論　文

選択的不感化ニューラルネットを用いた連続状態行動空間における Q 学習

小林　高彰[†a)]　　澁谷　長史[†b)]　　森田　昌彦[†c)]

Q-Learning in Continuous State-Action Space by Using a Selective Desensitization Neural Network

Takaaki KOBAYASHI[†a)], Takeshi SHIBUYA[†b)], and Masahiko MORITA[†c)]

あらまし　連続な状態空間において Q 学習を行う場合，価値関数の近似が不可欠であるが，強化学習において通常用いられる局所的な関数近似手法では，近似精度と学習効率とを両立させることができない．更に行動空間も連続な場合には，最適行動（Q 値の最大値）を求めるのに多大な計算コストを要するため，これまで実用性の高い方法はなかった．本研究では，選択的不感化ニューラルネット（SDNN）を行動価値関数の近似器として用い，多数の行動に関する Q 値を同時に出力層に分散表現することによってこの問題を解決する方法を提案する．この手法を，制約のあるアクロボットの振り上げ制御課題に適用したところ，既存の関数近似器を用いた場合に比べて，計算コストの増加が大幅に抑えられ，実時間制御に使えることが示された．また，離散行動の場合よりも環境の連続的な変化にも追従しやすいこと，従来手法では解くことができない条件下でも適用できることが分かった．本手法は，アルゴリズムが単純でパラメータ設定も容易であるなど実用性が高く，強化学習の適用範囲を大きく広げるものといえる．

キーワード　強化学習，行動価値関数，Q 値，関数近似

1. まえがき

強化学習 [1] は，ロボットなどの行動主体（エージェント）が環境との相互作用を繰り返し，その中で得られる報酬を最大化するような行動を試行錯誤的に獲得する行動学習の枠組みである．元々は動物の行動学習をモデル化したものであるが，制御設計の手間を省けることなどから工学的にも有用だと考えられている．また，動物が大量かつ多様な情報の存在する現実世界において行動を学習できることから，実環境やモデル化の難しい複雑な環境下で，ロボットなどを制御する手法として期待されている．

強化学習にはさまざまな手法が存在するが，本研究では，最も代表的なものの一つである Q 学習（方策オフ型 TD 学習）[2] を扱う．これは，各状態における各行動の価値（Q 値）を，ある更新式に従って逐次的に求めるというものであり，適当な条件下で最適な行動の獲得が理論的に保証されていること [3]，単純で実装しやすいことなどから，応用上もよく用いられる．

しかし，Q 学習を実空間など複雑な環境下での制御に応用するためには，解決すべき問題がある．Q 学習のアルゴリズムは，基本的に状態や行動が離散的であることを前提としているのに対し，実空間における状態変数は通常連続値をとり，行動も連続的な場合が多い．単に，制御目的を達成するのに十分な程度まで状態及び行動を細かく区切って離散化すると，状態・行動の数が爆発する．そのため，多くのメモリが必要となるだけでなく，経験すべき状態・行動の数が多いため学習や行動選択にかかる時間が非常に長くなってしまう．かといって，粗く離散化して状態・行動数の増加を抑えると，高精度の制御はできない．

したがって，連続な状態行動空間に Q 学習を適用する際には，経験したデータから未経験の状態・行動の価値を近似することが必要だと考えられる．しかし，

† 筑波大学大学院システム情報工学研究科，つくば市
Graduate School of System and Information Engineering, University of Tsukuba, Tsukuba-shi, 305–8573 Japan
a) E-mail: takaaki@bcl.esys.tsukuba.ac.jp
b) E-mail: shibuya@iit.tsukuba.ac.jp
c) E-mail: mor@bcl.esys.tsukuba.ac.jp
DOI:10.14923/transinfj.2014JDP7079

価値関数がどのような形をしているか事前には分からないので，あらかじめ設計した基底関数の線形和によって近似する手法 [4], [5] や，カーネル関数を用いる手法 [6] など，事前知識が必要な近似手法を一般的な対象に対して適用することは困難である．

理論上，任意の連続関数を任意精度で近似可能な関数近似器として多層パーセプトロン（MLP）がある [7], [8] が，中間層の素子数が少なすぎると表現力が不足し，多いと過剰適合が生じやすいなど，パラメータ依存性が強いため，使いこなすのが難しい．また，新たなサンプルのみを追加学習すると，既学習の入出力関係全体が壊れてしまうカタストロフィック干渉 [9] という現象があるため，近似した値を使って次々と Q 値を更新する場合には，古いサンプルを保存しておいて適宜再学習するといった対策が必要である．

同様の万能性をもち，Q 学習によく用いられるのが，タイルコーディング [1] や放射状基底関数ネットワーク（radial basis function network，以下 RBFN）[1], [10] といった局所的近似手法である．しかし，これらの手法は，局所領域が大きすぎると十分な精度で近似することができないし，逆に局所領域を小さくすると，汎化能力が低下する上に，必要なメモリや計算量が（特に状態行動空間が高次元の場合に）増大してしまう．そのため，しばしば現実的な計算コストではうまく近似できないし，できるとしても局所領域の設定（タイルや基底関数の配置）に熟練や事前知識が必要な場合が多い．

このように，既存の関数近似手法には問題点や制約条件が多いため，使いやすく，より幅広い問題に適用可能な手法が求められる．そのようなものとして期待されるのが，選択的不感化ニューラルネット（selective desensitization neural network，以下 SDNN）である．

新保ら [11] は，SDNN を用いて行動価値関数を行動ごとに近似する方法を提案し，アクロボットの振り上げ課題を対象として，4 次元の連続状態空間における Q 学習の数値実験を行った．この実験等から，SDNN が連続状態空間における価値関数の近似器として，数々の優れた性質（詳細は **5.1** 参照）をもつことが示されている．

しかしながら，新保らの手法は，行動が離散的であることを前提とし，行動ごとに別々の近似器を用意する必要がある．行動次元が連続な場合，計算コストが大きく増加し，学習効率も低下する．彼らの実験では，行動の種類を二つに限る代わりに制御周期を短くすることによって細かな制御を実現しているが，実時間で制御する場合，計算に要する時間やセンサ信号の遅れなどによっては制御周期をあまり短くできない場合がある．また，特定のタイミングでしか出力を選択できない課題（ジャンプするなど）の場合，この方法は無効である．

そこで本研究では，SDNN による価値関数の近似方法を改良し，状態空間だけでなく行動次元も連続な場合に Q 学習を適用できるようにする．具体的には，SDNN の出力層が全ての行動に関する Q 値を分散表現するようにすることで，異なる行動間の汎化を可能にするとともに，行動空間の連続化に伴う計算コストや学習時間の増加を抑える．

以下では，まず Q 学習及び SDNN によって行動ごとに価値関数を近似する方法について説明する．次に，提案する手法について述べた後，2 種類の課題について数値実験を行い，提案手法の有効性を検証するとともに，その性質や利点について考察する．

2. SDNN を用いた連続状態空間における Q 学習

2.1 Q 学習と価値関数

Q 学習は強化学習アルゴリズムの一つで，エージェントは Q 値（ある状態において，ある行動を選択することで，その後得られる総報酬の期待値）に基づいて行動を選択する．状態・行動から Q 値への写像（行動価値関数）を式 (1) に従って逐次的に更新することで選択行動の改善を行う．

$$Q(\boldsymbol{s}_t, a_t) \leftarrow (1-\alpha)Q(\boldsymbol{s}_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a' \in \mathcal{A}(\boldsymbol{s}_{t+1})} Q(\boldsymbol{s}_{t+1}, a')\right) \quad (1)$$

ここで $\boldsymbol{s}_t$，a_t，r_{t+1} は，それぞれ時刻 t における環境の状態，選択した行動，及びその行動に対する報酬である．また $\mathcal{A}(\boldsymbol{s})$ は状態 $\boldsymbol{s}$ において選択可能な全ての行動の集合，α は学習率，γ は割引率と呼ばれるパラメータである．

式 (1) から分かるように，Q 値を 1 回更新するためには，次状態 $\boldsymbol{s}_{t+1}$ において選択可能な全ての行動について Q 値を求める必要がある．一般的な価値関数近似器では行動ごとに Q 値を計算し直すことになるので，計算コストは行動数 $|\mathcal{A}(\boldsymbol{s}_{t+1})|$ の線形オーダーになる．また，行動を選択する場合にも，現在の状態

$\boldsymbol{s}_t$ において選択可能な全ての行動についての Q 値を用いるので，計算コストは行動数 $|\mathcal{A}(\boldsymbol{s}_t)|$ の線形オーダーになる．一般に，関数近似器の計算コストはテーブル参照などと比べて非常に大きいため，計算コストが行動数の線形オーダーになるという性質は，行動が連続な課題に Q 学習を用いる上で大きな制約となる．

2.2 SDNN-D による価値関数の近似

状態空間は連続的であるが，行動次元は離散的であり，行動数が比較的少ない場合，行動の種類ごとに別々の関数近似器を用意し，各近似器が特定の行動 a_i に関する Q 値 $Q_{a_i}(\boldsymbol{s}) = Q(\boldsymbol{s}, a_i)$ を近似することによって，近似する関数の入力次元を減らすことができる．

このような場合について，新保らは SDNN により構成した関数近似器を行動数の分だけ用意し，全体として連続状態空間における価値関数を近似した（本論文ではこれを SDNN-D と呼ぶ）．

以下では，まず SDNN の基礎となるパターンコーディングと選択的不感化 [12] について説明する．次に，SDNN を用いた一般的な関数近似器の構成と学習方法，及びそれを連続状態空間における Q 学習に適用したもの（SDNN-D）について説明する．

2.2.1 パターンコーディングと選択的不感化

連続な変数 x の値を，対応する 2 値パターン（コードパターン）$\bar{\boldsymbol{x}}$ によって表現することをパターンコーディングと呼ぶ．具体的には，x の変域を n 個の区間 $X_1, \ldots, X_n$ に等分割し，区間ごとに別々のコードパターンを対応させる．そして x が区間 $X_i (i = 1, \ldots, n)$ に含まれる場合，x を X_i に割り当てられたコードパターン $\bar{\boldsymbol{x}}_i$ によって表現する（$x \to \bar{\boldsymbol{x}}_i : x \in X_i$）．

ここで重要なのは，区間 X_i と X_j が近いほど，$\bar{\boldsymbol{x}}_i$ と $\bar{\boldsymbol{x}}_j$ の相関が大きくなるようなコードパターンを用いることである．すなわち，変数 x の値が連続的に変化するとき，コードパターン同士の相関が 1 から 0 へ連続的に減少し，十分に離れた変数値を表現するコードパターン同士の相関は 0 になるようにする．これによって，非局所的な汎化を実現すると同時に，無関係な点への学習の影響が小さくなる．

選択的不感化とは，2 種類のコードパターンがあるとき，一方のパターンに応じて他方の一部の要素を中立値（出力の期待値）に変えることによって，2 種類のパターンが表す情報を統合する手法である．具体的には，2 種類の m 次元の 2 値（要素の約半数が 1 で残りが -1）パターン $\bar{\boldsymbol{s}} = (\bar{s}_1, \ldots, \bar{s}_m)^T$ 及び $\bar{\boldsymbol{c}} = (\bar{c}_1, \ldots, \bar{c}_m)^T$ があるとき，$\bar{\boldsymbol{s}}$ の要素の一部（通常は約半数）をパターン $\bar{\boldsymbol{c}}$ に応じて選び 0 とする．

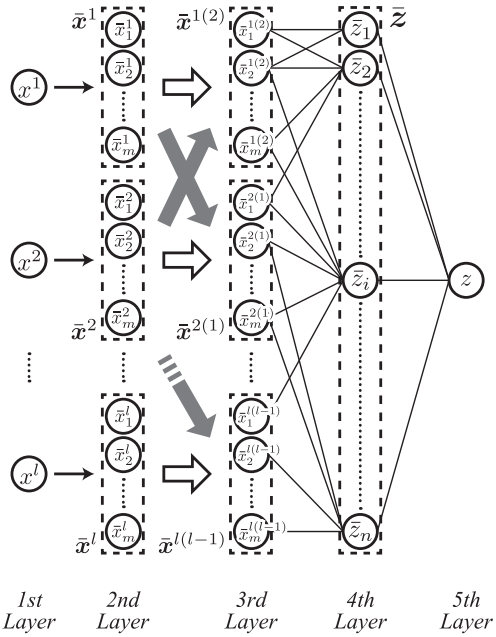


図 1　SDNN を用いた l 変数関数近似器の構成
Fig. 1　l-values function approximator using a selective desensitization neural network.

不感化する素子の選び方として最も単純なのは，

$$\bar{s}_i \leftarrow \frac{1 + \bar{c}_i}{2} \bar{s}_i \tag{2}$$

とする，すなわち $\bar{\boldsymbol{c}}$ の -1 の要素に対応する $\bar{\boldsymbol{s}}$ の要素を 0 にする方法である．ただし，異なる変数に同じコードパターンを用いる場合など，二つのパターン間に高い相関が生じる可能性があるときには，不感化する素子との対応関係をシャッフルする必要がある．

これにより，約半数が 0 で残りが ± 1 の 3 値パターンが得られるが，このパターンのことを「$\bar{\boldsymbol{c}}$ によって修飾された $\bar{\boldsymbol{s}}$」といい，$\bar{\boldsymbol{s}}(\bar{\boldsymbol{c}})$ で表す．同様にして，$\bar{\boldsymbol{s}}$ によって修飾された $\bar{\boldsymbol{c}}$ を考えることができる．二つのパターンが互いに修飾し合い，$\bar{\boldsymbol{s}}(\bar{\boldsymbol{c}})$ と $\bar{\boldsymbol{c}}(\bar{\boldsymbol{s}})$ を作ることを相互不感化という．

2.2.2 関数近似器の構成

SDNN によって l 変数関数 $z = f(\boldsymbol{x}) = f(x^1, \ldots, x^l)$ を近似する際，通常は図 1 に示す構成を用いる．この図で，第 1 層は各入力変数を表す l 個の素子からなる．また，第 2 層は入力変数に対応するコードパターンを表現する層で，m 個の素子からなる素子群を l 個含む．

コードパターンとして，本研究では新保らと同様，

次のように作成したものを用いている．まず m 次元の2値パターンをランダムに9個作成して $\bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_2, \ldots, \bar{\boldsymbol{p}}_9$ とする．(ただし，角度のように変域の両端の値が同じ状態を表すような変数については，$\bar{\boldsymbol{p}}_1 = \bar{\boldsymbol{p}}_9$ とする)．次に $\bar{\boldsymbol{p}}_i$ と $\bar{\boldsymbol{p}}_{i+1}$ の間 $(i = 1, \ldots, 8)$ を，それぞれ44個のパターンで補間し，全体として361個のコードパターンを作成する．このうち $\bar{\boldsymbol{p}}_9$ を除いた360個のパターンを，変域を360等分した区間に割り当てる．コードパターンのセットは変数ごとに異なるものを作成してもよいが，ここでは l 個の変数の全てについて共通のものを用いることにする．

第3層は，第2層で分散表現されたパターンを相互不感化する層で，$l \times (l-1)$ 個の素子群 $G^{\mu(\nu)}(\mu, \nu = 1, \ldots, l,\ \mu \neq \nu)$ で構成される．素子群 $G^{\mu(\nu)}$ は m 個の素子からなり，第2層の素子群の出力パターン $\bar{\boldsymbol{x}}^{\mu}$ を受け取るとともに，別の素子群の出力パターン $\bar{\boldsymbol{x}}^{\nu}$ による修飾を受けてパターン $\bar{\boldsymbol{x}}^{\mu(\nu)} = \bar{\boldsymbol{x}}^{\mu}(\bar{\boldsymbol{x}}^{\nu})$ を出力する．全ての μ, ν の組合せ(ただし $\mu \neq \nu$)について $\bar{\boldsymbol{x}}^{\mu(\nu)}$ を並べたものが第3層の出力パターン $\bar{\boldsymbol{x}}^{SD}$ である．

なお，異なる変数間で共通のコードパターンを用いるため，選択的不感化による修飾は式(2)ではなく，

$$\bar{x}_k^{\mu(\nu)} = \frac{1 + (\sigma^{\mu\nu}\bar{\boldsymbol{x}}^{\nu})_k}{2}\bar{x}_k^{\mu} \tag{3}$$

に従って行う($\bar{x}_k^{\mu(\nu)}$ は $\bar{\boldsymbol{x}}^{\mu(\nu)}$ の k 番目の要素)．ここで，$\sigma^{\mu\nu}$ は修飾する素子をランダムな順序に入れ換えるための m 次の置換行列であり，出力パターン間に意図しない相関が生じることのないよう，μ 及び ν ごとに異なるものを用いる．

第4層は第3層の出力 $\bar{\boldsymbol{x}}^{SD}$ を入力とし，0または1を出力する n 個のしきい値素子で構成されている．式で表すと，i 番目の素子の出力値 $\bar{z}_i$ は

$$\begin{aligned}\bar{z}_i &= \phi\left(\boldsymbol{w}_i^T\bar{\boldsymbol{x}}^{SD} - w_{i0}\right)\\ &= \phi\left(\sum_{j=1}^{ml(l-1)} w_{ij}\bar{x}_j^{SD} - w_{i0}\right)\end{aligned} \tag{4}$$

で計算される．ここで $\phi(u)$ は $u > 0$ のとき1を，それ以外のときは0を出力する関数，w_{ij} は $\bar{\boldsymbol{x}}^{SD}$ の j 番目の素子からの結合荷重，w_{i0} は素子のしきい値である．

第5層は出力層で，第4層の各素子の値に基づいて関数の出力 z を計算する1個の素子からなる．式で表すと

$$z = g\left(\sum_{i \in I}\bar{z}_i\right) \tag{5}$$

となる．ここで I は1から n までの自然数を要素とする添字集合，$g(u)$ はスケール変換を行う関数である．

関数近似器の学習は，第4層の素子への結合荷重としきい値を，一種の誤り訂正学習(p-delta 則)で修正することで行う．具体的には，もし出力値 $z = g(u)$ が目標値 $\hat{z} = g(\hat{u})$ より大きければ，第4層の素子で1を出力しているもののうち，$|\hat{u} - u|$ 個について0を出力するように修正する．逆に，出力値 z が目標値 $\hat{z}$ より小さければ，0を出力している素子について同様に修正する(このとき，なるべく内部電位の絶対値が小さい素子を選ぶとよい)．修正する素子を $\bar{z}_i$ とすると，結合荷重としきい値の更新式は

$$w_{ij} \leftarrow w_{ij} + c\,\mathrm{sgn}(\hat{u} - u)\,\bar{x}_j^{SD} \tag{6}$$

$$w_{i0} \leftarrow w_{i0} - c\,\mathrm{sgn}(\hat{u} - u) \tag{7}$$

となる．ここで $\mathrm{sgn}(u)$ は $u > 0$ のとき1を，それ以外のときは -1 を出力する符号関数，c は正定値の学習係数である．

新保らは，このような関数近似器を行動数の分だけ用意することで，全体として行動価値関数を近似する近似器(SDNN-D)を構成した．これを用いたQ学習の手順は，以下のとおりである．

(1) 現在の状態 $\boldsymbol{s}_t$ を入力変数としたときの i 番目のSDNNの出力値 z_i を，行動 a_i に関するQ値 $Q(\boldsymbol{s}_t, a_i)$ の推定値とする．

(2) 推定値に基づいて行動を選択する(greedy方策の場合であれば，z_i が最大となる a_i を選ぶ)．

(3) 行動後の新たな状態に関して，各近似器が推定したQ値と受け取った報酬 r_t から式(1)の右辺の値を計算する．

(4) 得られた値を目標値 $\hat{z}$，行動前の状態変数を入力として，選択した行動に対応する近似器の学習(結合荷重の更新)を行う．

(5) (1)に戻る．

3. 連続状態行動空間における価値関数近似

SDNN-Dは，Q値を表現するネットワークを行動ごとに用意するため，行動次元が連続である場合に適用すると，行動を細かく離散化すればするほど計算コ

ストが増加する．また，行動次元に関する汎化が起こらないため，学習効率が低下するという問題もある．

これを避ける一つの方法として，行動値を入力変数の一つとし，$Q(\boldsymbol{s}, a_i)$ を直接近似することが考えられる．こうすれば関数近似器は一つで済むように思われるが，行動選択の際に全ての行動に関する Q 値を求める必要があるため，結局は同程度の計算コストがかかる．また，SDNN のように非局所的な汎化能力をもつ近似器を用いた場合，同じ状態で行動値が変化しても，l が大きければ入力ベクトルはわずかしか変化しないため，行動の違いによる Q 値の違いをうまく学習できないおそれがある．

本章では，SDNN-D を改良することによって，計算コストの増加や学習効率の低下を抑えつつ，行動次元が連続な場合に Q 値を近似する手法（以下 SDNN-C）を提案する．

3.1 SDNN-C の構成

SDNN-C は，SDNN-D の第 4 層において Q 値が分散表現されていることに着目し，ある状態における全ての行動に関する Q 値を表すパターンを一度に求めよう，というアイディアに基づいている．求めたパターンに対して，行動値を分散表現したパターン（行動パターン）による選択的不感化を行うことによって，特定の行動値に関する Q 値を得る．詳しくは後述するが，これによって行動次元における汎化を実現するとともに，計算コストの増加を抑えている．

SDNN-C は行動価値関数 $Q(\boldsymbol{s}, a)$ の近似器であるが，一般の関数近似問題にも用いることができる．以下では **2.2.2** の構成と比較しやすいように，関数 $z = f(\boldsymbol{x}, y) = f(x^1, \ldots, x^l, y)$ を近似する関数近似器として説明する．なお，第 1 層から第 4 層については SDNN-D と全く同じ構成なので，説明は省略する．

第 5 層は，第 4 層の 2 値パターン $\bar{\boldsymbol{z}}$ を入力 y に対応したコードパターン $\bar{\boldsymbol{y}}$ によって修飾する層であり，n 個の素子から構成される．修飾は式 (2) に従って行い，i 番目の素子の出力 $\bar{z}_i(\bar{y}_i)$ は

$$\bar{z}_i(\bar{y}_i) = \frac{1 + \bar{y}_i}{2} \bar{z}_i \tag{8}$$

となる．

第 6 層は，第 5 層の全ての素子の出力値に基づいて関数の近似値 z を出力する 1 個の素子からなる．式で表すと次式のようになる．

$$z = f(\boldsymbol{x}, y)$$

$$= g\left(\sum_{i \in I} \bar{z}_i(\bar{y}_i) \right) \tag{9}$$

また，式 (9) は $\bar{\boldsymbol{y}}$ のうち値が 1 をとる素子の添字集合 $J(y) = \{j : \forall j \in I, \bar{y}_j = 1\}$ を用いることで，次式のように表すこともできる．

$$z = g\left(\sum_{j \in J(y)} \bar{z}_j \right) \tag{10}$$

入力 y についてのコードパターン $\bar{\boldsymbol{y}}$ は，y の値が近いほど，対応する $\bar{\boldsymbol{y}}$ の相関が大きく，十分に遠いと相関が 0 になる必要がある．上記の条件を満たすようなパターンコーディングの方法はさまざまなものが考えられるが，本論文では処理を簡略化するために以下のように作成する．

まず n 個の素子のうち 1 番目から n' 番目 $(n' < n)$ の素子が 1，それ以外の素子は -1 となるパターンを作成し，これを $\bar{\boldsymbol{q}}_1$ とする．次に 2 番目から $n'+1$ 番目の素子が 1，それ以外が -1 となるパターンを作成し，これを $\bar{\boldsymbol{q}}_2$ とする．以下同様にして，k 番目のパターン $\bar{\boldsymbol{q}}_k$ の i 番目の素子の値が

$$\bar{q}_{k,i} = \begin{cases} 1 & (\text{if } k \le i < k + n') \\ -1 & (\text{otherwise}) \end{cases} \tag{11}$$

となるように，$n - n' + 1$ 個のパターンを作成する．各パターンは y の変域を $n - n' + 1$ 等分した区間 $Y_k (k = 1, \ldots, n - n' + 1)$ と 1 対 1 で対応する $(y \to \bar{\boldsymbol{q}}_k : y \in Y_k)$．

$\bar{\boldsymbol{y}}$ として $\bar{\boldsymbol{q}}_1, \ldots, \bar{\boldsymbol{q}}_{n-n'+1}$ を用いた場合，$\bar{\boldsymbol{y}}$ によって不感化されない素子の添字集合 $J(y)$ は次式のように表すことができる．

$$J(y) = \{k, k+1, \ldots, k+n'-1 : y \in Y_k\} \tag{12}$$

3.2 SDNN-C の学習則

SDNN-C の学習は，SDNN-D の場合と同様に，誤った出力をしている $|\hat{u} - u|$ 個の出力素子について結合荷重としきい値を修正することで行う．しかし，修正する素子を単純に内部電位の絶対値が小さいものから順番に選ぶと，修正する素子の位置に偏りが生じ，汎化の広がり方が不均等になる可能性がある．修正する素子の偏りを抑える学習方法は幾つか考えられるが，予備実験を行った結果から，本論文では以下の方法を用いる．

まず出力素子 $\bar{z}_j (j \in J(y))$ を，添字の順序を保ったまま $|\hat{u}-u|$ 個の集合 $Z_1, Z_2, \ldots, Z_{|\hat{u}-u|}$ に分割する．このとき，各集合の要素数はできるだけ均等になるようにする．

次に，Z_i の中で，出力が誤っている素子（$u > \hat{u}$ ならば，1 を出力している素子，$u < \hat{u}$ ならば，0 を出力している素子）一つに対して，その結合荷重としきい値を式 (6)，(7) に従って修正する．この操作を各集合について行うことにより，全体として $|\hat{u}-u|$ 個の素子を修正する．なお，Z_i 内に条件を満たす素子が複数存在する場合は，その中で内部電位の絶対値が最も小さい素子を一つ選ぶ．また，条件を満たす素子が存在しない場合は，その集合内では修正を行わないものとする．

3.3 行動次元に関する汎化と計算コスト

SDNN-C では，以下のメカニズムにより，Q 値に関する汎化が行動次元においても生じると考えられる．

まず，Q 値を表現する素子 $\bar{z}_j$ は複数の行動間で共有されるが，行動値が近い行動ほど多数の素子が共有され，遠いと共有される素子は少なくなる（十分に遠ければ 0 になる）．具体的に，式 (11) のコードパターンを用いた場合，素子 $\bar{z}_j$ は区間 Y_{st} から Y_{end} に含まれる行動の Q 値を表現するために用いられる（ここで，$st = \max(1, j-n'+1)$ 及び $end = \min(n-n'+1, j)$ である）．そのため，この素子の出力が反転すると，これらの行動に関する Q 値全てが変化する．また前節で説明したように，修正される素子は位置が均等になるように選ばれる．

したがって，ある行動に関して Q 値を修正したとき，それに近い行動の Q 値もほぼ同様に修正される．つまり，ある行動を経験すると，それに類似した行動に関しても Q 値が更新されることになる．ただし，行動値が離れるに従って修正量は小さくなり，十分に離れた行動に関しては修正されないので，行動値がある程度違えば異なる Q 値を学習することが可能である．

次に，SDNN-C を用いたときの計算コストについて検討する．**2.1** で見たように，Q 学習ではある状態において，選択可能な全ての行動の中で Q 値が最大となる行動を求める必要がある．そこで，状態が p 変数，行動が q 通りの値を取るときに，$Q(\boldsymbol{s}, a_1), \ldots, Q(\boldsymbol{s}, a_q)$ の値を全て計算するために必要なコストを考える．

まず，第 1 層から第 4 層までの計算コストは，状態変数のコードパターンを相互不感化した全ての組合せについて計算するので，SDNN-D，SDNN-C ともに $O(p^2)$ となる．

SDNN-D の場合，行動値ごとに個別のネットワークを用いるので，全ての Q 値を求めるには，上記の計算を q 回行う必要がある．したがって計算コストは $O(p^2 q)$ のオーダーになる．一方，SDNN-C の場合，第 1 層から第 4 層までの計算は 1 回でよく，その結果を行動値ごとのコードパターンで不感化するので，計算コストは $O(p^2+q)$ となる．

上記の計算コストは，$O(p^2)$ を定数と見れば，どちらも行動変数に対して線形オーダーである．しかし，実際に数値計算を行う際の計算コストは，大部分が第 1 層から第 4 層までの計算にかかり，第 5 層以降の計算コストはそれよりもずっと小さい．そのため，行動数が十分に小さい場合を除いて $O(p^2+q) < O(p^2 q)$，すなわち SDNN-C は SDNN-D よりも，行動数の増加に対する計算コストの増加を抑えられることが分かる．

4. シミュレーション実験

提案手法が有意義なのは，状態空間及び行動次元が連続であり，かつ行動次元を粗く離散化すると目的の達成が難しくなるような場合だと考えられる．そこで，そのような状況を 2 種類用意してシミュレーション実験を行い，提案手法の有効性を検証する．

4.1 学習課題

実験 1，2 ともに，アクロボットの振り上げ課題 [1] を用いる．アクロボット [13] は図 2 に示すような 2 リンク 2 関節のマニピュレータで，アクチュエータが第 2 関節にのみ存在する劣駆動系である．非線形性が強く，制御が比較的難しいことから，連続状態空間における強化学習の課題としてしばしば用いられている．

学習の目的は，両リンクが吊り下がって静止した初期状態（$\theta_1 = \theta_2 = 0$ [rad], $\dot{\theta}_1 = \dot{\theta}_2 = 0$ [rad/s]）か

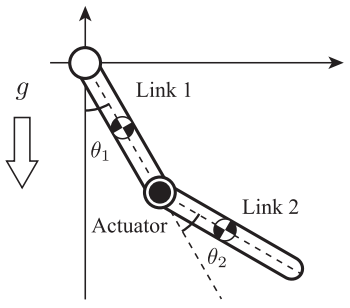


図 2 アクロボット
Fig. 2 The acrobot.

ら開始し，できるだけ早く，第 2 リンクの先端が一定の高さ以上にあり，更に一瞬静止した（$\dot{\theta}_1, \dot{\theta}_2$ の絶対値がある値以下になる）ゴール状態となるよう，アクロボットを制御することである．

アクロボットの物理パラメータは $m_1 = m_2 = 1$[kg]，$l_1 = l_2 = 1$[m]，$l_{c1} = l_{c2} = 0.5$[m]，$I_1 = I_2 = 1$[kg$\cdot$m^2] とした．ここで m_1，m_2 は各リンクの質量，l_1，l_2 は各リンクの長さ，l_{c1}，l_{c2} は根本の関節からリンクの重心までの距離，I_1，I_2 は慣性モーメントである．

アクロボットの状態は各リンクの角度と角速度 $\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}$ によって完全に規定される．そこで，この 4 変数を $\theta_1, \theta_2 \in [-\pi, \pi)$[rad]，$\dot{\theta_1} \in [-3\pi, 3\pi)$[rad/s]，$\dot{\theta_2} \in [-5\pi, 5\pi)$[rad/s] の範囲で $[0, 1]$ に正規化したものを強化学習の状態変数 s_1, s_2, s_3, s_4 とした．なお $\dot{\theta_1}, \dot{\theta_2}$ が正規化範囲の最大値よりも大きな値をとった場合は 1 を，最小値よりも小さな値をとった場合は 0 を s_3, s_4 の値とした．

エージェントは，第 2 関節に加えるトルク τ の値を $[-10, 10]$ [Nm] の範囲で選択する．すなわち，τ の値が行動値を表す．行動を離散化する際には，最小値及び最大値を含み，その間を等間隔に分ける（例えば行動数 $|\mathcal{A}|$ が 3 の場合，$\tau \in \{-10, 0, 10\}$ となる）．

ゴール状態は，第 2 リンクの先端が第 1 関節から見て鉛直上向きに 1.5 [m] 以上の位置にあり，かつ角速度が $|\dot{\theta}_1| < 3\lambda\pi$，$|\dot{\theta}_2| < 5\lambda\pi$ を満たした場合とした．1 回の試行（エピソード）は，初期状態から始まり，ゴール状態に到達するか，ゴール状態に到達しないまま規定のステップ数が経過した時点で終了するものとした．ここで λ は静止状態の許容範囲を設定する変数であり，第 k エピソードにおける値を次式のように設定した．

$$\lambda = \begin{cases} 10^{-\left(1+\frac{k-1}{10000-1}\right)} & (k \leq 10000) \\ 10^{-2} & (k > 10000) \end{cases} \qquad (13)$$

これによって λ は第 1〜10000 エピソードの間に 0.1 から 0.01 まで減少するため，課題は徐々に難化することになる．

報酬は，ゴール状態に到達した場合 $+10$ を，第 1 リンクの振れ角 $|\theta_1|$ が 5[deg] 未満の場合，及び角速度 $\dot{\theta_1}$，$\dot{\theta_2}$ の値が正規化範囲から外れた場合は -5 を与え，それ以外の場合は全て 0 とした．

4.2 実　験　1

実験 1 では，学習性能及び計算コストについて従来手法（SDNN-D 及び RBFN）と比較する．ただし，学習性能の差が明確に現れるよう，制御（行動選択）の周期を 1 秒間隔と長めに設定した．

4.2.1 方　　法

Q 学習では，行動価値関数を改善するためにさまざまな状態・行動の探索が必要である．一方，アクロボットの振り上げのような課題は，適切な行動を連続して選択しないとゴールするのが非常に難しい．そのため，価値関数の学習が正しく行われていても，探索行動を含んでいるとゴールするまでのステップ数が長くなる場合がある．そこで，実験では価値関数の学習を行うエピソード（学習エピソード）と，学習は行わずにその時点での価値関数を評価するエピソード（評価エピソード）とを分け，後者では常に Q 値が最大の行動を選択する greedy 方策を用いた．一方，学習エピソードでは，エピソード開始から 10 ステップまでは greedy 方策を，それ以降は ε の確率で選択可能な行動をランダムに選択し，それ以外の場合は Q 値が最大の行動を選択する ε-greedy 方策を用いた．また，1 回のエピソードは学習エピソード，評価エピソードともに最長 50 ステップとした．

SDNN-C と SDNN-D は，構成する素子の数を揃えて必要なメモリ量を同程度にした．具体的には，SDNN-D は $m = 200, n = 300, c = 0.1$，SDNN-C は $m = 200, n = 900, n' = 300, c = 0.1$ とした．このとき，SDNN-D で扱える行動数が $|\mathcal{A}| = 3$ であるのに対し，SDNN-C で扱える行動数は 601 であり，行動値を十分細かく離散化することが可能である．また出力値の範囲は $[-20, 20]$ とし，スケール変換関数 $g(u)$ は式 (14) とした．

$$g(u) = \begin{cases} 20 & (280 < u) \\ \frac{2}{13}u - \frac{300}{13} & (20 \leq u \leq 280) \\ -20 & (u < 20) \end{cases} \qquad (14)$$

もう一つの比較対象である RBFN は，ガウス関数のような放射状の基底関数の重み付き線形和によって目的の関数を近似するものである．ここでは，基底関数として 5 次元の状態行動空間（各次元は 0 から 1 の範囲に正規化する）中に等間隔に配置したガウス関数を用いる．

このとき，基底関数の間隔が広すぎると，表現能力が不足して十分に価値関数を近似することができず，逆に基底関数の間隔が狭すぎると，基底関数の数が大

きくなり，計算コストが増大する．そこで本研究では，最終的に 10 ステップ以内にゴールに到達できる範囲で，なるべく計算コストが小さくなるような配置にすることにした．行動次元と状態次元で，あるいは四つの状態次元ごとに基底関数の配置間隔を変えることも考えられるが，ガウス関数の各次元方向の広がりも含めて最適化するのは大変な作業であり，事前知識なしに行うのは極めて困難であるため，全ての次元について一様とした．

したがって，価値関数の近似は，j 番目の基底関数（ガウス関数）の中心座標を $(\hat{\boldsymbol{s}}_j, \hat{a}_j)$ として，

$$Q(\boldsymbol{s}, a) = \sum\nolimits_j w_j \exp\left(-\frac{\|\boldsymbol{s} - \hat{\boldsymbol{s}}_j\|^2 + (a - \hat{a}_j)^2}{\sigma^2}\right) \tag{15}$$

によって行う．ここで，$\|\boldsymbol{s} - \hat{\boldsymbol{s}}_j\|$ は $\boldsymbol{s}$ と $\hat{\boldsymbol{s}}_j$ とのユークリッド距離，w_j は j 番目の基底関数の荷重，σ はガウス関数の広がりを表すパラメータである．

また，**2.1** で述べたように，Q 学習では $Q(\boldsymbol{s}, a)$ が最大となる a を求める必要があり，そのための計算コストが問題となる．RBFN において状態 $\boldsymbol{s}$ を固定して a を変化させる場合，式 (15) 右辺の a に依存しない部分を記憶しておくことによって計算量が若干節約できるが，それでも行動数にほぼ比例して計算量が増加する（**3.3** の表記法で表すと，RBFN の計算コストは $O(N^p + Nq)$ となる．ここで，N は各次元に配置された基底関数の個数である）．$Q(\boldsymbol{s}, a)$ は一般に多峰性関数なので，勾配法などを用いて効率的に求めることも困難である．

そこで，ここでは，あらかじめ決めた数の行動値について順に Q 値を求めることにする．ただし，SDNN-C と同じ行動数 $|\mathcal{A}| = 601$ にすると，1 ステップの計算に時間がかかりすぎて実験ができないため，やはり最終的に 10 ステップ以内にゴールに到達できる範囲で，行動数を減らすことにした．

以上の方針に従って RBFN に関する予備実験を行った結果，基底関数は各次元に 15 個，全体で 15^5 個の格子状に配置し，ガウス関数のパラメータは $\sigma = 0.1$，行動数は $|\mathcal{A}| = 21$ に設定した．

いずれの関数近似器についても価値関数に関する事前知識は与えないものとし，初期状態において，全ての状態・行動に対して $Q(\boldsymbol{s}, a) \equiv 0$ とした．RBFN の場合，これは全ての基底関数の結合荷重を 0 とすれば実現できる．一方，SDNN-C 及び SDNN-D の場合，結合荷重を全て 0，しきい値を微小量にすると，初期段階のわずかな学習によって多くの出力素子の値が同時に変化してしまうため，学習が不安定になる．これを防ぐため，結合荷重はランダムな微小量とした上で，第 4 層素子の半分（奇数番目の素子）のしきい値を十分に大きな正の値で，残りの半分については十分小さな（絶対値が大きな）負の値に設定した．

表 1　強化学習 1 ステップの平均計算時間
Table 1　Average computation time for a step of reinforcement learning.

	行動数 $\|\mathcal{A}\|$	計算時間 [ms]
SDNN-C	601	28.4
SDNN-D	3	25.5
RBFN	21	919

実験は，関数近似器と行動数 $|\mathcal{A}|$ のみが異なる 3 条件，SDNN-C（$|\mathcal{A}| = 601$），SDNN-D（$|\mathcal{A}| = 3$），RBFN（$|\mathcal{A}| = 21$）について行った．

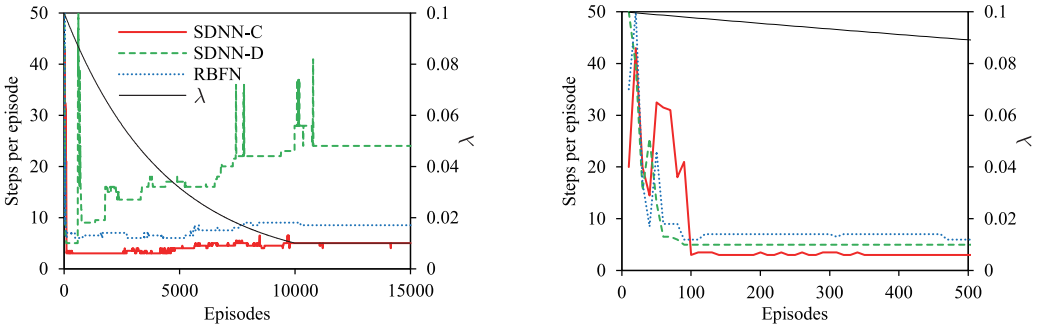
強化学習のパラメータは，$\alpha = 0.5, \gamma = 0.9, \varepsilon = 0.1$ とした．また，学習エピソードは 15000 エピソードまで行い，学習エピソード 10 回ごとに，評価エピソードを 1 回行うものとした．これを 1 実験試行として，試行ごとのばらつきの影響を抑えるため，各条件について乱数系列を変更して 10 試行ずつの実験を行った．

4.2.2　結　　果

まず各手法の計算時間を表 1 に示す．計算環境は，Dell Precision T7400 クアッドコア インテル(R)Xeon(R) プロセッサ X5482(2 × 6MB L2 キャッシュ，3.20GHz，1600MHzFSB)，8GB クアッドチャネル DDR2-SDRAM メモリである．

この表より，SDNN-C と SDNN-D の計算コストは同程度となることが分かる．一方，RBFN は行動数を 21 に減らしても，計算時間は SDNN の 30 倍以上かかっている．実験では強化学習 1 ステップを 1000[ms] としているので，実時間で制御を行うためには，これ以上基底関数の個数や行動数を増やすことはできない．したがって，性能をより高めることは困難である．

次に，評価エピソードの継続ステップ数，すなわちゴールに到達するまでのステップ数（到達しない場合は 50）の変化を図 3 に示す．グラフの横軸は学習エピソード数，左の縦軸はその際の評価エピソードの継続ステップ数であり，SDNN-C，SDNN-D，RBFN のそれぞれについて，10 試行の中央値をプロットしている．また，ゴール時の角速度の許容範囲を表す変数 λ の値（右の縦軸）も黒い細線でプロットした．(a) は全



(a) Episode 1 - 15000　　(b) Episode 1 - 500

図 3　実験 1 の学習過程
Fig. 3　Learning curves for experiment 1.

15000 エピソードについて示したグラフ，(b) はその最初の 500 エピソードの部分を拡大したものである．

学習の最も初期段階において，継続ステップ数が 10 以下の値に収束するまでのエピソード数に着目すると，SDNN-D と RBFN はともに 50 エピソード程度であるのに対して，SDNN-C は 100 エピソード程度と約 2 倍かかっている．しかし SDNN-D の行動数が 3 であるのに対し，SDNN-C の行動数は 601，RBFN の行動数は 21 であり，それぞれ約 200 倍と 7 倍の状態行動空間を探索していることになる．この差をふまえると，SDNN-C の学習効率が低いとはいえない．

エピソードが進んでゴールの条件が厳しくなると，SDNN-D では継続ステップ数が急激に大きくなることがしばしばあった．これは，それまでに獲得した行動系列ではゴールに到達することができなくなり，ゴールに到達可能な全く別の行動系列を獲得したためと考えられる．一方，SDNN-C と RBFN は，継続ステップ数が急激に変化することはなく，最終的なゴール条件でも 10 ステップ以下で振り上げる行動が学習されている．これは，行動を少しずつ変えることによってゴール条件の変化に対応できたことを示している．

最終的な継続ステップ数は，SDNN-D が 24，RBFN が 8.5，SDNN-C が 5 であった．これは，SDNN-C が最もよい行動を獲得できたことを示している．いずれも学習がほぼ収束していることから，結果の差は主に行動数，すなわち行動値の離散化の細かさの差を反映していると考えられる．

4.3 実　験　2

実験 1 において，制御周期を十分短くすれば，行動数の少ない SDNN-D でも短時間でゴールに到達することができる（ただし，行動選択の回数が増えるので，計算量及びステップ数は増加する）．これは，行動値を短時間で切り替えることによって，実質的に中間的な行動値を実現できるからだと考えられる．したがって，実験 1 の結果から，SDNN-C を用いると制御周期を長くすることができるといえるが，SDNN-C を用いることが不可欠だとはいえない．

そこで実験 2 では，制御周期を短くすることができない状況を考える．例えば，子どもが乗ったブランコを横に立った大人が揺らす場合，ブランコが鉛直線を通過する付近の一瞬しか力を加えられないため，そのときの力加減によって制御する．これと同様に，アクロボットの振り上げ課題において，エージェントはリンク 1 が鉛直線を通過した（$\theta_1 = 0$）タイミングでしか行動を選択できないという制約を設ける．ただし，アクチュエータは次の行動選択まで，選択されたトルクを出力し続けるものとする．

この場合，制御の間隔（1 ステップ）は，リンク 1 が鉛直線を通過してから次に通過するまでの時間に依存するので，一定ではない．また，行動選択の際の状態変数 s_1（θ_1 を正規化したもの）は常に同じ値なので，状態空間は実質的に三次元となる．

1 回のエピソードは学習エピソード，評価エピソードともに最長 1000 ステップとし，関数近似器に SDNN-C（$|\mathcal{A}| = 601$），SDNN-D（$|\mathcal{A}| = 3$）及び RBFN（$|\mathcal{A}| = 601$）を用いて実験を行った．ここで，SDNN-C と SDNN-D は実験 1 と同様の設定（ただし，入力変数は 3 個）とした．一方，RBFN は SDNN-C と行動数及び計算コストを揃えた場合の性能について比較を行うため，行動数を 601 とした上で，1 ステップの計算時間が SDNN-C と揃うように基底関数の個数を設定した．その上で学習性能が良くなるような σ の値

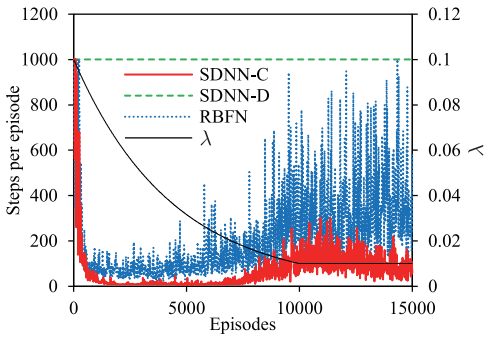


図 4　実験 2 の学習過程
Fig. 4　Learning curves for experiment 2.

を設定した．具体的には，4 次元の状態行動空間中に等間隔に 5^4 個の格子点をとり，各格子点を中心とした $\sigma = 0.2$ のガウス関数を基底関数として構成した．なお，これ以外の設定は実験 1 と共通である．

実験結果を図 4 に示す．図 3 と同様に，10 回の実験試行における各評価エピソードの継続ステップ数の中央値をプロットした．図から明らかなように，3 行動の SDNN-D の場合，最後まで 1 度も振り上げに成功していない．このことから，この振り上げ課題は離散行動では解けない，あるいは解くことが非常に困難な課題であるといえる．

これに対して SDNN-C の場合，1000〜2000 エピソードでいったん収束した後，7500 エピソード前後まで約 10 ステップで振り上げに成功している．その後，ゴール条件が厳しくなるに従い，継続ステップ数が上昇しているが，ほぼ 200 ステップ以内にはゴールしていることが分かる．なお，10 試行中の 2 試行において，10000〜15000 エピソードの間に 10 ステップ以下でゴールする行動を発見していた．

RBFN も SDNN-C と同様に 1000〜2000 エピソードでいったん収束し，7500 エピソード前後から継続ステップ数が上昇しているが，継続ステップ数は全体的に SDNN-C よりも長い．また SDNN-C と比べて継続ステップ数のばらつきが大きいが，これは実験試行間でのばらつきが大きいだけでなく，一つの実験試行の中でも継続ステップ数が収束していないためである．行動数は SDNN-C と共通であるから，この結果は関数近似器の表現能力が不十分だったためと考えられる．

5. 考　　察

5.1 提案手法の特徴

以下に，本研究で提案した SDNN-C の特徴についてまとめる．その大部分は元々の SDNN から受け継いだものである．

SDNN の基本的な特徴として，まず関数の表現能力と汎化能力との両立が挙げられる．SDNN は，並列パーセプトロン [14] にパターンコーディングと選択的不感化を導入したものであるから，並列パーセプトロン同様に万能の表現能力（任意の連続関数を任意の精度で近似可能）をもつ．万能の表現能力は，必ずしもサンプルからの学習によって精度よく近似できることを意味しないが，SDNN の場合，かなり複雑な関数であってもうまく学習できることが実験的に示されている [15]．この際，近似精度を上げるためには素子数を増やす必要があるが，いくら増やしても過剰適合による汎化誤差の増加は生じない．つまり，素子数を増やして表現能力を高めても，汎化能力が低下することはないという特徴がある．

また，SDNN は，どんな関数でも学習できるだけでなく，出力層への結合荷重しか修正しないため，誤差逆伝搬学習法などに比べて，一般に学習の収束が早い．また，学習則が単純で学習のパラメータが少ないため，学習の際の計算量が少なく，パラメータ依存性も低い．

更に，SDNN には，新たなサンプルを追加学習してもカタストロフィック干渉が生じないことが経験的に知られている [11]．実際，本実験でも過去のサンプルの再学習は一切行っていない．このことは，オンライン学習を行う上での利点の一つである．

そのほか，SDNN の大きな特徴として，入力変数の数 l の増加に対して，計算コストが l^2 のオーダーでしか増えないことが挙げられる．RBFN のように局所的な基底関数を用いる方法では，一般に計算コストが l の指数オーダーで増加するのと比べると，これは大きな利点である．

これに加えて，冗長次元の影響を受けにくいという性質がある．新保らは，SDNN-D の入力に，四つの状態変数以外に無関係な変数を幾つか追加する実験を行い，学習効率がわずかしか低下しないことを示した [11]．SDNN-C の場合も同様であるが，冗長だが意味がありそうな変数，例えば $l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$（アクロボットの足先の高さ）や $(\dot{\theta_1})^2 + (\dot{\theta_2})^2$（近似的な運動エネルギー）を追加する実験を行ったところ，学習効率が若干高まるという結果を得ている [16]．

このことは，計算コストの増加があまり大きくないこととあわせて，状態空間の設計をかなり容易にする．すなわち，事前知識（本実験の場合，アクロボットの

ダイナミックスに関する知識）が十分になくても，関係のありそうな変数やセンサ出力を全て入力変数とすれば強化学習を適用できると考えられる．

以上は全て SDNN-C と SDNN-D に共通の性質であるが，両者には不連続な価値関数の表現能力に関して違いがある．

価値関数は，全体として連続であっても，部分的に不連続となることがしばしばある．アクロボットの振り上げ課題の場合でいえば，現在状態がゴール領域かどうかで報酬が不連続的に変わるため，その境界で価値が不連続になるし，ゴールの 1 ステップ前の状態においても不連続な境界が生じるはずである．RBFN のように連続な基底関数を用いた場合，こうした境界付近での近似誤差が大きくなるため，実験 2 においてゴール条件が厳しいときの行動があまりうまく学習できなかったと考えられる．

このような不連続性のうち，状態次元（状態が少し変化したときに価値が不連続的に変化すること）に関しては，SDNN-C と SDNN-D はともに表現可能である．これは，第 4 層の各素子が 0 または 1 の値をとるしきい値素子であることと，状態変数のコードパターン同士で選択的不感化を行うためであり，野中ら [15] の実験結果が示すように，実際にサンプルから不連続な境界を学習することもできる．

しかしながら，SDNN-C の場合，行動次元に関する不連続性は表現できない．つまり，同じ状態で行動値が微妙に変わると価値が不連続的に変化する場合，境界付近での近似誤差が大きくなってしまう．この誤差を小さくするには，近い行動に対するコードパターン間の相関を下げる必要があるが，そうすると必要な素子数が増えるので計算コストは増加する．

一方，SDNN-D は，行動ごとに違う近似器を用いるので，そのような問題はない．ただし，微妙な行動の違いを表現しようとすると，それだけ行動を細かく離散化する必要があるため，多大な計算コストがかかる．また，一般に行動の違いは次の状態の違いに反映されるため，行動のわずかな差による価値の違いが表現できなくても，SDNN-C のように状態の差による価値の違いが表現できれば十分な場合も多いと思われる．

5.2 既存手法との比較

状態空間だけでなく行動空間も連続な場合の Q 学習の先行研究は，状態空間のみ連続な場合に比べて少ない．特に，理論的に検討するだけでなく，関数近似手法を用いて実際に何らかの課題に適用した例は限られる．そのうち主要と思われるものについて，提案手法の特徴を踏まえつつ，特徴や問題点を検討する．

Gaskett ら [17] は，各状態において幾つかの行動の Q 値を MLP によって学習し，それらを代表点としてワイヤーフィッティングを行うことによって，その他の行動に関する Q 値を近似する手法を提案した．この手法は，MLP の出力によって行動値の代表点を適応的に変化させ，その行動に関する Q 値について MLP が学習するという複雑なアルゴリズムを用いている．また，カタストロフィック干渉を防ぐために，過去のサンプルを全て保存しておいて適宜再学習しなければならないし，中間層の素子数のほか多数のパラメータの設定にも手間がかかる．更に，原理的に価値の不連続性は表現できないし，中間層の素子数をあまり増やすと過剰適合が生じるおそれがあるので，不連続点付近の誤差を減らすことも困難である．

Millán ら [18] は，状態空間を適応的に離散化した上で，幾つかの行動に関する Q 値のベクトルを各点において求め，そこから Q 値が最大となる行動値を推定する手法を提案した．この手法は，まず状態空間を局所領域に分割するため，局所的近似手法と同様の問題がある．また，決められた数の離散行動についてのみ Q 値を学習するため，離散化が細かいと計算コストが増大し，離散化が粗いと行動の違いによる価値の違いを十分に表現できない．特に，一定の離散行動の中からまず Q 値が最大となる行動を選び，その近傍の値に基づいて最大値を推定するため，Q 値のピークが鋭い課題だと適切な行動を全く獲得できない可能性がある．パラメータの数も多く，使いやすい方法とは言い難い．

木村 [19] は，状態空間と行動空間にそれぞれタイル（矩形状の局所領域）をランダムに配置し，それらを合成することによって価値関数を近似する手法を提案した．これはタイルコーディング（CMAC とも呼ぶ）の一種とみなせるが，一般にタイルコーディングではタイルの数や形状が関数近似能力に大きく影響する．特に，事前知識がなく完全にランダムにタイルを配置する場合，複雑な価値関数を十分な精度で近似するためには非常に多くのタイルが必要である．また，行動空間にもタイルを配置するため，Q 値が最大になる行動を求める際の計算コストが大きい（この問題に対して木村は Gibbs サンプリングを用いる方法を提案しているが，Gibbs サンプリング自体は関数近似手法とは独立であり，SDNN にも適用可能である）．

Pazis と Parr [20] は，状態価値関数及び行動価値関

数とは異なる価値関数（H 関数）を提案し，これを用いると，行動数が非常に多くても，価値が最大となる行動を少ない計算で求められることを示した．この手法は，最適行動を求めるのに適した H 関数を線形計画法によって構成するため，環境のモデル（任意の状態で任意の行動をとったとき，どの状態に遷移するか）が既知でなければならないが，モデルの代わりに状態遷移のサンプルを使うことによって価値関数に関する線形計画問題を解く別の手法と組み合わせることもできる．ただしこの場合には，基底関数の重み付き和によって価値関数を近似することとなり，そのほかの局所的近似手法と同様の問題を抱えることになると考えられる．

Carden [21] は，Nagraya-Watson カーネル回帰という方法で価値関数を近似し，あるアルゴリズムに従って学習を行ったとき，確率 1 で収束する（最適価値関数との誤差を任意に小さくできる）ことを理論的に示した．しかし，この近似手法は，理論的には扱いやすいが，決して実用的な方法ではない．例えば，ステップごとに新たな局所カーネルを生成するため，ステップ数とともに計算コストがどんどん増加してしまう．この論文では，計算コストの増加を抑えるための補助的な手法として，ある一定数の局所カーネルを用いる方法の検討もなされているが，十分な近似性能をもつよう設定するためには試行錯誤が必要となる．マウンテンカー課題に適用した実験例が示されているが，この課題は状態空間が二次元であり，また最適行動は最大または最小の行動値を常にとるものになる，かなり簡単な課題である．にもかかわらず，安定的にゴールに到達する行動が獲得できるまでには相当な時間がかかり，また獲得された行動は最適行動とはかなり離れたものであった．これは，連続性など理論上の仮定が，実際には満たされないからだと考えられる．

以上のように，連続状態行動空間における Q 学習の既存手法は，SDNN-C に比べると，いずれも実用上の問題点が多い．学習の性能や計算コストのほか，適用できる課題の範囲，パラメータ設定の容易さといった使いやすさを総合的に考慮すると，SDNN-C は既存手法よりも明らかに優れているといえよう．

Q 学習以外で，状態・行動がともに連続な場合に適用可能な強化学習として，actor-critic 法 [1] がある．この方法は，actor によって行動方策が明示的に表現されるため，行動選択の際の計算コストの問題は生じない．しかし，actor をどのように設計するかという問題がある上に，actor と critic の学習が相互に依存しているので，一般に Q 学習などの価値関数のみの方法と比べると学習の収束が遅い．また，学習に影響を与えるパラメータが増えるため，パラメータ設定に手間がかかるという問題もある

本研究においても，当初 actor-critic との比較実験を試みたが，実験した範囲では全て学習に失敗した．ただし，actor の設計やパラメータの設定が不適切だったため失敗した可能性も十分にあるため，手法自体の問題とはいえない．しかしながら，少なくとも使いやすさの点では SDNN-C の方が優れていると思われる．

6. むすび

選択的不感化ニューラルネット（SDNN）の出力層に，多数の行動に関する Q 値を同時に分散表現することによって，行動価値関数を効率的に近似する方法を提案した．また，制約条件を課して学習を難しくしたアクロボットの振り上げ課題についてシミュレーション実験を行い，提案手法の有効性を示した．

本手法は，同様に SDNN を用いた新保らの手法と比べて，学習効率の低下や計算コストの増加をほとんど生じることなく，行動値を十分細かく離散化することができる．また，RBFN のような局所的近似手法と比べても，状態空間が高次元の場合にはるかに少ない計算コストで済む．

また，SDNN はパラメータ依存性が小さいなど，実用上の利点を多く備える．Q 学習のアルゴリズムの単純さとあわせて，提案手法は連続な状態行動空間における強化学習の手法として既存のものより使いやすく，強化学習の適用範囲を拡大するものだといえる．

今後の課題として，まず本手法をより多くの課題に適用し，どのような場合に有効なのか明らかにすることが挙げられる．また，行動空間が多次元の場合への拡張は重要な課題である．原理的には行動次元が増えても現在の方法をほぼそのまま適用できるが，計算コストが大きく増加してしまう．そのため，関数近似器の改良とは別に，行動選択に Gibbs サンプリングを用いる [19], [22] 手法を併用するなどして計算コストを抑える必要があると思われる．そのほか，actor-critic 法など Q 学習以外の方法との比較実験についても，今後実施したいと考えている．

謝辞　本研究は，JSPS 科研費（課題番号 22300079, 24760308，26590173）の支援を受けた．

文　献

[1] R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.

[2] C.J.C.H. Watkins, “Learning from delayed rewards,” Ph.D. thesis, University of Cambridge, 1989.

[3] C.J.C.H. Watkins and P. Dayan, “Q–learning,” Machine Learning, vol.8, pp.279–292, 1992.

[4] G. Konidaris, S. Osentoski, and P. Thomas, “Value function approximation in reinforcement learning using the Fourier basis,” Proc. the Twenty-Fifth Conf. on Artificial Intelligence, pp.380–385, 2011.

[5] A. Geramifard, M. Bowling, and R.S. Sutton, “Incremental least-square temporal difference learning,” Proc. the twenty first Conf. on American Association for Artificial Intelligence, pp.356–361, 2006.

[6] G. Taylor and R. Parr, “Kernelized value function approximation for reinforcement learning,” Proc. the 26th Annual International Conf. on Machine Learning, pp.1017–1024, 2009.

[7] B. Irie and S. Miyake, “Capabilities of three-layered perceptrons,” Proc. IEEE International Conf. on Neural Networks, vol.1, pp.641–648, 1988.

[8] K. Funahashi, “On the approximate realization of continuous mappings by neural networks,” Neural Networks, vol.2, pp.183–192, 1989.

[9] M. McCloskey and N. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in The Psychology of Learning and Motivation, ed. G.H. Bower, vol.24, pp.109–164, Academic Press, NY, 1989.

[10] J. Park and I.W. Sandberg, “Universal approximation using radial-basis-function networks,” Neural Computation, vol.3, no.2, pp.246–257, 1991.

[11] 新保智之，山根　健，田中文英，森田昌彦，“選択的不感化ニューラルネットを用いた強化学習の価値関数近似,” 信学論（D），vol.J93-D, no.6, pp.837–847, June 2010.

[12] 森田昌彦，村田和彦，諸上茂光，末光厚夫，“選択的不感化法を適用した層状ニューラルネットの情報統合能力,” 信学論（D-II），vol.J87-D-II, no.12, pp.2242–2252, Dec. 2004.

[13] M.W. Spong, “The swing up control problem for the acrobot,” IEEE Control Systems, vol.15, no.1, pp.49–55, 1995.

[14] P. Auer, H. Burgsteiner, and W. Maass, “A learning rule for very simple universal approximators consisting of a single layer of perceptrons,” Neural Networks, vol.21, no.5, pp.786–795, June 2008.

[15] 野中和明，田中文英，森田昌彦，“階層型ニューラルネットの 2 変数関数近似能力の比較,” 信学論（D），vol.J94-D, no.12, pp.2114–2125, Dec. 2011.

[16] T. Kobayashi, T. Shibuya, and M. Morita, “Q-learning in continuous state-action space with redundant dimensions by using a selective desensitization neural network,” Proc. SCIS & ISIS 2014, pp.801–806, Kitakyushu, Japan, Dec. 2014.

[17] C. Gaskett, D. Wettergreen, and A. Zelinsky, “Q-learning in continuous state and action spaces,” Proc. the 12th Australian Joint Conf. on Articial Intelligence, pp.417–428, Sydney, Australia, Dec. 1999.

[18] J.D.R. Millán, D. Posenato, and E. Dedieu, “Continuous-action Q-learning,” Machine Learning, vol.49, no.2–3, pp.247–265, 2002.

[19] 木村　元，“ランダムタイリングと Gibbs-sampling を用いた多次元状態-行動空間における強化学習,” 計測自動制御学会論文集，vol.42, no.12, pp.1336–1343, 2006.

[20] J. Pazis and R. Parr, “Generalized value functions for large action sets,” Proc. the International Conf. on Machine Learning, 2011.

[21] S. Carden, “Convergence of a Q-learning variant for continuous states and actions,” J. Artificial Intelligence Research, vol.49, pp.705–731, 2014.

[22] B. Sallans and G.E. Hinton, “Reinforcement learning with factored states and actions,” J. Machine Learning Research, vol.5, pp.1063–1088, 2004.

（平成 26 年 5 月 14 日受付，8 月 23 日再受付，
10 月 9 日早期公開）

小林　高彰

平 22 筑波大・工学システム学類卒．現在，同大大学院博士課程システム情報工学研究科在学中．神経回路モデルの研究に従事．

澁谷　長史　（正員）

平 17 横浜国立大・工・電子情報工学科飛び級のため退学．平 22 同大学院工学府博士課程後期修了．平 19 より日本学術振興会特別研究員 DC1．平 22 より筑波大学大学院システム情報工学研究科助教．強化学習アルゴリズムの研究に従事．博士（工学）．

森田　昌彦　（正員）

昭 61 東大・工・計数卒．平 3 同大大学院博士課程了．日本学術振興会特別研究員，東京大学工学部助手を経て，平 4 筑波大学電子・情報工学系講師．同大機能工学系助教授などを経て，平 19 より同大大学院システム情報工学研究科教授．脳の情報処理機構及び神経回路網による情報処理の研究に従事．平 5 日本神経回路学会研究費，平 6 同学会論文賞，平 11 日本心理学会研究奨励賞受賞．