再利用アプローチ”であると言える。これに対し本研究の手法は分離学習した回避行動がほかのタスクでも常に使用できることを想定した“決定論的再利用アプローチ”である。この想定が正しい場合は後者が前者より優れていると考えられる。Zhang らは 2 本指のグリップによる複雑な物体の把持タスクを把持する方向の学習器と把持する位置の学習器に分解して学習し、方向の学習器が物体が変わったときにも共通に使えるとして再利用した [4]。さらに新しく把持しようとする物体の形状とすでに学習した物体の形状を比較することで既存の位置の学習器を再利用する方法も述べており、これは前述した推論の一種であると思われる。この研究はタスクを把持に特化しているものの、実ロボットにおいて再利用を試みた研究として興味深い。

本研究で実現する再利用は状態行動空間の制約となるようなタスク (A.1) の再利用であるが、これに前述の A.2 や B の再利用メカニズムが加わることで「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」の実現につながる。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や宇宙空間や地

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。

以下、2 章で強化学習に回避行動の再利用メカニズムを持たせる手法について述べ、3 章で連続状態空間へ拡張・回避行動の分離学習のための関数近似器修正法を提案する。続く 4 章ではこれらの手法を多関節ロボットの全身運動学習へ応用する手法について述べる。これらの手法が有効に機能することを 5 章 (格子状空間・二次元連続平面) および 6 章 (ロボットの全身運動学習) のシミュレーション実験で確認し、7 章でまとめる。

2. 回避行動の分離学習による再利用

強化学習における報酬設計として、本来のタスクの成功に対して適当な報酬を与える以外に、転倒などの望ましくない行動 (回避行動) を回避させる目的で罰 (多くの場合負の報酬) を与えると言ったことがしばしば行われる。このときある状態 $s \in \mathcal{S}$ (簡単のため離散集合とする) で行動 $a \in \mathcal{A}$ (同様) を取ったときの報酬関数を $R(s, a) \in \mathbb{R}$ と書くと、 $R(s, a)$ は本来のタスクの報酬 $R_0(s, a) \in \mathbb{R}$ と回避行動の報酬 $R_1(s, a) \in \mathbb{R}$ の和で与えられる。本章では回避行動の報酬関数がほかのタスクでも共通である場合に、回避行動を分離して学習し別の新たなタスクの学習で再利用する手法を提案する。

2.1 再利用の対象となる報酬

以下では 2.4 節で述べる回避行動の分離学習手法の設計を容易にするために、本来のタスクの成功に対しては正の報酬が与えられ回避行動に対しては負の報酬が与えられることを前提とする。この報酬の拘束により本来のタスクの目的を負の報酬で表現する (例えば文献 [4]) ができなくなる。しかしこのような報酬は適当なオフセットを加えるなどして正の報酬による表現に切り替えることが容易な場合が多く、強化学習の持つ汎用性を大きく損なわせるものではないと考えられる。さらに正負の報酬表現によって行動価値関数を正または負に拘束でき、連続状態空間における学習の数値解析的な安定性が増すなどの利点も得られる。

ただし本来のタスクの成功に与えられる正の報酬だけでは解が多数存在することが多く、このため学習エージェントの行動に対する“ステップコスト”が小さな負の報酬として与えられる場合がある。運動学習の場合は例えば入力トルクのノルムの時間積分などである。この報酬はタスク依存で再利用できないため R_0 に加算することが望ましい。これにより R_0 が負になることがあるが、正の報酬がステップコストに比べて十分大きい場合 2.4 節で述べる更新則を用いれば問題は生じない。

2.2 分離学習と再利用の定式化

再利用を目的とした分離学習を実現するために、本来のタスクの報酬関数と回避行動の報酬関数それぞれに対して強化学習器を用意し、それらを合成することで行動を決定するというアプローチを取る。より具体的には、個々の報酬関数に対して行動価値関数 $Q_i(s, a)$, $i \in \{0, 1\}$ を用意し (Q_0 は本来のタスク R_0 を学習し Q_1 は回避行動 R_1 を学習する)、その総和 $\sum_i Q_i(s, a)$ によって学習器の合成を実現するという方法を検討する。

分離学習の成功条件は「合成した行動価値関数 $\sum_i Q_i(s, a)$ によって、収益 $\sum_{n=1}^{\infty} \gamma^{n-1} \sum_{i \in \{0,1\}} R_i(s_{t+n}, a_{t+n})$ を最大にする

方策が表現できる」ことである (γ は割引率). つまり合成した行動価値関数がベルマンの方程式

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

を満たせば分離学習に成功したことになる.

再利用の成功条件は「回避行動を学習する行動価値関数 Q_1 が報酬 R_1 にのみ依存し R_0 に依存しない」こと, 言い替えると Q_1 がタスクに依存しないことである. よって

$$Q_1(s, a) = R_1(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) Q_1(s', \pi_1(s')) \quad (2a)$$

$$\pi_1(s) = \arg \max_{a \in \mathcal{A}} Q_1(s, a) \quad (2b)$$

を再利用可能な行動価値関数 Q_1 の定義とする. Q_1 が Q_1 のみから決定される方策に依存するのが特徴である.

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 $Q(s, a) \triangleq \sum_i Q_i(s, a)$ 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 はゼロに初期化されるが, すでに回避行動を学習した 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の回避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達成する行動を探すため, 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. このような要求を満たす行動選択の方法の一つに, ある状態における各行動価値に基づいて選択確率を決定するボルツマン選択がある. すなわち確率

$$P(a|s) = \frac{\exp\left(\frac{1}{\tau} Q(s, a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\tau} Q(s, a')\right)} \quad (3)$$

で行動 a を選択する. ただし温度パラメータ τ は, 学習が進むとグリーディ方策に近づくように $\tau = \tau_0 e^{-\delta_\tau N_{\text{eps}}}$ によって減少させている. τ_0 は初期値, δ_τ は時定数, N_{eps} はエピソード数である.

2.4 分離学習と再利用のための行動価値関数学習手法

「複数の報酬に対する強化学習手法」としては文献 [1] [10] をはじめとする多数の研究が報告されているが, 報酬を分離学習することによる学習時間の短縮を目的としたものが多い. このうち Russell ら [1] は Q-learning [11] を用いて個々の行動価値関数を更新した場合, 合成した行動価値関数が式 (1) を満たさないため分離学習に失敗することを示し, 代わりに Sarsa [12] を用いることで分離学習に成功することを証明, Sarsa による分離学習を Q-decomposition として提案した.

しかしながら Russell らの手法によって獲得される行動価値関数は式 (2) を満たさないため, 再利用には適さないと考えられる. 具体的には Sarsa によって個々の行動価値関数 $Q_i, i \in \{0, 1\}$ を更新した場合

$$Q_i(s, a) = R_i(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) Q_i(s', \pi(s')) \quad (4a)$$

$$\pi(s) = \arg \max_{a \in \mathcal{A}} \sum_{i \in \{0, 1\}} Q_i(s, a) \quad (4b)$$

に収束するため [12] Q_1 は式 (2) と一致せず, タスク依存となるからである.

一方個々の行動価値関数の学習に Q-learning を用いた場合は

$$Q_i(s, a) = R_i(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \max_{a' \in \mathcal{A}} Q_i(s', a') \quad (5)$$

に収束するため [11], Q_1 は式 (2) と一致し再利用可能となる. しかしながら, この場合は Russell ら [1] が示したように行動価値関数が

$$\forall s \in \mathcal{S};$$

$$\sum_{i \in \{0, 1\}} \max_{a \in \mathcal{A}} Q_i(s, a) = \max_{a \in \mathcal{A}} \sum_{i \in \{0, 1\}} Q_i(s, a) \quad (6)$$

を満たさない限り式 (1) を満足せず, 分離学習に成功しない. 式 (6) が満たされないのは, 二つの報酬が同時にゼロ以外の値を持つような場合などである.

そこで本研究では式 (6) が崩れると思われるような報酬を受け取った場合には「回避行動を優先する」ように Q-learning の更新則を変更することで, 近似的に分離学習が可能になるようにする. 具体的には Q-learning の更新に使う報酬を

$$R'_0(s_t, a_t) \leftarrow R_0(s_t, a_t) + w R_1(s_t, a_t) \quad (7a)$$

$$R'_1(s_t, a_t) \leftarrow R_1(s_t, a_t) \quad (7b)$$

のように変更し (w は適当な正の定数[†]), 通常の Q-learning

$$Q_{i,t+1}(s_t, a_t) \leftarrow Q_{i,t}(s_t, a_t) + \alpha \{ R'_i(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q_{i,t}(s_{t+1}, a) - Q_{i,t}(s_t, a_t) \} \quad (8)$$

によって更新を行った後,

$$\text{if } Q_{0,t+1}(s_t, a_t) < 0: \quad Q_{0,t+1}(s_t, a_t) \leftarrow 0 \quad (9a)$$

$$\text{if } Q_{1,t+1}(s_t, a_t) > 0: \quad Q_{1,t+1}(s_t, a_t) \leftarrow 0 \quad (9b)$$

によって各行動価値関数がそれぞれ 0 以上または 0 以下になることを強制する. 式 (7a) によって本来のタスクの報酬から回避行動の負の報酬が差し引かれているため, 式 (6) を崩すような報酬の組み合わせ R_0, R_1 を受け取った場合には $R'_0 < 0$ となる. しかし式 (9a) によって本来のタスクの行動価値がゼロ以上に制約されるため, 結果として R_1 が負の値を持つ場合には $Q_0 = 0, Q_1 < 0$ となって回避行動が優先されることになる. なお $R_0(s_t, a_t) \geq 0, R_1(s_t, a_t) = 0$ の場合は行動価値関数の学習に何ら影響を及ぼさない. この方法では式 (1) が近似的にしか満たされないが, 回避行動を優先した上で本来のタスクを学習することができるため, Russell らが指摘する Q-learning の問題 [1] を改善することができると言える. この更新則を以下では DQL+ (Decomposed Q-Learning +) と呼ぶ.

[†] $|w R_1|$ が $|R_0|$ より大きくなるように選択. $|R_1|$ が $|R_0|$ より大きい場合 1 でよい.

3. 連続状態空間への拡張

連続の状態空間を扱う場合適当な関数近似器を使って行動価値関数を近似する必要があるが、この関数近似器が原因で分離学習に失敗する場合、すなわち理想的には回避行動の行動価値関数を分離学習できるにもかかわらず、この関数近似器の近似精度によって得られる行動の性能に悪影響を及ぼす場合が存在する。本章では関数近似器として強化学習やロボットの運動学習でもしばしば用いられる（例えば文献 [2]）正規化ガウシアンネットワーク [13] (Normalized Gaussian Network, 以下 NGnet) を用いて連続状態空間で学習を行う方法を紹介し、さらに分離学習に失敗する原因を緩和するための基底関数修正法について述べる。

3.1 行動価値関数の関数近似

関数近似器 NGnet は入力 x 、出力 y の非線形関数を

$$y \approx f(x) \triangleq \sum_k N_k(x)(W_k x + b_k) \quad (10)$$

$$N_k(x) \triangleq \frac{G(x; \mu_k, \Sigma_k)}{\sum_{k'} G(x; \mu_{k'}, \Sigma_{k'})} \quad (11)$$

で近似する。ここで W_k, b_k は各基底関数と対になった線形関数のパラメタ、 $G(x; \mu, \Sigma)$ は中心 μ 、分散 Σ のガウス関数であり、 $N_k(x)$ 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。ある基底関数とそれに関連付けられた線形関数を合わせてユニットと呼ぶ。

ここでは行動価値関数は連続状態 x に対してのみ NGnet による近似を行い、行動 a は離散集合 \mathcal{A} から選択されることを想定してテーブルルックアップを用いる。すなわち

$$Q_i(x, a) \approx \tilde{Q}_i(x, a) \triangleq \sum_{k \in \mathcal{K}} N_k(x) Q_i(k, a) \quad (12)$$

によって行動価値関数が近似されているとする (\mathcal{K} はユニットの集合)。 $Q_i(k, a)$ 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。このパラメタの更新には学習を早めるために行動の履歴を用いる $Q(\lambda)$ -learning [14] を適用する (付録 A)。

3.2 「分離誤差」に基づく関数近似器修正法

一般的に基底関数の数が多いと学習が遅くなるため、また汎化性能を高めるために関数近似器の基底関数はある程度「粗く」状態空間に配置される。これが原因で収束が遅くなるなど、分離学習や再利用時の学習性能が悪くなる場合がある。

一例として Fig. 2 (a) に示すような、基底関数 k からの重み N_k がほぼ同じである二つの状態で選択した同一の行動が、異なる結果 (報酬) をもたらすような場合を考える。この基底関数 k に対応付けられた行動価値関数のパラメタ $Q_1(k, a)$ は状態 x_2 から行動 a を選択することによって負の方向に更新される。回避すべき行動を何度も選択しないように、負の報酬やステップサイズパラメタはある程度大きく設定されるので、この後状態 x_1 で同じ行動 a を選択する確率はボルツマン選択などにより小さくなる。結果として行動 a の選択が x_1 における最適方策の場合などでは、学習に時間が掛かるかほかの局所最適解に陥る可能性が高くなる。

これは関数近似器を用いると一般的に起こり得る問題で、当然分離学習でも起こり再利用においても問題となる。同じ例で

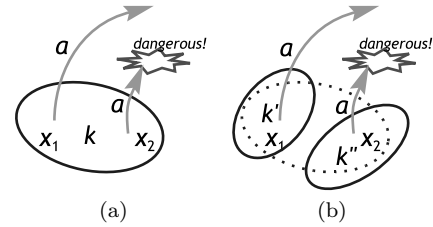


Fig. 2 An example of a basis function that will lead to failure of the learning (a), and improvement by dividing the basis function (b). Each ellipse denotes a basis function (normalized Gaussian), and a denotes an action

$Q_1(k, a)$ が状態 x_2 から行動 a を選択したことによって負の方向に更新され新たなタスクの学習で再利用された場合、その付近の状態 (x_1 など) から行動 a を安全であるにもかかわらず選択する確率が低くなってしまふ。

この問題が状態空間のどこで起きているか調べられれば、Fig. 2 (a) のような場合には基底関数を Fig. 2 (b) のように分割することで対処できる。DQL+は更新則に Q-learning を用いており、離散状態行動空間なら Q-learning で分離学習が可能である条件式 (6) が満たされるように学習されること、Fig. 2 (a) のような場合にはこの条件が崩れること、およびこの条件が分離学習した行動価値関数を使って評価できることに注意すると、次式で定義される「分離誤差」を評価することで問題となる基底関数が調べられる：

$$\tilde{C}(x) \triangleq \left(\sum_{i \in \{0,1\}} \max_{a \in \mathcal{A}} \tilde{Q}_i(x, a) - \max_{a \in \mathcal{A}} \sum_{i \in \{0,1\}} \tilde{Q}_i(x, a) \right)^2. \quad (13)$$

このとき“分離誤差を評価し大きければ基底関数を適当な方法で追加する”というアプローチによって連続状態空間における分離学習と再利用の失敗要因を減らすことができると考えられ、これを具体化すると以下のようなアルゴリズムになる。

Algorithm 1: Manipulation of basis-functions based on the separative error

```

 $\mathcal{X} \leftarrow$  分離誤差を評価する状態集合 *1
for each  $x \in \mathcal{X}$  do
   $C_x \leftarrow \left( \sum_{i \in \{0,1\}} \max_{a' \in \mathcal{A}} \tilde{Q}_i(x, a') - \max_{a' \in \mathcal{A}} \sum_{i \in \{0,1\}} \tilde{Q}_i(x, a') \right)^2$ 
  if  $C_x \geq \text{threshold}$  then
     $x$  に基底関数を追加 *2
  end if
end for
```

*1 の分離誤差を評価する状態集合としては、各基底関数の中心およびそれらの中間状態から選択した、より細かくすることも可能だが、その分 $\tilde{C}(x)$ の評価による計算時間が増加する。*2 の基底関数の追加方法については、Sato ら [13] の“unit division”を参考にした方法を用いた (付録 B)。このアルゴリズムの実行タイミングとしては行動価値関数の更新が終わった直後やエピソードの終了直後などが考えられるが、パラメタベクトルの再アロケートと言った実装上の都合から後者が適当である。なおこのアルゴリズムは NGnet を前提としたものだが、付録 B の基底関数の追加方法を変更すれば NGnet 以外の関数近似器 (ただし局所モデル) にも適用できると考えられる。

4. 多関節ロボットの運動学習への応用

本章では前章までの手法を多関節ロボットの全身運動学習に応用する方法について述べる。ここでは全身運動を「ロボットの一般化座標に相当する物理量の軌道によって表現可能な運動」と定義し、扱うロボットは土台が非固定の多関節ロボットを想定する。例えば跳躍や（ボールの初期位置を固定した）サーブ、起き上がり運動 [2] などが全身運動に該当する。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ロボットやタスクに依存したモデルは用いない。このときの課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、土台が非固定のため劣駆動系で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いことである。

この問題に対処するために以下のアプローチを取る (Fig. 3) :

- (1) ロボットの非線形ダイナミクスを各ユニット (以下シンボルと呼ぶ) が線形ダイナミクスを持つ NGnet で学習し (詳細は付録 C), このとき得られた基底関数を行動価値関数の推定にも使う。
- (2) 行動は NGnet から一つの目標シンボル k を選びその基底関数の中心 μ_k を目標とした制御を行うというものである。このときの制御は関節角 q の躍度のノルムを時間積分したものが最小になるように関節角軌道を計算し [15] (付録 D), PD 制御器で追従することで実現する。

このような手法を採用した裏には (a) ダイナミクスの非線形性が強くなるほどロボットの制御が困難になり、行動価値関数を推定するためにより多くの基底関数を配置する必要がある (逆に言うとダイナミクスが線形に近ければ比較的少数の基底関数でよい), (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二つの仮定がある。(a) の仮定によってロボットのダイナミクスを近似する NGnet の基底関数は行動価値関数が複雑な状態には比較的密に、そうでない状態には比較的粗く配置され行動価値関数の近似に適した関数近似器が得られると期待される。(b) の仮定により行動を限定しているため精密な動作は実現しにくい、学習時間を大幅に縮めるという利点がある。さらに基底関数のパラメータを固定した行動価値関数の学習は線形の重みのみになるので、比較的安定に強化学習が行われると期待できる。

5. シミュレーション実験 1: 移動タスク

本章では 2 章で述べた DQL+ の有効性を確認するため格子状

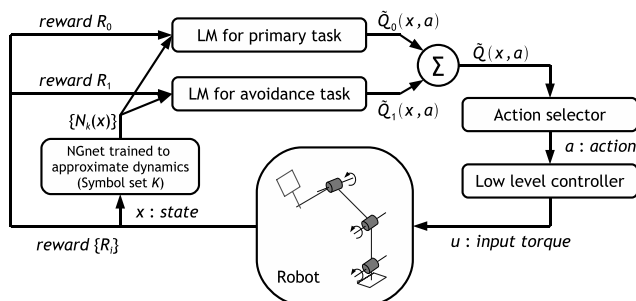


Fig. 3 Architecture for robots to learn whole-body motions. LM stands for Learning Module

空間および二次元連続平面上の移動タスクに適用し, Russell らの Sarsa による分離学習 [1] と比較して分離学習と再利用における性能の違いを検証する。

いずれのタスクもロボットの運動学習への応用を意識したもので、スタート状態 (“start”) からゴール状態 (“goal”) に移動すると報酬がもらえるというタスクである。一方で転倒などに相当する回避すべき状態 (“bad state”) があり、この状態に行くとき大きな負の報酬が与えられる。さらに「風」 (“wind”) が学習エージェントの行動を変化させ、必ず “bad state” に遷移してしまう状態が存在する (転倒しかけの状態を想定)。エピソードはスタートからはじまり、ゴールに到達するか “bad state” に遷移した時点で終了とする。

5.1 格子状空間における移動タスク

Fig. 4 に示す 7×10 の格子状空間における移動タスクに適用する。状態は $s = (x, y)$, 選択可能な行動は上下左右の 4 種類である。“start” から “goal” に移動できれば本来のタスクの報酬 $R_0 = 1$ が与えられ, “bad state” では大きい負の報酬 $R_1 = -10$ が与えられる。“wind” を表す小さい矢は +1, 大きい矢は +2 だけ矢印の方向に行動を変化させる。局所解に陥った場合を考慮して、エージェントの行動数が 1,000 を超えた場合はエピソードを終了させる。“trap” は本節の実験では機能せず、次節の実験で用いられる。なおエージェントの 1 回の行動につき -0.005 のステップコストを R_0 に加えている。

Fig. 5 (a) は 2 章で提案した DQL+ (DQL+) および Russell らの Sarsa による分離学習 (DSarsa) を移動タスクに適用した結果のエピソード-報酬曲線 (報酬はエピソード内の合計) で、いずれも再利用は行っていない。一方 Fig. 5 (b) はスタートの位置を (3, 0), ゴールの位置を (6, 8) に変えて Fig. 5 (a) の学習で最終的に (3,000 エピソード後) 得られた回避行動の行動価値関数を再利用して学習を行った結果のエピソード-報酬曲線で、DQL+による再利用 (DQL+ (reusing)), Russell らの手法による再利用 (DSarsa (reusing)), 再利用しない場合の DQL+ (DQL+ (no reusing)) および Russell らの Sarsa による分離学習 (DSarsa (no reusing)) の結果を示してある。Fig. 5 (b) の結果から再利用によって “bad state” に行く割合が減り、学習コストや学習時間が大幅に軽減していることが分かる。しかし分離学習 (Fig. 5 (a)) と再利用 (Fig. 5 (b)) のいずれの実験においても DQL+ と Russell らの Sarsa による分離学習の間に性能の差は見られなかった。これは回避行動と本来のタスクが完

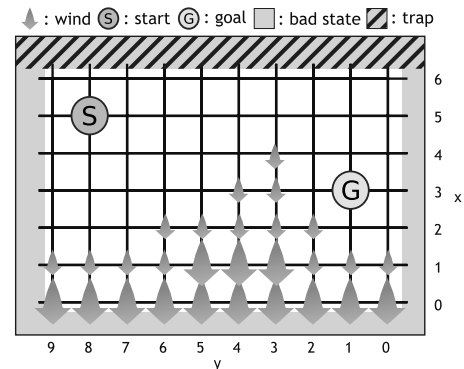


Fig. 4 Moving task in the grid world

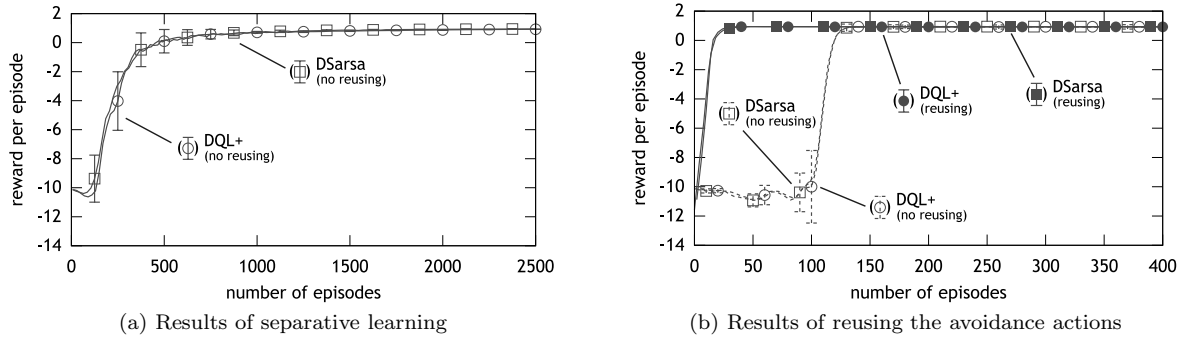


Fig. 5 Results of the moving task in the grid world: Each graph shows total rewards per episode (the mean and the ± 1 standard deviation (SD) around the mean over 100 runs). The labels of the curves denote the following: DQL+ (no reusing): separative learning without reusing by DQL+, DSarsa (no reusing): separative learning without reusing by Sarsa, proposed by Russell *et al.*, DQL+ (reusing): learning with reusing by DQL+, DSarsa (reusing): learning with reusing by the method of Russell *et al.*

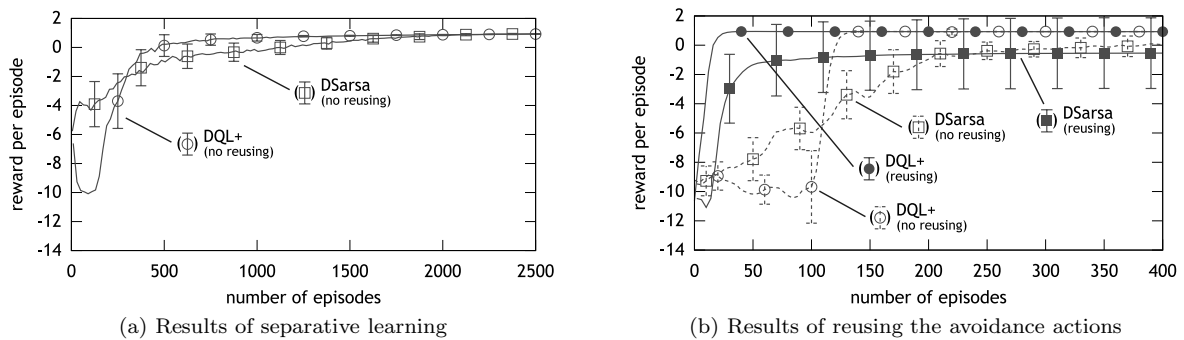


Fig. 6 Results of the moving task in the grid world with traps: Each graph shows total rewards per episode (the mean and the ± 1 SD around the mean over 100 runs). The labels of the curves denote the same learning methods as Fig. 5

全に分離している単純なタスクだからだと考えられる。

5.2 罠つき格子状空間における移動タスク

そこで Fig. 4 に示した格子状空間の上部 ($x > 6$) に “trap” を設け、この状態に移動すると $R_1 = -10$ と同時に $R_0 = +10$ も与えられるといった、やや極端な設定とした。この設定にしたとしても学習すべき最適な行動が “start” から “goal” に行くことであるのに変わりはないが、 R_0 と R_1 が同時に値を持つため (1) 提案手法の DQL+ が 2.4 節で述べた Q-learning による分離学習の問題を解決できているか、および (2) Russell らの Sarsa による分離学習で学習される Q_1 がタスク依存になり、再利用にどの程度影響を与えるかを検証できる。

“trap” 以外の条件は前節の実験と同じとし、分離学習および再利用の実験を行ったところ、それぞれ Fig. 6 (a), (b) の結果を得た。分離学習において DQL+ の報酬が一度下がっているのは、学習の初期 (0~100 エピソード) では探索のために “trap” に訪れて $R_0 = +10$ と $R_1 = -10$ を受け取るが、中期 (100 エピソード~) では R_1 を優先するという DQL+ の学習性質により “trap” に行く割合が DSarsa や学習初期と比較して少なくなっているため $R_0 = +10$ が受け取れないからである。一方再利用の結果では、DQL+ (reusing) は期待通り DQL+ (no reusing) と比較して学習コストや学習時間が軽減していることが分かるが、DSarsa (reusing) は最終的な分散が大きくなっており、ゴールに行かずスタート付近に留まる局所解や “trap” に行く局所解に陥っていた。2.4 節で述べたように Sarsa で学習さ

れる行動価値関数 Q_1 は一般的に再利用には不向きであり、この格子状空間における移動タスクはそれを示す例となっていると言える。

5.3 二次元連続平面における移動タスク

Fig. 7 に示す二次元連続平面での移動タスクに適用する。状態は位置 (x_1, x_2) 、制御入力は速度 (上限を持つ) であり、“wind” はこの入力値を矢の方向に増加させる。“wall” は通り抜けられず、ぶつかった場合 “wall” に沿って移動する。エージェントは “start” から “goal” に移動できれば本来のタスクの報酬 $R_0 = 1$ が与えられ、“bad state” に遷移すると発生する -10 から -5 の負の報酬 (大きさは “bad state” に留まった時間で決まる) が R_1 として与えられる。行動価値関数を近似する NGnet のパラメタ (中心 μ と分散 Σ) は Fig. 7 の楕円に示すように与えた。行動はなるべくロボットの運動学習と同じ状況にするために、4 章で述べた行動集合と類似のものを用いた。すなわち NGnet から一つの目標シンボル k を選びその基底関数の中心 μ_k を目標とした制御を行うというものである (詳細は付録 D)。ただし躍度ではなく加速度ノルム ($\ddot{x}_1^2 + \ddot{x}_2^2$) の時間積分を最小にするように軌道計画している。

Fig. 8 (a) は DQL+ (DQL+) および Russell らの Sarsa による分離学習 (DSarsa) を適用した結果のエピソード-報酬曲線 (報酬はエピソード内の合計) で、いずれも再利用は行っていない。一方 Fig. 8 (b) はスタートとゴールの位置を変更し Fig. 8 (a) の学習で最終的に得られた回避行動の行動価値関数を再利用して

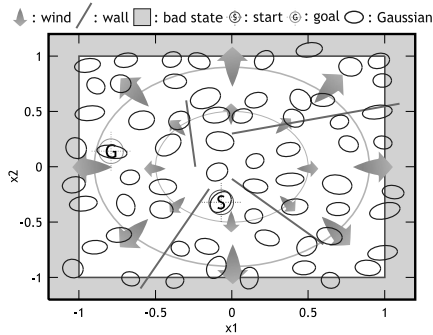


Fig. 7 Moving task in the 2-dimensional continuous plane. Each ellipse denotes a basis function

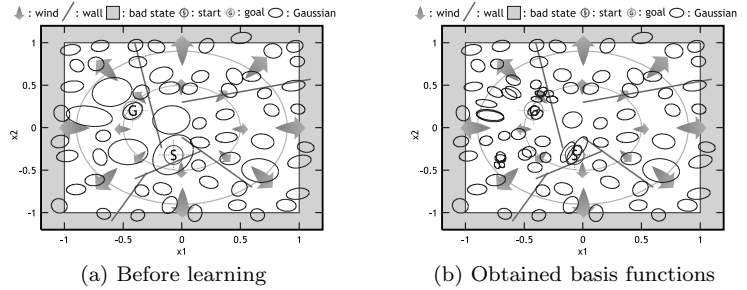
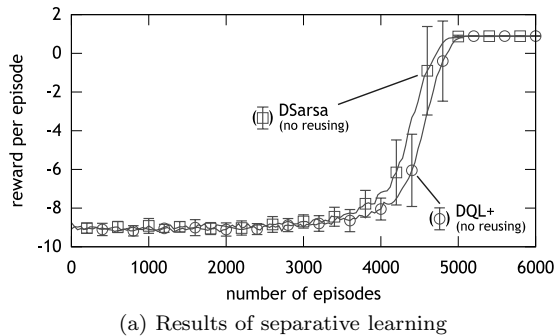
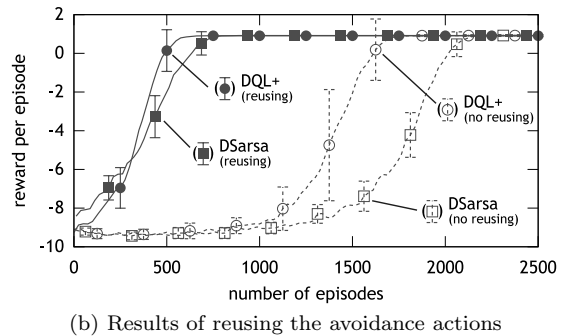


Fig. 9 Moving task in the 2-dimensional continuous plane to test the algorithm of sec. 3.2. Each ellipse denotes a basis function



(a) Results of separative learning



(b) Results of reusing the avoidance actions

Fig. 8 Results of the moving task in the 2-dimensional continuous plane: Each graph shows total rewards per episode (the mean and the ± 1 SD around the mean over 25 runs). The labels of the curves denote the same learning methods as Fig. 5

学習を行った結果のエピソード-報酬曲線で、DQL+による再利用 (DQL+ (reusing)), Russell らの手法による再利用 (DSarsa (reusing)), 再利用しない場合の DQL+ (DQL+ (no reusing)) および Russell らの Sarsa による分離学習 (DSarsa (no reusing)) の結果を示してある。Fig. 8 (a) の DQL+ と DSarsa を比較すると DSarsa のほうが収束が速く、Fig. 8 (b) の DQL+ (no reusing) と DSarsa (no reusing) を比較すると DQL+ (no reusing) のほうが収束が速いことから、分離学習における性能はある程度タスクによって変化することが分かる。一方 Fig. 8 (b) から再利用によって再利用しない場合と比べて大幅に収束が速くなっていることから、(低次元の) 連続状態空間においても回避行動の再利用が適切に機能すると言える。

次に 3.2 節で述べた基底関数の修正法が適切に機能するかを検証する。ただし Fig. 2 (a) のような問題を生じやすくするために、“start”, “goal”, “wall” および初期基底関数の配置を Fig. 9 (a) のように変更する。これにより “start” 付近などで Fig. 2 (a) のような問題が生じやすくなると考えられる。なお前述したように二次元連続平面における行動集合は基底関数の中心に依存しているが、この行動集合については基底関数の修正法を適用する際に学習パラメタの大幅な増加を防ぐため、また基底関数の修正法を使わない場合との比較における条件を同じにするために基底関数の修正に対して変化させない。基底関数の修正法を適用し分離学習を行わせた結果、Fig. 9 (b) に示すように基底関数が修正された。“start” 付近などで基底関数が分割されていることが確認され、所望の結果が得られていることが分かる。また修正法を用いない場合だと 20 回中 2 回しかゴー

ルにたどり着く解が得られなかったのに対し (スタート付近に留まる解に陥るなど)、修正法を用いた場合だと 20 回中 12 回まで向上することができた。

6. シミュレーション実験 2: ロボットの運動学習

4 章で述べた手法を 4 リンク 3 関節土台非固定型ロボット (Fig. 10) の全身運動学習に適用し、回避行動の再利用が行えるか検証する。学習させる全身運動はサーブおよび跳躍であり、サーブのためにラケットが手先に固定されている。このラケットはそれぞれの動作でロボットのダイナミクスを共通にするために、跳躍においても取り付けられたままである。各アクチュエータの軸はすべて同じ向き (y 軸方向) であるためロボットの動作は二次元平面内に限られる。このためロボットの状態は位置と姿勢で計三次元、関節角三次元、およびこれらの速度項六次元の合計十二次元 ($x, z, \theta, q_0, q_1, q_2, \dot{x}, \dot{z}, \dot{\theta}, \dot{q}_0, \dot{q}_1, \dot{q}_2$) で表現される (これらはセンサから観測できるとする)。ロボットの行動はダイナミクスを学習した NGnet のシンボル集合 \mathcal{K} から選んだ目標シンボル k と、関節角軌道の計画に使う k の中心 μ_k までの到達時間 $T_f \in \mathcal{I}$ (\mathcal{I} は離散集合) から成る。すなわち行動集合 $\mathcal{A} = \mathcal{K} \times \mathcal{I}$ (離散集合) から選択される (付録 D)。一つの行動は、計画された関節角軌道を PD 制御器で追従し μ_k に達するか T_f が経過することで終了する (シンボル遷移)。なおロボットのダイナミクスのシミュレーションは Open Dynamics Engine [16] を用いて行っている。

6.1 土台リンクの水平制御

ロボットの運動学習でいくつかの予備実験を行ったところ、ロ

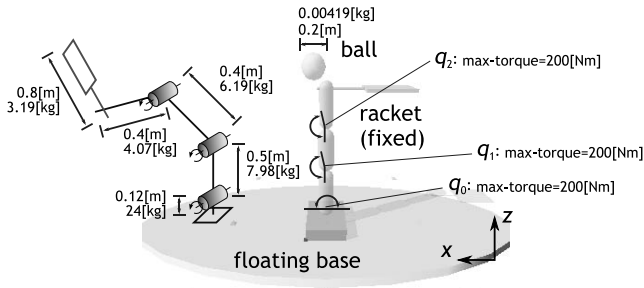


Fig. 10 4-link robot who has 3 joints and 3 actuators

ロボットの土台リンクの接触条件が変化するとき制御が不安定になり、学習に悪影響を与えることが分かった。ロボットの土台リンクがわずかに浮いた状態と接地している状態では行動価値関数を近似する関数器 (NGnet) の基底関数の出力 $\{N_k(x)|k \in \mathcal{K}\}$ にほとんど差がないが、それぞれの状態における行動の結果はダイナミクスの違いから大きく異なることが多い。この問題は事前に予測できたためロボットのダイナミクスを学習させた基底関数を用いる方法や分離誤差に基づく基底関数の修正法 (3.2 節) による解決を図ったが、一度土台リンクが浮くと明示的に土台を接地させる制御をしない限り接触状態の切り替わりが長く不安定な状態に陥るため、解決には至らなかった。

このため以下では土台リンクを水平に保つための制御を明示的に行うことで接触状態の切り替わりが続くことを阻止し、この問題を緩和している。すなわち前述の行動における制御信号 (各関節のトルク) のうち土台リンクの関節に入力されるものに、土台リンクを水平に保つ PD 制御信号を上乗せする。これにより接触状態の切り替わりはある程度緩和されるが転倒しなくなるわけではなく、次節で述べる回避行動の難易度は依然高い。また手法の汎用性は半減するものの、実現できるタスクにもサーブや跳躍といった多様性があるため、ロボットが自律的にタスクを達成することを阻害するものではない。

6.2 運動学習における回避行動

再利用される回避行動は転倒であり、リンク j にかかる力のノルム $\|f_j\|$ がある閾値 Th_j を超えたときそれを時間積分したものをダメージとして定義、これを回避する行動を各動作共通の回避行動とする。ただしダメージの分散が閾値と比較してかなり大きくなってしまふのを抑えるために、ダメージ D を次式によって 1 から 2 の間に抑えこれを $-1,000$ 倍したものを回避行動の報酬 R_1 として用いている。

$$d \triangleq \sum_j \frac{[\|f_j\| - Th_j]_+}{Th_j} \quad (14)$$

$$\text{for } d > 0, D = 0: D' = D + 1 + (1 - D) \left\{ \frac{2}{1 + e^{-\tau_{\text{dmg}} d}} - 1 \right\} \quad (15)$$

$$\text{for } d > 0, D \neq 0: D' = D + (2 - D) \left\{ \frac{2}{1 + e^{-\tau_{\text{dmg}} d}} - 1 \right\} \quad (16)$$

ただし $[\cdot]_+$ は正なら値をそのまま返し負ならゼロを返す関数、 d は各関節に加わる閾値以上の力の合計 (各閾値で割って正規化したものの和)、 τ_{dmg} は適当な時定数を表し、ダメージ D は各行動 (シンボル遷移) が始まる前にゼロに初期化される。 d と D の変化の一例を Fig. 11 に示す。

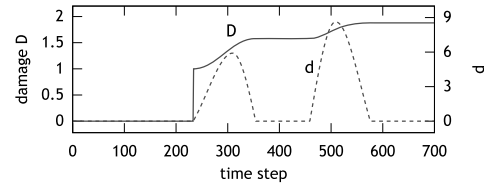


Fig. 11 An example of a trajectory of the damage D over the total normalized force d during a symbol transition

6.3 サーブと回避行動の分離学習

ロボットの運動学習における分離学習アルゴリズムの性能を検証するために、サーブタスクをロボットに学習させる。ここでいうサーブは宙に静止したボールを水平方向 (x 軸方向) に可能な限り速く打つことを目的としたもので、この報酬は

$$R_0 = \begin{cases} 10 \left[v_x - \frac{\sqrt{v_z^2}}{2} - r_{0\text{max}} \right]_+ & \text{for } \theta_{\text{ball}} < 15^\circ \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

によって与えられる。ただし $v_{\{x,z\}}$ はボールの速度の水平・垂直成分、 θ_{ball} はボールの速度ベクトルと x 軸のなす角、 $r_{0\text{max}}$ はそのエピソード内の $(v_x - \sqrt{v_z^2}/2)$ の最大値である。 $r_{0\text{max}}$ を引いておくことでそのエピソードの累積報酬がボールの速度に比例する。さらにエピソードが成功として終了した場合には $+10$ の報酬を R_0 に加え、トルク変化のノルムをシンボル遷移内で時間積分したものをステップコストとして R_0 から引いている。なおサーブのようにボールを扱うタスクではマルコフ性を成り立たせるためにボールの状態とロボットの状態を合わせて強化学習の「状態」とするべきだが、ここではボールに重力相当の力を常に上向きに加え初期状態では静止させることによってロボットの状態のみで学習できるようにしている。

一つのエピソードはロボットが直立姿勢・静止状態かつボールが初期位置・静止状態からはじまり、一つのシンボル遷移が終了した時点で

- 転倒によるダメージを受けている。
- エピソード開始から 20 秒以上経過している。
- ボールが指定された方向以外に飛び去った。
- ボールが指定された方向に十分飛び (2 [m] 以上)、ロボットが静止している。

のいずれかを満たしていれば終了とする。(d) のみがタスクの成功であり、この場合のみ前述したように $R_0 = +10$ を与えてから終了する。

行動価値関数がすべての状態でゼロである初期状態から、DQL+ (DQL+) および Russell らの Sarsa による分離学習 (DSarsa) を適用した。この結果は Fig. 12 に示すエピソード報酬曲線 (報酬はエピソード内の合計) となり、DQL+ のほうが DSarsa よりも速く収束していることが分かる。この理由としては $Q(\lambda)$ -learning が持つ収束の速さ、 \tilde{Q}_1 をゼロ以下に拘束することによる DQL+ の学習の安定さなどが効果的に機能したからなどが考えられる。DQL+ で最終的に (3,000 エピソード後) 得られたサーブの一例を Fig. 13 に示す。

6.4 遅いサーブと速いサーブの間の再利用

次に、ロボットの運動学習における回避行動の再利用の有効性を検証するために、「遅いサーブ」を学習し、回避行動の行動価値関数を再利用して「速いサーブ」を学習、再利用しない場

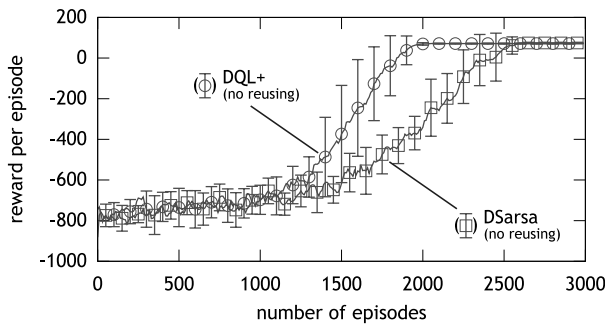


Fig. 12 Results of separative learning the serve task: Each graph shows total rewards per episode (the mean and the ± 1 SD around the mean over 14 runs). The labels of the curves denote the same learning methods as Fig. 5

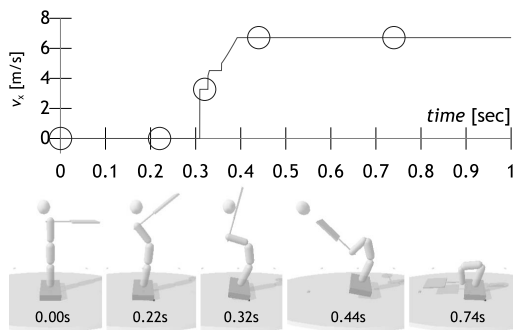


Fig. 13 Trajectory of ball-velocity v_x and snapshots of an acquired serve motion. Each circle on the trajectory corresponds to the snapshot

合と比較する. ここで速いサーブとは 6.3 節で定義したサーブと同様のタスク, 遅いサーブとは式 (17) の報酬をボールの速度の水平成分 v_x がある値 (5 [m/s]) を超えた場合にはゼロとするタスクである (それ以外は速いサーブと同様). 実験はまず遅いサーブを学習させて回避行動の行動価値関数 \tilde{Q}_1 を得, これを再利用して速いサーブを学習するという流れで行った.

この結果 **Fig. 14** に示すエピソード-報酬曲線を得た. ここでは DQL+ (DQL+ (reusing)), Russell らの Sarsa による分離学習 (DSarsa (reusing)), およびこれらの手法を \tilde{Q}_1 を再利用せずに用いた DQL+ (no reusing) と DSarsa (no reusing) を比較している. DQL+ (reusing) は再利用しない場合 (DQL+ (no reusing)) に比べてわずかに収束が速くなっているが, DSarsa (reusing) は再利用しない場合 (DSarsa (no reusing)) よりも収束が遅くなっている. これは 2.4 節で述べたように Sarsa で学習される行動価値関数 Q_1 がタスク依存であるため, 再利用した場合にかえて性能を悪化させてしまったからだと考えられる. 一方 DQL+ (reusing) が再利用によって向上した性能は格子状空間や二次元連続平面における移動タスクと比較するとわずかであり, この原因としては

- (1) 状態行動空間が大きいいため回避行動の行動価値関数が十分に学習されていない.
- (2) 連続状態空間へ拡張する際の関数近似器が本来のタスクと回避行動を適切に分離していない.

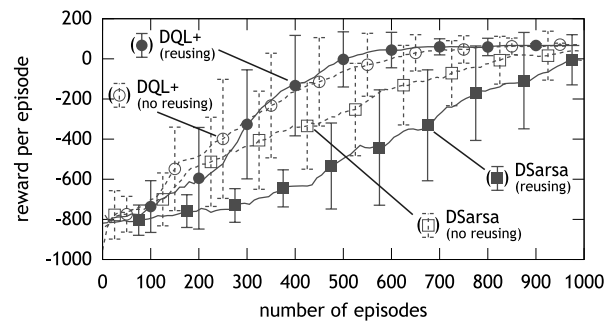


Fig. 14 Results of learning a fast serve reusing the avoidance actions obtained from learning a slow serve: Each graph shows total rewards per episode (the mean and the ± 1 SD around the mean over 25 runs). The labels of the curves denote the same learning methods as Fig. 5

などが複合的に絡んでいると考えられ, ロボットの運動学習における再利用の困難さを表している. (1) の解決には学習時間を増やすか基底関数の数を減らしてパラメタを削減などが考えられるが, 基底関数を減らすことは (2) の問題をより大きくするため注意が必要である.

6.5 跳躍

3.2 節で述べた基底関数の修正法が運動学習においても有効に機能することを検証するため, 比較的難易度が高く Fig. 2(a) のような問題が生じる可能性がある跳躍に適用する. ここでの跳躍はロボットの重心の垂直成分 z_{com} をできるだけ高い位置に移動させることを目的としたものであり, あるシンボル遷移 (行動) に対する報酬 R_0 は次式で定義される:

$$R_0 = 200 \left[z_{\text{com-max}} - z_{\text{com-max}}^{\text{old}} \right]_+ \quad (18)$$

ここで $z_{\text{com-max}}$ はシンボル遷移内の最大重心位置, $z_{\text{com-max}}^{\text{old}}$ はそのシンボル遷移以前の最大重心位置をそれぞれ表し, $z_{\text{com-max}}^{\text{old}}$ の初期値は初期姿勢での重心位置である. さらに爪先立ちの解を避けるために土台リンクが接地しているときは報酬 R_0 を与えず, エピソードが成功として終了した場合には +10 の報酬を R_0 に加え, トルク変化のノルムをシンボル遷移内で時間積分したものをステップコストとして R_0 から引いている.

一つのエピソードは直立姿勢・静止状態からはじまり, 一つの行動 (シンボル遷移) が終了した時点で

- (a) 転倒によるダメージを受けている.
- (b) エピソード開始から 20 秒以上経過している.
- (c) エピソード内で獲得した本来のタスクの報酬が正で, 土台リンクが接地かつロボットが静止している.

のいずれかを満たしていれば終了とする. (c) のみがタスクの成功であり, この場合のみ前述したように $R_0 = +10$ を与えてから終了する.

3.2 節で述べた基底関数の修正法を適用した DQL+ (DQL+ (manip)) と適用しない DQL+ (DQL+ (no manip)), および Russell らの Sarsa を用いた分離学習 (DSarsa) のそれぞれに跳躍を学習させたところ, **Fig. 15** に示すエピソード-報酬曲線を得た. 基底関数の追加によるパラメタの増加があるにもかかわらず DQL+ (manip) が学習がもっとも速く収束していることから, 跳躍では 3.2 節で述べた関数近似器の問題による学習の

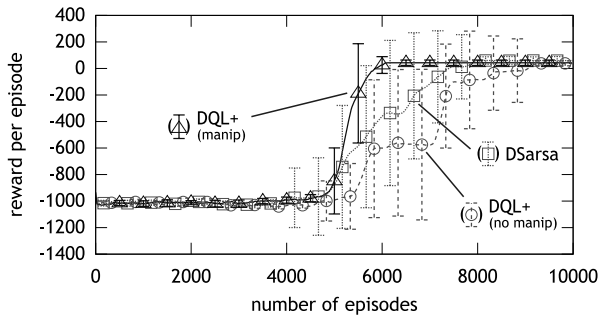


Fig. 15 Results of separative learning the jump task: Each graph shows total rewards per episode (the mean and the ± 1 SD around the mean over 15 runs). The labels of the curves denote the following: DQL+ (manip): learning by DQL+ with the modification algorithm of basis functions mentioned in sec. 3.2, DQL+ (no manip): learning by normal DQL+, DSarsa: learning by the method of Russell *et al.*

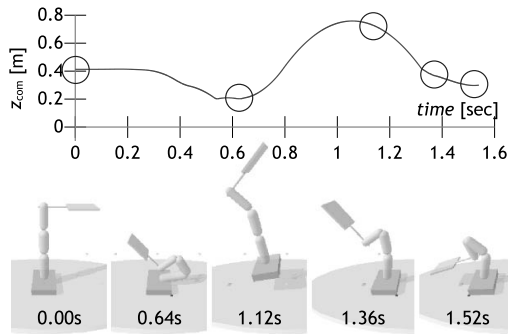


Fig. 16 Trajectory of z_{com} and snapshots of acquired jump motion. Each circle on the trajectory corresponds to the snapshot

遅延が生じており、関数近似器の修正法によってそれが改善されたと考えられる。DQL+ (manip) で最終的に (10,000 エピソード後) 獲得された跳躍の一例を Fig. 16 に示す。

7. 結 論

本稿では回避行動を分離して学習し新たなタスクの学習で再利用するための強化学習手法として、既存の代表的な分離学習手法である Russell らの Sarsa による分離学習 [1] が再利用には適さないことを示し、再利用・分離学習のいずれも可能にするような学習手法 DQL+ を提案した。5 章、6 章の実験から、DQL+ は分離学習手法としては Russell らの手法と同等の性能であったが、再利用の条件下では大きく上回った。特にいくつかの実験では Russell らの手法を用いて再利用を行うとかえって学習性能が悪化する場面があったが、DQL+ では再利用しない場合と比べて性能を向上させることができた。よって分離学習と回避行動の再利用をとともに実現する手法として、DQL+ は Russell らの手法よりも優れていると考えられる。さらに、DQL+ の学習結果から学習に悪影響を及ぼしている関数近似器の基底関数を判定可能であることに着目すると、基底関数の修正法が比較的容易に設計できるといった利点もある。この修正法により跳躍における学習の遅延を改善させることができることを、実験的に示した。しかしながら DQL+ では回避行動を優先するため

分離学習を近似的にしか達成できない場合もあり、分離学習に限ると Russell らの手法の方が優れていると考えられるため、今後分離学習も制限なしに実現できるよう改良する必要がある。

さらに遅いサーブと速いサーブの間の再利用では、二次元連続平面の移動タスクほど性能が向上しなかった。この原因としては状態行動空間が大きい学習が不十分であることも大きい。6.1 節で述べたように土台リンクの接触状態が切り替わる前後で同じ行動を選択したときのふるまいが大きく異なり、ある程度粗く配置された関数近似器の基底関数では両者の状態を明確に区別することが困難であるのが主要因であると考えられる。このようにマルコフ決定過程から逸脱した状況下で安定な学習を行うためには方策勾配法などの導入が必要だと考えられ、このうち再利用の条件を満たす可能性がある方法として「方策オフ型 Natural Actor-Critic 法」[17] などが有望であり、今後検討して行きたい。一方で仮に接触状態の切り替わりに対して安定に学習できたとしても、動作そのものが安定になるとは限らない。つまり土台リンクが振動を始めた場合などに、安定な状態に戻すような行動、あるいは行動の組合せが存在する必要がある。本研究では土台リンクを水平に保つ制御を明示的に行ったが、この方法はロボット依存であり、土台リンクを積極的に使った跳躍 (人間が跳ぶときに足で地面を蹴るような跳躍) ができなくなるなど汎用的とは言いがたく、今後改善を要する。

人間やマウスなどの哺乳類においても異なる報酬のそれぞれに対して個別の学習器を持つ場合があることが、生理学的な実験から示唆されている。例えば Kobayakawa ら [18] は後天的に学習されると考えられていた匂いに対する行動には先天的に備わっているものもあり、それらは別々の神経回路によって処理されていることを実験的に示した。このような知見は本稿で述べた手法の妥当性を支持すると考えられ、さらに「先天的に備わっている学習器」は類似の構造を持つロボット間で回避行動を再利用することに相当するなど、生理学的な知見からロボットの運動学習メカニズムを発展させるアプローチも期待される。

参 考 文 献

- [1] S.J. Russell and A. Zimdars: "Q-decomposition for reinforcement learning agents," the Twentieth International Conference on Machine Learning (ICML 2003), pp.656-663, 2003.
- [2] J. Morimoto and K. Doya: "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," Robotics and Autonomous Systems, vol.36, no.1, pp.37-51, 2001.
- [3] 木村, 山下, 小林: "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.
- [4] J. Zhang and B. Rössler: "Self-valuing learning and generalization with application in visually guided grasping of complex objects," Robotics and Autonomous Systems, vol.47, no.2-3, pp.117-127, 2004.
- [5] 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, vol.42, no.7, pp.814-821, 2006.
- [6] T. Matsubara, J. Morimoto, J. Nakanishi, S.-H. Hyon, J.G. Hale and G. Cheng: "Learning to acquire whole-body humanoid CoM movements to achieve dynamic tasks," ICRA, pp.2688-2693, 2007.
- [7] K. Doya, K. Samejima, K. Katagiri and M. Kawato: "Multiple model-based reinforcement learning," Neural Comput., vol.14,

- no.6, pp.1347–1369, 2002.
- [8] 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, vol.21, no.2, pp.164–171, 2003.
- [9] F. Fernández and M. Veloso: “Probabilistic policy reuse in a reinforcement learning agent,” AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp.720–727, 2006.
- [10] 内部, 銅谷: “複数報酬のもとでの階層強化学習”, 日本ロボット学会誌, vol.22, no.1, pp.120–129, 2004.
- [11] C.J.C.H. Watkins: “Learning from Delayed Rewards,” PhD thesis, Cambridge University, 1989.
- [12] G.A. Rummery and M. Niranjan: “On-line Q-learning using connectionist systems,” Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
- [13] M. Sato and S. Ishii: “On-line EM algorithm for the normalized gaussian network,” Neural Computation, vol.12, no.2, pp.407–432, 2000.
- [14] J. Peng and R.J. Williams: “Incremental multi-step Q-learning,” International Conference on Machine Learning, pp.226–232, 1994.
- [15] T. Flash and N. Hogan: “The coordination of arm movements: an experimentally confirmed mathematical model,” Journal of Neuroscience, vol.5, no.7, pp.1688–1703, 1985.
- [16] R. Smith: Open dynamics engine (ODE), 2006. <http://www.ode.org/>
- [17] 森, 中村, 石井: “重点サンプリングに基づく natural actor-critic 法による効果的なサンプルの再利用”, 電子情報通信学会論文誌 D, vol.J89-D, no.5, pp.954–966, 2006.
- [18] K. Kobayakawa, R. Kobayakawa, H. Matsumoto, Y. Oka, T. Imai, M. Ikawa, M. Okabe, T. Ikeda, S. Itoharu, T. Kikusui, K. Mori and H. Sakano: “Innate versus learned odour processing in the mouse olfactory bulb,” Nature, vol.450, no.7169, pp.503–508, 2007.
- [19] J.N. Tsitsiklis and B.V. Roy: “An analysis of temporal-difference learning with function approximation,” IEEE Transactions on Automatic Control, vol.42, no.5, pp.674–690, 1997.
- [20] R.S. Sutton: “Learning to predict by the methods of temporal differences,” Machine Learning, vol.3, no.1, pp.9–44, 1988.
- [21] N. Ueda, R. Nakano, Z. Ghahramani and G.E. Hinton: “SMEM algorithm for mixture models,” Neural Computation, vol.12, no.9, pp.2109–2128, 2000.

付録 A. 行動価値関数の近似関数の Q(λ)-learning

行動価値関数を近似した式 (12) のパラメタ更新に Q(λ)-learning [14] を用いる場合の更新則を DQL+ に対して導出すると以下ようになる (文献 [19] [20] を参考にした).

Algorithm 2: Update rule of the Q(λ)-learning

```

 $R'_0(x_t, a_t) \leftarrow R_0(x_t, a_t) + wR_1(x_t, a_t)$ 
 $R'_1(x_t, a_t) \leftarrow R_1(x_t, a_t)$ 
for each  $i \in \{0, 1\}$  do
   $e_{i,t} \leftarrow R'_i(x_t, a_t) + \gamma \tilde{V}_{i,t}(x_{t+1}) - \tilde{V}_{i,t}(x_t)$ 
   $e'_{i,t} \leftarrow R'_i(x_t, a_t) + \gamma \tilde{V}'_{i,t}(x_{t+1}) - \tilde{Q}_{i,t}(x_t, a_t)$ 
  for each  $(k, a) \in \mathcal{K} \times \mathcal{A}$  do
     $Tr(k, a) \leftarrow (\gamma\lambda) Tr(k, a)$ 
     $Q_{i,t+1}(k, a) \leftarrow Q_{i,t}(k, a) + \alpha Tr(k, a)e_{i,t}$ 
    if  $i = 0 \wedge Q_{i,t+1}(k, a) < 0$  then  $Q_{i,t+1}(k, a) \leftarrow 0$ 
    if  $i = 1 \wedge Q_{i,t+1}(k, a) > 0$  then  $Q_{i,t+1}(k, a) \leftarrow 0$ 
  end for
for each  $k \in \mathcal{K}$  do
   $Q_{i,t+1}(k, a_t) \leftarrow Q_{i,t+1}(k, a_t) + \alpha e'_{i,t} N_k(x_t)$ 
   $Tr(k, a_t) \leftarrow Tr(k, a_t) + N_k(x_t)$ 
  if  $i = 0 \wedge Q_{i,t+1}(k, a_t) < 0$  then  $Q_{i,t+1}(k, a_t) \leftarrow 0$ 
  if  $i = 1 \wedge Q_{i,t+1}(k, a_t) > 0$  then  $Q_{i,t+1}(k, a_t) \leftarrow 0$ 

```

end for
end for

ここで $Tr(k, a)$ は activity trace と呼ばれ, 行動の履歴を表す. $\tilde{V}_i(x)$ はグリーディ方策における状態価値関数の近似であり, $\tilde{V}_i(x) = \max_{a \in \mathcal{A}} \tilde{Q}_i(x, a)$ で与えられる. この行動価値関数の更新は選択された離散行動 $a \in \mathcal{A}$ が終了した時点で行われる.

ここで注意すべきなのは, 回避行動の学習 ($i = 1$) では $\lambda = 0$ としなければならない点である. なぜなら $\lambda > 0$ だとあるエピソードにおいて取った行動すべてが更新対象となるので, 「回避する必要はないがたまたま負の報酬を得る結果につながった行動」までもが負の方向に更新されてしまうからである. 本来のタスクの学習にはこのほうが学習を早める効果が期待できるが, 回避行動で Q(λ)-learning 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.

付録 B. 関数近似器修正法の基底関数追加

基底関数を追加する際に 4 章や 5.3 節で述べたように行動が基底関数の中心状態 μ に依存する場合, すでに学習した行動価値を保持するために分割もとの基底関数の中心状態を変化させるべきではない. この点に留意して, x が基底関数 k の中心である場合は Fig. 2 (b) のような分割として Sato ら [13] の “unit division” を参考にした分割

$$\mu'_k = \mu_k, \quad \mu_{k_{\text{new}}} = \mu_k + \sqrt{\lambda_{k,1}} u_{k,1} \quad (\text{B.1a})$$

$$\Sigma_k'^{-1} = \Sigma_{k_{\text{new}}}^{-1} = 4\lambda_{k,1}^{-1} u_{k,1} u_{k,1}^\top + \sum_{n=2} \lambda_{k,n}^{-1} u_{k,n} u_{k,n}^\top \quad (\text{B.1b})$$

を用いる ($\lambda_{k,n}$, $u_{k,n}$ は Σ_k の固有値, 固有ベクトルをそれぞれ表し, $n = 1$ は固有値が最大のものを特に表す. この操作は Σ_k の最大固有値に対応する軸で 2 分割にする).

一方 x が二つの基底関数の中間である場合はその状態にほかの基底関数が存在する可能性があるため, $N_k(x)$ が適当な閾値を超える基底関数 $\mathcal{K}_{\text{prn}} = \{k_i\}$ をもとにして新たな基底関数の生成を行う:

$$\mu'_{k_i} = \mu_{k_i}, \quad \Sigma_{k_i}'^{-1} = \sum_{n=1} \lambda_{k_i,n}^{-1} \beta_{k_i,n}^{-2} u_{k_i,n} u_{k_i,n}^\top \quad (\text{B.2a})$$

$$\beta_{k_i,n} \triangleq 1 - \frac{1}{|\mathcal{K}_{\text{prn}}|} \left| u_{k_i,n}^\top \frac{x - \mu_{k_i}}{\|x - \mu_{k_i}\|} \right| \quad (\text{B.2b})$$

$$\mu_{k_{\text{new}}} = x, \quad \Sigma_{k_{\text{new}}}^{-1} = \sum_{k_i \in \mathcal{K}_{\text{prn}}} N_{k_i}(x) \Sigma_{k_i}^{-1}. \quad (\text{B.2c})$$

基底関数の追加を行う際の行動価値関数の学習パラメタについては, 式 (B.1) の場合分割対象 k のパラメタ $Q_i(k, a)$ を $i \in \{0, 1\}$, $a \in \mathcal{A}$ についてコピーし, 式 (B.2) の場合は $N_{k_i}(x)$ の重み付け和を初期値とする. ただしこのままだと分離誤差までも引き継いでしまうため, 分割にかかわるすべてのパラメタについて

for each $a \in \mathcal{A}$:

if $\prod_{i \in \{0,1\}} Q_i(k,a) \neq 0$: $Q_0(k,a) \leftarrow 0$, $Q_1(k,a) \leftarrow 0$

によって分離誤差を発生させるパラメタをゼロに初期化する。これはすべてのパラメタをゼロに初期化する場合に比べてはるかに効率的である。

付録 C. ダイナミクスの学習

ロボットのダイナミクスは、

$$A_s(\xi)\ddot{\xi} + b_s(\xi, \dot{\xi}) = \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} \quad (\text{C.3})$$

によって表されるとする。ここで $\xi \in \mathbb{R}^{D_x \times 1}$ はベースリンク(土台)の位置・姿勢と関節角 $q \in \mathbb{R}^{D_q \times 1}$ をまとめたベクトル、 $u \in \mathbb{R}^{D_q \times 1}$ は関節トルク、 $A_s(\xi)$ は慣性行列、 $b_s(\xi, \dot{\xi})$ はコリオリ力・遠心力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[†]。このダイナミクスを

$$\frac{\dot{q}(t+\Delta t) - \dot{q}(t)}{\Delta t} = B \begin{bmatrix} \xi \\ \dot{\xi} \end{bmatrix} + d + Au \quad (\text{C.4})$$

のように線形化・離散化したもの ($A \in \mathbb{R}^{D_q \times D_q}$, $B \in \mathbb{R}^{D_q \times 2D_x}$, $d \in \mathbb{R}^{D_q \times 1}$) を NGnet の各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。式 (C.3) の非線形項は ξ と $\dot{\xi}$ のみに依存するから、ロボットの状態は $x = [\xi^T \ \dot{\xi}^T]^T \in \mathbb{R}^{2D_x \times 1}$ である。

NGnet のパラメタはランダムにロボットを動かしてサンプルを取り、それをもとに EM アルゴリズムで最適化する。具体的

には Ueda らの SMEM [21] 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。6章の運動学習の実験では獲得されるユニット数が100個前後になるようにパラメタを設定した。

付録 D. ロボットの行動

4章で導入したロボットの行動の詳細について述べる。関節角 q の躍度最小軌道は

$$\int_0^{T_f} \left\| \frac{d^3 q}{dt^3} \right\|^2 dt \rightarrow \min, \quad (\text{D.5})$$

の解であり、変分法を用いて解くと5次方程式が得られる。すなわち関節 j の関節角軌道を $q_j(t) = \sum_{m=0}^5 c_{jk} t^m$ として境界条件を $q_j(0) = q_j^i, \dot{q}_j(0) = \dot{q}_j^i, \ddot{q}_j(0) = \ddot{q}_j^i, q_j(T_f) = q_j^f, \dot{q}_j(T_f) = \dot{q}_j^f, \ddot{q}_j(T_f) = \ddot{q}_j^f$ とすることによって係数 c_{jk} を求める。具体的には境界条件として現在の関節状態 (q_j^i, \dot{q}_j^i) および目標シンボルの関節状態 (q_j^f, \dot{q}_j^f) を用い、加速度はゼロ $\ddot{q}_j^i = \ddot{q}_j^f = 0$ とする。行動の時間幅 T_f はその大きさによって動作の俊敏さが変化するため、 $\mathcal{I} = \{0.2\|q^f - q^i\|, 0.4\|q^f - q^i\|, 0.8\|q^f - q^i\|\}$ の3段階から選択することにした。目標関節角と現在の関節角の差のノルム $\|q^f - q^i\|$ を掛けているのは目標シンボルの違いによる動作の俊敏さの差異 (T_f が同じなら遠いシンボルほど俊敏な動作になる) を軽減するためである。したがって強化学習エージェントの行動集合は $\mathcal{A} = \mathcal{K} \times \mathcal{I}$ であり、離散行動である。なお \mathcal{I} を連続にすることも可能だが、本稿では簡単のため離散とした。

[†]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 (C.3) で表されるシステムなら、本手法はそのまま適用可能である。



山口明彦 (Akihiko Yamaguchi)

2006年京都大学工学部電気電子工学科卒業。同年奈良先端科学技術大学院大学情報科学研究科博士前期課程入学。ロボットの運動学習の研究に従事。電子情報通信学会会員。(日本ロボット学会学生会員)



杉本徳和 (Norikazu Sugimoto)

2001年金沢大学工学部機能機械工学科卒業。2006年奈良先端科学技術大学院大学博士課程修了(工学博士)。同年ATR脳情報研究所研究員。強化学習の研究に従事する。



川人光男 (Mitsuo Kawato)

1976年東京大学理工学部物理学科卒業。1981年大阪大学博士課程修了。同年助手。1987年同講師。1988年(株)ATRに移る。2003年よりATR脳情報研究所所長。2004年よりATRフェロー。IEICEフェロー。およびJST国際共同研究『計算脳プロジェクト』研究総括兼任。1996~2001年『川人学習動態脳プロジェクト』総括責任者兼任。1994, 2000, 2002, 2004, 2007年より金沢工業大学、奈良先端大連携講座、阪大生命機能研究科、生理学研究所、京都府立医大の客員教授。2006年より富山県立大学の特任教授。計算論的神経科学の研究に従事。米澤賞、大阪科学賞、科学技術長官賞、塚原賞、時実賞、中日文化賞、志田林三郎賞、朝日賞などを受賞。著書に「脳の仕組み」、「脳の計算理論」等。HFSP Journal 編集委員、日本神経学会理事。