社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

回避行動の再利用メカニズムを備えた強化学習のための関数近似器修正手法と多関節ロボットへの応用

山口　明彦[†,††]　　杉本　徳和[††]　　川人　光男[††,†]

† 奈良先端科学技術大学院大学　〒 630–0192 奈良県生駒市高山町 8916–5
†† ATR 脳情報研究所　〒 619–0288 けいはんな学研都市 光台 2–2–2
E-mail: †akihiko-y@is.naist.jp, ††{xsugi,kawato}@atr.jp

あらまし　強化学習などの学習手法をロボットの運動学習に適用する際に問題となる学習コスト（転倒によるダメージなど）を軽減する一手法として，我々はあるタスクの学習中に回避行動を分離して学習しこれをほかのタスクの学習で再利用する手法を強化学習の枠組で提案，4 リンク程度の土台非固定型ロボットへの応用を行ってきた [1]．本稿では分離学習における分離性能を向上させることを目的として基底関数を修正する手法を提案し，運動学習における有効性を示す．さらに回避行動を再利用することによって運動学習における転倒ダメージが軽減するかを検討する．
キーワード　運動学習，強化学習，再利用，回避行動，跳躍，サーブ

A Modification Algorithm of Function Approximator for the Reinforcement Learning with Reusing Mechanism of Avoidance Actions

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

Akihiko YAMAGUCHI[†,††], Norikazu SUGIMOTO[††], and Mitsuo KAWATO[††,†]

† Nara Institute of Science and Technology　8916–5, Takayama, Ikoma, Nara 630–0192, JAPAN
†† ATR Computational Neuroscience Laboratories　2–2–2 Hikaridai, Seika, Souraku, Kyoto 619–0288, JAPAN
E-mail: †akihiko-y@is.naist.jp, ††{xsugi,kawato}@atr.jp

Abstract　Applying a learning method, such as reinforcement learning, to learning motions of multi-link robots requires large cost, such as damage from falling down. To overcome this problem, we proposed a reusing mechanism for reinforcement learning where the avoidance actions, such as not to fall down, are learned separately from primary actions, then they are reused in learning new tasks [1]. A method to apply it to learning whole-body motions of 4-link robot whose base is not fixed to a ground was also developed. In this paper, we propose a new method to modify basis functions of a function approximator of an action value function to improve the separative performance, and demonstrate the method works effectively in learning whole-body motions of a multi-link robot. Furthermore, we investigate a learning cost of damage from falling down in learning whole-body motions is reduced by reusing avoidance actions.
Key words　motion learning, reinforcement learning, reusing, avoidance actions, jumpping, serve

1. 再利用メカニズムの導入による強化学習の性能向上

ロボットがより自律的・臨機応変に行動するために不可欠な「目的から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，高次元の状態空間を扱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やコスト（転倒によるダメージなど）が掛かることがあげられる．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 [2] 階層構造を導入したり [3], [4] することでこの問題に対処している．前者の方法は効率的に状態の次元を減らすが問題の依存性が強く，後者の方法としては状態空間を分割するものが多い．

これらに対し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学習の性能を向上させることを目指しており，文献 [1] において回避行動の再利用メカニズムを強化学習に導入，多関節ロボットの運動学習への応用方法を示した．ここで言う回避行動とは転倒につながる行動や人を傷つける行動など，避けられるべき行動のことである．

文献 [1] では学習させたいタスク（以下「本来のタスク」）に

対して与える正の報酬と回避行動に対して与える負の報酬のそれぞれを分離学習することで再利用を実現するアプローチを取り，各報酬の学習に Q-learning を用いる場合を検討した．本稿では (1) 文献 [1] で仮定した「本来のタスクと回避行動が完全に分離可能」という条件が満たされない場合の対処方法を示し，(2) 連続空間に拡張する際に用いる基底関数を分離学習した行動価値関数から評価できる「分離誤差」を基準に修正することで回避行動が本来のタスクに与える悪影響を改善する手法を提案，(3) 多関節ロボットの運動学習に応用する．(2) の手法は 2 次元連続平面での移動タスク及び運動学習のいずれにおいても良好な結果を示したが，回避行動を再利用してほかの運動を学習する場合に関しては 2 次元連続平面ほどの結果が得られなかった．本稿の最後にこの原因を議論する．

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（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，$R(s,a)$ は本来のタスクの報酬と回避行動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．

ここでは回避行動の分離学習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 [2]）ことができなくなるが，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ことから，正負の報酬表現で本来のタスクと回避行動の分離学習が比較的容易になり回避行動の再利用がしやすくなるという利点を優先する．

この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（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．このようなアプローチのうちもっともシンプルなもののひとつは，学習器として Q-learning を用い，個々の学習器が獲得した行動価値関数 $Q_i(s,a)$, $i \in \{0,\,1\}$ の総和 $\sum_i Q_i(s,a)$ によって学習器を合成するというものである．我々は既に [1] でこの方法を検討したが, [1] の方法だと正の報酬と負の報酬が同時に値を持つときなど獲得される行動価値関数が分離可能条件を満たさなくなる場合が存在する．以下ではこのような問題が発生する原因を議論し，回避するための行動価値関数の更新方法を導出，シミュレーション実験においてその有効性を確認する．

2.1 Q-learning を用いた正と負の報酬の分離学習

個々の報酬関数 $R_i(s,a)$, $i \in \{0,\,1\}$ 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．ここで $\mathcal{S}$ 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' へ遷移する確率，γ は割引率，* は最適行動価値関数であることを表す．行動価値関数をテーブルルックアップ $Q_i(s,a)$ を用いて学習する場合の更新則は α をステップサイズパラメタとすると $i = 0,\,1$ について

$$Q_{i,t+1}(s_t,a_t) \leftarrow Q_{i,t}(s_t,a_t) + \alpha\big\{R_i(s_t,a_t) + \gamma \max_{a \in \mathcal{A}} Q_{i,t}(s_{t+1},a) - Q_{i,t}(s_t,a_t)\big\} \tag{2}$$

となる．学習エージェントの行動選択は合成した行動価値関数 $\sum_i Q_i(s,a)$ をもとに Boltzmann 選択などで行う．

2.2 分離学習が可能な条件と分離誤差

式 (1) を $i = 0,\,1$ について足し合わせると

$$\sum_{i \in \{0,\,1\}} Q_i^\star(s,a) = \sum_{i \in \{0,\,1\}} R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \sum_{i \in \{0,\,1\}} \max_{a' \in \mathcal{A}} Q_i^\star(s',a'). \tag{3}$$

が得られる．ここから

$$^\forall s \in \mathcal{S}:$$
$$\sum_{i \in \{0,\,1\}} \max_{a \in \mathcal{A}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} \sum_{i \in \{0,\,1\}} Q_i^\star(s,a) \tag{4}$$

が成立するとき式 (3) は行動価値関数の和 $\sum_i Q_i^\star(s,a)$ が報酬関数の和 $\sum_i R_i(s,a)$ 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，分離学習した結果から通常の Q-learning によって得られる行動価値関数を合成できることがわかる．

この条件式は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して，どの程度この条件が満たされないかを評価する「分離誤差」を次式で定義する：

$$C(s) \triangleq \left(\sum_{i \in \{0,1\}} \max_{a' \in \mathcal{A}} Q_i(s,a') - \max_{a' \in \mathcal{A}} \sum_{i \in \{0,1\}} Q_i(s,a') \right)^2. \tag{5}$$

2.3 式 (4) を満たす行動価値関数

本来のタスクの行動価値関数と回避行動の行動価値関数が同時に値を持たないとき，すなわち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には式 (4) が成立する．厳密には

$$^\forall (s,a) \in \mathcal{S} \times \mathcal{A}:$$
$$\prod_{i \in \{0,\,1\}} Q_i^\star(s,a) = 0 \wedge Q_0^\star(s,a) \geqslant 0 \wedge Q_1^\star(s,a) \leqslant 0 \tag{6}$$

が成立するような行動価値関数に対しては式 (4) が成立する．以下ではこの条件を満たすように行動価値関数を更新する方法について述べる．なおこの逆は成立せず式 (4) と等価な行動価値関数は $^\forall s \in \mathcal{S} : \arg\max_{a \in \mathcal{A}} Q_0^\star(s,a) \bigcap \arg\max_{a \in \mathcal{A}} Q_1^\star(s,a) \neq \emptyset$ であるが，これが成立するように更新則を定めるのは困難なので式 (6) を学習するようにしている．

2.4 分離学習のための行動価値関数更新則

文献 [1] では転倒などの回避行動が本来のタスクから完全に分離されると考えて式 (6) を満たし再利用可能であるとしたが，より複雑な問題設定（例えば後述するサーブ動作など）ではこれを満足しない回避行動も存在する．状態集合 s が離散の場合に行動価値関数が式 (6) の条件を満たさなくなる原因として，

(a) 報酬 $R_i(s_t,a_t)$, $i = 0,\,1$ が同時に 0 以外の値を持つとき
このような状態 s_t，行動 a_t については $\prod_i Q_{i,t+1}(s_t,a_t) \neq 0$ となるように更新される．

(b) 負の報酬を受け取り，遷移先の最大行動価値が正のとき
$R_0(s_t,a_t) = 0$, $R_1(s_t,a_t) < 0$ でかつ $\max_{a \in \mathcal{A}} Q_{0t}(s_{t+1},a) > 0$ の場合，式 (2) によって $Q_{0t+1}(s_t,a_t)$ は正に更新される一方で $Q_{1t+1}(s_t,a_t)$ は負に更新され，$\prod_i Q_{i,t+1}(s_t,a_t) \neq 0$ となる．

などが考えられる．

Russell ら [5] は Sarsa を使うことでこの問題を解決できると主張しているが，このときに学習される各々の行動価値関数は合成された行動価値関数から得られる方策に依存する．したがって回避行動を学習する行動価値関数 Q_1 はタスクに依存してしまい，再利用には適していないと考えられる．

そこで方策に依存せず最適行動価値関数を近似する Q-learning において式 (6) が満たされるように更新則を修正する．まず式 (2) の更新に使う報酬を

$$R_0'(s_t,a_t) \leftarrow R_0(s_t,a_t) + wR_1(s_t,a_t) \tag{7}$$
$$R_1'(s_t,a_t) \leftarrow R_1(s_t,a_t) \tag{8}$$

のように変更し（w は適当な正の定数(注1)），さらに式 (2) の更新を行った後に

$$\texttt{if} \quad Q_{0t+1}(s_t,a_t) < 0: \quad Q_{0t+1}(s_t,a_t) \leftarrow 0 \tag{9}$$
$$\texttt{if} \quad Q_{1t+1}(s_t,a_t) > 0: \quad Q_{1t+1}(s_t,a_t) \leftarrow 0 \tag{10}$$

(注1)：$|wR_1|$ が $|R_0|$ より十分大きくなるように選択．$|R_1|$ が $|R_0|$ より大きい場合 1 でよいが，両者の差が極端だと後述の Q(λ)-learning によって行動履歴の価値関数が大幅に減らされるため，10 倍程度の違いになるように w で調整．

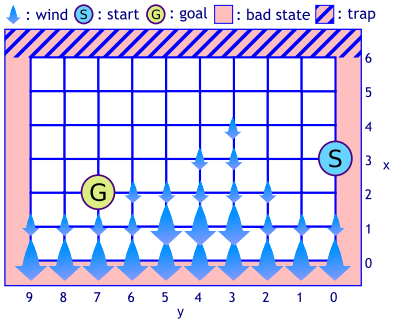


図 1　格子状空間における移動タスク.
Fig. 1　Moving task in the grid world.

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．これにより (a) や (b) の場合に式 (7) を用いると $|wR_1|$ が $|R_0|$ より十分に大きい場合 $Q_{0t+1}(s_t, a_t) \leqslant 0$ となり，式 (9),(10) によって式 (6) が満たされるようになる．$R_0(s_t, a_t) \geqslant 0$, $R_1(s_t, a_t) = 0$ の場合は行動価値関数の影響に何ら影響を及ぼさない．この方法でいかなる場合にも式 (4) が満たされるようになるわけではないが，本研究で想定している回避行動は必ず避けられるべきものであるので上記更新則で実用に耐え得るものと考えられる．

2.5　ステップコスト

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，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上述の更新則を用いれば問題は生じない．

2.6　格子状空間でのシミュレーション

上述した更新則によって回避行動が適切に分離して学習されることを確認するため，図 1 に示すような 7×10 の格子状空間における移動タスクに適用した．学習エージェントの行動は上下左右，スタート S からゴール G に移動できれば正の報酬 1 が与えられ，格子の外側 (“bad state”) では大きい負の報酬 -10 が与えられる（転倒などに相当）．矢印 (“wind”) はエージェントの行動を変化させる「風」（小さい矢は $+1$，大きい矢は $+2$）を表しておりどのような行動を取っても必ず “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）．さらに “bad state” の上部 (“trap”) は負の報酬に加えて正の報酬 $+2$ も与えられる状態であり，2.4 節の (a) に該当するため同節で述べた行動価値関数の更新則を使わないと格子上部をうろつく解に収束してしまう．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する．なおエージェントの 1 回の行動につき -0.01 の報酬を与えている．

比較のため通常の Q-learning (QL)，2.4 節の更新則を適用しない分離学習 (DQL(1))，及び適用した分離学習 (DQL(2)) を使って学習させたところ図 2 に示すエピソード–報酬曲線を得た（報酬はエピソード内の合計）．この結果から 2.4 節の更新則を適用した分離学習 (DQL(2)) は通常の Q-learning (QL) と同程度の性能であることがわかるが，2.4 節の更新則を使わない分離学習 (DQL(1)) は前述した “trap” の効果によって最適解に収束できていない．なお DQL(1) で一度報酬が上昇しているのは学習途中でのランダムなサンプリングによって “trap” から抜け出てゴール G に到達しているためである．

次にスタートとゴールの位置を変えて上のシミュレーションで最終的に（3000 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学習を行ったところ，図 3 の結果を得た．ここでも通常の Q-learning （QL，再利用は行わない），DQL(1), DQL(2) を比較している．このグラフから再利用によって学習の初期段階で “bad state” から受ける負の報酬が大幅に減少していることがわかる．なお 2.4 節の更新則を適用しない分離学習 (DQL(1)) でも再利用に成功しているのは，学習の初期段階で “trap” が回避されているためである．

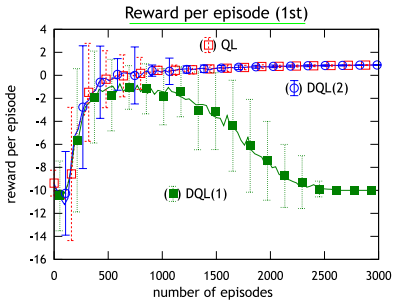


図 2　エピソードごとの累積報酬（100 回の実験の平均と標準偏差）．QL は通常の Q-learning，DQL(1) は単純な Q-learning を使った分離学習，DQL(2) は 2.4 節の更新則を使った分離学習．
Fig. 2　Cumulative rewards per episode (the mean and the ± 1 standard deviation around the mean over 100 runs). QL is a normal Q-learning, DQL(1) is a separative learning by normal Q-learning, DQL(2) is a separative learning with the update rule of section 2.4.

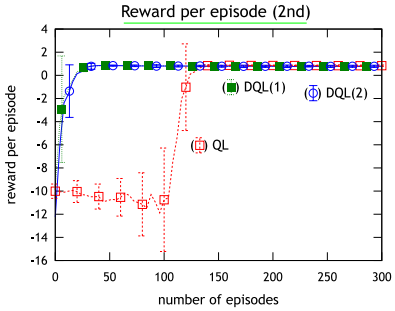


図 3　エピソードごとの累積報酬（100 回の実験の平均と標準偏差）．QL, DQL(1), (2) は図 2 と同様．DQL(1), (2) は回避行動を再利用した場合の学習結果.
Fig. 3　Cumulative rewards per episode (the mean and the ± 1 standard deviation around the mean over 100 runs). QL, DQL(1), (2) denotes the same update rule as Fig. 2. DQL(1) and (2) are reusing the avoidance actions.

3.　分離誤差に基づく基底関数修正法

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，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関わらず，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．本節では関数近似器としてロボットの運動学習でもしばしば用いられる（例えば [3]）正規化ガウシアンネットワーク [6] (Normalized Gaussian Network，以下 NGnet）を想定し，この問題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

$$y \approx f(x) \triangleq \sum_s N_s(x)\big(W_s x + b_s\big) \tag{11}$$

$$N_s(x) \triangleq \frac{G(x; \mu_s, \Sigma_s)}{\sum_{s'} G(x; \mu_{s'}, \Sigma_{s'})} \tag{12}$$

で近似する．ここで W_s, b_s 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，$N_s(x)$ 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．

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

$$Q_i^*(x, a) \approx \tilde{Q}_i(x, a) \triangleq \sum_{s \in \mathcal{S}} \sum_{a' \in \mathcal{A}} N_s(x) \delta_{a'}(a) Q_i(s, a') \tag{13}$$

によって行動価値関数が近似されているとする．なお $\delta_{a'}(a)$ 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

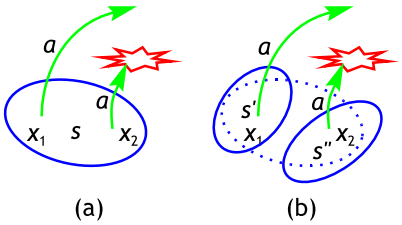


図 4　分離学習が可能な条件を満足するように学習できない基底関数と行動の例 (a)，及び基底関数の分割による改善 (b)．楕円は基底関数（正規化ガウス関数），a は行動をそれぞれ表す．

Fig. 4　An example of a basis function that cannot learn to satisfy the condition of separative learning (a), and improvement by dividing the basis function (b). Each ellipse denotes a basis function (normalized Gaussian), and a denotes an action.

ブルであり，これが学習対象のパラメタとなる．

3.2　分離誤差に基づく基底関数修正法

一般的に基底関数の数が多いと学習が遅くなるため，関数近似器はある程度「粗く」状態空間に配置される．これが原因で学習した行動価値関数が分離学習可能な条件式 (4) を満たさなくなることがある．例えば各基底関数からの出力がほぼ同じである 2 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などである（図 4(a)）．

分離誤差が分離学習した行動価値関数を使って評価できることから，分離誤差を計算することで状態空間のどこでこの問題が起きているかを調べられる．図 4(a) のような場合は基底関数を図 4(b) のように分割すればこの問題が解決できるから，「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．

$\mathcal{X} \leftarrow$ 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 $C_x \geqslant threshold$ **then**
x に基底関数を追加 ※2
end if
end for

※1, 2 及びこのアルゴリズムの実行タイミングについては任意性がある．本稿の具体的な実装例については付録 1. に記載する．

3.3　2 次元連続平面でのシミュレーション

本節で述べたアルゴリズムの有効性を検証するために，図 5 に示す 2 次元連続平面での移動タスクに適用した．状態は位置 (x_1, x_2) と速度，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” に移動できれば正の報酬 1 が与えられ（ただしゴール時のずれに応じて若干減少させる），“bad state” に遷移するか “wall” にぶつかると発生する 1 から 2 のダメージ（大きさは “bad state” に留まった時間や “wall” に接した時間で決まる）を -100 倍したものが負の報酬として与えられる．行動価値関数を近似する NGnet のパラメタ（中心 μ と分散 Σ）は図 5 の楕円に示すように与えた．行動は基底関数の中からひとつ選び，その中心を目標として加速度のノルムの時間積分を最小にするように軌道計画，その微分を制御入力とする離散行動である．この行動集合 $\mathcal{A}$ は前述の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．

学習は式 (13) に Q(λ)-learning を適用して得られる更新則（詳細は [1] 参照）に 2.4 節で述べた方法を追加したものを用いて行う．エピソードの終了後に本節で述べた基底関数の修正法を適用した場合 (“manip”) としなかった場合 (“no manip”) のそれぞれについてエピソードごとの正の報酬及び学習開始からの累積ダメージをエピソードに対してプロットしたところ図

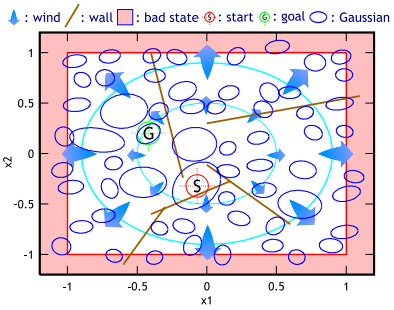
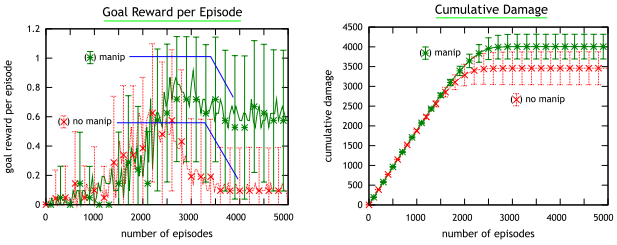


図 5　2 次元連続平面での移動タスク．楕円は基底関数（正規化ガウス関数）を表す．

Fig. 5　Moving task in the 2-dimensional continuous plane. Each ellipse denotes a basis function.



(a) 正の報酬．　(b) 学習開始からの累積ダメージ．

図 6　再利用しない場合の結果．“no manip” は基底関数の修正法を使わない場合の分離学習，“manip” は使った場合の分離学習による結果をそれぞれ表す．

Fig. 6　Result of learning without reusing. “manip”/“no manip” denotes the separative learning with/without using the modification of basis functions.

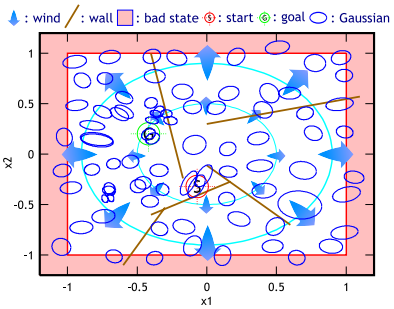


図 7　3.2 節のアルゴリズムを適用して得られた基底関数．

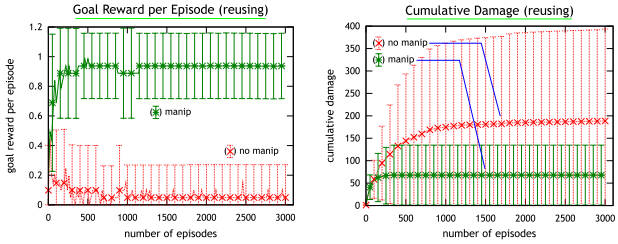
Fig. 7　Obtained basis functions by the algorithm of section 3.2.

6(a)，6(b) を得た．スタート付近など図 4(a) のような問題が生じやすい領域が存在するためゴールを目指さずにある安全な場所に留まる解に収束する可能性が高くなるが，正の報酬のグラフから基底関数の修正を行うことによってこの問題が解消されていることがわかる．一方で累積ダメージは基底関数の修正を行った方がやや大きい．これは基底関数を追加することによって学習すべきパラメタが増加したためである．図 7 の楕円は最終的に得られた NGnet の基底関数を表しており，スタート付近などで基底関数が生成されていることがわかる．

次にゴールの位置を変更して回避行動の行動価値関数を再利用し学習を行ったところ，エピソードごとの正の報酬及び学習開始からの累積ダメージのグラフ図 8(a)，8(b) を得た．基底関数の修正を行っていない場合だと学習の初期段階から既に図 4(a) のような問題に陥っていると考えられ，正の報酬が大幅に小さい．累積ダメージについても基底関数の修正を行った方が小さくなっているが，これは回避行動の行動価値関数がより正確に近似できているためであると考えられる．

4.　多関節ロボットへの応用

本節では前節までの手法を多関節ロボットの全身運動学習に



(a) 正の報酬. (b) 学習開始からの累積ダメージ.

図 8 回避行動を再利用して学習を行った場合の結果. “no manip” は基底関数の修正法を使わない場合の分離学習, “manip” は使った場合の分離学習による結果をそれぞれ表す.

Fig. 8 Result of learning with reusing the avoidance actions. “manip”/“no manip” denotes the separative learning with/without using the modification of basis functions.

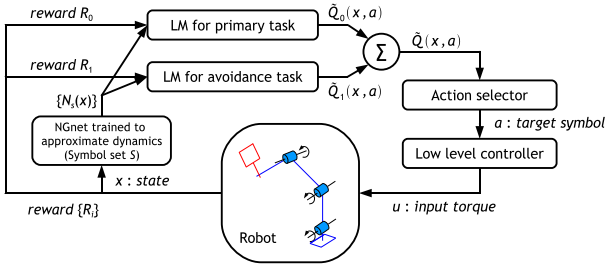


図 9 全身運動を学習するアーキテクチャ. LM は学習器を表す.

Fig. 9 Architecture for robots to learn whole-body motions. LM stands for Learning Module.

適用し，回避行動の再利用が行えるか検証する．ここでは全身運動を「ロボットの一般化座標(注2)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（ボールの位置を固定した）サーブ，起き上がり運動 [3] 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（図 9）：

(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，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 からひとつのシンボルを選びその基底関数の中心（平均）を目標とした制御を行うというもので，離散行動である．このときの制御は関節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 [7]，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 2 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離散にしているため精密な動作は実現しにくいが，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．

(注2)：系を記述するために必要な最小の次元数の座標.

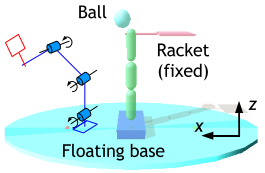


図 10 4 リンク 3 関節のロボット.

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

ロボットのダイナミクスの学習は EM アルゴリズムを使って行い，行動価値関数の学習は Q(λ)-learning を用いる（詳細は [1] 参照）.

4.1 シミュレーション実験

この手法を 4 リンク 3 関節土台非固定型ロボット（図 10）の全身運動学習に適用する．学習させる全身運動は跳躍，サーブの 2 種類であり，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．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うのを抑えるために，ダメージ D を次式によって 1 から 2 の間に抑えこれを -1000 倍したものを回避行動の報酬 R_1 として用いている．

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

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

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

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，ダメージ D はシンボル遷移が始まる前にゼロに初期化される．

4.2 跳　　躍

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：

$$R_0 = 200\left[z_{\text{com-max}} - z^{\text{old}}_{\text{com-max}}\right]_+ \tag{17}$$

ここで $z_{\text{com-max}}$ はシンボル遷移内の最大重心位置，$z^{\text{old}}_{\text{com-max}}$ 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，$z^{\text{old}}_{\text{com-max}}$ 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．

ひとつのエピソードは直立姿勢・静止状態からはじまり，ひとつの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) エピソード内で正の報酬を獲得していて，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る．

3.2 節で述べた基底関数の修正法を用いた場合 (“manip”) と用いなかった場合 (“no manip”) のそれぞれについて学習を行わせたときの正の報酬の変化をエピソードに対しプロットしたところ図 11 に示す結果が得られた．基底関数の修正法を用いた方が得られた動作の性能が向上していることから，3.2 節で述べた問題がロボットの運動学習でも起きており，基底関数を修正することでこれが解決されたと言える．

4.3 跳躍とサーブの間の再利用

回避行動が再利用できることを示すために跳躍で学習した回避行動の行動価値関数を再利用してサーブを学習，再利用しない場合と比較した（サーブ動作の報酬はボールの速さが水平方

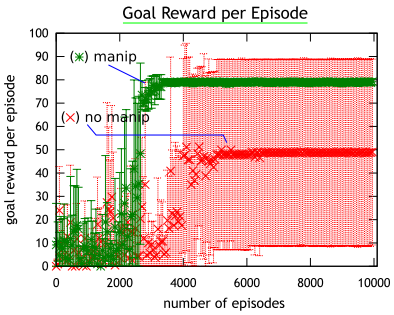


図 11　エピソードごとの正の報酬.
Fig. 11　Positive rewards per episode.

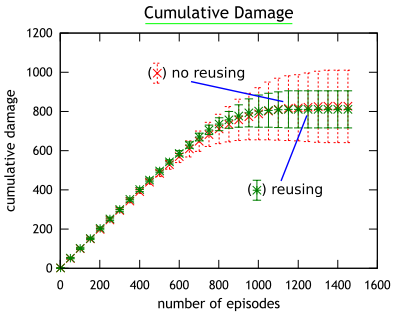


図 12　学習開始からの累積ダメージ.
Fig. 12　Cumulative damages.

向に大きいほど高くなるように与える．詳細は [1] を参照．回避行動の報酬は 4.1 節のものを用いる）．この結果の累積ダメージは図 12 のようになり，再利用した場合 (“reusing”) は再利用しない場合 (“no reusing”) よりも累積ダメージの分散が小さくなっていることや平均がわずかに小さくなっていることがわかるが，大差はない．したがって回避行動の再利用によるダメージの軽減に失敗していると言える．

5.　議　　論

本稿では [1] で提案した行動価値関数の分離学習による回避行動の再利用方法における問題点と改善方法を示し，格子状空間の移動タスク，2 次元連続平面の移動タスク，多関節ロボットの運動学習のそれぞれで検証した．この結果から提案した分離学習手法はいずれの場合でも有効に機能することが確認できた．しかしながらこのとき分離学習した回避行動の行動価値関数を再利用してほかのタスクを行わせる実験では，格子状空間・2 次元連続平面の場合は回避されるべき行動を選択したことによる負の報酬を軽減することが確認できたが，多関節ロボットの運動学習の場合は再利用しなかった場合とほとんど差が見られなかった．

この原因としては (1) 一度サーブで空振りすると立ち位置がわずかにずれ，再びもとの位置に戻ってサーブを打つのはかなり困難であること，(2) ロボットの土台リンクの接触条件が変化したときに生じる制御の不安定さが主に考えられる．もし前者の要因が強ければほかの全身運動学習では再利用に成功する可能性があり，起き上がり運動 [3] に似たタスクで検討したが期待した性能は得られなかった．よってこの問題を解決するには図 9 に示したロボットの運動学習への適用方法を改善する必要がある．

付　　録

1.　分離誤差に基づく基底関数修正法の詳細

3.2 節で述べたアルゴリズムの具体的な実装例を以下に示す．

分離誤差を評価する状態集合

分離誤差を評価する状態としては様々な候補が考えられるが，連続状態空間においては分離誤差も近似となるため，また計算量を軽減するためにある程度候補を限定した方がよい．そこで分離誤差を評価する状態集合として，基底関数の中心状態 $\{\mu_s \mid s \in \mathcal{S}\}$ と 2 つの基底関数の中間 $\{(\mu_{s_1} + \mu_{s_2})/2 \mid s_1,\, s_2 \in \mathcal{S}\}$ からランダムに選択することにする．

基底関数の追加方法

基底関数を追加する際に 3.3 節や 4. 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，既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 s の中心である場合は図 4(b) のような分割として Sato ら [6] の “unit division” を参考にした分割

$$\mu'_s = \mu_s, \quad \mu_{s_{\mathrm{new}}} = \mu_s + \sqrt{\lambda_{s1}}u_{s1} \tag{A·1a}$$

$$\Sigma'^{-1}_s = \Sigma^{-1}_{s_{\mathrm{new}}} = 4\lambda^{-1}_{s1}u_{s1}u^{\top}_{s1} + \sum_{n=2}\lambda^{-1}_{sn}u_{sn}u^{\top}_{sn} \tag{A·1b}$$

を用いる（λ_{sn}, u_{sn} は Σ_s 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_s の最大固有値に対応する軸で 2 分割にする）．

一方 x が 2 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，$N_s(x)$ が適当な閾値を超える基底関数 $\mathcal{S}_{\mathrm{prn}} = \{s_i\}$ をもとにして新たな基底関数の生成を行う：

$$\mu'_{s_i} = \mu_{s_i}, \quad \Sigma'^{-1}_{s_i} = \sum_{n=1}\lambda^{-1}_{s_in}\beta^{-2}_{s_in}u_{s_in}u^{\top}_{s_in} \tag{A·2a}$$

$$\beta_{s_in} \triangleq 1 - \frac{1}{|\mathcal{S}_{\mathrm{prn}}|}\left|u^{\top}_{s_in}\frac{x - \mu_{s_i}}{\|x - \mu_{s_i}\|}\right| \tag{A·2b}$$

$$\text{for} \quad s_i \in \mathcal{S}_{\mathrm{prn}}$$

$$\mu_{s_{\mathrm{new}}} = x, \quad \Sigma^{-1}_{s_{\mathrm{new}}} = \sum_{s_i \in \mathcal{S}_{\mathrm{prn}}} N_{s_i}(x)\Sigma^{-1}_{s_i}. \tag{A·2c}$$

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 (A·1) の場合分割対象 s のパラメタ $Q_i(s,a)$ を $i \in \{0,1\}$, $a \in \mathcal{A}$ についてコピーし，式 (A·2) の場合は $N_{s_i}(x)$ 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，分割に関わるすべてのパラメタについて

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

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

によって式 (6) を満たさない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．

実行のタイミング

このアルゴリズムを実行するタイミングとしては行動価値関数の更新が終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本稿では後者を採用する．

文　　献

[1] 山口, 杉本, 川人：“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”, 第 25 回日本ロボット学会学術講演会, p. 3N37 (2007).

[2] J. Zhang and B. Rössler: “Self-valuing learning and generalization with application in visually guided grasping of complex objects”, Robotics and Autonomous Systems, **47**, 2-3, pp. 117–127 (2004).

[3] J. Morimoto and K. Doya: “Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning”, Robotics and Autonomous Systems, **36**, pp. 37–51(15) (31 July 2001).

[4] K. Doya, K. Samejima, K. Katagiri and M. Kawato: “Multiple model-based reinforcement learning”, Neural Comput., **14**, 6, pp. 1347–1369 (2002).

[5] 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).

[6] M. Sato and S. Ishii: “On-line EM algorithm for the normalized gaussian network”, Neural Computation, **12**, 2, pp. 407–432 (2000).

[7] T. Flash and N. Hogan: “The coordination of arm movements: an experimentally confirmed mathematical model”, Journal of Neuroscience, **5**, 7, pp. 1688–1703 (1985).